МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФГБОУ ВО «Уральский государственный экономический университет»

Одобрена
на заседании кафедры

09.12.2025 г.

протокол № 12

И.о. зав. кафедрой Кольева Н.С.

Утверждена
Советом по учебно-методическим вопросам и качеству образования

16 декабря 2025 г.

протокол № 4

Председатель _____ Карх Д.А.
(подпись)

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ

Наименование дисциплины	Алгоритмы и структуры данных
Направление подготовки	09.03.03 Прикладная информатика
Профиль	Инжиниринг предприятий и информационных систем
Форма обучения	заочная
Год набора	2026

Разработана:
Доцент, к.п.н.
Кольева Н.С.

Екатеринбург
2025 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Рабочая программа дисциплины является частью основной профессиональнойобразовательной программы высшего образования - программы бакалавриата, разработанной всоответствии с ФГОС ВО

ФГОС ВО	Федеральный государственный образовательный стандарт высшего образования- бакалавриат по направлению подготовки 09.03.03 Прикладная информатика(приказ Минобрнауки России от 19.09.2017 г. №

1. ЦЕЛЬ ОСВОЕНИЯ ДИСЦИПЛИНЫ

Целью освоения дисциплины "Алгоритмы и структуры данных" является изучениеприменяемых в программировании и информатике структур данных, их спецификации иреализации, алгоритмов обработки данных и анализа этих алгоритмов, взаимосвязь алгоритмов иструктур данных.

2. МЕСТО ДИСЦИПЛИНЫ В СТРУКТУРЕ ОПОП

Дисциплина относится к обязательной части учебного плана.

3. ОБЪЕМ ДИСЦИПЛИНЫ

Промежуточнаяаттестация	Часов					З.е.
	Всего засеместр	Контактная работа .(поуч.зан.)			Самостоятельная работа в том числеподготовкаконтрольных икурсовых	
		Всего	Лекции	Лабораторные		
Семестр 4						
	36	4	4	0	32	1
Семестр 5						
Экзамен, Контрольнаяработа	144	12	4	8	123	4
	180	16	8	8	155	5

4.ПЛАНИРУЕМЫЕ РЕЗУЛЬТАТЫ ОСВОЕНИЯ ОПОП

В результате освоения ОПОП у выпускника должны быть сформированы компетенции,установленные в соответствии ФГОС ВО.

Шифр и наименованиекомпетенции	Индикаторы достижения компетенций
ОПК-1 Способен применятьестественнонаучные иобщеинженерные знания,методы математическогоанализа и моделирования,теоретического иэкспериментальногоисследования впрофессиональнойдеятельности;	ИД-1.ОПК-1 Знать: основы высшей математики, физики, основывычислительной техники и программирования.

ОПК-1 Способен применятьестественнонаучные иобщеинженерные знания,методы математическогоанализа и моделирования,теоретического иэкспериментальногоисследования впрофессиональнойдеятельности;	ИД-2.ОПК-1 Уметь: решать стандартные профессиональные задачи сприменением естественнонаучных и общеинженерных знаний,методов математического анализа и моделирования.
	ИД-3.ОПК-1 Иметь практический опыт: теоретического иэкспериментального исследования объектов профессиональнойдеятельности
ОПК-7 Способенразрабатывать алгоритмы ипрограммы, пригодные дляпрактического применения;	ИД-1.ОПК-7 Знать: основные языки программирования и работы сбазами данных, операционные системы и оболочки, современныепрограммные среды разработки информационных систем итехнологий.
	ИД-2.ОПК-7 Уметь: применять языки программирования и работы сбазами данных, современные программные среды разработкиинформационных систем и технологий для автоматизации бизнес-процессов, решения прикладных задач различных классов, ведения базданных и информационных хранилищ
	ИД-3.ОПК-7 Иметь практический опыт: программирования, отладки итестирования прототипов программно-технических комплексов задач

5. ТЕМАТИЧЕСКИЙ ПЛАН

Тема	Часов						
	Наименование темы	Всего часов	Контактная работа .(по уч.зан.)			Самост.работа	Контрольсамостоятельной работы
			Лекции	Лабораторные	Практическиезанятия		
Семестр 4		36					
Тема 1.	Введение в алгоритмы и структурыданных.	36	4			32	
Семестр 5		135					
Тема 2.	Алгоритмы сортировки	12	1	1		10	
Тема 3.	Элементарные структуры данных (ОПК-1, ОПК-7)	11	1	1		9	
Тема 4.	Алгоритмы поиска и строки (ОПК-1,ОПК-7)	20	1	1		18	

Тема 5.	Сбалансированные и специальныедеревья (ОПК-1, ОПК-	27	1	1		25	
Тема 6.	Обобщенный быстрый поиск и хеш-функции (ОПК-1, ОПК-7)	6		1		5	
Тема	Жадные алгоритмы (ОПК-1, ОПК-	6		1		5	
Тема 8.	Динамическое программирование (ОПК-1, ОПК-7)	22		1		21	
Тема	Алгоритмы на графах (ОПК-1,	31		1		30	

6. ФОРМЫ ТЕКУЩЕГО КОНТРОЛЯ И ПРОМЕЖУТОЧНОЙ АТТЕСТАЦИИ ШКАЛЫ ОЦЕНИВАНИЯ

Раздел/Тема	Вид оценочногосредства	Описание оценочного средства	Критериоценивания
Текущий контроль (Приложение 4)			
Тема 1-3.	Контрольнаяработа (приложение 4)	Контрольная работа состоит из 5 заданий. Вкаждом	10 баллов
Тема 4-6.	Контрольнаяработа (приложение 4)	Контрольная работа состоит из 5 заданий. Вкаждом	10 баллов
Тема 7-9.	Контрольнаяработа (приложение 4)	Контрольная работа состоит из 5 заданий. Вкаждом	10 баллов
Промежуточная аттестация(Приложение 5)			
5 семестр(Эк)	Экзаменационныйбилет (приложение5)	Экзаменационный билет состоит из 2-хтеоретических вопросов и одногопрактическог о задания.	Теоретическиевопросы - по 25баллов,практическоезадание - 50баллов.

ОПИСАНИЕ ШКАЛ ОЦЕНИВАНИЯ

Показатель оценки освоения ОПОП формируется на основе объединения текущего контроляи промежуточной аттестации обучающегося.

Показатель рейтинга по каждой дисциплине выражается в процентах, который показываетуровень подготовки студента.

Текущий контроль.Используется 100-балльная система оценивания. Оценка работы студента втечении семестра осуществляется преподавателем в соответствии с разработанной им системойоценки учебных достижений в процессе обучения по данной дисциплине.

В рабочих программах дисциплин и практик закреплены виды текущего контроля,планируемые результаты контрольных мероприятий и критерии оценки учебный достижений.

В течение семестра преподавателем проводится не менее 3-х контрольных мероприятий, поооценке деятельности студента. Если посещения занятий по дисциплине включены в рейтинг, тоданный показатель составляет не более 20% от максимального количества баллов по дисциплине.

Промежуточная аттестация. Используется 5-балльная система оценивания. Оценка работыстудента по окончанию дисциплины (части дисциплины) осуществляется преподавателем всоответствии с разработанной им системой оценки достижений студента в процессе обучения поданной дисциплине. Промежуточная аттестация также проводится по окончанию формированиякомпетенций.

Порядок перевода рейтинга, предусмотренных системой оценивания, по дисциплине, впятибалльную систему.

Высокий уровень – 100% - 70% - отлично, хорошо.

Средний уровень – 69% - 50% - удовлетворительно.

