Факультет	Математики, физики и информатики	
Кафедра	Информатики и информационных технологий	
Направление подготовки	09.03.03 Прикладная информатика	
Направленность (профиль)	Прикладная информатика в здравоохранении	
Параллельное программирование		Б1.В.ДВ.12

Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Тульский государственный педагогический университет им. Л.Н. Толстого»
ФГБОУ ВО «ТГПУ им. Л.Н. Толстого»

УТВЕРЖДЕНА
на заседании Ученого совета университета
протокол № 2 от 11 февраля 2016 г.

Рабочая программа дисциплины «Параллельное программирование»

Трудоемкость: 3 зачетные единицы

Квалификация (степень) выпускника: бакалавр

Форма обучения: очная

Рассмотрена на заседании кафедры
информатики и информационных технологий
протокол № 4 от 24 декабря 2015 г.

Заведующий кафедрой _____А.В. Якушин

Одобрена на заседании Ученого совета факультета
Математики, физики и информатики
протокол № 6 от 21 января 2016 г.

Декан _____И.Ю. Реброва

СОДЕРЖАНИЕ

1. ПЕРЕЧЕНЬ ПЛАНИРУЕМЫХ РЕЗУЛЬТАТОВ ОБУЧЕНИЯ ПО ДИСЦИПЛИНЕ, СООТНЕСЕННЫХ С ПЛАНИРУЕМЫМИ РЕЗУЛЬТАТАМИ ОСВОЕНИЯ ОБРАЗОВАТЕЛЬНОЙ ПРОГРАММЫ

Достижение планируемых результатов обучения, соотнесенных с общими целями и задачами ОПОП, является целью освоения дисциплины.

Планируемые результаты освоения образовательной программы (код и название компетенции)	Планируемые результаты обучения	Этапы формирования компетенции в процессе освоения образовательной программы
готовностью к концептуальному, функциональному и логическому проектированию систем среднего и крупного масштаба и сложности (ДПК-8)	**Выпускник знает:** архитектурные принципы реализации параллельной обработки в вычислительных машинах критерии оценки эффективности параллельных программ и их ограничения методы и языковые механизмы конструирования параллельных программ параллельные вычислительные методы **Умеет:** разрабатывать параллельные программы с использованием библиотеки MPI осуществлять перенос реализованных программных средств на параллельные платформы проводить сравнительный анализ последовательных и параллельных программных средств и оценивать их эффективность **Владеет**: навыками конструирования и настройки высокопроизводительных вычислительных систем	2 этап из 2 (6 семестр)

2. МЕСТО ДИСЦИПЛИНЫ В СТРУКТУРЕ ОПОП БАКАЛАВРИАТА

Дисциплина «Параллельное программирование» относится к дисциплинам по выбору вариативной части образовательной программы. Изучение данной дисциплины осуществляется в 6 семестре. Изучение данной дисциплины базируется на освоении студентами дисциплин «Вычислительная математика», «Структуры и алгоритмы компьютерной обработки данных».

К началу изучения дисциплины студенты должны владеть:

- знаниями

основных методов, способов и средств обработки данных

- умениями

работать с системами управления базами данных

навыками и (или) опытом деятельности

методами программирования на алгоритмическом языке C/C++

Дисциплина «Параллельное программирование» является базовой для дисциплин «Системы компьютерной математики» и «Теория систем и системный анализ».

3. ОБЪЕМ ДИСЦИПЛИНЫ И ВИДЫ УЧЕБНОЙ РАБОТЫ

Очная форма обучения

Вид учебной работы	**Объем зачетных единиц / часов по формам обучения**
Максимальная учебная нагрузка (всего)	108/3
Контактная работа обучающихся с преподавателем (всего)	22
в том числе:	
лекции	8
лабораторные занятия (включая защиту отчета по лабораторным работам)	
семинарские занятия	
практические занятия	12
контрольные работы	
другие виды контактной работы (КСРС)	2
Самостоятельная работа студента (всего)	86
в том числе:	
внеаудиторная самостоятельная работа по подготовке к лекционным занятиям	22
внеаудиторная самостоятельная работа по подготовке к лабораторным занятиям и защите отчета	
внеаудиторная самостоятельная работа при подготовке к семинарским и/или практическим занятиям	30
подготовка учебного проекта	
подготовка к контрольной работе	
выполнение заданий для самостоятельной работы в системе управления обучением MOODLE	30
выполнение курсового проекта (работы)	
подготовка к зачету	4
подготовка к экзамену	
другие виды самостоятельной работы студента	
Промежуточная аттестация в форме зачета	

4. СОДЕРЖАНИЕ ДИСЦИПЛИНЫ, СТРУКТУРИРОВАННОЕ ПО ТЕМАМ (РАЗДЕЛАМ) С УКАЗАНИЕМ ОТВЕДЕННОГО НА НИХ КОЛИЧЕСТВА АКАДЕМИЧЕСКИХ ИЛИ АСТРОНОМИЧЕСКИХ ЧАСОВ И ВИДОВ УЧЕБНЫХ ЗАНЯТИЙ

Наименование тем (разделов).	Количество академических или астрономических часов по видам учебных занятий			
	Занятия лекционного типа	Занятия семинарского типа	Другие виды учебных занятий	Самостоятельная работа обучающихся
Тема 1. Введение в предмет	2	4		20
Тема 2. Вычислительные системы с массовым параллелизмом	2	2		20
Тема 3. Параллельное программирование в MPI.	2	4		22
Тема 4. Реализация параллельных алгоритмов	2	2		20
Контроль самостоятельной работы студентов			2	
Индивидуальные консультации				
Подготовка к зачету				4
Групповые консультации				
ИТОГО	8	12	2	86

Тема 1. Введение в предмет

Формальные модели параллельного программирования. Параллельные схемы программ. Информационный базис и схема управления. Асинхронные вычислительные процессы над информационным базисом.

Неуправляемые вычислительные процессы. Схема управления параллельной схемы программы.

Управляющий автомат и особенности автоматных функций. Эквивалентность вычислительных процессов по произвольному отношению. Детерминизм и отношение "большей параллельности" схем управления.

Тема 2. Вычислительные системы с массовым параллелизмом

Определение и примеры. Особенности применения: моделирующие вычислительные системы и системы реального времени. Примеры. Характерные особенности таких систем.

Классификация ВС по Флину. Реконфигурируемые системы MIMD архитектуры.

Статическая и динамическая реконфигурация. Оценка производительности массивно параллельных ВС. Причины снижения производительности.

Тема 3. Параллельное программирование в MPI.

Структура библиотеки MPI. Организация обмена сообщениями. Модели вычислений

Кластерные технологии. Организация кластера на базе локальной сети.

Тема 4. Реализация параллельных алгоритмов

Параллельные алгоритмы. Общие способы распараллеливания алгоритмов. Организация параллельного исполнения рекурсивных вычислений. Параллельные численные алгоритмы линейной алгебры. Способы разбиения матриц (горизонтальная, вертикальная, блочные схемы).

Методы вычисления произведения матриц с использованием разных схем разбиения матриц. Обеспечение предельно допустимого параллелизма. Обращение матриц. Параллельные методы решения систем линейных уравнений. Параллельные численные алгоритмы решения дифференциальных уравнений в частных производных.

5. ПЕРЕЧЕНЬ УЧЕБНО-МЕТОДИЧЕСКОГО ОБЕСПЕЧЕНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ ОБУЧАЮЩИХСЯ ПО ДИСЦИПЛИНЕ

Преподавание дисциплины предполагает использование следующего учебно-методического обеспечения.

Комплекта мультимедийных презентаций для лекционных занятий.

Теоретического курса и информационных приложений, размещенных в электронной образовательной среде MOODLe.

Комплекса заданий для практических занятий, размещенных в электронной образовательной среде MOODLe.

Виды самостоятельной работы обучающихся: выполнение заданий на практические занятия, выполнение индивидуального проектного задания.

При подготовке к занятиям и выполнении самостоятельной работы студентам доступны следующие учебно-методические ресурсы, перечисленные в п.7 рабочей программы, а также электронный учебный ресурс размещенный в среде электронного обучения ТГПУ им. Л.Н. Толстого (http://moodle.tsput.ru)

6. ФОНД ОЦЕНОЧНЫХ СРЕДСТВ ДЛЯ ПРОВЕДЕНИЯ ПРОМЕЖУТОЧНОЙ АТТЕСТАЦИИ ОБУЧАЮЩИХСЯ ПО ДИСЦИПЛИНЕ

6.1. Перечень компетенций с указанием этапов их формирования в процессе освоения образовательной программы

Перечень планируемых результатов обучения по дисциплине, соотнесенных с планируемыми результатами освоения образовательной программы представлен в таблице пункта 1 рабочей программы.

Формирование компетенции “готовность к концептуальному, функциональному и логическому проектированию систем среднего и крупного масштаба и сложности (ДПК-8)” осуществляется в течение двух этапов освоения основной образовательной программы.

Первый этап формирования компетенции осуществляется в процессе освоения дисциплин «Системы искусственного интеллекта», «Математические методы и модели».

Второй этап формирования компетенции осуществляется в процессе освоения дисциплин «Компьютерное моделирование», «Компьютерная графика», «Параллельное программирование», «Обработка и представление результатов исследований».

6.2. Описание показателей и критериев оценивания компетенций на различных этапах их формирования, описание шкал оценивания

Дескриптор компетенций	Показатели оценивания	Критерии оценивания

Знания	архитектурных принципов реализации параллельной обработки в вычислительных машинах; критерий оценки эффективности параллельных программ и их ограничения; методов и языковых механизмов конструирования параллельных программ; параллельных вычислительных методов.	Отметка «зачтено» выставляется, если студент в целом за семестр набрал от 61 до 100 баллов (с учетом баллов, набранных на промежуточной аттестации (зачете)). Отметка «незачтено» выставляется, если студент в целом за семестр набрал менее 61 балла (с учетом баллов, набранных на промежуточной аттестации (зачете)).
Умения	разрабатывать параллельные программы с использованием библиотеки MPI; осуществлять перенос реализованных программных средств на параллельные платформы; проводить сравнительный анализ последовательных и параллельных программных средств и оценивать их эффективность	
Навыки и опыт деятельности	конструирования и настройки высокопроизводительных вычислительных систем.	

