МИНОБРНАУКИ РОССИИ

–––––––—————––––––––––––—————–––––––––––––

Санкт-Петербургский государственный электротехнический

университет «ЛЭТИ» им. В. И. Ульянова (Ленина)

––––––———––––––––––––—————–––––––––

**МЕТОДИЧЕСКИЕ УКАЗАНИЯ**

**К ЛАБОРАТОРНЫМ РАБОТАМ, ПРАКТИЧЕСКИМ**

**ЗАНЯТИЯМ И КУРСОВОЙ РАБОТЕ**

Учебно-методическое пособие

Санкт-Петербург

2017

УДК 004.02

Методические указания к лабораторным работам, практическим занятиям и курсовой работе : учеб.-метод. пособие / сост.: М.А. Фирсов. СПб, 2016. 18 с.

Представлены материалы по дисциплине «Построение и анализ алгоритмов». Рассматриваются вопросы программной реализации классических алгоритмов. Приводятся примеры типовых заданий и возможные варианты их выполнения.

Предназначено для студентов направлений «Программная инженерия» и «Прикладная математика и информатика».

Одобрено

Методической комиссией факультета компьютерных технологий и информатики в качестве учебно-методического пособия

© СПбГЭТУ «ЛЭТИ», 2016

**ВВЕДЕНИЕ**

Дисциплина «Построение и анализ алгоритмов» входит в учебные планы подготовки бакалавров по направлениям «Прикладная математика и информатика» и «Программная инженерия» по профилю «Разработка программно-информационных систем». Рабочая программа дисциплины предусматривает лекционные, лабораторные и практические занятия с преподавателем, а также большой объем самостоятельной работы студентов, в т.ч. выполнение курсовой работы.

Учебно-методическое пособие поддерживает всю практическую деятельность студента при изучении дисциплины, включая практические и лабораторные занятия, а также выполнение курсовой работы.

Цель практических занятий – помочь студентам при подготовке к выполнению курсовой и лабораторных работ. На практических занятиях преподаватель на примерах поясняет лекционный материал, рассматривает постановку задачи и общую схему ее решения, возможные варианты реализации алгоритмов на языке С++. Студент выбирает по каждому разделу индивидуальное задание из предложенного в пособии списка и реализует его в рамках часов, отведенных на лабораторные занятия и самостоятельную работу. Задания необходимо выполнять в соответствии с требованиями, указанными в каждом разделе пособия и с учетом рекомендаций преподавателя.

Обсуждаются возможные варианты выполнения типовых заданий. Приложение к каждому разделу содержит список предлагаемых студентам индивидуальных заданий.

Информационные технологии (операционные системы, программное обеспечение общего и специализированного назначения, информационные справочные системы) и материально-техническая база, используемые при осуществлении образовательного процесса по дисциплине, соответствуют требованиям федерального государственного образовательного стандарта высшего образования.

Для обеспечения образовательного процесса по дисциплине используются следующие информационные технологии:

1. Операционные системы: Microsoft Windows 10 Version 1607 Education, Ubuntu Desktop 16.04.1.

2. Программное обеспечение общего и специализированного назначения: Microsoft Visual Studio Professional 2015, Qt 5.5.1, TDM64-GCC 5.1.0, Code::Blocks 16.01, LibreOffice 5.1.4.2, WPS Office (9.1.0.5240).

3. Информационные справочные системы: международная ассоциация сетей «Интернет», электронные библиотечные системы и ресурсы удаленного доступа библиотеки СПбГЭТУ «ЛЭТИ».

Образовательный процесс обеспечивается на следующей материально-технической базе (один из трех вариантов):

1) Персональный компьютер (системный блок RAMEC STORM, монитор LG L1953S, клавиатура, мышь, ИБП APC), персональный компьютер (системный блок RAMEC STORM, монитор LG L1953S, клавиатура, мышь, ИБП APC), сервер (RAMEC, монитор LG L1953S, клавиатура, мышь, ИПБ APC), проектор Mitsubishi XD430U, экран проекционный настенный;

2) Персональный компьютер (системный блок RAMEC STORM, монитор LG L1953S, клавиатура, мышь, ИБП APC), персональный компьютер (системный блок RAMEC STORM, монитор LG L222WS, клавиатура, мышь, ИБП APC), принтер НР Laser Jet, принтер НР Color Laser Jet, сервер SuperMicro, сканер планшетный HP LaserJet, проектор Mitsubishi XD430U, экран проекционный настенный подпружиненный ScreenMedia Goldview 213x213 MW;

3) Персональный компьютер (системный блок Hewlett-Packard, монитор Samsung SyncMaster 913N, клавиатура, мышь), персональный компьютер (системный блок Hewlett-Packard, монитор Samsung SyncMaster E2220, клавиатура, мышь), проектор Vivitek D555, компьютер-сервер (системный блок Universal KOMPUMIR, монитор Samsung SyncMaster 913N, клавиатура, мышь).

**РАЗДЕЛ 1. ТРЕБОВАНИЯ**

1. Интерфейс программы с внешним окружением

1.1. Ввод данных

Входные данные программы целесообразно разделить на две части:

1. основной набор данных (как правило, относительно большого объема, т.е. те данные, которые обременительно вводить при каждом запуске программы) должен вводиться из файла;
2. небольшое количество входных параметров, которые предполагается изменять относительно часто при запусках программы, и которые должны вводится с клавиатуры; эти параметры могут быть либо параметрами алгоритма, либо могут, например, определять текущий размер входных данных, которые будут составлять только часть размещенных в файле.

Ввод данных с клавиатуры должен сопровождаться соответствующим диалогом (запрос, подтверждение).

Желательно, чтобы программа проверяла *корректность* входных данных (кроме случаев, оговоренных в задании, или указанных преподавателем).

Введенные корректные данные должны отображаться до начала основного вычислительного процесса и сохраняться в выходном наборе для последующего анализа результатов работы программы.

В некоторых случаях (по рекомендации преподавателя и с учетом режима тестирования) *все* входные данные должны быть прочитаны из входного файла (с оговоренным форматированием).

1.2. Вывод промежуточных (вспомогательных) данных и результатов

Вывод данных должен осуществляться в форме, удобной для анализа и интерпретации.

Кроме собственно результирующих данных в процессе работы программы **должны выводиться промежуточные (вспомогательные) данные, характеризующие вычислительный процесс**. Это требуется, во-первых, для демонстрации способа получения результата, а во-вторых, позволяет преподавателю оценить степень понимания студентом алгоритма и порождаемого алгоритмом вычислительного процесса. Количество и состав выводимых промежуточных данных, а также форма и «динамика» вывода выбираются студентом, исходя из существа задачи, а также с целью аргументировано убедить преподавателя в правильности и эффективности разработанной программы. Даже если программа очень простая, какие-то промежуточные данные обязательно должны выводиться.