Показатель оценки	По 5-балльной системе	Характеристика показателя
100% - 85%	отлично	обладают теоретическими знаниями в полном объеме,понимают, самостоятельно умеют применять,исследовать, идентифицировать, анализировать,систематизировать, распределять по категориям,рассчитать показатели, классифицировать,разрабатывать модели, алгоритмизировать, управлять,организовать, планировать процессы исследования,осуществлять оценку результатов на высоком уровне
84% - 70%	хорошо	обладают теоретическими знаниями в полном объеме,понимают, самостоятельно умеют применять,исследовать, идентифицировать, анализировать,систематизировать, распределять по категориям,рассчитать показатели, классифицировать,разрабатывать модели, алгоритмизировать, управлять,организовать, планировать процессы исследования,осуществлять оценку результатов. Могут быть допущены недочеты, исправленныестудентом самостоятельно в процессе работы (ответаи т.д.)
69% - 50%	удовлетворительно	обладают общими теоретическими знаниями, умеютприменять, исследовать, идентифицировать,анализировать, систематизировать, распределять покатегориям, рассчитать показатели,классифицировать, разрабатывать модели,алгоритмизировать, управлять, организовать,планировать процессы исследования, осуществлятьоценку результатов на среднем уровне. Допускаются ошибки, которые студент затрудняетсяисправить самостоятельно.
49 % и менее	неудовлетворительно	обладают не полным объемом общих теоретическимизнаниями, не умеют самостоятельно применять,исследовать, идентифицировать, анализировать,систематизировать, распределять по категориям,рассчитать показатели, классифицировать,разрабатывать модели, алгоритмизировать, управлять,организовать, планировать процессы исследования,осуществлять оценку результатов. Не сформированы умения и навыки для
100% - 50%	зачтено	характеристика показателя соответствует «отлично»,«хорошо», «удовлетворительно»
49 % и менее	не зачтено	характеристика показателя соответствует«неудовлетворительно»

7. СОДЕРЖАНИЕ ДИСЦИПЛИНЫ

7.1. Содержание лекций

Тема 1. Введение в алгоритмы и структуры данных. Рекурсия (ОПК-1, ОПК-7) Сложность алгоритма. Исполнитель. Инварианты. Индуктивное программирование. Понятиеабстракции. Интерфйс абстракции. Рекурсия. Принцип разделения (разделяй и властвуй).Представление чисел в алгоритмах. Основная теорема о рекурсии. Быстрое возведение в степень.
Тема 2. Алгоритмы сортировки (ОПК-1, ОПК-7) Задача сортировки. Сортировка сравнением. Нижняя оценка сложности алгоритмов сортировкисравнениями. Сортировка с использованием свойств элементов. Внешняя сортировка. Сортировка ипараллельные вычисления. Сравнительный анализ методов сортировки.
Тема 3. Элементарные структуры данных (ОПК-1, ОПК-7) Структура данных "список". Структура данных "дерево". Бинарная куча и абстракция "приоритетнаяочередь". HeapSort.
Тема 4. Алгоритмы поиска и строки (ОПК-1, ОПК-7) Задача поиска. Абстракция поиска. Поиск с сужением зоны. Распределяющий поиск. Поиск побинарному дереву. Префиксное дерево. Строки. Z-функция.
Тема 5. Сбалансированные и специальные деревья (ОПК-1, ОПК-7) Абстракция "отображение". Бинарные деревья поиска. Дисбаланс. Рандомизированное дерево.Декартовы деревья. Сбалансированные деревья поиска. Списки с пропусками. Внешний поиск. B-деревья. Дерево отрезков.
Тема 6. Обобщенный быстрый поиск и хеш-функции (ОПК-1, ОПК-7) Обобщенный быстрый поиск. Хеш-функции. Применение хеш-функций. Хеш-таблицы. Хеш-таблицы во внешней памяти.
Тема 7. Жадные алгоритмы (ОПК-1, ОПК-7) Экстремальные задачи. Жадные алгоритмы. Задача об интервалах. Задача о резервных копиях.Применимость жадных алгоритмов. Приближенное решение экстремальных задач. Сжатиеинформации: алгоритм Хаффмана.

7.2 Содержание практических занятий и лабораторных работ

Тема 3. Элементарные структуры данных (ОПК-1, ОПК-7) Программирование списков и деревьев.
Тема 4. Алгоритмы поиска и строки (ОПК-1, ОПК-7) Обратные задачи для монотонных функций. Решение уравнений. Задача о проводах. Реализациясортировки через списки. Построение деревьев поиска.
Тема 5. Сбалансированные и специальные деревья (ОПК-1, ОПК-7) Хранение полных бинарных деревьев в массиве. Оценка сложности операций с Heap. Быстраясортировка с применением бинарных деревьев. Быстрая сортировка на месте.

Тема 6. Обобщенный быстрый поиск и хеш-функции (ОПК-1, ОПК-7) Оценка вероятности коллизии. Универсальные семейства хэш-функций — проверить некотороесемейство на универсальность. Исследование свойств хеш-функций.
Тема 7. Жадные алгоритмы (ОПК-1, ОПК-7) Задача про банкомат. Применимость жадных алгоритмов. Задача об аудиториях. Задача про атлетов.Задача про минимальный вес множества отрезков.
Тема 8. Динамическое программирование (ОПК-1, ОПК-7) Решение задач на динамическое программирование.
Тема 9. Алгоритмы на графах (ОПК-1, ОПК-7) Корректность алгоритмов Беллмана-Форда и Дейкстры. Направленный ациклический граф, егосвойства и связь с динамическим программированием. Решение задач на графы. Алгоритм Флойда-Уоршалла. Корректность алгоритма. Комбинированные задачи. Связь алгоритма DFS стопологической сортировкой

7.3. Содержание самостоятельной работы

Тема 2. Алгоритмы сортировки (ОПК-1, ОПК-7) Быстрая сортировка. Сортировка за линейное время. Медианы и порядковые статистики.
Тема 3. Элементарные структуры данных (ОПК-1, ОПК-7) Стеки и очереди. Связанные списки. Реализация указателей и объектов. Представление корневыхдеревьев. Пирамиды. Поддержка свойств пирамиды. Построение пирамиды. Алгоритмпирамидальной сортировки. Очереди с приоритетами.
Тема 4. Алгоритмы поиска и строки (ОПК-1, ОПК-7) Бинарное дерево поиска. Работа с бинарным деревом поиска. Вставка и удаление. Случайноепостроение бинарных деревьев поиска. Простейший алгоритм поиска подстрок. Алгоритм Рабина-Карпа. Поиск подстрок с помощьюконечных автоматов. Алгоритм Кнута-Морриса-Пратта.
Тема 5. Сбалансированные и специальные деревья (ОПК-1, ОПК-7) Красно-черные деревья. B-деревья. Расширение структур данных. Фибоначчиевы пирамиды.Деревья ван Эмде Боаса.
Тема 6. Обобщенный быстрый поиск и хеш-функции (ОПК-1, ОПК-7) Хеширование и хеш-таблицы. Поиск подстрок.
Тема 7. Жадные алгоритмы (ОПК-1, ОПК-7) Задача о выборе процессов. Элементы жадной стратегии. Коды Хаффмана. Матроиды и жадныеметоды. Планирование заданий как матроид.

Тема 8. Динамическое программирование (ОПК-1, ОПК-7) Разрезание стержня. Перемножение цепочки матриц. Элементы динамического программирования.Наидлиннейшая общая подпоследовательность. Оптимальные бинарные деревья поиска. NP-полнота. Приближенные алгоритмы.
Тема 9. Алгоритмы на графах (ОПК-1, ОПК-7) Элементарные алгоритмы для работы с графами. Минимальные остовные деревья. Кратчайшие путиизодной вершины. Кратчайшие пути между всеми парами вершин. Задача о максимальном потоке.

7.3.1. Примерные вопросы для самостоятельной подготовки к зачету/экзамену
Приложение 1.

7.3.2. Практические задания по дисциплине для самостоятельной подготовки кзачету/экзамену
Приложение 2.

7.3.3. Перечень курсовых работ
Не предусмотрено.

7.4. Электронное портфолио обучающегося
Размещаются контрольные работы

7.5. Методические рекомендации по выполнению контрольной работы
Приложение 6

7.6 Методические рекомендации по выполнению курсовой работы
Не предусмотрено.

8. ОСОБЕННОСТИ ОРГАНИЗАЦИИ ОБРАЗОВАТЕЛЬНОГО ПРОЦЕССА ПО ДИСЦИПЛИНЕ ДЛЯ ЛИЦ С ОГРАНИЧЕННЫМИ ВОЗМОЖНОСТЯМИ ЗДОРОВЬЯ

По заявлению студента

В целях доступности освоения программы для лиц с ограниченными возможностями здоровья при необходимости кафедра обеспечивает следующие условия:

- особый порядок освоения дисциплины, с учетом состояния их здоровья;
- электронные образовательные ресурсы по дисциплине в формах, адаптированных к ограничениям их здоровья;
- изучение дисциплины по индивидуальному учебному плану (вне зависимости от формы обучения);
- электронное обучение и дистанционные образовательные технологии, которые предусматривают возможности приема-передачи информации в доступных для них формах.
- доступ (удаленный доступ), к современным профессиональным базам данных и информационным справочным системам, состав которых определен РПД.