Критерии оценивания компетенций формируются на основе балльно-рейтинговой системы с помощью всего комплекса методических материалов, определяющих процедуры оценивания знаний, умений, навыков и опыта деятельности, характеризующих данный этап формирования компетенций.

Баллы, набранные студентом в течение семестра	Баллы за промежуточную аттестацию (зачет)	Общая сумма баллов за модуль в семестр	Отметка
21 – 60	0 – 40	61-100	Зачтено
0 – 20	0 – 40	0 – 60	Не зачтено

Оценка «*зачтено*» ставится, если студент освоил программный материал всех разделов, последователен в изложении программного материала, достаточно последовательно и логически стройно его излагает, умеет увязывать теорию с практикой, успешно прошел текущий контроль успеваемости по дисциплине, продемонстрировал индивидуальные знания, умениями и навыки практической работы.

Оценка «*не зачтено*» ставится, если студент не знает значительной части программного материала, допускает существенные ошибки, непоследователен в его изложении, не прошел текущий контроль успеваемости, не в полной мере владеет необходимыми знаниями, умениями и навыками при выполнении практических заданий, то есть студент не может продолжить обучение без дополнительной подготовки по соответствующей дисциплине.

6.3. Типовые контрольные задания или иные материалы, необходимые для оценки знаний, умений, навыков и (или) опыта деятельности, характеризующих этапы формирования компетенций в процессе освоения образовательной программы

Образцы заданий к практическим занятиям:

1. Написать программу, используя коммуникационные функции (MPI_Ssend, MPI_Bsend, MPI_Rsend, MPI_Isend, MPI_Irecv), передающие одномерные и двумерные массивы (вектора и матрицы) между двумя процессорами
2. Провести сравнение по скорости передачи данных в зависимости от применяемых функций и размера передаваемых данных

3. Написать программу, используя коммуникационную функцию (MPI_Bcast), реализующую алгоритм передачи данных от 0 процесса всем остальным
4. Написать программу, используя коммуникационную функцию (MPI_Gather), реализующую алгоритм передачи частей массива от всех процессоров на 0.
5. Написать программу, используя коммуникационную функцию (MPI_Allgather), реализующую алгоритм передачи частей массива от всех процессоров на все процессора.

Вопросы к зачету

1. Что такое параллельное программирование и суперкомпьютеры. Области в которых может возникать потребность в параллельных вычислениях. Особенности параллельных вычислений.
2. Увеличение производительности при параллельных вычислениях. К каким областям задач может быть применено распараллеливание к каким нет. Закон Амдала.
3. Ускорение параллельного алгоритма.
4. Эффективность параллельного алгоритма.
5. Распараллеливание арифметических выражений. Алгоритм Винограда.
6. Разделение на подзадачи. Установление связей между отдельными подзадачами. Объединение мелких подзадач в большие, законченные счетные единицы (агломерация).
7. Средства поддержки параллельной работы: параллелизм процессов, механизм синхронизации и механизм разделения ресурсов, семафоры, обмен сообщениями.
8. Библиотека MPI. Инициализация, завершение. Базовые функции.
9. Точечные обмены. Синхронный и асинхронный режим.
10. Буферизация сообщений.
11. Отложенные операции. Функции инициализации и завершения операции.
12. Коллективные взаимодействия. Передача сообщений
13. Управление группами и коммуникатором.
14. Коллективные взаимодействия. Прием сообщений
15. Создание и освобождение коммуникатора.
16. Типы данных в MPI.
17. Пакет mpich. Установка и настройка кластера под управлением Linux.
18. Простейшие параллельные алгоритмы и их степень параллелизма.
19. Параллельный алгоритм скалярного умножения векторов и его ускорение по сравнению с последовательным алгоритмом.
20. Параллельный алгоритм умножения матрицы на вектор и его ускорение по сравнению с последовательным алгоритмом.
21. Параллельный алгоритм умножения матрицы на матрицу и его ускорение по сравнению с последовательным алгоритмом.
22. Параллельный алгоритм решения СЛАУ прямым методом Гаусса и его ускорение по сравнению с последовательным алгоритмом.
23. Технологии параллельного программирования.
24. Кластерные технологии.

Индивидуальное проектное задание заключается в разработке параллельной программы:

Каждый проект независимо от темы и задания должен предоставлять пользователю выбор количества узлов, на которых будет запущен процесс вычислений, а также выводить, время, за которое они произошли.

1. Нахождение кратчайших путей для ориентированного графа

Исходной информацией для задачи является взвешенный граф, G=(V,R) содержащий n вершин (|V|=n), в котором каждому ребру графа приписан неотрицательный вес. Граф будем полагать ориентированным, т.е., если из вершины i есть ребро в вершину j, то из этого не следует наличие ребра из j в i. В случае, если вершины все же соединены взаимообратными ребрами, то веса, приписываемые им, могут не совпадать. Для имеющегося графа G требуется найти минимальные длины путей между каждой парой вершин графа.

Входные данные начинаются N количеством вершин от 0 до 10000 и M количеством ребер в графе от 0 до 10000 через пробел, затем в M строках записаны номера вершин соединенных i-ым ребром и его вес от 0 до 10000 , где i – номер строки, начиная счет со второй строки.

Выходные данные должны содержать все кратчайшие пути для всех пар вершин, если такой путь существует для пары. Каждый путь должен быть записан на отдельной строке в виде последовательности номеров ребер через пробел в том порядке, в которм они включены в путь, и к ним добавить последнее после знака «=» число, показывающее вес пути, в начале строки должны быть помещены номера вершин, соединенных этим путем («I J : »). В случае отсутствия пути между вершинами поставить вместо последовательности вершин прочерк.

Пример входных(input.txt) и выходных(output.txt) данных:

input.txt	**output.txt**
5 4 1 2 1 1 3 8 1 4 3 2 3 2 3 1 3 4 2 1	1 2 : 1 =1 1 3 : 1 4 = 4 1 4 : 3 = 3 1 5 : - 2 1 : - 2 3 : 4 = 3 2 4 : - 2 5 : - 3 1 : 5 = 3 3 2 : - 3 4 : - 3 5 : - 4 1 : - 4 2 : 6 = 1 4 3 : - 4 5 :- 5 1 : - 5 2 : - 5 3 : - 5 4 : -

Примечание: в случае если наименьший путь между парой вершин не является единственным, считать кратчайшим тот, который содержит наименьшее количество вершин, если же данное условие не дает единственное решение, вывести любой из них.

2. Решение систем линейных уравнений методом алгоритма Гаусса, рекурсивное удвоение Стоуна

Задача: дана расширенная (добавлен веткор-столбец свободных членов) матрица коэффициентов системы линейных уравнений, найти решение системы.

Входной файл в первой строке содержит N количество строк и M количество столбцов матрицы через пробел, далее в N содержится M элементов (числа от -10000 до 10000) матрицы через пробел. Порядок расположения элементов матриц внутри файла соответствует порядку расположения элементов в самой матрице.

$-10000 \leq N, M \leq 10000$

Выходной файл должен содержать значение переменных, совокупность которых является решением данной системы линейных уравнений, через пробел. Порядок расположения значений переменных должен соответствовать порядку номеров их переменных. В случае невозможности нахождения единственного решения вывести в файл сообщение «Error Data». В случае решения системы равного пустому множеству вывести сообщение «No Solutions»

Пример входных(input.txt) и выходных(output.txt) данных:

input.txt	**output.txt**
1 2 -7 21	-3

input.txt	**output.txt**
2 3 1 1 5 1 -2 -4	2 3

Критерии оценки проектов

Составляющие проекта	Критерии для оценивания	Максимальное количество балов
Постановка проблемы и ее обоснованность, формулирование целей и задач	• общественная значимость и актуальность выдвинутых проблем; • соответствие темы, цели и задач проекта; разумность масштаба работ.	10
Содержание проекта/ проектной разработки	• логичность, взаимосвязь и последовательность этапов проекта; • адекватность предлагаемых мероприятий решению поставленных задач; • корректность используемых методов работы; • четкость определения целевой группы и обоснованность её участия при реализации проекта; • соответствие теоретической, эмпирической и проектной частей, их связь с практикой и выбранным видом профессиональной деятельности; • соблюдение заявленных временных рамок реализации проекта; самостоятельность и активность участника проекта.	10
Результат выполнения прикладного проекта	• соответствие ожиданий от проекта / планируемого результата полученному продукту; • степень решения заявленной проблемы; • успешность преодоления трудностей в реализации проекта; • оценка участников целевой группы;	10

	• перспективы развития проекта после завершения проекта; возможность тиражирования проекта.	
Презентация результатов работы над прикладным проектом	• ясность, логичность, профессионализм изложения доклада; • наглядность и структурированность материала презентации; умение корректно использовать профессиональную лексику и понятийно-категориальный аппарат.	10
Ответы на вопросы	• степень владения темой; • ясность аргументации взглядов студента, презентующего результаты выполнения проекта; четкость и лаконичность ответов на вопросы.	10

6.4. Методические материалы, определяющие процедуры оценивания знаний, умений, навыков и (или) опыта деятельности, характеризующих этапы формирования компетенций

Описание балльно-рейтинговой системы по дисциплине.

Итоговая рейтинговая оценка по дисциплине складывается из следующих составляющих:

1) В течении семестра за выполнение заданий по курсу студент может максимально получить 60 баллов.;

2) Обязательной формой текущей аттестации знаний является выполнение индивидуального проектного задания 20 баллов.

3) На зачёте ответ студента может быть максимально оценен в 40 баллов.

При этом, для получения положительной итоговой оценки на зачете необходимо получить не менее 60% по каждой составляющей и выполнить все задания для практических занятий. Шкала перевода баллов в оценку: до 60 - «не зачтено»; 61 - 100 - «зачтено».