Выходные данные должны выводиться *на экран* для экспресс-анализа (в режиме *online*) в ходе работы программы и *в файл* для сохранения, документирования, последующего анализа и интерпретации результатов работы программы (в режиме off-line).

В случае использования «автоматического» тестирования все выходные данные должны помещаться в выходной файл. При этом должны быть заранее определены их состав и формат вывода.

1.3. Организация рабочих и отчетных материалов, предъявляемых студентом:

- исходный код;

- запускаемый файл + файлы входных данных;

- отчет, включающий полный исходный код и тестирование;

2. Внутреннее устройство программы

2.1. Требования к разработке и структуре программы

Общие требования аналогичны использованным в предыдущих семестрах (дисциплина «Программирование»).

При разработке программ следует придерживаться идеологии абстрактных типов данных (АТД) и опираться на триаду «интерфейс-реализация-клиент». В разных лабораторных работах преподавателем могут быть рекомендованы различные парадигмы («процедурно-модульная»или «объектно-ориентированная»), а также использование элементов обобщенного программирования (шаблоны функций и классов).

2.2. Оформление текста программы (форматирование кода)

Рекомендуется при оформлении текстов программ использовать правила форматирования кода, объявленные на 1 курсе, или аналогичные им (по согласованию с преподавателем).

3. Тестирование и отладка

При защите лабораторной работы студент предъявляет преподавателю проект отчета, набор тестов, краткий протокол испытаний программы (тестирования и отладки). Количество тестов и их наполнение студент определяет самостоятельно с конечной целью убедить преподавателя в корректности разработанной программы, её эффективности и, возможно, надежности. Преподаватель может предложить свои тесты или рекомендовать студенту подготовить дополнительные, а затем провести соответствующие испытания программы. Успешным является тест, с помощью которого были обнаружены дефекты программы.

4. Требования к отчету

Отчет представляется в электронной форме в виде текстового файла (например, формата odt), пригодного для оперативного вывода на бумагу.

В разделе «Задание» должен быть указан номер варианта.

Отчет должен быть оформлен по шаблону, приведенному на сайте СПбГЭТУ «ЛЭТИ».

Содержание и форма отчета должны соответствовать требованиям, использованным в предыдущих семестрах (дисциплина «Программирование»). В отчёте должен быть раздел «Описание алгоритма». В приложении должны быть представлены тестовые наборы, результаты их обработки и краткий протокол испытаний программы (тестирования и отладки). Блок-схемы приводить не требуется.

**РАЗДЕЛ 2. ЗАДАНИЯ НА КУРСОВУЮ РАБОТУ**

**1-3. Минимальное остовное дерево**. Реализовать один из трех алгоритмов построения минимального остовного дерева заданного графа (алгоритмы: жадный [Kruskal], ближайшего соседа [Prim; Dijkstra], алгоритм со сложностью O(m log n) [Bоruvka]). Возможны вариации задания по реализациям используемых структур данных. Провести экспериментальный анализ их эффективности на графах разной степени насыщенности ребрами. Продумать и предложить механизм генерации необходимых для эксперимента тестовых данных.

**1. Минимальное остовное дерево. Алгоритм Краскала.**

**2. Минимальное остовное дерево. Алгоритм ЯПД.**

**3. Минимальное остовное дерево. Алгоритм Борувки.**

**4. Двусвязные компоненты.** Реализовать алгоритм нахождения двусвязных компонент графа. Провести экспериментальный анализ его эффективности на графах разной структуры. Продумать и предложить механизм генерации необходимых для эксперимента (или визуализации) тестовых данных.

**5. Мосты графа.** Мост в связном неориентированном графе – это ребро, удаление которого делает граф несвязным. Реализовать алгоритм нахождения мостов графа (модификацией поиска в глубину: ребро {v,w} графа будет мостом тогда и только тогда, когда {v,w} является ребром дерева и Low(w)>=NumVert(w)). Провести экспериментальный анализ его эффективности на графах разной структуры. Продумать и предложить механизм генерации необходимых для эксперимента тестовых данных.

**6. Сильно связные компоненты орграфа.** Реализовать алгоритм нахождения сильно связных компонент орграфа. Провести экспериментальный анализ его эффективности на графах разной структуры. Продумать и предложить механизм генерации необходимых для эксперимента тестовых данных.

**7. Клики графа.** Реализовать алгоритм порождения клик графа. Провести экспериментальный анализ его эффективности на графах разной (в том числе специальной) структуры (например, на графах Муна-Мозера). Продумать и предложить механизм генерации необходимых для эксперимента тестовых данных.

**8-10. Кратчайшие пути в графе** (Dijkstra, Ford-Bellman, Floyd-Warshall):

- визуализация;

- или эффективная реализация и исследование на графах разной структуры.

**8. Кратчайшие пути в графе. Алгоритм Дейкстры.**

**9. Кратчайшие пути в графе. Алгоритм Форда-Беллмана.**

**10. Кратчайшие пути в графе. Алгоритм Флойда-Уоршалла.**

**11. Транзитивное замыкание (Warshall)**

**12. Топологическая сортировка.**

**РАЗДЕЛ 3. ЛАБОРАТОРНЫЕ РАБОТЫ**

**3.1. Программирование и исследование алгоритмов сортировки**

**Цель** работы – познакомиться с часто используемыми на практике алгоритмами сортировок, исследовать сложность алгоритмов.

**Задания:**

Простые методы

1. Сортировка выбором (варианты: выбор максимума и одновременный выбор максимума и минимума).

2. Сортировка вставками (простые вставки (просмотр назад/вперед), бинарные вставки).

3. Пузырьковая сортировка (не менее 2 вариантов).

Усовершенствованные методы

4. Быстрая сортировка (варианты процедуры разделения).

5. Быстрая сортировка (варианты процедуры разделения – трехчастное разделение).

6. Быстрая сортировка (варианты отсечения малых подмассивов: прямая сортировка подмассивов или всего массива).

7. Быстрая сортировка (отсечение малых подмассивов: экспериментальный выбор параметра).

8. Быстрая сортировка (рекурсивный и итеративный варианты). См. Седж., 2.3.20.\*

9. Пирамидальная сортировка (варианты: построение пирамиды сверху вниз или снизу вверх).

10. Сортировка массивов слиянием (простое слияние). См. Вирт, 2.4.1\*, или Седж.

11. Сортировка массивов слиянием (естественное слияние). См. Вирт, 2.4.2, или Седж.

12. Сортировка Шелла.

13. Генерация входа для сортировки (см. \*\*).

14. Визуализация выхода сортировки (см. \*\*).

15. Интегральное сравнение набора алгоритмов сортировки.

16. Поразрядная сортировка.

Порядковые статистики (медиана и т.п.)

17. k-й по порядку или k-й минимум («малое» k (*k* ≤ *n* / log2*n*) – через пирамиду). См. Седж.