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

Сайт библиотеки УрГЭУ
http://lib.usue.ru/

Основная литература:

2. Дадян Э.Г. Данные: хранение и обработка [Электронный ресурс]:Учебник. - Москва: ООО"Научно-издательский центр ИНФРА-М", 2026. - 205 – Режим доступа:https://znanium.com/catalog/product/2214875

3. Гданский Н.И. Основы теории и алгоритмы на графах [Электронный ресурс]:Учебноепособие. - Москва: ООО "Научно-издательский центр ИНФРА-М", 2025. - 206 – Режим доступа:https://znanium.com/catalog/product/2166200

4. Ахмад И., Чикин Р. 40 алгоритмов, которые должен знать каждый программист наPython:производственно-практическое издание. - Санкт-Петербург [и др.]: Питер, 2024. - 362

5. Черняк А. А., Богданович С. А., Черняк Ж. А., Метельский Ю. М. Методы оптимизации:теория и алгоритмы [Электронный ресурс]:учебное пособие для вузов. - Москва: Юрайт, 2024. - 357– Режим доступа: https://urait.ru/bcode/539155

Дополнительная литература:

2. Кислицын Е. В. Алгоритмы и структуры данных [Электронный ресурс]:учебное пособие. -Екатеринбург: Издательство УрГЭУ, 2020. - 281 – Режим доступа:http://lib.usue.ru/resource/limit/ump/20/p493245.pdf

3. Дорогов В.Г., Теплова Я.О. Введение в методы и алгоритмы принятия решений[Электронный ресурс]:Учебное пособие. - Москва: Издательский Дом "ФОРУМ", 2022. - 240 –Режим доступа: https://znanium.com/catalog/product/1841773

4. Колдаев В.Д. Структуры и алгоритмы обработки данных [Электронный ресурс]:Учебноепособие. - Москва: Издательский Центр РИО, 2020. - 296 – Режим доступа:https://znanium.com/catalog/product/1054007

5. Седжвик Р., Уэйн К., Моргунов А. А., Артеменко Ю. Н. Алгоритмы на Java:научноеиздание. - Москва: Вильямс, 2017. - 843

6. Веретехина С.В., Симонов В.Л. Модели, методы, алгоритмы и программные решениявычислительных машин, комплексов и систем [Электронный ресурс]:Учебник. - Москва: ООО"Научно-издательский центр ИНФРА-М", 2020. - 268 – Режим доступа:https://znanium.com/catalog/product/1210403

7. Белов В. В., Чистякова В.И. Алгоритмы и структуры данных [Электронныйресурс]:учебник. - Москва: ООО "КУРС", 2020. - 240 – Режим доступа:https://znanium.ru/catalog/product/1057212

8. Новиков А.И. Исследование операций в экономике [Электронный ресурс]:Учебник. - Москва: Издательско-торговая корпорация "Дашков и К", 2022. - 352 – Режим доступа:https://znanium.com/catalog/product/2082697

10. ПЕРЕЧЕНЬ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, ВКЛЮЧАЯ ПЕРЕЧЕНЬ

ЛИЦЕНЗИОННОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ И ИНФОРМАЦИОННЫХ СПРАВОЧНЫХ СИСТЕМ, ОНЛАЙН КУРСОВ, ИСПОЛЬЗУЕМЫХ ПРИ ОСУЩЕСТВЛЕНИИ ОБРАЗОВАТЕЛЬНОГО ПРОЦЕССА ПО ДИСЦИПЛИНЕ

Перечень лицензионного программного обеспечения:

Microsoft Visual Studio Community. Лицензия для образовательных учреждений. Срок действия лицензии - без ограничения срока.

Microsoft Windows 10 .Договор № 52/223-ПО/2020 от 13.04.2020, Акт № Tr000523459 от 14.10.2020. Срок действия лицензии -Без ограничения срока.

Microsoft Office 2016.Договор № 52/223-ПО/2020 от 13.04.2020, Акт № Tr000523459 от 14.10.2020 Срок действия лицензии -Без ограничения срока.

Intellij IDEA.

Перечень информационных справочных систем, ресурсов информационно-телекоммуникационной сети «Интернет»:

Справочно-правовая система Консультант +. Договор № 143/223-У/2025 от 02.12.2025 Срок действия лицензии до 31.12.2026

Справочно-правовая система Гарант. Договор № 58419 от 22 декабря 2015. Срок действия лицензии -без ограничения срока

Алгоритмы программирования и структуры данных
https://openedu.ru/course/ITMOUniversity/PADS/
Методы вычислительной математики
https://openedu.ru/course/spbstu/NUMMETH/

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

Реализация учебной дисциплины осуществляется с использованием материально-технической базы УрГЭУ, обеспечивающей проведение всех видов учебных занятий и научно-исследовательской и самостоятельной работы обучающихся:

Специальные помещения представляют собой учебные аудитории для проведения всех видов занятий, групповых и индивидуальных консультаций, текущего контроля и промежуточной аттестации.

Помещения для самостоятельной работы обучающихся оснащены компьютерной техникой с возможностью подключения к сети "Интернет" и обеспечением доступа в электронную информационно-образовательную среду УрГЭУ.

Все помещения укомплектованы специализированной мебелью и оснащены мультимедийным оборудованием спецоборудованием (информационно-телекоммуникационным, иным компьютерным), доступом к информационно-поисковым, справочно-правовым системам, электронным библиотечным системам, базам данных действующего законодательства, иным информационным ресурсам служащими для представления учебной информации большой аудитории.

Для проведения занятий лекционного типа презентации и другие учебно-наглядные пособия, обеспечивающие тематические иллюстрации.

Приложение 1
к рабочей программе

7.3.1. Примерные вопросы для самостоятельной подготовки к экзамену

К экзамену

Раздел 1

1. Свойства алгоритма. Сложность алгоритма.
2. Инварианты. Индуктивное программирование.
3. Понятие абстракции. Абстракция «Последовательность». Абстракция «Массив».
4. Интерфейс абстракции «Стек». Интерфейс абстракции «Множества».
5. Рекурсия. Принцип «разделяй и властвуй».
6. Представление чисел в алгоритмах. Длинные числа.
7. Основная теорема о рекурсии.
8. Жадные алгоритмы.
9. Двоичные деревья. Алгоритм Хаффмана.
10. Префиксное дерево. Задача о покрытии строки.
11. Нахождение k-порядковой статистики.
12. Структура данных «Список».
13. Структура данных «Дерево». Представление деревьев.
14. Структура данных «Дерево». Обход деревьев.
15. Бинарная куча. Абстракция «Очередь с приоритетом».
16. Абстракция «Отображение».
17. Бинарные деревья поиска.
18. Декартовы деревья. Операции над Декартовыми деревьями.
19. Сбалансированные деревья поиска.
20. Списки с пропусками.
21. Дерево отрезков.
22. Внешний поиск. B-деревья.
23. Хеш-функции. Хеш-таблицы с прямой адресацией.
24. Хеш-таблицы с открытой адресацией.
25. Хеш-таблицы во внешней памяти.
26. Графы и их представление.
27. Обход графа. Поиск в ширину. Алгоритм BFS.
28. Обход графа. Поиск в глубину. Алгоритм DFS.
29. Поиск компонент связности в графе.
30. Остовные деревья. Свойства MST.

Раздел 2

1. Алгоритм Карацубы
2. Алгоритм быстрого возведения в степень
3. Задача об интервалах.
4. Задача о резервных копиях.
5. Задача о рюкзаке.
6. Сортировка сравнением. Понятие инверсии. Сортировка пузырьком.

7. Сортировка вставками. Сортировка Шелла.
8. Сортировка выбором.
9. Быстрая сортировка (сортировка Хоара).
10. Сортировка слиянием.
11. Сортировка подсчетом.
12. Поразрядная сортировка.
13. Внешняя сортировка слиянием.
14. Сортировка сериями.
15. Задача поиска. Абстракция поиска. Последовательный поиск.
16. Поиск с сужением зоны.
17. Распределяющий поиск.
18. HeapSort.
19. Обобщенный быстрый поиск.
20. Алгоритм Карпа-Рабина.
21. Задача о количестве маршрутов. Принцип Беллмана.
22. Задача о возрастающей подпоследовательности наибольшей длины.
23. Задача о банкомате.
24. Задача о счастливых билетах.
25. Задача о вторичной структуре РНК.
26. Топологическая сортировка.
27. Алгоритм Прима.
28. Алгоритма Краскала.
29. Алгоритм Дейкстры.
30. Алгоритм Флойда-Уоршалла.