№ п/п	Критерии оценивания	Максимальное количество баллов	Баллы, полученные студентом
1.	Выполнение заданий:	60	
1.1.	Практические занятия	40	
1.2.	Индивидуальное проектное задание	20	
3.	Зачет	40	
	ИТОГО:	100	

7. ПЕРЕЧЕНЬ ОСНОВНОЙ И ДОПОЛНИТЕЛЬНОЙ УЧЕБНОЙ ЛИТЕРАТУРЫ, НЕОБХОДИМОЙ ДЛЯ ОСВОЕНИЯ ДИСЦИПЛИНЫ

7.1. Основная литература

1. Основы параллельного программирования [Электронный ресурс] : учебное пособие / К. Ю. Богачёв. - М : [б. и.], 2013. - 344 с. - ISBN 978-5-9963-0939-9 : Б. ц. URL:http://biblioclub.ru/index.php?page=book&id=214157
2. Мартынюк, Ю. М. Методы программирования [Текст] : учебное пособие / Ю. М. Мартынюк, С. С. Гербут, В. С. Ванькова ; рец.: Е. Г. Торина, Е. А. Снижко ; ФГБОУ

ВПО "Тульский государственный педагогический университет им. Л. Н. Толстого". - Тула : Изд-во ТГПУ им. Л. Н. Толстого, 2013. - 70 с.

7.2. Дополнительная литература

1. Основы информатики и начала программирования [Текст] : учебное пособие / В. Г. Куперман, Е. Г. Торина, Изд. 2-е, перераб. и доп. - Тула : Изд-во ТГПУ им. Л. Н. Толстого, 1997. - 264 с. - ISBN 5879541169
2. Объектно-ориентированное программирование на C++ [Текст] / Айра Пол. - 2-е изд. - М. : Бином, 1999. - 462 с. : ил. - ISBN 5798901408
3. Технологии параллельного программирования [Текст] : учебное пособие для студентов вузов / С. А. Лупин, М. А. Посыпкин. - М : Форум, 2008. - 208 с. - ISBN 9785819903360
4. C++.Объектно-ориентированное программирование [Текст] : учебное пособие для студентов вузов / В. В. Лаптев, А. В. Морозов, А. В. Бокова. - С П б. : Питер, 2008. - 464 с. : ил. - ISBN 9785911802004

8. ПЕРЕЧЕНЬ РЕСУРСОВ ИНФОРМАЦИОННО-ТЕЛЕКОММУНИКАЦИОННОЙ СЕТИ «ИНТЕРНЕТ», НЕОБХОДИМЫХ ДЛЯ ОСВОЕНИЯ ДИСЦИПЛИНЫ

1. Math-Net.Ru [Электронный ресурс] : общероссийский математический портал / Математический институт им. В. А. Стеклова РАН ; Российская академия наук, Отделение математических наук. - М. : [б. и.], 2010. - Загл. с титул. экрана. - Б. ц.
   URL: http://www.mathnet.ru
2. ИКТ [Электронный ресурс] : федеральный образовательный портал / ФГАУ ГНИИ ИТТ "Информика". - М. : [б. и.], 2003. - Загл. с титул. экрана. - Б. ц.
   URL: http://www.ict.edu.ru
3. Университетская библиотека Online [Электронный ресурс] : электронная библиотечная система / ООО "Директ-Медиа" . - М. : [б. и.], 2001. - Загл. с титул. экрана. - Б. ц.
   URL: www.biblioclub.ru
4. Универсальные базы данных East View [Электронный ресурс] : информационный ресурс / East View Information Services. - М. : [б. и.], 2012. - Загл. с титул. экрана. - Б. ц.
   URL: www.ebiblioteka.ru
5. Научная электронная библиотека eLIBRARY.RU [Электронный ресурс] : информационный портал / ООО "РУНЭБ" ; Санкт-Петербургский государственный университет. - М. : [б. и.], 2005. - Загл. с титул. экрана. - Б. ц.
   URL: www.eLibrary.ru

9. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ДЛЯ ОБУЧАЮЩИХСЯ ПО ОСВОЕНИЮ ДИСЦИПЛИНЫ

Приступая к изучению новой учебной дисциплины, студенты должны ознакомиться с учебной программой, учебной, научной и методической литературой, имеющейся в библиотеке университета, встретиться с преподавателем, ведущим дисциплину, получить в библиотеке рекомендованные учебники и учебно-методические пособия, осуществить запись на соответствующий курс в среде электронного обучения университета.

Глубина усвоения дисциплины зависит от активной и систематической работы студента на лекциях и практических занятиях, а также в ходе самостоятельной работы, по изучению рекомендованной литературы.

На лекциях важно сосредоточить внимание на ее содержании. Это поможет лучше воспринимать учебный материал и уяснить взаимосвязь проблем по всей дисциплине. Основное содержание лекции целесообразнее записывать в тетради в виде ключевых фраз, понятий, тезисов, обобщений, схем, опорных выводов. Необходимо обращать внимание на термины, формулировки, раскрывающие содержание тех или иных явлений и процессов, научные выводы и практические рекомендации. Желательно оставлять в конспектах поля, на которых делать пометки из рекомендованной литературы, дополняющей материал прослушанной лекции, а также подчеркивающие особую важность тех или иных теоретических положений. С целью уяснения теоретических положений, разрешения спорных ситуаций необходимо задавать преподавателю уточняющие вопросы. Для закрепления содержания лекции в памяти, необходимо во время самостоятельной работы внимательно прочесть свой конспект и дополнить его записями из учебников и рекомендованной литературы. Конспектирование читаемых лекций и их последующая доработка способствует более глубокому усвоению знаний, и поэтому являются важной формой учебной деятельности студентов.

Прочное усвоение и долговременное закрепление учебного материала невозможно без продуманной самостоятельной работы. Такая работа требует от студента значительных усилий, творчества и высокой организованности. В ходе самостоятельной работы студенты выполняют следующие задачи: дорабатывают лекции, изучают рекомендованную литературу, готовятся к практическим занятиям, к коллоквиуму, контрольным работам по отдельным темам дисциплины. При этом эффективность учебной деятельности студента во многом зависит от того, как он распорядился выделенным для самостоятельной работы бюджетом времени.

Результатом самостоятельной работы является прочное усвоение материалов по предмету согласно программы дисциплины. В итоге этой работы формируются профессиональные умения и компетенции, развивается творческий подход к решению возникших в ходе учебной деятельности проблемных задач, появляется самостоятельности мышления.

Целью практических занятий по данной дисциплине является закрепление теоретических знаний, полученных при изучении дисциплины.

При подготовке к практическому занятию целесообразно выполнить следующие рекомендации: изучить основную литературу; ознакомиться с дополнительной литературой, новыми публикациями в периодических изданиях: журналах, газетах и т. д.; при необходимости доработать конспект лекций. При этом учесть рекомендации преподавателя и требования учебной программы.

При выполнении практических занятий основным методом обучения является самостоятельная работа студента под управлением преподавателя. На них пополняются теоретические знания студентов, их умение творчески мыслить, анализировать, обобщать изученный материал, проверяется отношение студентов к будущей профессиональной деятельности.

Оценка выполненной работы осуществляется преподавателем комплексно: по результатам выполнения заданий, устному сообщению и оформлению работы. После подведения итогов занятия студент обязан устранить недостатки, отмеченные преподавателем при оценке его работы.

Преподавание дисциплины должно включать в себя следующие образовательные технологии:

1) Проведение лекций с использованием презентаций на основе мультимедийных технологий;
2) Обеспечение студентов сопутствующими материалами, размещенными среде Moodle;

Примерная тематика практических занятий по дисциплине.

Полные варианты практических занятий размещены в в системе управления обучением MOODLE.

№	Наименование практических занятий	Объем в часах
1	Кластерные технологии	2
2	Основы MPI	2

3	Простейшие параллельные алгоритмы	4
4	Распараллеливание вычислений.	4
	Итого	12

Типовые задания для самостоятельной работы по дисциплине

1. Нахождение кратчайших путей для ориентированного графа
2. Нахождение кратчайших путей для неориентированного графа
3. Нахождение минимального остового дерева
4. Сортировка Шелла
5. Быстрая сортировка
6. Умножение матриц делением на полосы (строки или столбцы) — ленточная схема
7. Умножение матриц делением на блоки (подматрицы) — блочное представление, алгоритм Фокса
8. Умножение матриц делением на блоки (подматрицы) — блочное представление, алгоритм Кэннона
9. Решение систем линейных уравнений методом алгоритма Гаусса, рекурсивное удвоение Стоуна
10. Последовательная и каскадная суммы
11. Вычисление определенного интеграла: метод Гаусса
12. Вычислить сумму ряда

10. ПЕРЕЧЕНЬ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, ИСПОЛЬЗУЕМЫХ ПРИ ОСУЩЕСТВЛЕНИИ ОБРАЗОВАТЕЛЬНОГО ПРОЦЕССА ПО ДИСЦИПЛИНЕ, ВКЛЮЧАЯ ПЕРЕЧЕНЬ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ И ИНФОРМАЦИОННЫХ СПРАВОЧНЫХ СИСТЕМ

Материально-техническое обеспечение дисциплины:

1. Специально оборудованные аудитории и компьютерные классы: персональные компьютеры (модели: Intel Pentium4, AMD Athlon, AMD Duron), мультимедийные проекторы, аудиовизуальные устройства;
2. Программное обеспечение в соответствии с программой курса;
3. Методические пособия и литература в библиотеке университета и на кафедре.
4. Студентам обеспечен доступ к сети Internet.