18. k-й минимум (разделение Хоора).

19. k-й минимум (линейный по сложности алгоритм – prune and search).

Применение сортировок\*\*\*

20. Пересечение множеств. Заданы два множества (размеров *n* и *m*). Являются ли они пересекающимися?

А. Решение может опираться на сортировку (одного или обоих) множеств и бинарный поиск, если надо. Рассмотреть случай *m<<n* и различные варианты решения.

Б. Решение посредством хеширования. Создается хеш-таблица, содержащая элементы обоих множеств, и проверяется, что коллизии являются результатом хеширования идентичных элементов. На практике это может быть лучшим решением.

21. В наборе *S* имеется *n* вещественных чисел. Задано также вещественное число *x*. Содержатся ли в *S* два таких элемента, что их сумма равна *x*.

Указание. Если набор *S* отсортирован, то решить задачу можно за время *O*(*n*).

22. Заданы *k* отсортированных списков c общим количеством *n* элементов. Требуется осуществить их слияние в один отсортированный список.

Указание. Простейший алгоритм требует времени *O*(*kn*). Использовать пирамиду и получить время выполнения *O*(*n* log *k*).

23. «Болты и гайки». Имеется куча перемешанных *n* болтов и *n* гаек, отличающихся диаметрами. Нужно *быстро* найти все соответствующие пары болтов и гаек. Известно, что каждая гайка подходит по диаметру ровно к одному болту, и наоборот. Нет ни двух болтов одинакового диаметра, ни двух гаек одинакового диаметра. Попробовав навинтить гайку на болт, можно определить, что из них больше (или они соответствуют друг другу), но невозможно (а в алгоритме – нельзя) непосредственно сравнить два болта или две гайки.

[Простой алгоритм – *O*(*n*2). Это *не быстро*. Реализовать аналог рандомизированной быстрой сортировки – *O*(*n* log *n*).]

Примечания.

\* Литература:

- Вирт Н. Алгоритмы и структуры данных / пер. с англ. Д. Б. Подшивалова. – М. : Мир, 1989.

- Седжвик Р. Фундаментальные алгоритмы на С. Ч. 1-5 : Анализ структуры данных. Сортировка. Поиск : пер. с англ. – СПб. : Диа-СофтЮП, 2003.

\*\* Обеспечивающие задания 13 и 14. Другие студенты при выполнении своих заданий могут использовать разработанные в заданиях 13-14 программы (компоненты).

Генерация: при случайной генерации задаются тип элементов, размер массива, диапазон значений элементов, наличие (или требование отсутствия) и доля совпадающих значений элементов, характеристика существующей упорядоченности (параметр упорядоченности); при генерации массива регулярной структуры также задаются соответствующие параметры (регулярная структура: уже упорядочен, обратно упорядочен, чередование регулярных участков, смесь регулярных участков и т.п.)

Визуализация: на усмотрение преподавателя. Можно использовать любые простейшие графические средства (в т.ч. с учетом будущих заданий на графах).

Указания.

Все задания 1-19 (кроме 13-14) используют известные алгоритмы. Конечно, алгоритмы имею разную «сложность» (по их пониманию и реализации). Основной упор в задании делается на проведение экспериментального испытания (исследования) алгоритма (программной реализации). При этом оценивается эффективность алгоритма (его вариаций, разновидностей, модификаций). Основной показатель эффективности в этих заданиях – это время выполнения. Время выполнения должно измеряться, во-первых, с использованием компьютерного таймера. Во-вторых, требуется в ходе выполнения программы подсчитывать количество произведенных базовых операций алгоритма, например, операций сравнения и операций присваивания (перемещения элементов). По результатам испытаний должны анализироваться:

- зависимость времени выполнения и количества операций от размера массива,

- соотношение измеренных и теоретически предсказанных значений,

- сравнительные характеристики эффективности разных вариаций (модификаций) алгоритмов,

- и др., в зависимости от конкретного задания.

Во всех случаях следует провести сравнения (по тем показателям, по которым это возможно) с каким-либо «эталонным» алгоритмом (реализацией), например, с системной функцией qsort или с какой-либо ещё библиотечной функцией. Это необходимо в том числе и для проверки корректности сортировки.

\*\*\* Задания 20-23 имеют творческий характер в том смысле, что алгоритм нужно самостоятельно предложить, обосновать и испытать. Как и для других заданий, должны быть проведены экспериментальные исследования реализованных алгоритмов.

**3.2. Программирование алгоритма поиска с возвращением**

**Цель** работы – практическое освоение решения задач методом поиска с возвращением (полного перебора).

**Задания:**

1. **Лабиринты.** Найдите путь между двумя выделенными точками. Найдите кратчайший такой путь. Найдите все такие пути.

2. **Задача о почтовых марках.** В некоторой стране почтовое отделение выпускает марки n различных достоинств и запрещает наклеивать на одно письмо более чем m марок. Стоимость почтового отправления может быть 1, 2 и т.д. единиц. Для различных n и m определите наибольшую последовательность стоимостей от 1 и выше и все возможные наборы достоинств марок так, чтобы любая стоимость отправления могла быть оплачена с соблюдением указанных выше требований. Например, для марок четырех различных достоинств и при условии, что на одно письмо разрешается наклеивать не более 5 марок, любые почтовые расходы из последовательности 1, 2, …, 71 можно оплатить с помощью набора {1, 4, 12, 21} или с помощью набора {1, 5, 12, 28}.

3. **Покрытие несравнимыми прямоугольниками.** Два прямоугольник называются несравнимыми, если ни один из них не может быть размещен внутри другого, когда они расположены так, что соответственные стороны параллельны. Найдите покрытия различных прямоугольных областей наборами попарно несравнимых прямоугольников. Например, наименьшее возможное решение представляет собой прямоугольник 12×22, покрытый следующим образом (рис. 1):

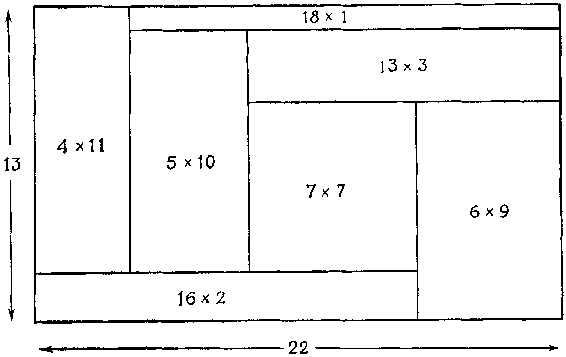


Рисунок 1

Наименьший из известных квадратов, который можно покрыть несравнимыми прямоугольниками, имеет размер 27×27. Существует ли еще меньший?