Приложение 2
к рабочей программе

7.3.2. Практические задания по дисциплине для самостоятельной подготовки к экзамену

Примерные вопросы закрытого типа (ОПК-1, ОПК-7)

1.Какова сложность алгоритма быстрой сортировки (Quick Sort)?

1. O(n)
2. O(n log n)
3. O(n^2)
4. O(log n)

2.Какая структура данных используется для хранения данных в виде "последним пришел, первым ушел" (Last In First Out, LIFO)?

1. Очередь (Queue)
2. Стек (Stack)
3. Дерево (Tree)
4. Граф (Graph)

3.Какая структура данных используется для представления связанных узлов в виде цепочки?

1. Массив (Array)
2. Список (List)
3. Дерево (Tree)
4. Хеш-таблица (Hash Table)

4.Какая операция выполняется быстрее всего на хеш-таблице с методом открытой адресации?

1. Поиск элемента
2. Вставка элемента
3. Удаление элемента
4. Обход всех элементов

5.Какой алгоритм сортировки имеет наихудшую сложность O(n^2)?

1. Сортировка пузырьком (Bubble Sort)
2. Сортировка вставками (Insertion Sort)
3. Сортировка выбором (Selection Sort)
4. Сортировка слиянием (Merge Sort)

Примерные вопросы открытого типа (ОПК-1, ОПК-7)

1. Какие основные принципы лежат в основе выбора оптимальной структуры данных для конкретной задачи?
2. Какие алгоритмы сортировки вы знаете и в каких случаях каждый из них эффективен?
3. Как работает алгоритм поиска в ширину (BFS) в графе? Какие задачи можно решить с его помощью?
4. Какие преимущества и недостатки у деревьев поиска по сравнению с хеш-таблицами?

5. Какие методы оптимизации алгоритмов и структур данных вы знаете и как они могут повлиять на производительность программы?

Примерные практические задания к экзамену

Примерные практические задания расположены на сайте:
http://acm.timus.ru/problemset.aspx
Количество баллов, полученных за практическое задание рассчитывается как (Сложность задания) / 50.

Примеры заданий:

1048. Сверхдлинные суммы

Ограничение времени: 2.0 секунды
Ограничение памяти: 16 МБ

Создатели нового языка программирования D++ поняли, что какое бы большое ограничение на тип SuperLongInt они ни наложили, некоторым программистам потребуется работать с ещё бо́льшими числами. Ограничение в 1000 цифр так мало… Вам нужно найти сумму двух целых чисел размером до 1 000 000 цифр.

Исходные данные

Первая строка содержит целое число N — длину чисел ($1 \le N \le 1\ 000\ 000$). В следующих N строках следуют записанные в столбик числа, каждая строка содержит по две цифры, разделённые пробелом. Каждое из двух чисел не меньше 1, а длина суммы этих чисел не превосходит N. Числа могут содержать ведущие нули.

Результат

Выведите в одной строке ровно N цифр, представляющих сумму этих двух чисел.

Пример

исходные данные	результат
4 0 4 4 2 6 8 3 7	4750

1003. Чётность

Ограничение времени: 2.0 секунды
Ограничение памяти: 64 МБ

Вы играете со своим другом в следующую игру. Ваш друг записывает последовательность, состоящую из нулей и единиц. Вы выбираете непрерывную подпоследовательность (например, подпоследовательность от третьей до пятой цифры включительно) и спрашиваете его, чётное или нечётное количество единиц содержит эта подпоследовательность. Ваш друг отвечает, после чего вы можете спросить про другую подпоследовательность, и так далее.

Ваша задача — угадать всю последовательность чисел. Но вы подозреваете, что некоторые из ответов вашего друга могут быть неверными, и хотите уличить его в обмане. Вы решили написать программу, которая получит наборы ваших вопросов вместе с ответами друга и найдет первый ответ, который гарантированно неверен. Это должен быть такой ответ, что существует последовательность, удовлетворяющая ответам на предыдущие вопросы, но никакая последовательность не удовлетворяет этому ответу.

Исходные данные

Ввод содержит несколько тестов. Первая строка каждого теста содержит одно число, равное длине последовательности нулей и единиц. Эта длина не превосходит 10^9. Во второй строке находится одно неотрицательное целое число — количество заданных вопросов и ответов на них. Количество вопросов и ответов не превышает 5 000. Остальные строки содержат вопросы и ответы. Каждая строка содержит один вопрос и ответ на этот вопрос: два целых числа (позиции первой и последней цифр выбранной подпоследовательности) и одно слово — “even” или “odd” — ответ, сообщающий чётность количества единиц в выбранной подпоследовательности, где “even” означает чётное количество единиц, а “odd” означает нечётное количество. Ввод заканчивается строкой, содержащей −1.

Результат

Каждая строка вывода должна содержать одно целое число X. Число X показывает, что существует последовательность нулей и единиц, удовлетворяющая первым X условиям чётности, но не существует последовательности, удовлетворяющей $X + 1$ условию. Если существует последовательность нулей и единиц, удовлетворяющая всем заданным условиям, то число X должно быть равно количеству всех заданных вопросов.

Пример

исходные данные	результат
10 5 1 2 even 3 4 odd 5 6 even 1 6 even 7 10 odd -1	3

1085. Встреча

Ограничение времени: 2.0 секунды
Ограничение памяти: 64 МБ

K друзей решили отпраздновать свою победу на олимпиаде по программированию. Но в связи с повышением цен на билеты возникла следующая проблема: все они живут в разных частях города, поэтому им нужно выбрать место встречи так, чтобы на поездки не пришлось тратить слишком много денег. Вы должны помочь им сделать наилучший выбор.

Пусть остановки пронумерованы целыми числами от 1 до N включительно, а в городе ходит M маршрутов трамвая (все друзья ездят исключительно на трамваях и не ходят пешком между остановками). Для каждого маршрута известны номера составляющих его остановок. Для каждого человека известно, сколько у него денег и есть ли у него проездной на трамвай. Цена билета равна 4 рублям.

Вам требуется найти номер такой остановки, чтобы все могли доехать до неё, и сумма денег, потраченных ими на проезд, была минимальной. Естественно, можно делать пересадки с маршрута на маршрут, но учтите, что каждый раз, делая пересадку, требуется покупать новый билет: друзья зайцами не ездят. За дорогу до места встречи каждый платит сам. Денег на обратную дорогу оставлять не требуется.

Исходные данные

В первой строке даны два целых числа N и M; $1 \le N, M \le 100$. В следующих M строках идёт описание маршрутов трамвая следующим образом: в начале строки находится целое число L ($2 \le L \le 100$), задающее число остановок в маршруте. Затем идут L целых чисел, задающих номера остановок в маршруте. Все числа в строке разделены пробелами. Затем следует строка с целым числом K ($1 \le K \le 100$). В следующих K строках дана информация для каждого из них, по строке на человека. В начале строки указано целое положительное число, задающее количество денег в рублях у человека. Затем указан номер остановки, до которой он доходит от дома пешком. За ним следует либо число 0, если этот человек не имеет проездного, либо 1, если имеет. Числа в строке разделены пробелами. Никто из друзей не имеет больше 1000 рублей.

Результат

Выведите два числа: номер остановки, на которой друзья должны встретиться (если таких номеров несколько, выведите наименьший), и суммарное количество рублей, затраченное на поездки друзьями. Числа должны быть разделены пробелом. Если друзья не смогут все встретиться на одной остановке, выведите единственное число 0.

Пример

исходные данные	результат
4 3 2 1 2 2 2 3 2 3 4 3 27 1 0 15 4 0 45 4 0	4 12

7.5. Методические рекомендации по выполнению контрольной работы

Приложение 6
к рабочей программе

Федеральное государственное бюджетное образовательное учреждение высшего образования
УРАЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ЭКОНОМИЧЕСКИЙ УНИВЕРСИТЕТ

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ВЫПОЛНЕНИЮ

КОНТРОЛЬНОЙ РАБОТЫ
по дисциплине
Алгоритмы и структуры данных

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ КОНТРОЛЬНОЙ РАБОТЫ

В качестве контрольной работы студентам предлагается решить задачи по дисциплине «Алгоритмы и структуры данных».