Перечень лицензионного программного обеспечения, используемого при освоении дисциплины:

1. Подписка Microsoft DreamSpark Premium - Сублицензионный договор № S-2042626/M18 от 04.06.2013:
   - 1.1. Средства для разработки и проектирования Visual Studio 2008, 2010, 2012 и 2013 Professional Editions;
   - 1.2. Операционная система Windows 7 Professional;
   - 1.3. Операционная система Windows 8 Pro;
   - 1.4. Операционная система Windows 8.1 Pro;
   - 1.5. Отдельные программы из Office 2007, Office 2010, Office 2013 (в том числе Access, Visio, Project и др.);
2. Свободное программное обеспечение по лицензии GNU
   - 2.1. Debian Linux Weezy
   - 2.2. Apache Web Server
   - 2.3. MySQL
   - 2.4. PHP 5.0
   - 2.5. Domain Technologie Control Контрольная панель локального хостинга

У обучающихся имеется доступ (удаленный доступ), в том числе в случае применения электронного обучения, дистанционных образовательных технологий, к современным профессиональным базам данных и информационным справочным системам, состав которых определяется в рабочих программах дисциплин и подлежит ежегодному обновлению:

1. Компьютерная информационно-правовая система «Гарант» - регистрационный номер клиента 71-70685-000033.
2. Официальный интернет-портал правовой информации http://pravo.gov.ru.
3. Портал Федеральных государственных образовательных стандартов высшего образования http://fgosvo.ru.
4. Math-Net.Ru [Электронный ресурс] : общероссийский математический портал / Математический институт им. В. А. Стеклова РАН ; Российская академия наук, Отделение математических наук. - М. : [б. и.], 2010. - Загл. с титул. экрана. - Б. ц. URL: http://www.mathnet.ru
5. ИКТ [Электронный ресурс] : федеральный образовательный портал / ФГАУ ГНИИ ИТТ "Информика". - М. : [б. и.], 2003. - Загл. с титул. экрана. - Б. ц. URL: http://www.ict.edu.ru
6. Университетская библиотека Online [Электронный ресурс] : электронная библиотечная система / ООО "Директ-Медиа" . - М. : [б. и.], 2001. - Загл. с титул. экрана. - Б. ц. URL: www.biblioclub.ru
7. Универсальные базы данных East View [Электронный ресурс] : информационный ресурс / East View Information Services. - М. : [б. и.], 2012. - Загл. с титул. экрана. - Б. ц. URL: www.ebiblioteka.ru
8. Научная электронная библиотека eLIBRARY.RU [Электронный ресурс] : информационный портал / ООО "РУНЭБ" ; Санкт-Петербургский государственный университет. - М. : [б. и.], 2005. - Загл. с титул. экрана. - Б. ц. URL: www.eLibrary.ru

11. ОПИСАНИЕ МАТЕРИАЛЬНО-ТЕХНИЧЕСКОЙ БАЗЫ, НЕОБХОДИМОЙ ДЛЯ ОСУЩЕСТВЛЕНИЯ ОБРАЗОВАТЕЛЬНОГО ПРОЦЕССА ПО ДИСЦИПЛИНЕ

1. Учебные аудитории для проведения занятий лекционного типа, оборудованные мультимедийными средствами обучения.
2. Учебные аудитории для проведения практических занятий.
3. Компьютерные классы с доступом в интернет для работы с информационно-правовыми системами, в том числе «Гарант» и с доступом к электронно-библиотечной системе.
4. Аудитории для самостоятельной работы студентов, оснащенные компьютерной техникой, имеющей доступ к информационно-телекоммуникационной сети «Интернет», электронной информационно-образовательной среде ТГПУ им. Л.Н. Толстого, внутривузовскому сетевому окружению.

12. АННОТАЦИЯ РАБОЧЕЙ ПРОГРАММЫ ДИСЦИПЛИНЫ.

1.Планируемые результаты обучения при освоении дисциплины, соотнесенные с планируемыми результатами освоения образовательной программы.

В результате освоения дисциплины у студента должна быть сформирована следующая компетенция: готовность к концептуальному, функциональному и логическому проектированию систем среднего и крупного масштаба и сложности (ДПК-8).

В результате освоения дисциплины студент должен приобрести:

знания архитектурных принципов реализации параллельной обработки в вычислительных машинах; критерий оценки эффективности параллельных программ и их ограничения; методов и языковых механизмов конструирования параллельных программ; параллельных вычислительных методов.

умения разрабатывать параллельные программы с использованием библиотеки MPI; осуществлять перенос реализованных программных средств на параллельные платформы; проводить сравнительный анализ последовательных и параллельных программных средств и оценивать их эффективность

навыки конструирования и настройки высокопроизводительных вычислительных систем.

2.Место дисциплины в структуре ОПОП.

Дисциплина «Параллельное программирование» относится к дисциплинам по выбору вариативной части образовательной программы. Изучение данной дисциплины осуществляется в 6 семестре.

3.Объем дисциплины: 3 зачетные единицы.

4.Образовательный процесс осуществляется на русском языке.

5.Разработчик: Якушин А.В., к.п.н., доцент, зав. кафедрой И и ИТ.

13. ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ К РАБОЧЕЙ ПРОГРАММЕ ДИСЦИПЛИНЫ

1) Внесены изменения в п.7 «Перечень основной и дополнительной учебной литературы, необходимой для освоения дисциплины».

2) Обновлен п.10 «Перечень информационных технологий, используемых при осуществлении образовательного процесса по дисциплине, включая перечень лицензионного программного обеспечения и информационных справочных систем» на основании действующих лицензионных соглашений

Заведующий кафедрой ИиИТ _____ А.В. Якушин

«26» августа 2016 г..

ПРОГРАММА СОСТАВЛЕНА В СООТВЕТСТВИИ С ТРЕБОВАНИЯМИ ФГОС ВО.

Разработчик (и):

Фамилия, имя, отчество	Учёная степень	Учёное звание	Должность	Дата разработки	Подпись
Якушин Алексей Валериевич	к.п.н.	Доцент	Зав. кафедрой информатики и информационных технологий		

ФОНД ОЦЕНОЧНЫХ СРЕДСТВ
по дисциплине «Параллельное программирование»

Состав:

1. ПЕРЕЧЕНЬ КОМПЕТЕНЦИЙ С УКАЗАНИЕМ ЭТАПОВ ИХ ФОРМИРОВАНИЯ В ПРОЦЕССЕ ОСВОЕНИЯ ОБРАЗОВАТЕЛЬНОЙ ПРОГРАММЫ

.

Планируемые результаты освоения образовательной программы (код и название компетенции)	Планируемые результаты обучения	Этапы формирования компетенции в процессе освоения образовательной программы
готовностью к концептуальному, функциональному и логическому проектированию систем среднего и крупного масштаба и сложности (ДПК-8)	Выпускник знает: архитектурные принципы реализации параллельной обработки в вычислительных машинах критерии оценки эффективности параллельных программ и их ограничения методы и языковые механизмы конструирования параллельных программ параллельные вычислительные методы Умеет: разрабатывать параллельные программы с использованием библиотеки MPI осуществлять перенос реализованных программных средств на параллельные платформы проводить сравнительный анализ последовательных и параллельных программных средств и оценивать их эффективность Владеет: навыками конструирования и настройки высокопроизводительных вычислительных систем	2 этап из 2 (6 семестр)

Формирование компетенции “готовность к концептуальному, функциональному и логическому проектированию систем среднего и крупного масштаба и сложности (ДПК-8)” осуществляется в течение двух этапов освоения основной образовательной программы.

Первый этап формирования компетенции осуществляется в процессе освоения дисциплин «Системы искусственного интеллекта», «Математические методы и модели».

Второй этап формирования компетенции осуществляется в процессе освоения дисциплин «Компьютерное моделирование», «Компьютерная графика», «Параллельное программирование», «Обработка и представление результатов исследований».

2. ОПИСАНИЕ ПОКАЗАТЕЛЕЙ И КРИТЕРИЕВ ОЦЕНИВАНИЯ КОМПЕТЕНЦИЙ НА РАЗЛИЧНЫХ ЭТАПАХ ИХ ФОРМИРОВАНИЯ, ОПИСАНИЕ ШКАЛ ОЦЕНИВАНИЯ

Дескриптор	**Показатели оценивания**	**Критерии оценивания**

компетенций		
Знания	архитектурных принципов реализации параллельной обработки в вычислительных машинах; критерий оценки эффективности параллельных программ и их ограничения; методов и языковых механизмов конструирования параллельных программ; параллельных вычислительных методов.	Отметка «зачтено» выставляется, если студент в целом за семестр набрал от 61 до 100 баллов (с учетом баллов, набранных на промежуточной аттестации (зачете)). Отметка «незачтено» выставляется, если студент в целом за семестр набрал менее 61 балла (с учетом баллов, набранных на промежуточной аттестации (зачете)).
Умения	разрабатывать параллельные программы с использованием библиотеки MPI; осуществлять перенос реализованных программных средств на параллельные платформы; проводить сравнительный анализ последовательных и параллельных программных средств и оценивать их эффективность	
Навыки и опыт деятельности	конструирования и настройки высокопроизводительных вычислительных систем.	

Критерии оценивания компетенций формируются на основе балльно-рейтинговой системы с помощью всего комплекса методических материалов, определяющих процедуры оценивания знаний, умений, навыков и опыта деятельности, характеризующих данный этап формирования компетенций.

Баллы, набранные студентом в течение семестра	Баллы за промежуточную аттестацию (зачет)	Общая сумма баллов за модуль в семестр	Отметка
21 – 60	0 – 40	61-100	Зачтено
0 – 20	0 – 40	0 – 60	Не зачтено

3. КОНТРОЛЬНЫЕ ЗАДАНИЯ ИЛИ ИНЫЕ МАТЕРИАЛЫ, НЕОБХОДИМЫЕ ДЛЯ ОЦЕНКИ ЗНАНИЙ, УМЕНИЙ, НАВЫКОВ И ОПЫТА ДЕЯТЕЛЬНОСТИ, ХАРАКТЕРИЗУЮЩИХ ЭТАПЫ ФОРМИРОВАНИЯ КОМПЕТЕНЦИЙ В ПРОЦЕССЕ ОСВОЕНИЯ ОБРАЗОВАТЕЛЬНОЙ ПРОГРАММЫ

3.1. Вопросы к зачету

Вопросы к зачету

1. Что такое параллельное программирование и суперкомпьютеры. Области в которых может возникать потребность в параллельных вычислениях. Особенности параллельных вычислений.
2. Увеличение производительности при параллельных вычислениях. К каким областям задач может быть применено распараллеливание к каким нет. Закон Амдала.
3. Ускорение параллельного алгоритма.
4. Эффективность параллельного алгоритма.
5. Распараллеливание арифметических выражений. Алгоритм Винограда.

6. Разделение на подзадачи. Установление связей между отдельными подзадачами. Объединение мелких подзадач в большие, законченные счетные единицы (агломерация).
7. Средства поддержки параллельной работы: параллелизм процессов, механизм синхронизации и механизм разделения ресурсов, семафоры, обмен сообщениями.
8. Библиотека MPI. Инициализация, завершение. Базовые функции.
9. Точечные обмены. Синхронный и асинхронный режим.
10. Буферизация сообщений.
11. Отложенные операции. Функции инициализации и завершения операции.
12. Коллективные взаимодействия. Передача сообщений
13. Управление группами и коммуникатором.
14. Коллективные взаимодействия. Прием сообщений
15. Создание и освобождение коммуникатора.
16. Типы данных в MPI.
17. Пакет mpich. Установка и настройка кластера под управлением Linux.
18. Простейшие параллельные алгоритмы и их степень параллелизма.
19. Параллельный алгоритм скалярного умножения векторов и его ускорение по сравнению с последовательным алгоритмом.
20. Параллельный алгоритм умножения матрицы на вектор и его ускорение по сравнению с последовательным алгоритмом.
21. Параллельный алгоритм умножения матрицы на матрицу и его ускорение по сравнению с последовательным алгоритмом.
22. Параллельный алгоритм решения СЛАУ прямым методом Гаусса и его ускорение по сравнению с последовательным алгоритмом.
23. Технологии параллельного программирования.
24. Кластерные технологии.