4. **Игра солитер.** Тридцать две фишки расставляются, как показано на рис. 2, на все клетки доски, кроме центральной. Фишка переносится на свободную клетку через любую соседнюю фишку, которая при этом снимается с доски (ходить можно только по вертикали и горизонтали).

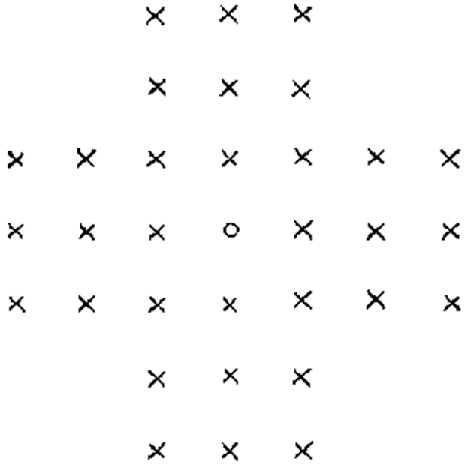


Рисунок 2

Найдите последовательность прыжков такую, чтобы на доске осталась всего одна фишка в центральной клетке.

5. **Симметричная n×n-задача о не бьющих друг друга ферзях.** Найдите все решения n×n-задачи о не бьющих друг друга ферзях, инвариантные относительно поворотов на 90˚, 180˚ и 270˚. В такой постановке каждый ферзь, помещенный не в центре, определяет положение трех других ферзей.

6. **Минимальное множество не атакующих друг друга ферзей и доминирующих на шахматной доске размера n×n.** Определите минимальное число не атакующих друг друга ферзей и все возможные их распределения так, чтобы каждая незанятая клетка шахматной доски находилась под боем.

7. **Маршрут коня на шахматной доске размера n×n.** Существует ли способ передвижения коня по шахматной доске такой, что конь посещает каждую клетку точно один раз и возвращается в исходную точку?

8. **Пентамино.** Имеется 12 фигур пентамино (жестких, связных, плоских фигур, каждая из которых составлена из пяти квадратов одного размера, соединенных по ребрам), которые можно складывать в виде мозаики многими способами для получения прямоугольника размера 6×10. Например, одно решение имеет вид:

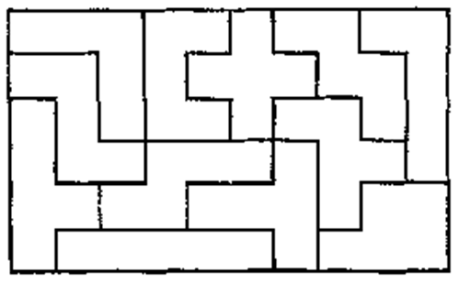


Рисунок 3

Найдите все решения для прямоугольников размера 6×10 и 5×12.

9. **Кубики сома.** Следующие 7 элементов можно сложить вместе многими способами для того, чтобы составить куб размером 3×3×3:

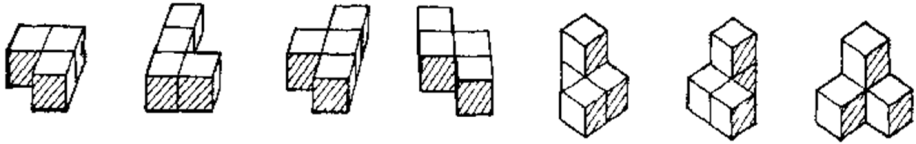


Рисунок 4

Найдите число неэквивалентных (относительно вращения и/или отражения) решений.

10. **Куб Микусинского.** Из следующих элементов можно составить куб размером 3×3×3 только несколькими способами:

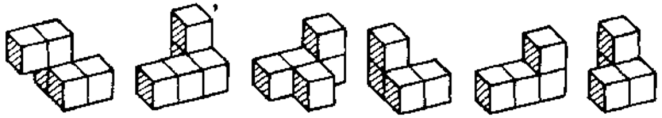


Рисунок 5

Найдите все неэквивалентные решения.

11. **Y-пентамино.** Найти наименьшее n, такое, что область размера 12×5n можно покрыть элементами Y-пентамино вида, показанного на рисунке 6:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |

Рисунок 6

Найдите все значения n, для которых это возможно.

12. **Сумма квадратов.** Из тождества 12+22+32+…+242 = 4900 = 702 можно предположить, что имеется возможность покрыть квадрат размера 70×70, если использовать 24 квадрата размеров 1×1, 2×2, 3×3, …, 24×24 соответственно. Существует ли решение?

13. **Минимизация числа линий в кубе.** Дан куб размера n×n×n, состоящий из n3 ячеек; заполнить ячейки куба белыми и черными шарами так, чтобы минимизировать число линий, включая диагональные линии, в которых все n шаров имеют одинаковый цвет. Вычислите как общее число решений, так и общее число решений, неэквивалентных относительно вращений и отражений.

14. **Магические квадраты.** Магический квадрат порядка n представляет собой таблицу размера n×n, заполненную числами от 1 до n2, которые расположены так, что суммы в каждой строке, в каждом столбце и двух диагоналях одинаковы. Поскольку

сумма должна быть равна 0.5n(n2+1). Так, например, таблица 1 является магическим квадратом порядка 5:

Таблица 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 15 | 24 | 8 | 17 |
| 23 | 7 | 16 | 5 | 14 |
| 20 | 4 | 13 | 22 | 6 |
| 12 | 21 | 10 | 19 | 3 |
| 9 | 18 | 2 | 11 | 25 |

В каждой строке, каждом столбце и на главных диагоналях сумма равна 0.5∙5(52+1)=65. Определите число неэквивалентных (относительно вращений и/или отражений) магических квадратов порядка n.

15. **Пандиагональные магические квадраты.** Приведенный в задании 14 магический квадрат имеет дополнительное свойство: «разорванные диагонали» также дают в сумме 65. Когда пандиагональный магический квадрат порядка n используется для покрытия плоскости, любой n×n-квадрат является пандиагональным магическим квадратом. Определите число неэквивалентных пандиагональных магических квадратов порядка n.

16. **Латинские квадраты.** Латинский квадрат порядка n есть таблица размера n×n, в которой в каждой клетке находится целое число от 1 до n, такая, что каждая ее строка и каждый столбец является перестановкой чисел из множества {1, 2, …, n}. Так, таблица 2 является латинским квадратом порядка 4.

Таблица 2

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| 3 | 4 | 1 | 2 |
| 4 | 3 | 2 | 1 |
| 2 | 1 | 4 | 3 |

Латинский квадрат называется приведенным, если в первых строке и столбце элементы стоят в естественном порядке 1, 2, …, n. Определите число приведенных латинских квадратов порядка n.