Задачи должны быть решены на одном из языков программирования: C#, Java (Kotlin) или C++.

Перед выполнением задания студент должен изучить материалы по темам «Введение в алгоритмы и структуры данных. Рекурсия», «Алгоритмы сортировки», «Элементарные структуры данных».

По итогам решенных задач студент должен подготовить текстовый файл, в котором содержится отчет о проделанной работе. По каждой задаче в отчете должно быть представлено: постановка задачи, ход решения, алгоритм решения (код программы), пример работы программы, тестирование программы.

Постановка задачи копируется из условий. Ход решения является описательным пунктом – здесь описывается логика решения задачи на естественном языке.Код программы обязательно должен содержать комментарии. Пример работы программы должен содержать скриншоты программы при выполнении.

В пункте «Тестирование» должны быть представлены наборы тестовых данных (не менее 10-ти наборов, соответствующие условиям задачи). Хотя бы один из тестовых наборов должен отражать крайние значения (например, если максимально введенное число 1000000). Также, по каждому тестовому набору должны быть представлены время выполнения программы и затраченная память. Рекомендуется, оформить данный пункт в виде таблицы, содержащий следующие поля: номер, входные данные, выходные данные (полученные программой), корректность (да/нет), время выполнения (в миллисекундах), память (в Мб). Для измерения времени и памяти необходимо использовать возможности выбранного языка (например, в C# для измерения времени используется StopWatch).

Для измерения времени выполнения метода на языке C# следует использовать следующую конструкцию:

```
Stopwatch stopWatch = new Stopwatch();
stopWatch.Start();
// здесь пишется тестируемый метод
stopWatch.Stop();
Console.WriteLine("RunTime: " + stopWatch.Elapsed);
```

Для измерения объема затраченной дополнительной памяти в методе на языке C# можно использовать следующую конструкцию:

```
long before = GC.GetTotalMemory(false);
// здесь пишется тестируемый метод
long after = GC.GetTotalMemory(false);
long consumedInMegabytes = (after - before) / (1024 * 1024);
Console.WriteLine("Memory is {0} Mb.", consumedInMegabytes);
```

При выполнении заданий необходимо обязательно делать такие измерения.

Подготовленный отчет оформляется по требованиям УрГЭУ к оформлению и сдается преподавателю.

Контрольная работа состоит из 3-х заданий. Вариант выбирается по первой букве фамилии студента:

		Номера задач		
Фамилия	Вариант	Задание 1	Задание 2	Задание 3
А	1	1	7	24
БА-БИ	2	3	8	23
БК-БЯ	3	4	8	13
В	4	3	11	15
Г	5	6	11	21
Д	6	5	10	23
Е, Ё	7	2	8	22

Ж	8	6	7	13
З	9	5	11	19
И	10	3	10	20
КА-КО	11	1	9	21
КП-КЯ	12	4	11	25
Л	13	4	9	19
МА-МЗ	14	6	7	16
МИ-МЯ	15	5	10	18
НА-НМ	16	2	11	22
НН-НЯ	17	6	7	24
О	18	5	9	25
П	19	3	8	18
Р	20	5	7	12
С	21	3	10	22
Т	22	6	9	15
У	23	4	11	18
Ф	24	6	8	20
Х	25	6	10	14
Ц, Ч	26	1	10	21
Ш, Щ	27	4	8	17
Э	28	2	7	16
Ю	29	4	9	23
Я	30	4	11	19

Для получения оценки «удовлетворительно» достаточно выполнить 1 задание, для получения оценки «хорошо» – 2, «отлично» – 3.

Задача 1. Симметрическая разность.

Ограничение по времени: 2 секунды

Ограничение по памяти: 64 мегабайта

На вход подается множество чисел в диапазоне от 1 до 20000, разделенных пробелом. Они образуют множество A. Затем идет разделитель – число 0 и на вход подается множество чисел B, разделенных пробелом, 0 – признак конца описания множества (во множество не входит). Необходимо вывести множество A∆B – симметрическую разность множеств A и B в порядке возрастания элементов. В качестве разделителя используйте пробел. В случае, если множество пусто, вывести 0.

Формат входных данных:

1 2 3 4 5 0 1 7 5 8 0

Формат выходных данных:

2 3 4 7 8

Примеры:

Стандартный ввод	Стандартный вывод
1 2 6 8 7 3 0 4 1 6 2 3 9 0	4 7 8 9

Замечание. Для вывода можно использовать любой алгоритм сортировки.

Задача 2. Два массива.

Ограничение по времени: 2 секунды

Ограничение по памяти: 64 мегабайта

Даны два упорядоченных по неубыванию массива. Требуется найти количество таких элементов, которые присутствуют в обоих массивах. Например, в массивах (0, 0, 1, 1, 2, 3) и (0, 1, 1, 2) имеется четыре общих элемента – (0, 1, 1, 2).

Первая строка содержит размеры массивов N1 и N2. В следующих N1 строках содержатся элементы первого массива, в следующих за ними N2 строках – элементы второго массива.

Программа должна вывести ровно одно число – количество общих элементов.

Формат входных данных:

N_a, N_b

a_1

a_2

…

a_{Na}

b_1

b_2

…

b_{Nb}

Формат выходных данных:

Одно целое число – количество общих элементов

Примеры:

Стандартный ввод	Стандартный вывод
5 5 1 1 2 2 3 0 1 3 3 4	2

Задача 3. Длинное сложение и вычитание

Ограничение по времени: 2 секунды

Ограничение по памяти: 64 мегабайта

На вход подается три строки. Первая содержит представление длинного десятичного числа (первый операнд), вторая – представление операции, строки + и -, третья – представление второго операнда.

Длина первой и третьей строки ограничены 1000 символами. Вторая строка содержит ровно один символ.

Требуется исполнить операцию и вывести результат в десятичном представлении.

Формат входных данных:

123

+

999

Формат выходных данных:

1122

Примеры:

Стандартный ввод	Стандартный вывод
232 + -100	132
-100 - 199	-299

Замечание. Постарайтесь реализовать программу таким образом, чтобы ей можно было воспользоваться в дальнейшем. В других работах нашего курса имеются задачи, в которых потребуется длинная арифметика.

Задача 4. Вычисление полинома.

Ограничение по времени: 1 секунда

Ограничение по памяти: 16 мегабайт

Вычисление полинома – необходимая операция для многих алгоритмов. Нужно вычислить значение полинома

$$a_n x^n + a_{n-1} x^{n-1} + ... + a_2 x^2 + a_1 x^1 + a_0$$

Так как число n может быть достаточно велико, требуется вычислить значение полинома по модулю M. Сделать это предлагается для нескольких значений аргумента.

Формат входных данных:

Первая строка файла содержит три числа – степень полинома $2 \le N \le 100000$, количеств вычисляемых значений аргумента $1 \le M \le 10000$ и модуль $10 \le MOD \le 10^9$.

Следующие N+1 строк содержат значения коэффициентов полинома $0 \le a_i \le 10^9$

В очередных M строках содержатся значения аргументов $0 \le x_i \le 10^9$.

Формат выходных данных:

Выходной файл должен состоять из ровно M строк – значений данного полинома при заданных значениях аргументов по модулю MOD.

Примеры:

Стандартный ввод	Стандартный вывод
2 5 10 1 5 4 0 1 2 3 4	4 0 8 8 0
5 9 10 1 0 0 0 0 0 1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9

Задача 5. Считаем комментарии.

Ограничение по времени: 1 секунда

Ограничение по памяти: 256 мегабайт

Комментарием в языке Object Pascal является любой текст, находящийся между последовательностью символов, начинающих комментарий определенного вида и последовательностью символов, заканчивающей комментарий этого вида.

Виды комментариев могут быть следующие:

1. Начинающиеся с набора символов (* и заканчивающиеся набором символов *).

2. Начинающиеся с символа { и заканчивающиеся символом }.

3. Начинающиеся с набора символов // и заканчивающиеся символом новой строки.

Еще в языке Object Pascal имеются литеральные строки, начинающиеся с символа одиночной кавычки ‘ и заканчивающиеся этим же символом. В корректной программе строки не могут содержать символа перехода на новую строку.