Критерии оценки зачета по дисциплине

Оценка «*зачтено*» ставится, если студент освоил программный материал всех разделов, последователен в изложении программного материала, достаточно последовательно и логически стройно его излагает, умеет увязывать теорию с практикой, успешно прошел текущий контроль успеваемости по дисциплине, продемонстрировал индивидуальные знания, умениями и навыки практической работы.

Оценка «*не зачтено*» ставится, если студент не знает значительной части программного материала, допускает существенные ошибки, непоследователен в его изложении, не прошел текущий контроль успеваемости, не в полной мере владеет необходимыми знаниями, умениями и навыками при выполнении практических заданий, то есть студент не может продолжить обучение без дополнительной подготовки по соответствующей дисциплине.

3.2. Индивидуальные проектные задания

3.2.1. Список индивидуальных заданий

1. Нахождение кратчайших путей для ориентированного графа
2. Нахождение кратчайших путей для неориентированного графа
3. Нахождение минимального остового дерева
4. Сортировка Шела
5. Быстрая сортировка
6. Умножение матриц делением на полосы (строки или столбцы) — ленточная схема
7. Умножение матриц делением на блоки (подматрицы) — блочное представление, алгоритм Фокса

8. Умножение матриц делением на блоки (подматрицы) — блочное представление, алгоритм Кэннона
9. Решение систем линейных уравнений методом алгоритма Гаусса, рекурсивное удвоение Стоуна
10. Последовательная и каскадная суммы
11. Вычисление определенного интеграла: метод Гаусса

12-13. Вычислить сумму ряда

$$\sum_{i=1}^{6}\sum_{j=1}^{6}\frac{(-1)^{i+1}\cos^{j+1}x}{j(i+1)}$$

$$\sum_{i=1}^{12}\sum_{j=1}^{10}\frac{(-1)^{j-1}}{(i+1)(3/2)^{i}}$$

14-24. Вычислить сумму ряда

$\sum_{n=1}^{10}\frac{n^3+2}{n^5+\sin 2^n}$
$\sum_{n=1}^{15}\frac{1}{\sqrt[5]{n+1}}\sin\frac{1}{\sqrt{n}}$
$\sum_{n=1}^{12}\frac{\arcsin\frac{3+(-1)^n}{4}}{2^n+n}$
$\sum_{n=1}^{10}\frac{\sin\frac{\pi}{2n+1}}{n\left(3+\sin\frac{\pi n}{4}\right)}$
$\sum_{n=1}^{13}\frac{\ln n}{n^3+n+1}$
$\sum_{n=1}^{15}\frac{n^2+3}{n^3(2+\sin(n\pi/2))}$
$\sum_{n=2}^{10}\frac{\arcsin\frac{n-1}{n}}{\sqrt[3]{n^3-3n}}$
$\sum_{n=1}^{13}\frac{\cos^2(n\pi/2)}{n(n+1)(n+2)}$
$\sum_{n=1}^{20}\frac{\ln n}{\sqrt[3]{n^7}}$

$\sum_{n=2}^{15} \frac{n+1}{\left(\sqrt[3]{n}-1\right)\left(n\sqrt[4]{n^3}-1\right)}$
$\sum_{n=1}^{10} \frac{(-1)^n}{\cos\frac{\pi}{3\sqrt{n}}\sqrt[3]{3n+\ln n}}$

3.2.2.Требования к индивидуальным заданиям

1) Постановка задачи.
2) Обзор литературы по задаче
3) Последовательная реализация
4) Параллельная реализация
5) Сравнение времени выполнения
6) Оценка ускорения работы алгоритма
7) Анализ сложности алгоритма (и прогноз на расширение ресурсов и объемов данных)
8) Описание используемых функций библиотеки MPI
9) Список литературы

Предложены задачи низкого и высокого уровня сложности:

- Задачи низкого уровня сложности (до 10 баллов)
- Задачи высокого уровня сложности (до 20 баллов)

3.2.3. Перечень индивидуальных проектных заданий

Задачи высокого уровня сложности (до 20 баллов)
Нахождение кратчайших путей для ориентированного графа

Исходной информацией для задачи является взвешенный граф, G=(V,R) содержащий n вершин (|V|=n), в котором каждому ребру графа приписан неотрицательный вес. Граф будем полагать ориентированным, т.е., если из вершины i есть ребро в вершину j, то из этого не следует наличие ребра из j в i. В случае, если вершины все же соединены взаимообратными ребрами, то веса, приписываемые им, могут не совпадать. Для имеющегося графа G требуется найти минимальные длины путей между каждой парой вершин графа.

Входные данные начинаются N количеством вершин от 0 до 10000 и M количеством ребер в графе от 0 до 10000 через пробел, затем в M строках записаны номера вершин соединенных i-ым ребром и его вес от 0 до 10000 , где i – номер строки, начиная счет со второй строки.

Выходные данные должны содержать все кратчайшие пути для всех пар вершин, если такой путь существует для пары. Каждый путь должен быть записан на отдельной строке в виде последовательности номеров ребер через пробел в том порядке, в которм они включены в путь, и к ним добавить последнее после знака «=» число, показывающее вес пути, в начале строки должны быть помещены номера вершин, соединенных этим путем («I J : »). В случае отсутствия пути между вершинами поставить вместо последовательности вершин прочерк.

Пример входных(input.txt) и выходных(output.txt) данных:

input.txt	**output.txt**

5 4 1 2 1 1 3 8 1 4 3 2 3 2 3 1 3 4 2 1	1 2 : 1 =1 1 3 : 1 4 = 4 1 4 : 3 = 3 1 5 : - 2 1 : - 2 3 : 4 = 3 2 4 : - 2 5 : - 3 1 : 5 = 3 3 2 : - 3 4 : - 3 5 : - 4 1 : - 4 2 : 6 = 1 4 3 : - 4 5 :- 5 1 : - 5 2 : - 5 3 : - 5 4 : -

Примечание: в случае если наименьший путь между парой вершин не является единственным, считать кратчайшим тот, который содержит наименьшее количество вершин, если же данное условие не дает единственное решение, вывести любой из них.

Нахождение кратчайших путей для неориентированного графа

Исходной информацией для задачи является взвешенный граф, G=(V,R) содержащий n вершин (|V|=n), в котором каждому ребру графа приписан неотрицательный вес. Граф будем полагать неориентированным, т.е., если из вершины i есть ребро в вершину j, то из этого следует наличие ребра из j в i. В случае, если вершины все же соединены взаимообратными ребрами, то веса, приписываемые им, могут не совпадать. Для имеющегося графа G требуется найти минимальные длины путей между каждой парой вершин графа.

Входные данные начинаются N количеством вершин от 0 до 10000 и M количеством ребер в графе от 0 до 10000 через пробел, затем в M строках записаны номера вершин соединенных i-ым ребром и его вес от 0 до 10000, где i – номер строки, начиная счет со второй строки.

Выходные данные должны содержать все кратчайшие пути для всех пар вершин, если такой путь существует для пары. Каждый путь должен быть записан на отдельной строке в виде последовательности номеров ребер через пробел в том порядке, в котором они включены в путь, и к ним добавить последнее после знака «=» число, показывающее вес пути, в начале строки должны быть помещены номера вершин, соединенных этим путем («I J : »). В случае отсутствия пути между вершинами поставить вместо последовательности вершин прочерк.

Пример входных(input.txt) и выходных(output.txt) данных:

input.txt	**output.txt**
5 4 1 2 1 1 3 8 1 4 3 2 3 2 3 1 3	1 2 : 1 =1 1 3 : 5 = 3 1 4 : 1 6 =2 1 5 : - 2 1 : 1 = 1 2 3 : 4 = 3

4 2 1	2 4 : 6 = 1 2 5 : - 3 1 : 5 = 3 3 2 : 4 = 3 3 4 : 4 6 = 4 3 5 : - 4 1 : 6 1 = 2 4 2 : 6 = 1 4 3 : 6 4 = 4 4 5 :- 5 1 : - 5 2 : - 5 3 : - 5 4 : -

Примечание: в случае если наименьший путь между парой вершин не является единственным, считать кратчайшим тот, который содержит наименьшее количество вершин, если же данное условие не дает единственное решение, вывести любой из них.

Нахождение минимального остового дерева

Охватывающим деревом или остовом неориентированного графа G называется подграф T графа G, который является деревом и содержит все вершины из G. Определив вес подграфа для взвешенного графа как сумму весов входящих в подграф дуг, под минимально охватывающим деревом будем понимать охватывающие дерево минимального веса. Для графа G найдите МОД.

Входные данные начинаются N количеством вершин от 0 до 10000 и M количеством ребер в графе от 0 до 10000 через пробел, затем в M строках записаны номера вершин соединенных i-ым ребром и его вес от 0 до 10000, где i – номер строки, начиная счет со второй строки.

Выходные данные должны содержать МОД в виде последовательности ребер и общий вес МОД после знака «=», в случае отсутствия МОД ответ «-».

Пример входных(input.txt) и выходных(output.txt) данных:

input.txt	**output.txt**
44 1 2 1 1 3 8 1 4 3 2 3 2 3 1 3 4 2 1	2 4 6 = 6

input.txt	**output.txt**
5 4 1 2 1 1 3 8 1 4 3 2 3 2 3 1 3 4 2 1	-

Примечание: в случае если МОД не являться единственным, вывести любое из них.

Сортировка Шелла

Общая идея сортировки Шелла состоит в сравнении на начальных стадиях сортировки пар значений, располагаемых достаточно далеко друг от друга в упорядочиваемом наборе данных. Такая модификация метода сортировки позволяет быстро переставлять далекие неупорядоченные пары значений (сортировка таких пар обычно требует большого количества перестановок, если используется сравнение только соседних элементов).