17. **Ортогональные латинские квадраты.** Два латинских квадрата порядка n называются *ортогональными*, если при наложении одного из них на другой каждая из n2 возможных пар элементов встречается ровно один раз. Например, латинские квадраты на рисунке 7 ортогональны квадрату из задания 16 и ортогональны друг другу.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |  |  | 1 | 2 | 3 | 4 |
| 2 | 1 | 4 | 3 |  |  | 4 | 3 | 2 | 1 |
| 3 | 4 | 1 | 2 |  |  | 2 | 1 | 4 | 3 |
| 4 | 3 | 2 | 1 |  |  | 3 | 4 | 1 | 2 |

Рисунок 7

Найдите все такие множества взаимно ортогональных латинских квадратов порядка n.

18. **Слоны.** Слон – это шахматная фигура, которая из текущей позиции может ходить только по диагонали. Два слона атакуют друг друга, если один из них находится на пути другого. На рисунке 8 темными клетками обозначены поля, доступные для слона В1 из его текущей позиции. Слоны В1 и В2 атакуют друг друга, а слоны В1 и В3 – нет. Слоны В2 и В3 также не атакуют друг друга.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |
|  |  |  | В3 |  |  | В2 |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  | В1 |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

Рисунок 8

Для заданных чисел n и k определите число способов, которыми можно расставить k слонов на доске размером n×n так, чтобы никакие два из них не атаковали друг друга.

*Входные данные:* два целых числа – n (1≤n≤8) и k(0≤k≤n2).

*Выходные данные:* полное число способов, которыми можно расставить заданное число слонов на доске заданного размера так, чтобы ни один из них не атаковал другого. Вы можете считать, что это число будет меньше 1015.

*Примеры:*

- n=8, k=6, результат – 5599888;

- n=4, k=4, результат – 260.

19. **Задача про пятнашки.** Пятнашки – это очень популярная головоломка, и вы наверняка видели ее, хотя, быть может, и не под таким именем. Она состоит из 15 квадратиков, которые могут скользить вдоль стороны и на каждом из которых написан свой номер от 1 до 15, причем все эти квадратики помещены в рамку размером 4×4 и в результате одно место остается незанятым (мы будем называть его пустышкой). Цель головоломки – упорядочить квадратики так, чтобы они шли в порядке, изображенном в таблице 3.

Таблица 3

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| 5 | 6 | 7 | 8 |
| 9 | 10 | 11 | 12 |
| 13 | 14 | 15 |  |

Единственное, что можно делать, – это менять местами пустышку и один из 2, 3 или 4 квадратиков, имеющих с ней общую сторону. Рассмотрим последовательность ходов, изображенную на рисунке 9.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 2 | 12 | 11 | 14 |  | 2 | 12 | 11 | 14 |  | 2 | 12 | 11 | 14 |  | 2 | 12 | 11 | 14 |
| 6 | 15 | 10 | 5 |  | 6 | 15 | 10 | 5 |  | 6 | 15 |  | 5 |  | 6 |  | 15 | 5 |
| 3 |  | 9 | 13 |  | 3 | 9 |  | 13 |  | 3 | 9 | 10 | 13 |  | 3 | 9 | 10 | 13 |
| 8 | 7 | 1 | 4 |  | 8 | 7 | 1 | 4 |  | 8 | 7 | 1 | 4 |  | 8 | 7 | 1 | 4 |
| а | | | |  | б | | | |  | в | | | |  | г | | | |

Рисунок 9. а – случайное состояние головоломки; б – пустышка движется вправо (R); в – пустышка движется вверх (U); г – пустышка движется влево (L).

Мы обозначаем ходы исходя из соседа пустышки, с которым она меняется местами. Разрешенными являются значения «R», «L», «U», «D» для перемещений пустышки вправо, влево, вверх и вниз.

Вы должны по начальному состоянию головоломки определить последовательность шагов, ведущую к итоговому состоянию. Для каждого *решаемого* начального расположения судейскому решению требуется максимум 45 ходов; вы ограничены 50 ходами для решения головоломки.

*Входные данные:* 4 строки из 4 чисел, задающие начальную позицию (пустышка обозначается нулем).

*Выходные данные:* если головоломка имеет решение, то последовательность решающих головоломку ходов в формате, описанном выше; если не имеет – то строка «Эта задача нерешаема.».

*Примеры:*

1)

2 3 4 0

1 5 7 8

9 6 10 12

13 14 11 15

Результат: LLLDRDRDR

2)

13 1 2 4

5 0 3 7

9 6 10 12

15 8 11 14

Результат: Эта задача нерешаема.

20. **Шеренга.** Рассмотрим шеренгу из N людей разного роста. Каждый из них видит все слева, если он выше всех людей, стоящих слева; иначе поле зрения перекрыто. Аналогично каждый из них видит все справа, если он выше всех людей, стоящих справа.

Было совершено преступление: человек, стоящий слева от шеренги, убил бумерангом человека, стоящего справа от шеренги. Ровно P человек в шеренге видели все слева и ровно R человек видели все справа, и, таким образом, они могут служить свидетелями.

Защита наняла вас, чтобы определить, сколько перестановок N людей обладают таким свойством для заданных P и R.

*Входные данные:* 3 целых числа. Первое число – N – задает число людей в шеренге (1≤N≤13). Второе число – P – количество людей, которые видят все слева. Третье число – R – количество людей, видящих все справа.

*Выходные данные:* число перестановок N людей таких, что P людей видят все слева и R людей видят все справа.

*Примеры:*

- N=10, P=4, R=4, результат – 90720;

- N=11, P=3, R=1, результат – 1026576;

- N=3, P=1, R=2, результат – 1.

21. **Станции техобслуживания.** Компания занимается продажей персональных компьютеров в N городах (3≤N≤35), обозначаемых 1, 2, …, N. M пар этих городов соединены прямыми дорогами. Компания решила построить несколько станций техобслуживания так, чтобы для любого города X станция находилась либо непосредственно в X, либо в каком-то городе, напрямую соединенном с X.

Напишите программу, определяющую минимальное число станций, необходимых компании.

*Входные данные:* несколько строк, каждая из которых состоит из двух целых чисел, разделенных пробелом. В первой строке содержатся число городов N и число пар M; каждая из следующих M строк содержит пару чисел, определяющих соединенные города, причем на одну строку приходится ровно одна пара чисел.

*Выходные данные:* число необходимых станций техобслуживания.

*Пример входных данных:*

8 12

1 2

1 6

1 8

2 3

2 6

3 4

3 5

4 5

4 7

5 6

6 7

6 8

*Соответствующие выходные данные:* 2.

22. **Перетягивание каната.** Перетягивание каната – это состязание в грубой силе, когда две группы людей тянут канат в противоположные стороны. Та команда, которая сумела утянуть канат в свою сторону, объявляется победителем.

На корпоративном пикнике решили посостязаться в перетягивании каната. Участников пикника нужно честно разбить на две команды. Каждый человек должен попасть в одну или другую команду, число человек в одной команде не должно превышать число человек в другой более чем на одного, и суммарные веса людей каждой команды должны быть близки, насколько это возможно.