Будьте внимательны, в задаче используются только символы с кодами до 128, то есть, кодировка ASCII. При тестировании своего решения будьте внимательны. Код одиночной кавычки – 39, двойной – 34.

Формат входных данных:

На вход программы подается набор строк, содержащих фрагмент корректной программы на языке Object Pascal.

Формат выходных данных:

Выходом программы должно быть 4 числа – количество комментариев первого, второго и третьего типов, а также количество литеральных строк.

Примеры:

Стандартный ввод	Стандартный вывод

program test;
(*just for testing *)
var
(* variables
note that
// here is not comment
and (* here is
not a begin of
another comment
*)
x: integer; (* *)
begin
write(‘(*is not comment//’);
write(‘ and (*here*) ‘
,x // y);
End. // It is comment
	3 0 2 2

Задача 6. Две кучи.

Ограничение по времени: 2 секунды

Ограничение по памяти: 64 мегабайт

Имеется $2 \le N \le 23$ камня с целочисленными весами W_1, W_2, … W_N. Требуется разложить их на две кучи таким образом, чтобы разница в весе куч была минимальной. Каждый камень должен принадлежать ровно одной куче.

Формат входных данных:

N

W1 W2 W3 … WN

Формат выходных данных:

Минимальная неотрицательная разница в весе куч

Примеры:

Стандартный ввод	Стандартный вывод
5 8 9 6 9 8	4
6 14 2 12 9 9 8	2

Задача 7. Максимальная тройка

Ограничение по времени: 1.5 секунд

Ограничение по памяти: 8 мегабайт

Имеется не более 1000000 целых чисел, каждое из которых лежит в диапазоне от -1000000 до 1000000. Найти максимально возможное значение произведений любых трех различных по номерам элементов массива.

Формат входных данных:

N
A1
A2
…
AN

Формат выходных данных:

MaxPossibleProduct

Примеры:

Стандартный ввод	Стандартный вывод
10 -1 2 3 -4 -2 5 -1 5 -3 -2	75

Задача 8. Сортировка по многим полям

Ограничение по времени: 2 секунды

Ограничение по памяти: 64 мегабайта

В базе данных хранится N записей, вида (Name, a_1, a_2, …, a_k) – во всех записях одинаковое число параметров. На вход задачи подается приоритет полей – перестановка на числах 1, …, k – записи нужно вывести по невозрастанию в соответствии с этим приоритетом. В случае, если приоритет полей таков: 3 4 2 1, то это следует воспринимать так: приоритет значений из 3 колонки самый высокий, приоритет значений из колонки 4 ниже, приоритет значений из колонки 2 еще ниже, а приоритет значений из колонки 1 самый низкий.

Формат входных данных:

$N \leq 1000$

k: $1 \leq k \leq 10$

p_1 p_2 … p_k – перестановка на k числах, разделитель – пробел

N строк вида

Name a_1 a_2 … a_k

Формат выходных данных:

N строк с именами в порядке, согласно приоритету

Примеры:

Стандартный ввод	Стандартный вывод
3 3 2 1 3 A 1 2 3	B A C

B 3 2 1 C 3 1 2	

Замечание. Так как колонка под номером 2 самая приоритетная, то переставить записи можно только двумя способами: (A, B, C) и (B, A, C). Следующий по приоритетности столбец – первый, и он позволяет выбрать из возможных перестановок только (B, A, C). Так как осталась ровно одна перестановка, третий приоритет не имеет значения.

Задача 9. Оболочка.

Ограничение по времени: 2 секунды

Ограничение по памяти: 64 мегабайта

Имеется массив из N целочисленных точек на плоскости.

Требуется найти периметр наименьшего охватывающего многоугольника, содержащего все точки.

Формат входных данных:

N

x1 y1

x2 y2

…

xn yn

$5 \le N \le 500000$

$-10000 \le x_i, y_i \le 10000$

Формат выходных данных:

Одно вещественное число – периметр требуемого многоугольника с двумя знаками после запятой.

Примеры:

Стандартный ввод	Стандартный вывод
5 2 1 2 2 2 3 3 2 1 2	5.66

Задача 10. Очень быстрая сортировка.

Ограничение по времени: 1.5 секунд

Ограничение по памяти: 512 мегабайт

Имеется рекуррентная последовательность A1, A2, …, AN, строящаяся по следующему правилу:

$A1 = K$

$A_{i+1} = (A_i \times M) \% (2^{32} - 1) \% L$

Требуется найти сумму всех нечетных по порядку элементов в отсортированной по неубыванию последовательности по модулю L.

Для входных данных

5 7 13 100

последовательность будет такой:

{7; 7 × 13%100 = 91; 91 × 13%100 = 83; 83 × 13%100 = 79; 79 × 13%100 = 27}, то есть {10; 91; 83; 79; 27}.

Отсортированная последовательность {7; 27; 79; 83; 91}.

Сумма элементов на нечетных местах = (7 + 79 + 91)%100 = 77.

Формат входных данных:

N K M L

$5000000 \le N \le 60000000$, $0 \le K, L, M \le 2^{32} - 1$

Формат выходных данных:

RESULT

Примеры:

Стандартный ввод	Стандартный вывод
5 7 13 100	77

Замечание. Для представления элементов последовательности необходимо использовать тип данных unsigned int.

Для получения массива используйте цикл:

```
a[0] = K;
for (int i = 0; i < N-1; i++)
a[i+1] = (unsigned int) ((a[i]*unsigned long long)M)&0xFFFFFFFFU)%L;
```

Внимание! Изменение типа данных и/или метода генерации элементов массива может привести (и на различных компиляторах приводит) к другой последовательности!

Задача 11. Внешняя сортировка.

Ограничение по времени: 2 секунды

Ограничение по памяти: 2 мегабайта

В файле «input.txt» содержатся строки символов, длина каждой строки не превышает 10000 байт. Файл нужно отсортировать в лексикографическом порядке и вывести результат в файл «output.txt». Вот беда, файл занимает много мегабайт, а в Вашем распоряжении оказывается вычислительная система с очень маленькой оперативной памятью. Но файл должен быть отсортирован!

Примеры:

input.txt	output.txt
qwertyuiopasdffghhj qpoiuytredgfhfd asdfghjjklvcvx alkjghcdysdfgsr pquytrgsdjdsa akjhfgdghshhfuushvdfs	akjhfgdghshhfuushvdfs alkjghcdysdfgsr asdfghjjklvcvx pquytrgsdjdsa qpoiuytredgfhfd qwertyuiopasdffghhj

Задача 12. Стек

Ограничение по времени – 2 секунды. Ограничение по памяти – 256 мегабайт.

Реализуйте работу стека. Для каждой операции изъятия элемента выведите ее результат.

На вход программе подаются строки, содержащие команды. Каждая строка содержит одну команду. Команда — это либо "+ N", либо "−". Команда "+ N" означает добавление в стек числа N, по модулю не превышающего 109. Команда "−" означает изъятие элемента из стека. Гарантируется, что не происходит извлечения из пустого стека. Гарантируется, что размер стека в процессе выполнения команд не превысит 106 элементов.

Формат входного файла

В первой строке входного файла содержится M (1≤M≤106) — число команд. Каждая последующая строка исходного файла содержит ровно одну команду.

Формат выходного файла

Выведите числа, которые удаляются из стека с помощью команды "−", по одному в каждой строке. Числа нужно выводить в том порядке, в котором они были извлечены из стека. Гарантируется, что изъятий из пустого стека не производится.

Задача 13. Очередь

Ограничение по времени – 2 секунды. Ограничение по памяти – 256 мегабайт.

Реализуйте работу очереди. Для каждой операции изъятия элемента выведите ее результат.

На вход программе подаются строки, содержащие команды. Каждая строка содержит одну команду. Команда — это либо «+ N», либо «−». Команда «+ N» означает добавление в очередь числа N, по модулю не превышающего 109. Команда «−» означает изъятие элемента из очереди. Гарантируется, что размер очереди в процессе выполнения команд не превысит 106 элементов.

Формат входного файла

В первой строке содержится M (1≤M≤106) — число команд. В последующих строках содержатся команды, по одной в каждой строке.

Формат выходного файла

Выведите числа, которые удаляются из очереди с помощью команды «−», по одному в каждой строке. Числа нужно выводить в том порядке, в котором они были извлечены из очереди. Гарантируется, что извлечения из пустой очереди не производится.