Общая схема метода состоит в следующем. На первом шаге алгоритма происходит упорядочивание элементов *n/2* пар (a_i, $a_{n/2+i}$) для $1 \le i \le n/2$. Далее на втором шаге упорядочиваются элементы в *n/4* группах из четырех элементов (a_i, $a_{n/4+1}$, $a_{n/2+1}$, $a_{3n/4+1}$) для $1 \le i \le n/4$. На третьем шаге упорядочиваются элементы уже в *n/4* группах из восьми элементов и т.д. На последнем шаге упорядочиваются элементы сразу во всем массиве (a_1, a_2,..., a_n). На каждом шаге для упорядочивания элементов в группах используется метод сортировки вставками. Как можно заметить, общее количество итераций алгоритма Шелла является равным $log_2 n$.

```
ShellSort(double A[], int n){
    int incr = n/2;
    while( incr > 0 ) {
        for ( int i=incr+1; i<n; i++ ) {
            j = i-incr;
            while ( j > 0 )
                if ( A[j]> A[j+incr] ){
                    swap(A[j], A[j+incr]);
                    j = j - incr;
                }
                else j = 0;
        }
        incr = incr/2;
    }
}
```

Для алгоритма Шелла может быть предложен параллельный аналог метода, если топология коммуникационный сети может быть представлена в виде N-мерного гиперкуба (т.е. количество процессоров равно $p=2^N$). Выполнение сортировки в таком случае может быть разделено на два последовательных этапа. На первом этапе (N итераций) осуществляется взаимодействие процессоров, являющихся соседними в структуре гиперкуба (но эти процессоры могут оказаться далекими при линейной нумерации; для установления соответствия двух систем нумерации процессоров может быть использован, код Грея). Формирование пар процессоров, взаимодействующих между собой при выполнении операции "сравнить и разделить", может быть обеспечено при помощи следующего простого правила – на каждой итерации i, $0 \le i < N$, парными становятся процессоры, у которых различие в битовых представлении их номеров имеется только в позиции N-i.

Второй этап состоит в реализации обычных итераций параллельного алгоритма четнечетной перестановки. Итерации данного этапа выполняются до прекращения фактического изменения сортируемого набора и, тем самым, общее количество L таких итераций может быть различным - от 2 до p.

С учетом представленного описания параллельного варианта алгоритма Шелла базовая подзадача для организации параллельных вычислений, может быть определена на основе операции "сравнить и разделить". Как результат, количество подзадач всегда совпадает с числом имеющихся процессоров (размер блоков данных в подзадачах равен *n/p*) и не возникает проблемы масштабирования. Распределение блоков упорядочиваемого набора данных по процессорам должно быть выбрано с учетом возможности эффективного выполнения операций "сравнить и разделить" при представлении топологии сети передачи данных в виде гиперкуба.

Входные данные представляют собой две строки: первая строка содержит количество чисел от 0 до 100000 во второй строке, вторая строка содержит неупорядоченную последовательность случайных действительных чисел от -10000 до 10000 через пробел.

Выходные данные должны отсортированную по возрастанию вводную последовательность через пробел.

Пример входных(input.txt) и выходных(output.txt) данных:

input.txt	**output.txt**
5 3 2 -4 6 999.01	-4 2 3 6 999.01

Быстрая сортировка

Параллельное обобщение алгоритма быстрой сортировки наиболее простым способом может быть получено для вычислительной системы с топологией в виде *N*-мерного гиперкуба (т.е. $p=2^N$). Пусть, как и ранее, исходный набор данных распределен между процессорами блоками одинакового размера *n/p*; результирующее расположение блоков должно соответствовать нумерации процессоров гиперкуба. Возможный способ выполнения первой итерации параллельного метода при таких условиях может состоять в следующем:

- выбрать каким-либо образом ведущий элемент и разослать его по всем процессорам системы (например, в качестве ведущего элемента можно взять среднее арифметическое элементов, расположенных на выбранном ведущем процессоре);
- разделить на каждом процессоре имеющийся блок данных на две части с использованием полученного ведущего элемента;
- образовать пары процессоров, для которых битовое представление номеров отличается только в позиции *N*, и осуществить взаимообмен данными между этими процессорами; в результате таких пересылок данных на процессорах, для которых в битовом представлении номера бит позиции *N* равен 0, должны оказаться части блоков со значениями, меньшими ведущего элемента; процессоры с номерами, в которых бит *N* равен 1, должны собрать, соответственно, все значения данных, превышающие значение ведущего элемента.

В результате выполнения такой итерации сортировки исходный набор оказывается разделенным на две части, одна из которых (со значениями меньшими, чем значение ведущего элемента) располагается на процессорах, в битовом представлении номеров которых бит *N* равен 0. Таких процессоров всего *p/2* и, таким образом, исходный *N* -мерный гиперкуб также оказывается разделенным на два гиперкуба размерности *N-1*. К этим подгиперкубам, в свою очередь, может быть параллельно применена описанная выше процедура. После *N* - кратного повторения подобных итераций для завершения сортировки достаточно упорядочить блоки данных, получившиеся на каждом отдельном процессоре вычислительной системы

В качестве *базовой подзадачи* для организации параллельных вычислений может быть выбрана операция "сравнить и разделить", а количество подзадач совпадает с числом используемых процессоров. Распределение подзадач по процессорам должно производиться с учетом возможности эффективного выполнения алгоритма при представлении топологии сети передачи данных в виде гиперкуба.

Входные данные представляют собой две строки: первая строка содержит количество чисел от 0 до 100000 во второй строке, вторая строка содержит неупорядоченную последовательность случайных действительных чисел от -10000 до 10000 через пробел.

Выходные данные должны отсортированную по возрастанию вводную последовательность через пробел.

Пример входных(input.txt) и выходных(output.txt) данных:

input.txt	output.txt
5 3 2 -4 6 999.01	-4 2 3 6 999.01

Умножение матриц делением на полосы (строки или столбцы) — ленточная схема

При ленточной схеме разделения данных исходные матрицы разбиваются на горизонтальные (для матрицы *A*) и вертикальные (для матрицы *B*) полосы. Получаемые полосы распределяются по процессорам, при этом на каждом из имеющегося набора процессоров располагается только по одной полосе матриц *A* и *B*. Перемножение полос (а выполнение процессорами этой операции может быть выполнено параллельно) приводит к получению части блоков результирующей матрицы *C*. Для вычисления оставшихся блоков матрицы *C* сочетания полос матриц *A* и *B* на процессорах должны быть изменены. В наиболее простом виде это может быть обеспечено, например, при кольцевой топологии вычислительной сети (при числе процессоров, равном количеству полос) – в этом случае необходимое для матричного умножения изменение положения данных может быть обеспечено циклическим сдвигом полос матрицы *B* по кольцу. После многократного выполнения описанных действий (количество необходимых повторений является равным числу процессоров) на каждом процессоре получается набор блоков, образующий горизонтальную полосу матрицы *C*.

Рассмотренная схема вычислений позволяет определить параллельный алгоритм матричного умножения при ленточной схеме разделения данных как итерационную процедуру, на каждом шаге которой происходит параллельное выполнение операции перемножения полос и последующего циклического сдвига полос одной из матриц по кольцу. Метод основан на представлении матрицы непрерывными наборами (горизонтальными полосами) строк (или столбцов). При таком способе разделения данных в качестве базовой подзадачи может быть выбрана операция скалярного умножения одной строки матрицы на вектор, столбец второй матрицы.

Задача: даны 2 матрицы, найти их произведение, используя ленточную схему.

Входной файл в первой строке содержит N_1 количество строк и M_1 количество столбцов первой матрицы через пробел, далее в N_1 содержится M_1элементов (числа от -10000 до 10000) первой матрицы через пробел, затем следующая строка содержит N_2 количество строк и M_2 количество столбцов второй матрицы через пробел, далее в N_2 содержится M_2элементов (числа от -10000 до 10000) первой матрицы через пробел. Порядок расположения элементов матриц внутри файла соответствует порядку расположения элементов в самой матрице.

$-10000 \le N_1, N_2, M_1, M_2 \le 10000$

Выходной файл должен содержать в первой строке N_3 количество строк и M_3 количество столбцов в результирующей матрице через пробел, далее в N_3 содержится M_3элементов этой матрицы через пробел. Порядок расположения элементов матрицы внутри файла должен соответствовать порядку расположения элементов в самой матрице.

Пример входных(input.txt) и выходных(output.txt) данных:

input.txt	output.txt
3 2 1 2 3 4 5 6 2 3 7 8 9 -1 -2 -3	3 3 5 4 3 17 16 15 29 8 27

Умножение матриц делением на блоки (подматрицы) — блочное представление, алгоритм Фокса

При блочном представлении данных параллельная вычислительная схема матричного умножения в наиболее простом виде может быть представлена, если топология вычислительной сети имеет вид прямоугольной решетки (если реальная топология сети имеет иной вид, представление сети в виде решетки можно обеспечить на логическом уровне). Основные положения параллельных методов для блочно представленных матриц состоят в следующем:

- Каждый из процессоров решетки отвечает за вычисление одного блока матрицы *C*,
- В ходе вычислений на каждом из процессоров располагается по одному блоку исходных матриц *A* и *B*,
- При выполнении итераций алгоритмов блоки матрицы *A* последовательно сдвигаются вдоль строк процессорной решетки, а блоки матрицы *B* − вдоль столбцов решетки,
- В результате вычислений на каждом из процессоров вычисляется блок матрицы *C*, при этом общее количество итераций алгоритма равно *p* (где *p* – число процессоров),

При *блочном* разделении матрица делится на прямоугольные наборы элементов – при этом, как правило, используется разделение на непрерывной основе. Пусть количество процессоров составляет $p=s\cdot q$, количество строк матрицы является кратным *s*, а количество столбцов – кратным *q*, то есть $m=k\cdot s$ и $n=l\cdot q$. Представим исходную матрицу *A* в виде набора прямоугольных блоков следующим образом:

$$A=\begin{pmatrix} A_{00} & A_{02} & \dots A_{0q-1} \\ & \dots & \\ A_{s-11} & A_{s-12} & \dots A_{s-1q-1} \end{pmatrix},$$

где A_{ij} - блок матрицы, состоящий из элементов:

$$A_{ij}=\begin{pmatrix} a_{i_0j_0} & a_{i_0j_1} & \dots a_{i_0j_{l-1}} \\ & \dots & \\ a_{i_{k-1}j_0} & a_{i_{k-1}j_1} & a_{i_{k-1}j_{l-1}} \end{pmatrix}, i_v=ik+v, 0\le v<k, k=m/s,\ j_u=jl+u, 0\le u\le l, l=n/q$$

При таком подходе целесообразно, чтобы вычислительная система имела физическую или, по крайней мере, логическую топологию процессорной решетки из *s* строк и *q* столбцов. В этом случае при разделении данных на непрерывной основе процессоры, соседние в структуре решетки, обрабатывают смежные блоки исходной матрицы. Следует отметить, однако, что и для блочной схемы может быть применено циклическое чередование строк и столбцов.