*Входные данные.* Первая строка содержит n, число участников пикника. Каждая из следующих n строк содержит вес одного из участников пикника, причем вес задается целым числом между 1 и 450. Число участников пикника не превышает 100.

*Выходные данные.* Два числа: суммарный вес людей в одной команде и суммарный вес людей в другой команде. Если эти числа отличны, выведите сначала меньшее.

*Пример входных данных:*

3

100

90

200

*Соответствующие выходные данные:* 190 200.

23. **Эдемский сад.** Клеточные автоматы – это математические идеализации физических систем, в которых время и пространство дискретны и физические величины рассматриваются на конечном множестве дискретных значений. Клеточный автомат состоит из решетки (или массива) переменных, принимающих дискретные значения. Состояние такого автомата полностью определяется значением всех переменных решетки. Клеточные автоматы развиваются дискретными временными шагами, причем на значение каждой позиции (ячейки) влияют значения переменных в соседних клетках в предыдущий квант времени. Для каждого автомата определен набор правил, задающий его развитие.

Для большинства клеточных автоматов существуют недостижимые конфигурации (состояния): не существует такого состояния, из которого к ним можно прийти, пользуясь заданными правилами развития. Такие состояния принято называть Эдемскими садами, так как автомат может их принимать только в начальный момент времени. В качестве примера рассмотрим тривиальный набор правил, переводящий все ячейки в 0. Для такого автомата любое состояние с ненулевыми ячейками – это Эдемский сад.

В общем случае нахождение прообраза заданного состояния (или доказательство того, что прообраза не существует) является очень сложной задачей, требующей большого количества вычислений. В целях упрощения задачи мы ограничимся рассмотрением конечных одномерных бинарных клеточных автоматов. Другими словами, число ячеек конечно, ячейки выстроены в одну линию, и они могут находиться в двух состояниях: «0» и «1». Чтобы упростить задачу еще больше, будем считать, что состояние каждой ячейки зависит только от предыдущих состояний самой ячейки и ее непосредственных (правого и левого) соседей.

Ячейки расположены по кругу, то есть последняя ячейка является соседом первой.

Условие задачи. Для заданного кругового бинарного клеточного автомата вы должны определить, является данное состояние Эдемским садом или достижимым состоянием. Клеточный автомат задается через правила развития. Например, таблица 4 задает правила развития автомата: Cell = XOR(Left, Right).

Таблица 4

Сумма чисел в правом столбце равна 90. Это идентификатор автомата.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Left [i-1] | Cell [i] | Right [i+1] | Новое состояние |  |
| 0 | 0 | 0 | 0 | 0\*20 |
| 0 | 0 | 1 | 1 | 1\*21 |
| 0 | 1 | 0 | 0 | 0\*22 |
| 0 | 1 | 1 | 1 | 1\*23 |
| 1 | 0 | 0 | 1 | 1\*24 |
| 1 | 0 | 1 | 0 | 0\*25 |
| 1 | 1 | 0 | 1 | 1\*26 |
| 1 | 1 | 1 | 0 | 0\*27 |

С учетом всех наложенных ограничений существует только 256 различных автоматов. Мы можем получить идентификатор каждого автомата, взяв вектор *новых состояний* и рассмотрев его как двоичное число, как показано в таблице. Идентификатор автомата, взятого в качестве примера, равен 90, тогда как идентификатор *тождественного* автоата (в котором каждое состояние переходит само в себя) равен 204.

*Входные данные.* Одной строкой задается клеточный автомат и состояние. Первым на строке идет идентификатор автомата, с которым вы будете работать. Вторым на строке идет положительное целое число N (4≤N≤32), задающее число ячеек. И наконец, третьим на строке идет состояние, которое представляется строкой ровно из N нулей и единиц. Ваша программа должна считывать входные данные, пока она не достигнет символа конца файла.

*Выходные данные.* Если состояние автомата задает Эдемский сад, выведите строку «САД ЭДЕМА». Если же нет (заданное состояние достижимо), то выведите строку «ДОСТИЖИМО».

*Примеры:*

- 0 4 1111, результат – САД ЭДЕМА;

- 204 5 10101, результат – ДОСТИЖИМО;

- 255 6 000000, результат – САД ЭДЕМА;

- 154 16 1000000000000000, результат – САД ЭДЕМА.

24. **Color hash.** Эта головоломка состоит из двух колес. Оба колеса могут вращаться по и против часовой стрелки. Они содержат 21 цветной фрагмент, 10 из которых – это скругленные треугольники и 11 – это разделители. Слева на рисунке 10 показано конечное состояние головоломки. Обратите внимание, что, для того чтобы перевести колесо в следующее состояние, вы должны вращать колесо, пока не продвинетесь на один треугольник и один разделитель.



Рисунок 10. Конечное состояние головоломки (слева) и головоломка после того, как левое колесо повернуто по часовой стрелке на один шаг (справа).

Ваша задача состоит в том, чтобы написать программу, которая считывает начальное состояние головоломки и определяет кратчайшую последовательность ходов, ведущую к конечному состоянию. Для обозначения частей мы будем использовать следующие значения:

0 – серый разделитель;

1 – желтый треугольник;

2 – желтый разделитель;

3 – голубой треугольник;

4 – голубой разделитель;

5 – фиолетовый треугольник;

6 – фиолетовый разделитель;

7 – зеленый треугольник;

8 – зеленый разделитель;

9 – красный треугольник;

10 – красный разделитель.

Состояние головоломки описывается 24 целыми числами; первые 12 задают состояние левого колеса; следующие 12 задают состояние правого колеса. Первое число задает нижний правый разделитель левого колеса, а следующие 11 чисел задают части левого колеса по часовой стрелке. 13-е число задает нижний левый разделитель правого колеса, а следующие 11 чисел задают части правого колеса против часовой стрелки.

Тогда конечное состояние описывается так:

0 3 4 3 0 5 6 5 0 1 2 1 0 7 8 7 0 9 10 9 0 1 2 1

Если мы повернем левое колесо на одну позицию по часовой стрелке (как показано справа на рисунке), то состояние головоломки будет описываться так:

2 1 0 3 4 3 0 5 6 5 0 1 0 7 8 7 0 9 10 9 0 5 0 1

*Входные данные.* Строка, содержащая 24 целых числа, разделенных одним пробелом, и задающая начальное состояние головоломки, как описано выше.

*Выходные данные.* Программа должна вывести одну строку, содержащую одно число, определяющее решение. Каждое изменение головоломки задается одной цифрой от 1 до 4:

- 1 – поворот левого колеса по часовой стрелке;

- 2 – поворот правого колеса по часовой стрелке;

- 3 – поворот левого колеса против часовой стрелки;

- 4 – поворот правого колеса против часовой стрелки.