Задача 14. Скобочная последовательность

Последовательность A, состоящую из символов из множества «(», «)», «[» и «]», назовем *правильной скобочной последовательностью*, если выполняется одно из следующих утверждений:

A — пустая последовательность;

первый символ последовательности A — это «(», и в этой последовательности существует такой символ «)», что последовательность можно представить как A=(B)C, где B и C — правильные скобочные последовательности;

первый символ последовательности A — это «[», и в этой последовательности существует такой символ «]», что последовательность можно представить как A=[B]C, где B и C — правильные скобочные последовательности.

Так, например, последовательности «(())» и «()[]» являются правильными скобочными последовательностями, а последовательности «[)» и «((» таковыми не являются.

Входной файл содержит несколько строк, каждая из которых содержит последовательность символов «(», «)», «[» и «]». Для каждой из этих строк выясните, является ли она правильной скобочной последовательностью.

Формат входных данных:

Первая строка входного файла содержит число N (1≤N≤500) - число скобочных последовательностей, которые необходимо проверить. Каждая из следующих N строк содержит скобочную последовательность длиной от 1 до 104 включительно. В каждой из последовательностей присутствуют только скобки указанных выше видов.

Формат выходных данных:

Для каждой строки входного файла выведите в выходной файл «YES», если соответствующая последовательность является правильной скобочной последовательностью, или «NO», если не является.

Задача 15. Очередь с приоритетами

Реализуйте очередь с приоритетами. Ваша очередь должна поддерживать следующие операции: добавить элемент, извлечь минимальный элемент, уменьшить элемент, добавленный во время одной из операций.

Формат входных данных:

В первой строке входного файла содержится число n (1≤n≤106) - число операций с очередью.

Следующие n строк содержат описание операций с очередью, по одному описанию в строке. Операции могут быть следующими:

A x — требуется добавить элемент x в очередь.

X — требуется удалить из очереди минимальный элемент и вывести его в выходной файл. Если очередь пуста, в выходной файл требуется вывести звездочку «*».

D x y — требуется заменить значение элемента, добавленного в очередь операцией A в строке входного файла номер x+1, на y. Гарантируется, что в строке x+1 действительно находится операция A, что этот элемент не был ранее удален операцией X, и что y меньше, чем предыдущее значение этого элемента.

В очередь помещаются и извлекаются только целые числа, не превышающие по модулю 109.

Формат выходных данных:

Выведите последовательно результат выполнения всех операций X, по одному в каждой строке выходного файла. Если перед очередной операцией X очередь пуста, выведите вместо числа звездочку «*».

Задача 16. Куча ли?

Структуру данных «куча», или, более конкретно, «неубывающая пирамида», можно реализовать на основе массива. Для этого должно выполнятся основное свойство неубывающей пирамиды, которое заключается в том, что для каждого $1 \le i \le n$ выполняются условия:

- если $2i \le n$, то $a[i] \le a[2i]$;
- если $2i+1 \le n$, то $a[i] \le a[2i+1]$.

Дан массив целых чисел. Определите, является ли он неубывающей пирамидой.

Формат входных данных:

Первая строка входного файла содержит целое число n ($1 \le n \le 106$). Вторая строка содержит n целых чисел, по модулю не превосходящих $2 \cdot 109$.

Формат выходных данных:

Выведите «YES», если массив является неубывающей пирамидой, и «NO» в противном случае.

Задача 17. Очередь с минимумом

Реализуйте работу очереди. В дополнение к стандартным операциям очереди, необходимо также отвечать на запрос о минимальном элементе из тех, которые сейчас находится в очереди. Для каждой операции запроса минимального элемента выведите ее результат.

На вход программе подаются строки, содержащие команды. Каждая строка содержит одну команду. Команда — это либо «+ N», либо «−», либо «?». Команда «+ N» означает добавление в очередь числа N, по модулю не превышающего 109. Команда «−» означает изъятие элемента из очереди. Команда «?» означает запрос на поиск минимального элемента в очереди.

Формат входных данных:

В первой строке содержится M ($1 \le M \le 106$) — число команд. В последующих строках содержатся команды, по одной в каждой строке.

Формат выходных данных:

Для каждой операции поиска минимума в очереди выведите её результат. Результаты должны быть выведены в том порядке, в котором эти операции встречаются во входном файле. Гарантируется, что операций извлечения или поиска минимума для пустой очереди не производится.

Задача 18. Quack

Ограничение по времени – 2 секунды. Ограничение по памяти – 256 мегабайт.

Язык Quack — забавный язык, который фигурирует в одной из задач с Internet Problem Solving Contest. В этой задаче вам требуется написать интерпретатор языка Quack.

Виртуальная машина, на которой исполняется программа на языке Quack, имеет внутри себя очередь, содержащую целые числа по модулю 65536 (то есть, числа от 0 до 65535, соответствующие беззнаковому 16-битному целому типу). Слово get в описании операций означает извлечение из очереди, put — добавление в очередь. Кроме того, у виртуальной машины есть 26 регистров, которые обозначаются буквами от 'a' до 'z'. Изначально все регистры хранят нулевое значение. В языке Quack существуют следующие команды (далее под α и β подразумеваются некие абстрактные временные переменные):

+	Сложение: get α, get β, put (α+β) mod 65536
-	Вычитание: get α, get β, put (α−β) mod 65536
*	Умножение: get α, get β, put (α·β) mod 65536
/	Целочисленное деление: get α, get β, put α div β (будем считать, что α div 0 = 0)
%	Взятие по модулю: get α, get β, put α mod β (будем считать, что α mod 0 = 0)
>[register]	Положить в регистр: get α, установить значение [register] в α
<[register]	Взять из регистра: put значение [register]
P	Напечатать: get α, вывести α в стандартный поток вывода и перевести строку
P[register]	Вывести значение регистра [register] в стандартный поток вывода и перевести строку
C	Вывести как символ: get α, вывести символ с ASCII-кодом α mod 256 в стандартный поток вывода
C[register]	Вывести регистр как символ: вывести символ с ASCII-кодом α mod 256 (где α — значение регистра [register]) в стандартный поток вывода
:[label]	Метка: эта строка программы имеет метку [label]
J[label]	Переход на строку с меткой [label]
Z[register][label]	Переход если 0: если значение регистра [register] равно нулю, выполнение программы продолжается с метки [label]
E[register1][register2][label]	Переход если равны: если значения регистров [register1] и [register2] равны, исполнение программы продолжается с метки [label]
G[register1][register2][label]	Переход если больше: если значение регистра [register1] больше, чем значение регистра [register2], исполнение программы продолжается с метки [label]
Q	Завершить работу программы. Работа также завершается, если выполнение доходит до конца программы
[number]	Просто число во входном файле — put это число

Формат входного файла

Входной файл содержит синтаксически корректную программу на языке Quack. Известно, что программа завершает работу не более чем за 105 шагов. Программа содержит не менее одной и не более 105 инструкций. **Метки имеют длину от 1 до 10 и состоят из цифр и латинских букв.**

Формат выходного файла

Выведите содержимое стандартного потока вывода виртуальной машины в выходной файл.

Задача 19. Расстояние в дереве

Дано взвешенное дерево. Найти кратчайшее расстояние между заданными вершинами.

Формат входных данных:

Первая строка содержит целое число n — количество вершин в дереве ($1 \le n \le 50000$). Вершины нумеруются целыми числами от 0 до n – 1. В следующих n – 1 строках содержится по три целых числа u, v, w, которые соответствуют ребру весом w ($0 \le w \le 1000$), соединяющему вершины u и v. В следующей строке содержится целое число m — количество запросов ($1 \le m \le 75000$). В следующих m строках содержится по два числа — номера вершин, расстояние между которыми необходимо вычислить.

Формат выходных данных:

Для каждого запроса выведите на отдельной строке одно число — искомое расстояние.

Задача 20. Вложенные отрезки

Ограничение времени: 1 секунда. Ограничение памяти: 64 МБ.

На прямой лежат n отрезков. Для каждой пары отрезков известно, что они либо не имеют общих точек, либо все точки одного из них также принадлежат и другому отрезку.

Дано m запросов. Каждый запрос представляет собой точку на прямой. Найдите для каждого запроса отрезок минимальной длины, которому принадлежит эта точка.