При использовании блочного представления матрицы *A* базовые подзадачи целесообразно определить на основе вычислений, выполняемых над матричными блоками. Для нумерации подзадач могут использоваться индексы располагаемых в подзадачах блоков матрицы *A*, т.е. подзадача *(i,j)* содержит блок A_{ij}. Помимо блока матрицы *A* каждая подзадача должна содержать также и блок вектора *b*. При этом для блоков одной и той подзадачи должны соблюдаться определенные правила соответствия – операция умножения блока матрицы A_{ij} может быть выполнена только, если блок вектора *b'(i,j)* имеет вид

$b'(i,j)=(b'_0(i,j),\dots,b'_{l-1}(i,j))$, где $b'_u(i,j)=b_{j_u}$, $j_u=jl+u, 0\le u<l, l=n/q$

Рассмотрим общую схему параллельных вычислений для операции умножения матрицы на вектор при блочном разделении исходных данных. После перемножения блоков матрицы *A* и вектора *b* каждая подзадача *(i,j)* будет содержать вектор частичных результатов *c'(i,j)*, определяемый в соответствии с выражениями

$$c'_v(i,j) = \sum_{u=0}^{l-1} a_{i_v j_u} b_{j_u},\ i_v = ik + v,\ 0 \le v < k,\ k = m/s,\ \ j_u = jl + u,\ 0 \le u \le l,\ l = n/q$$

Поэлементное суммирование векторов частичных результатов для каждой горизонтальной полосы блоков матрицы *A* позволяет получить результирующий вектор *c*

$$c_\eta = \sum_{j=0}^{q-1} c'_v(i,j),\ 0 \le \eta < m,\ i = \eta / s,\ v = \eta - i \cdot s$$

Для размещения вектора *c* применим ту же схему, что и для исходного вектора *b*: организуем вычисления таким образом, чтобы при завершении расчетов вектор *c* располагался поблочно в каждой из вертикальных полос блоков матрицы *A* (тем самым, каждый блок вектора *c* должен быть продублирован по каждой горизонтальной полосе).

Рассмотрев представленную схему параллельных вычислений, можно сделать вывод, что информационная зависимость базовых подзадач проявляется только на этапе суммирования результатов перемножения блоков матрицы *A* и блоков вектора *b*. Выполнение таких расчетов может быть выполнено по обычной каскадной схеме и, как результат, характер имеющихся информационных связей для подзадач одной и той же горизонтальной полосы блоков соответствует структуре двоичного дерева.

Задача: даны 2 матрицы, найти их произведение, используя блочную схему и алгоритм Фокса

Входной файл в первой строке содержит N_1 количество строк и M_1 количество столбцов первой матрицы через пробел, далее в N_1 содержится M_1элементов (числа от -10000 до 10000) первой матрицы через пробел, затем следующая строка содержит N_2 количество строк и M_2 количество столбцов второй матрицы через пробел, далее в N_2 содержится M_2элементов (числа от -10000 до 10000) первой матрицы через пробел. Порядок расположения элементов матриц внутри файла соответствует порядку расположения элементов в самой матрице.

$-10000 \le N_1, N_2, M_1, M_2 \le 10000$

Выходной файл должен содержать в первой строке N_3 количество строк и M_3 количество столбцов в результирующей матрице через пробел, далее в N_3 содержится M_3элементов этой матрицы через пробел. Порядок расположения элементов матрицы внутри файла должен соответствовать порядку расположения элементов в самой матрице. В случае невозможности данного произведения вывести в файл сообщение «Error Matrix Format».

Пример входных(input.txt) и выходных(output.txt) данных:

input.txt	**output.txt**
3 2 1 2 3 4 5 6 2 3 7 8 9 -1 -2 -3	3 3 5 4 3 17 16 15 29 28 27

Умножение матриц делением на блоки (подматрицы) — блочное представление, алгоритм Кэннона

При блочном представлении данных параллельная вычислительная схема матричного умножения в наиболее простом виде может быть представлена, если топология вычислительной сети имеет вид прямоугольной решетки (если реальная топология сети имеет иной

вид, представление сети в виде решетки можно обеспечить на логическом уровне). Основные положения параллельных методов для блочно представленных матриц состоят в следующем:

- Каждый из процессоров решетки отвечает за вычисление одного блока матрицы *C*,
- В ходе вычислений на каждом из процессоров располагается по одному блоку исходных матриц *A* и *B*,
- При выполнении итераций алгоритмов блоки матрицы *A* последовательно сдвигаются вдоль строк процессорной решетки, а блоки матрицы *B* − вдоль столбцов решетки,
- В результате вычислений на каждом из процессоров вычисляется блок матрицы *C*, при этом общее количество итераций алгоритма равно *p* (где *p* – число процессоров),

При *блочном* разделении матрица делится на прямоугольные наборы элементов – при этом, как правило, используется разделение на непрерывной основе. Пусть количество процессоров составляет $p=s\cdot q$, количество строк матрицы является кратным *s*, а количество столбцов – кратным *q*, то есть $m=k\cdot s$ и $n=l\cdot q$. Представим исходную матрицу *A* в виде набора прямоугольных блоков следующим образом:

$$A=\begin{pmatrix} A_{00} & A_{02} & \dots A_{0q-1} \\ & \dots & \\ A_{s-11} & A_{s-12} & \dots A_{s-1q-1} \end{pmatrix},$$

где A_{ij} - блок матрицы, состоящий из элементов:

$$A_{ij}=\begin{pmatrix} a_{i_0j_0} & a_{i_0j_1} & \dots a_{i_0j_{l-1}} \\ & \dots & \\ a_{i_{k-1}j_0} & a_{i_{k-1}j_1} & a_{i_{k-1}j_{l-1}} \end{pmatrix}, i_v=ik+v, 0\le v<k, k=m/s,\ j_u=jl+u, 0\le u\le l, l=n/q$$

При таком подходе целесообразно, чтобы вычислительная система имела физическую или, по крайней мере, логическую топологию процессорной решетки из *s* строк и *q* столбцов. В этом случае при разделении данных на непрерывной основе процессоры, соседние в структуре решетки, обрабатывают смежные блоки исходной матрицы. Следует отметить, однако, что и для блочной схемы может быть применено циклическое чередование строк и столбцов.

При использовании блочного представления матрицы *A* базовые подзадачи целесообразно определить на основе вычислений, выполняемых над матричными блоками. Для нумерации подзадач могут использоваться индексы располагаемых в подзадачах блоков матрицы *A*, т.е. подзадача *(i,j)* содержит блок A_{ij}. Помимо блока матрицы *A* каждая подзадача должна содержать также и блок вектора *b*. При этом для блоков одной и той подзадачи должны соблюдаться определенные правила соответствия – операция умножения блока матрицы A_{ij} может быть выполнена только, если блок вектора *b'(i,j)* имеет вид

$$b'(i,j)=(b'_0(i,j),\dots,b'_{l-1}(i,j)),\ \text{где}\ b'_u(i,j)=b_{j_u},\ j_u=jl+u, 0\le u<l, l=n/q$$

Рассмотрим общую схему параллельных вычислений для операции умножения матрицы на вектор при блочном разделении исходных данных. После перемножения блоков матрицы *A* и вектора *b* каждая подзадача *(i,j)* будет содержать вектор частичных результатов *c'(i,j)*, определяемый в соответствии с выражениями

$$c'_v(i,j)=\sum_{u=0}^{l-1} a_{i_vj_u}b_{j_u},\ i_v=ik+v, 0\le v<k, k=m/s,\ j_u=jl+u, 0\le u\le l, l=n/q$$

Поэлементное суммирование векторов частичных результатов для каждой горизонтальной полосы блоков матрицы *A* позволяет получить результирующий вектор *c*

$$c_\eta = \sum_{j=0}^{q-1} c'_\nu (i, j),\ 0 \le \eta < m,\ i = \eta / s,\ \nu = \eta - i \cdot s$$

Для размещения вектора *c* применим ту же схему, что и для исходного вектора *b*: организуем вычисления таким образом, чтобы при завершении расчетов вектор *c* располагался поблочно в каждой из вертикальных полос блоков матрицы *A* (тем самым, каждый блок вектора *c* должен быть продублирован по каждой горизонтальной полосе).

Рассмотрев представленную схему параллельных вычислений, можно сделать вывод, что информационная зависимость базовых подзадач проявляется только на этапе суммирования результатов перемножения блоков матрицы *A* и блоков вектора *b*. Выполнение таких расчетов может быть выполнено по обычной каскадной схеме и, как результат, характер имеющихся информационных связей для подзадач одной и той же горизонтальной полосы блоков соответствует структуре двоичного дерева.

Задача: даны 2 матрицы, найти их произведение, используя блочную схему и алгоритм Кэннона

Входной файл в первой строке содержит N_1 количество строк и M_1 количество столбцов первой матрицы через пробел, далее в N_1 содержится M_1элементов (числа от -10000 до 10000) первой матрицы через пробел, затем следующая строка содержит N_2 количество строк и M_2 количество столбцов второй матрицы через пробел, далее в N_2 содержится M_2элементов (числа от -10000 до 10000) первой матрицы через пробел. Порядок расположения элементов матриц внутри файла соответствует порядку расположения элементов в самой матрице.