Между цифрами не должно быть пробелов. Так как может существовать несколько решений, вы должны вывести такое решение, которое задается минимальным числом. Для решения никогда не потребуется более 16 поворотов.

Если вы не нашли решения, вы должны вывести «НЕ БЫЛО НАЙДЕНО РЕШЕНИЯ ЗА 16 ШАГОВ». Если вам задана конечная позиция, вы должны вывести «ЗАДАЧА УЖЕ РЕШЕНА».

Примеры:

- 0 3 4 3 0 5 6 5 0 1 2 1 0 7 8 7 0 9 10 9 0 1 2 1, результат - ЗАДАЧА УЖЕ РЕШЕНА;

- 0 3 4 5 0 3 6 5 0 1 2 1 0 7 8 7 0 9 10 9 0 1 2 1, результат - 1434332334332323;

- 0 9 4 3 0 5 6 5 0 1 2 1 0 7 8 7 0 9 10 3 0 1 2 1, результат - НЕ БЫЛО НАЙДЕНО РЕШЕНИЯ ЗА 16 ШАГОВ.

25. **Большой квадрат.** У Томи есть много бумажных квадратиков. Длина их стороны (размер) изменяется от 1 до N-1, и у него есть неограниченное число квадратов любого размера. Но ему очень хочется получить большой квадрат – квадрат размера N.

Он может получить такой квадрат, построив его из уже имеющихся квадратов. Например, квадрат размера 7 может быть построен из 9 меньших квадратов, как показано на рисунке 11.

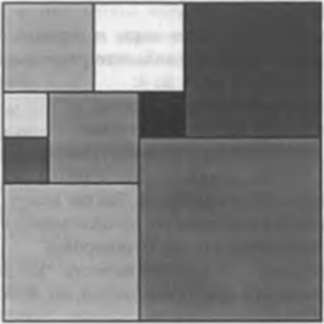


Рисунок 11

Внутри квадрата не должно быть пустого места, меньшие квадраты не должны выходить за пределы большего и не должны перекрываться. Кроме того, Томи хочет истратить минимально возможное число квадратов. Поможете?

*Входные данные:* целое число N, причем 2≤N≤50.

*Выходные данные:* строка, содержащая одно число K, задающее минимальное количество квадратов, из которых можно построить квадрат заданного размера; далее должны идти K строк, каждая из которых должна содержать 3 целых числа x, y и m, задающие координаты левого верхнего угла (1≤x, y≤N) и длину стороны соответствующего квадрата.

*Примеры входных и соответствующих выходных данных* показаны в таблице 5.

Таблица 5

|  |  |  |  |
| --- | --- | --- | --- |
| Входные данные | 4 | 3 | 7 |
| Соответствующие выходные данные | 4  1 1 2  1 3 2  3 1 2  3 3 2 | 6  1 1 2  1 3 1  2 3 1  3 1 1  3 2 1  3 3 1 | 9  1 1 2  1 3 2  3 1 1  4 1 1  3 2 2  5 1 3  4 4 4  1 5 3  3 4 1 |

**Подсказки:**

1) (К заданию 18.) Как мы можем изменить наше решение задачи n ферзей для решения задачи о слонах? Может ли помочь разбиение задачи на отдельные задачи по размещению белопольных и чернопольных слонов?

2) (К заданию 19.) Как мы можем избежать повторяющихся состояний головоломки, ведущих и к уменьшению эффективности, и к получению чрезмерно длинных последовательностей ходов?

3) (К заданию 20.) Как мы можем представить решение для эффективного поиска? Что лучше, составлять перестановки или распознавать подмножества людей с неперекрытым полем зрения?

4) (К заданию 22.) Возможный размер этой задачи делает ее сложной и, пожалуй, слишком большой даже для качественного отсечения вариантов. Можем ли мы отслеживать все *командные веса*, реализуемые некоторым подмножеством, состоящим из первых i людей, без явного рассмотрения всех 2i подмножеств? Обратите внимание, что число различных командных весов гораздо меньше, чем 2i.

5) (К заданию 25.) Окупится ли попытка разместить большие квадраты первыми?

**3.3. Программирование метода ветвей и границ и алгоритмы динамического программирования**

В лабораторных работах 3 и 4 необходимо решить одну и ту же задачу разными методами: методом ветвей и границ в работе 3 и методом динамического программирования в работе 4.

**Цель:** практическое освоение решения задач методом ветвей и границ (лабораторная работа 3), методом динамического программирования (лабораторная работа 4).

***Задания:***

1. **Задача о назначениях.** Имеется n человек, которых нужно назначить на n работ. Стоимость назначения i-го человека на j-ю работу равна Cij. Задача состоит в отыскании назначения, при котором каждая работа выполняется некоторым человеком и которое минимизирует общую стоимость назначения.

2. **Платы.** Имеются компоненты, которые нужно расположить в n ячейках на плате. Число соединений между парами компонент задается матрицей C, в которой Cij – число связей между i-й и j-й компонентами. Расстояние между парами мест задается матрицей D, в которой Dkl – расстояние между k-й и l-й ячейками. Таким образом, в терминах общей длины использованного провода размещение i-й компоненты в k-й ячейке и j-й компоненты в l-й ячейке стоит CijDkl. В каждой ячейке можно поместить только одну компоненту, и каждая компонента может находиться только в одной ячейке. Найдите размещение компонент в ячейках, минимизирующее общую длину использованного провода.

3. **Расписание работ.** Работы J1, J2, …, Jn должны быть выполнены последовательно одним оператором. Работы требуют τ1, τ1, …, τn единиц времени соответственно, и для каждой работы Jk существует монотонно возрастающая функция стоимости ck(t), определяющая стоимость, связанную с завершением работы Jk ко времени t. Найдите расписание {, , …, }, минимизирующее

4. **Расписание для параллельно работающих операторов.** Имеется n работ и k операторов, параллельно выполняющих эти работы; время, требуемое для выполнения i-й работы, равно ti. Более того, некоторые работы не могут выполняться, пока не окончены другие: существует строго верхняя треугольная матрица P=(pij), такая, что pij=-1, если работа i должна быть выполнена перед работой j, pij=+1, если работа j должна быть выполнена перед работой i, и pij=0, если работы выполняются независимо друг от друга. Найдите расписание работ операторов, минимизирующее время всей работы от начала до конца.

5. **Задача упаковки.** Требуется упаковать предметы в ящик. Ящик вмещает не более W фунтов. Имеется не более mi штук предметов i-го типа, каждый из которых стоит vi и весит wi фунтов. Найдите набор предметов, максимизирующий общую стоимость содержимого в ящике.

6. **Редакционное расстояние.** Даны две строки A и B. Со строкой A разрешено выполнять следующие операции:

- удалить символ;

- вставить символ;

- заменить один символ на другой.