Формат входных данных:

В первой строке записано целое число n — количество отрезков ($1 \le n \le 10^5$). i-я из следующих n строк содержит целые числа a_i и b_i — координаты концов i-го отрезка ($1 \le a_i < b_i \le 10^9$). Отрезки упорядочены по неубыванию a_i, а при $a_i = a_j$ — по убыванию длины. Совпадающих отрезков нет. В следующей строке записано целое число m — количество запросов ($1 \le m \le 10^5$). В j-й из следующих m строк записано целое число c_j — координата точки ($1 \le c_j \le 10^9$). Запросы упорядочены по неубыванию c_j.

Формат выходных данных:

Для каждого запроса выведите номер искомого отрезка в отдельной строке. Если точка не принадлежит ни одному отрезку, выведите «-1». Отрезки пронумерованы числами от 1 до n в том порядке, в котором они перечислены во входных данных.

Задача 21. Столовая

Ограничение времени: 1 секунда. Ограничение памяти: 64 МБ

Опасно есть в столовой — можно отравиться несвежими продуктами, и вообще — опасно. Причём один человек может впасть в кому от столовской курицы, а другому хоть бы хны. И наоборот. Еду в столовой готовят из M разных продуктов. Всего в меню N продуктов, но не все они есть на раздаче. Допустим, что едят эту еду $K + 1$ студентов и для каждого из них известно, какими продуктами он может отравиться. Первым, естественно, ест самый хитрый — тот, кто пролез без очереди. И он, допустим, не отравился. Как тогда обед подействует на остальных?

Формат входных данных:

В первой строке находится одно число N ($1 \le N \le 100$). В следующих N строках названия продуктов — непустые последовательности латинских букв и цифр длиной не более 40 символов. Затем следует число K ($1 \le K \le 100$), за которым идёт $K + 1$ блоков, описывающих продукты из меню, опасные для посетителей столовой. i-й такой блок начинается строкой с числом N_i — количеством продуктов, вслед за которым идёт N_i строк с названиями опасных продуктов ($0 \le N_i \le N$). Первый блок описывает продукты, опасные для самого хитрого студента, следующие K блоков — для всех остальных. Вход заканчивается строкой, содержащей число M ($0 \le M \le N$).

Формат выходных данных:

Выведите K строк — в i-й строке:

YES, если обед будет полностью безвреден для $(i + 1)$-го студента,

NO, если среди имеющихся продуктов есть вредный для $(i + 1)$-го студента,

MAYBE, если при таких исходных данных возможна и та, и другая ситуация

Задача 22. Четность

Ограничение времени: 2 секунды. Ограничение памяти: 64 МБ

Вы играете со своим другом в следующую игру. Ваш друг записывает последовательность, состоящую из нулей и единиц. Вы выбираете непрерывную подпоследовательность (например, подпоследовательность от третьей до пятой цифры включительно) и спрашиваете его, чётное или нечётное количество единиц содержит эта подпоследовательность. Ваш друг отвечает, после чего вы можете спросить про другую подпоследовательность, и так далее.

Ваша задача — угадать всю последовательность чисел. Но вы подозреваете, что некоторые из ответов вашего друга могут быть неверными, и хотите уличить его в обмане. Вы решили написать программу, которая получит наборы ваших вопросов вместе с ответами друга и найдет первый ответ, который гарантированно неверен. Это должен быть такой ответ, что существует последовательность, удовлетворяющая ответам на предыдущие вопросы, но никакая последовательность не удовлетворяет этому ответу.

Формат входных данных:

Ввод содержит несколько тестов. Первая строка каждого теста содержит одно число, равное длине последовательности нулей и единиц. Эта длина не превосходит 10^9. Во второй строке находится одно неотрицательное целое число — количество заданных вопросов и ответов на них. Количество вопросов и ответов не превышает 5 000. Остальные строки содержат вопросы и ответы. Каждая строка содержит один вопрос и ответ на этот вопрос: два целых числа (позиции первой и последней цифр выбранной подпоследовательности) и одно слово — “`even`” или “`odd`” — ответ, сообщающий чётность количества единиц в выбранной подпоследовательности, где “`even`” означает чётное количество единиц, а “`odd`” означает нечётное количество. Ввод заканчивается строкой, содержащей −1.

Формат выходных данных:

Каждая строка вывода должна содержать одно целое число X. Число X показывает, что существует последовательность нулей и единиц, удовлетворяющая первым X условиям чётности, но не существует последовательности, удовлетворяющей $X + 1$ условию. Если существует последовательность нулей и единиц, удовлетворяющая всем заданным условиям, то число X должно быть равно количеству всех заданных вопросов.

Задача 23. Дерево

Ограничение времени: 1 секунда. Ограничение памяти: 64 МБ.

Рассмотрим дерево, состоящее из n вершин. Назовём *расстоянием* между двумя вершинами минимальное количество рёбер в пути, соединяющем эти вершины. По вершине v_i и расстоянию d_i найдите такую вершину u_i, что расстояние между v_i и u_i равняется d_i.

Формат входных данных:

В первой строке записано количество вершин n $(1 \le n \le 20000)$ и количество запросов q $(1 \le q \le 50000)$. Каждая из следующих $n - 1$ строк описывает ребро и содержит номера вершин, соединённых этим ребром. Вершины занумерованы числами от 1 до n. В следующих q строках заданы запросы. Каждый запрос представляет собой строку, в которой записаны числа v_i $(1 \le v_i \le n)$ и d_i $(0 \le d_i \le n)$.

Формат выходных данных:

Выведите q строк. В i-й строке выведите номер вершины u_i — ответ на i-й запрос. Если существует несколько возможных ответов, выведите любой из них. Если искомой вершины не существует, выведите 0.

Задача 24. Постфиксная запись

В постфиксной записи (или обратной польской записи) операция записывается после двух операндов. Например, сумма двух чисел `A` и `B` записывается как `A B +`. Запись `B C + D *` обозначает привычное нам `(B + C) * D`, а запись `A B C + D * +` означает `A + (B + C) * D`. Достоинство постфиксной записи в том, что она не требует скобок и дополнительных соглашений о приоритете операторов для своего чтения.

Дано выражение в обратной польской записи. Определите его значение.

Формат входных данных:

В первой строке входного файла дано число N (1≤N≤106) - число элементов выражения. Во второй строке содержится выражение в постфиксной записи, состоящее из N элементов. В выражении могут содержаться неотрицательные однозначные числа и операции +, −, *. Каждые два соседних элемента выражения разделены ровно одним пробелом.

Формат выходных данных:

Необходимо вывести значение записанного выражения. Гарантируется, что результат выражения, а также результаты всех промежуточных вычислений, по модулю будут меньше, чем 231.

Задача 25. Миллиардеры

Ограничение времени: 3 секунды. Ограничение памяти: 64 МБ

Возможно, вы знаете, что из всех городов мира больше всего миллиардеров живёт в Москве. Но, поскольку работа миллиардера подразумевает частые перемещения по всему свету, в определённые дни какой-то другой город может занимать первую строчку в таком рейтинге. Ваши приятели из ФСБ, ФБР, MI5 и Шин Бет скинули вам списки перемещений всех миллиардеров за последнее время. Ваш работодатель просит посчитать, сколько дней в течение этого периода каждый из городов мира был первым по общей сумме денег миллиардеров, находящихся в нём.

Формат входных данных:

В первой строке записано число n — количество миллиардеров ($1 \le n \le 10000$). Каждая из следующих n строк содержит данные на определённого человека: его имя, название города, где он находился в первый день данного периода, и размер состояния. В следующей строке записаны два числа: m — количество дней, о которых есть данные ($1 \le m \le 50000$), k — количество зарегистрированных перемещений миллиардеров ($0 \le k \le 50000$). Следующие k строк содержат список перемещений в формате: номер дня (от 1 до $m - 1$), имя человека, название города назначения. Вы можете считать, что миллиардеры путешествуют не чаще одного раза в день, и что они отбывают поздно вечером и прибывают в город назначения рано утром следующего дня. Список упорядочен по возрастанию номера дня. Все имена и названия городов состоят не более чем из 20 латинских букв, регистр букв имеет значение. Состояния миллиардеров лежат в пределах от 1 до 100 миллиардов.

Формат выходных данных:

В каждой строке должно содержаться название города и, через пробел, количество дней, в течение которых этот город лидировал по общему состоянию миллиардеров, находящихся в нём. Если таких дней не было, пропустите этот город. Города должны быть отсортированы по алфавиту (используйте обычный порядок символов: ABC...Zabc...z).