$-10000 \le N_1, N_2, M_1, M_2 \le 10000$

Выходной файл должен содержать в первой строке N_3 количество строк и M_3 количество столбцов в результирующей матрице через пробел, далее в N_3 содержится M_3элементов этой матрицы через пробел. Порядок расположения элементов матрицы внутри файла должен соответствовать порядку расположения элементов в самой матрице. В случае невозможности данного произведения вывести в файл сообщение «Error Matrix Format».

Пример входных(input.txt) и выходных(output.txt) данных:

input.txt	**output.txt**
3 2 1 2 3 4 5 6 2 3 7 8 9 -1 -2 -3	3 3 5 4 3 17 16 15 29 28 27

Решение систем линейных уравнений методом алгоритма Гаусса, рекурсивное удвоение Стоуна

Задача: дана расширенная (добавлен веткор-столбец свободных членов) матрица коэффициентов системы линейных уравнений, найти решение системы.

Входной файл в первой строке содержит N количество строк и M количество столбцов матрицы через пробел, далее в N содержится M элементов (числа от -10000 до 10000) матрицы через пробел. Порядок расположения элементов матриц внутри файла соответствует порядку расположения элементов в самой матрице.

$-10000 \le N, M \le 10000$

Выходной файл должен содержать значение переменных, совокупность которых является решением данной системы линейных уравнений, через пробел. Порядок расположения значений переменных должен соответствовать порядку номеров их переменных. В слу-

чае невозможности нахождения единственного решения вывести в файл сообщение «Error Data». В случае решения системы равного пустому множеству вывести сообщение «No Solutions»

Пример входных(input.txt) и выходных(output.txt) данных:

input.txt	output.txt
1 2 -7 21	-3

input.txt	output.txt
2 3 1 1 5 1 -2 -4	2 3

Последовательная и каскадная суммы

Задача: дана последовательность случайных вещественных чисел, найти их сумму последовательным и каскадным методом, произвести анализ результатов.

Входной файл в первой строке содержит N количество чисел в последовательности, в следующей строке N вещественных чисел от -50000 до 50000 через пробел.

$0 \le N \le 1000000$

Выходной файл должен содержать результат суммирования и время суммирования в секундах через пробел.

Пример входных(input.txt) и выходных(output.txt) данных:

input.txt	output.txt
2 20 5	25 0.0000001

input.txt	output.txt
2 20 5	25 .0000003

Вычисление определенного интеграла: метод Гаусса

Численное — вычисление значения определённого интеграла (как правило, приближённое). Под численным интегрированием понимают набор численных методов отыскания значения определённого интеграла.

Численное интегрирование применяется, когда:

- Сама подынтегральная функция не задана аналитически. Например, она представлена в виде таблицы (массива) значений в узлах некоторой расчётной сетки.
- Аналитическое представление подынтегральной функции известно, но её первообразная не выражается через аналитические функции. Например, $f(x) = exp(-x2)$.

Если мы можем выбирать точки, в которых мы вычисляем значения функции $f(x)$, то можно при том же количестве вычислений подынтегральной функции получить методы более высокого порядка точности. Так для двух (как в методе трапеций) вычислений значений подынтегральной функции, можно получить метод уже не 2-го, а 3-го порядка точности:

$$I \approx \frac{b-a}{2}\left(f\left(\frac{a+b}{2}-\frac{b-a}{2\sqrt{3}}\right)+f\left(\frac{a+b}{2}+\frac{b-a}{2\sqrt{3}}\right)\right).$$

В общем случае, используя n точек, можно получить метод с порядком точности $2n - 1$. Значения узлов метода Гаусса по n точкам являются корнями полинома Лежандра степени n.

Значения узлов метода Гаусса и их весов приводятся в справочниках специальных функций. Наиболее известен метод Гаусса по пяти точкам.

Задача: дана таблица значений аргумента и значений функции в этих точка, нужно найти величину интеграла функции на заданном интервале.

Входной файл в первой строке содержит через пробел числа A и B – границы интервала определения интеграла, во второй строке записано N количество значений аргументов функции и самой функции (значения от -100000 до 100000).

$1 \le N \le 100000$

$-100000 \le A, B \le 100000$

Выходной файл должен содержать число равное интегралу данной функции. В случае невозможности достоверного интегрирования вывести сообщение «Doubtful»

Пример входных(input.txt) и выходных(output.txt) данных:

input.txt	**output.txt**
1 3.5 10 0 0 1 1 1.5 2.25 1.7 2.89 2 4 2.5 6.25 3 9 12.25 3.5 4 16 5 25	13.9583

12-13. Вычислить сумму ряда

3. $$\sum_{i=1}^{6}\sum_{j=1}^{6}\frac{(-1)^{i+1}\cos^{j+1}x}{j(i+1)}$$

4. $$\sum_{i=1}^{12}\sum_{j=1}^{10}\frac{(-1)^{j-1}}{(i+1)(3/2)^{i}}$$

Задачи низкого уровня сложности (до 10 баллов)

14-24. Вычислить сумму ряда

5. $$\sum_{n=1}^{10}\frac{n^3+2}{n^5+\sin 2^n}$$

6. $$\sum_{n=1}^{15}\frac{1}{\sqrt[5]{n+1}}\sin\frac{1}{\sqrt{n}}$$

7. $\sum_{n=1}^{12} \frac{\arcsin \frac{3+(-1)^n}{4}}{2^n+n}$
8. $\sum_{n=1}^{10} \frac{\sin \frac{\pi}{2n+1}}{n\left(3+\sin \frac{\pi n}{4}\right)}$
9. $\sum_{n=1}^{13} \frac{\ln n}{n^3+n+1}$
10. $\sum_{n=1}^{15} \frac{n^2+3}{n^3\left(2+\sin\left(n\pi/2\right)\right)}$
11. $\sum_{n=2}^{10} \frac{\arcsin \frac{n-1}{n}}{\sqrt[3]{n^3-3n}}$
12. $\sum_{n=1}^{13} \frac{\cos^2(n\pi/2)}{n(n+1)(n+2)}$
13. $\sum_{n=1}^{20} \frac{\ln n}{\sqrt[3]{n^7}}$
14. $\sum_{n=2}^{15} \frac{n+1}{\left(\sqrt[3]{n}-1\right)\left(n\sqrt[4]{n^3}-1\right)}$
15. $\sum_{n=1}^{10} \frac{(-1)^n}{\cos \frac{\pi}{3\sqrt{n}} \sqrt[3]{3n+\ln n}}$

3.2.4.Критерии оценки индивидуальных проектных заданий

Составляющие проекта	Критерии для оценивания	Максимальное количество балов
Постановка проблемы и ее обоснованность, формулирование целей и задач	• общественная значимость и актуальность выдвинутых проблем; • соответствие темы, цели и задач проекта; разумность масштаба работ.	4
Содержание проекта/	• логичность, взаимосвязь и последовательность этапов проекта;	4

	• адекватность предлагаемых мероприятий решению поставленных задач; • корректность используемых методов работы; • четкость определения целевой группы и обоснованность её участия при реализации проекта; • соответствие теоретической, эмпирической и проектной частей, их связь с практикой и выбранным видом профессиональной деятельности; • соблюдение заявленных временных рамок реализации проекта; самостоятельность и активность участника проекта.	
Результат выполнения прикладного проекта	• соответствие ожиданий от проекта / планируемого результата полученному продукту; • степень решения заявленной проблемы; • успешность преодоления трудностей в реализации проекта; • оценка участников целевой группы; • перспективы развития проекта после завершения проекта; возможность тиражирования проекта.	4
Презентация результатов работы над прикладным проектом	• ясность, логичность, профессионализм изложения доклада; • наглядность и структурированность материала презентации; умение корректно использовать профессиональную лексику и понятийно-категориальный аппарат.	4
Ответы на вопросы	• степень владения темой; • ясность аргументации взглядов студента, презентующего результаты выполнения проекта; четкость и лаконичность ответов на вопросы.	4

3.3. Содержание и типовые задания к практическим занятиям

Полные варианты практических занятий размещены в в системе управления обучением MOODLE.

№	Наименование практических занятий	Объем в часах
1	Кластерные технологии	2
2	Основы MPI	2
3	Простейшие параллельные алгоритмы	4
4	Распараллеливание вычислений.	4
	Итого	12

Образцы заданий к практическим занятиям:

1. Написать программу, используя коммуникационные функции (MPI_Ssend, MPI_Bsend, MPI_Rsend, MPI_Isend, MPI_Irecv), передающие одномерные и двумерные массивы (вектора и матрицы) между двумя процессорами
2. Провести сравнение по скорости передачи данных в зависимости от применяемых функций и размера передаваемых данных

3. Написать программу, используя коммуникационную функцию (MPI_Bcast), реализующую алгоритм передачи данных от 0 процесса всем остальным
4. Написать программу, используя коммуникационную функцию (MPI_Gather), реализующую алгоритм передачи частей массива от всех процессоров на 0.
5. Написать программу, используя коммуникационную функцию (MPI_Allgather), реализующую алгоритм передачи частей массива от всех процессоров на все процессора.

4. МЕТОДИЧЕСКИЕ МАТЕРИАЛЫ, ОПРЕДЕЛЯЮЩИЕ ПРОЦЕДУРЫ ОЦЕНИВАНИЯ ЗНАНИЙ, УМЕНИЙ, НАВЫКОВ И ОПЫТА ДЕЯТЕЛЬНОСТИ, ХАРАКТЕРИЗУЮЩИХ ЭТАПЫ ФОРМИРОВАНИЯ КОМПЕТЕНЦИЙ

.

Описание балльно-рейтинговой системы по дисциплине.

Итоговая рейтинговая оценка по дисциплине складывается из следующих составляющих:

1) В течении семестра за выполнение заданий по курсу студент может максимально получить 60 баллов.;

2) Обязательной формой текущей аттестации знаний является выполнение индивидуального проектного задания 20 баллов.

3) На зачёте ответ студента может быть максимально оценен в 40 баллов.

При этом, для получения положительной итоговой оценки на зачете необходимо получить не менее 60% по каждой составляющей и выполнить все задания для практических занятий. Шкала перевода баллов в оценку: до 60 - «не зачтено»; 61 - 100 - «зачтено».

№ п/п	Критерии оценивания	Максимальное количество баллов	Баллы, полученные студентом
1.	Выполнение заданий:	60	
1.1.	Практические занятия	40	
1.2.	Индивидуальное проектное задание	20	
3.	Зачет	40	
	ИТОГО:	100	