Какое наименьшее количество операций необходимо применить к строке A, чтобы получить строку B?

Примечание. Это количество операций принято называть редакционным расстоянием.

7. **Почтовые отделения.** Вдоль прямой дороги расположены деревни. Дорога представляется целочисленной осью, а расположение каждой деревни задается целым числом – координатой на этой оси. Никакие две деревни не имеют одинаковых координат. Расстояние между двумя деревнями вычисляется как модуль разности их координат.

В некоторых, не обязательно во всех, деревнях будут построены почтовые отделения. Деревня и расположенное в ней почтовое отделение имеют одинаковые координаты. Почтовые отделения необходимо расположить в деревнях таким образом, чтобы общая сумма расстояний от каждой деревни до ближайшего к ней почтового отделения была минимальной.

Требуется по заданным координатам деревень и количеству почтовых отделений найти такое их расположение по деревням, при котором общая сумма расстояний от каждой деревни до её ближайшего почтового отделения будет минимальной.

8. **Скульптор.** Скульптор готовится к выполнению заказа. Для этого ему требуются в некотором количестве мраморные прямоугольные плитки заданных размеров w1\*h1, w2\*h2, …, wn\*hn. Имея большие прямоугольные мраморные плиты, скульптор хочет разрезать их так, чтобы получать плитки заданных размеров. Мраморная плита или плитка может быть разрезана только вертикальным или горизонтальным разрезом на две прямоугольные плитки. Разрез осуществляется от края до края, и это единственный доступный способ разрезания. Вырезанные плитки соединять нельзя. Так как мрамор имеет рисунок, плитки нельзя вращать. Из одной отдельной большой плиты скульптор может вырезать ноль или более плиток каждого из требуемых размеров, и при этом могут получаться «обрезки», т.е. плитки других размеров, которые не будут использоваться. Скульптор хочет знать, как нужно разрезать заданную большую плиту, чтобы суммарная площадь обрезков была минимальна.

Например, пусть ширина исходной плиты равна 21, высота равна 11, а заданные размеры плиток: 10\*4, 6\*2, 7\*5 и 10\*15. Тогда возможная минимальная суммарная площадь обрезков равна 10. На рисунке 12 показан один из способов разрезания исходной плиты.

10\*4

10\*4

6\*2

6\*2

6\*2

7\*5

7\*5

7\*5

Рисунок 12

9. **Палиндром.** Палиндром – строка, которая одинаково читается слева направо и справа налево. Требуется по заданной строке определить минимальное количество символов, которые необходимо вставить в строку для преобразования ее в палиндром.

*Вход*: строка длиной n (3<=n<=5000), состоящая из прописных и строчных букв латинского алфавита и цифр от 0 до 9. Прописные и строчные буквы различаются.

*Выход*: целое число – минимальное количество вставляемых символов.

10. **Длинный палиндром**. (Cм. задачу 9). Требуется найти самый длинный палиндром P, получающийся из заданной строки S удалением некоторого (какого потребуется; возможно, нулевого) количества символов. Длина S не превышает 1000.

11. **Последовательность.** Дана последовательность целых положительных чисел a1, a2, …, an. За одну операцию разрешается удалить любое (кроме крайних) число. Стоимость операции равна произведению этого числа на сумму соседних чисел. Требуется удалить все числа, кроме крайних, так, чтобы суммарная стоимость была минимальной. Ограничения n<=100, ai <=100.

12. **Отбор по росту.** Дана последовательность целых положительных чисел a1, a2, …, an. За одну операцию разрешается удалить из последовательности число (кроме крайних) только в двух случаях: 1) оба соседа строго меньше этого числа или 2) оба соседа строго больше этого числа. Требуется выработать план удаления чисел, такой, чтобы итоговая последовательность стала неубывающей, при этом должно быть удалено минимальное количество чисел. Необходимо указать количество удаляемых чисел и порядок их удаления. Если получить неубывающую последовательность невозможно, то вывести сообщение об этом.

13. **Покрытие прямоугольниками.** Клетчатое прямоугольное поле размером n\*m, некоторые клетки которого вырезаны, требуется покрыть минимальным количеством прямоугольников. Прямоугольники не могут пересекаться, выходить за пределы (границы) поля и должны быть параллельны осям координат. Ограничения: 1<=n, m <=13. На рисунке 13 дан пример покрытия поля размера 5\*7 прямоугольниками. Вырезанные клетки показаны черным цветом.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

Рисунок 13

14. **Минимизация значения многочлена из строки.** Строка S состоит из десятичных цифр. Её можно разбить на последовательные непустые подстроки S0, S1, S2, …, Sq (при их конкатенации (соединении) получается стока S). Каждая подстрока Si определяет число ai. Из чисел ai образуется многочлен a0 + a1\*x + a2 \*x2 +…+ aq\*xq . Требуется разбить строку так, чтобы при заданном значении x значение многочлена было бы минимальным.

15. **Покрытие отрезками.** Задано некоторое множество отрезков [xi, yi] в виде натуральных координат своих концов (1<= xi < yi <=1000, i=1, 2, …, n, n<=100), а также отрезок [u, v] с аналогичными ограничениями на его координаты. Требуется найти покрытие отрезка [u, v] подмножеством отрезков с минимальной суммарной длиной.

16. **Покрытие непересекающимися отрезками.** Задано некоторое множество отрезков [xi, yi] в виде натуральных координат своих концов (1<= xi < yi <=1000, i=1, 2, …, n, n<=100), а также отрезок [u, v] с аналогичными ограничениями на его координаты. Каждый отрезок [xi, yi] имеет стоимость wi. Отрезки могут пересекаться. Требуется найти покрытие отрезка [u, v] подмножеством непересекающихся отрезков с максимальной суммарной стоимостью.

17. **Возрастающая подпоследовательность с наибольшей суммой.** Дана последовательность целых положительных чисел a1, a2, …, an. Найти такую неубывающую подпоследовательность этой последовательности, что сумма составляющих подпоследовательности максимальна.

Пример: входная последовательность – 3, 2, 1, 2, 4, выходная подпоследовательность – 2, 2, 4.

**Содержание**

Введение 3

РАЗДЕЛ 1. Требования 4

РАЗДЕЛ 2. ЗАДАНИЯ НА КУРСОВУЮ РАБОТУ 7

РАЗДЕЛ 3. ЛАБОРАТОРНЫЕ РАБОТЫ 8

3.1. Сортировки 8

3.2. Программирование алгоритма поиска с возвращением 11

3.3. Программирование метода ветвей и границ и алгоритмы динамического программирования 25

Учебно-методическое пособие

Фирсов Михаил Александрович

**Методические указания**

**к лабораторным работам, практическим занятиям и курсовой работе**

Издание публикуется в авторской редакции

СПбГЭТУ «ЛЭТИ»

197376, Санкт-Петербург, ул. Проф. Попова, 5