Лабораторная работа № 4

по дисциплине “Алгоритмы и структуры данных”

Тема: Динамическое программирование.

Последовательности.

1. По данному натуральному *n* определите количество последовательностей длины *n* из 0 и 1, не содержащих двух единиц подряд. Гарантируется, что ответ не превосходит 231-1.
2. По данному натуральному *n* определите количество последовательностей длины *n* из 0, 1 и 2, не содержащих двух единиц подряд. Гарантируется, что ответ не превосходит 231-1.
3. По данному натуральному *n* определите количество последовательностей длины *n* из 0, 1 и 2, не содержащих двух единиц подряд и двух двоек подряд. Гарантируется, что ответ не превосходит 231-1.
4. По данным натуральным *n* и *k* определите количество последовательностей длины *n* из 0 и 1, не содержащих *k* единиц подряд. *n* ≤ 106, 1 ≤ *k* ≤ *n*+1. Гарантируется, что ответ не превосходит 231-1.
5. По данным натуральным *n* и *k* определите количество последовательностей длины *n* из 0 и 1, не содержащих *k* одинаковых символов подряд. *n* ≤ 106, 2 ≤ *k* ≤ *n*+1. Гарантируется, что ответ не превосходит 231-1.

Разбиение на слагаемые.

1. Для данных натуральных чисел *n* и *k* определите количество способов представить число *n* в виде суммы натуральных слагаемых, не превосходящих *k*, если способы, отличающиеся только порядком слагаемых считать одинаковыми.

Программа получает на вход два натуральных числа *n* и *k*, не превосходящих 120.

1. Для данных натуральных чисел *n* и *k* определите количество способов представить число *n* в виде суммы *k* натуральных слагаемых, если способы, отличающиеся только порядком слагаемых считать одинаковыми.

Программа получает на вход два натуральных числа *n* и *k*, не превосходящих 150.

1. Плавные числа.

Назовем число плавным, если его две соседние цифры различаются не более, чем на 1. По данному натуральному *n* определите количество плавных натуральных чисел, имеющих длину *n*. Гарантируется, что ответ не превосходит 231-1.

1. Пилообразная последовательность

Назовем последовательность *пилообразной*, если каждый ее элемент строго больше, (либо строго меньше) своих соседей. По данным числам *n* и *k* определите число пилообразных последовательностей длины *n*, составленных из чисел 1 .. *k*.

Программа получает на вход два натуральных числа *n* и *k*,  1 ≤ *n* ≤ 1000, 1 ≤ *k* ≤ 1000.

1. Лесенка.

Лесенкой называется набор кубиков, в котором каждый горизонтальный слой содержит меньше кубиков, чем слой под ним.

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

Вводится одно число *N (1 ≤ N ≤ 150)*. Подсчитать количество различных лесенок, которые могут быть построены из N кубиков

1. Бросание кубика.

Кубик, грани которого помечены цифрами от 1 до 6, бросают *N* раз. Определите вероятность того, что произведение выпавших чисел будет в точности равна *Q*. Программа получает на вход два целых числа: *N* и *Q* (1 ≤ *N* ≤ 50, 0 ≤ *Q* ≤ 100). Программа должна вывести единственное действительное число: искомую вероятность с точностью не менее чем 10-6.

1. Радиоактивность.

При переработке радиоактивных материалов образуются отходы двух видов — особо опасные (тип *A*) и неопасные (тип *B*). Для их хранения используются одинаковые контейнеры. После помещения отходов в контейнеры последние укладываются вертикальной стопкой. Стопка считается взрывоопасной, если в ней подряд идет более одного контейнера типа *A*. Для заданного количества контейнеров *N* определить число безопасных стопок.

Входные данные: Количество контейнеров *N  (1 ≤ N ≤ 20).*

Выходные данные: Количество безопасных вариантов формирования стопки.

1. Платная лестница.

Мальчик подошел к платной лестнице. Чтобы наступить на любую ступеньку, нужно заплатить указанную на ней сумму. Мальчик умеет перешагивать на следующую ступеньку, либо перепрыгивать через ступеньку. Требуется узнать, какая наименьшая сумма понадобится мальчику, чтобы добраться до верхней ступеньки.

Входные данные:

* Натуральное число *N < 100* — количество ступенек.
* *N* натуральных чисел, не превосходящих 100 — стоимость каждой ступеньки (снизу вверх).

Выходные данные: Наименьшая возможная стоимость прохода по лесенке.

1. Гвоздики.

В дощечке в один ряд вбиты гвоздики. Любые два гвоздика можно соединить ниточкой. Требуется соединить некоторые пары гвоздиков ниточками так, чтобы к каждому гвоздику была привязана хотя бы одна ниточка, а суммарная длина всех ниточек была минимальна.

Входные данные:

* Число *N* — количество гвоздиков *(2 ≤ N ≤ 100)*.
* *N* чисел — координаты всех гвоздиков (неотрицательные целые числа, не превосходящие 10000).

Выходные данные: Минимальная суммарная длина всех ниточек.

1. Покупка билетов.

За билетами на премьеру нового мюзикла выстроилась очередь из *N* человек, каждый из которых хочет купить 1 билет. На всю очередь работала только одна касса, поэтому продажа билетов шла очень медленно. Самые сообразительные быстро заметили, что, как правило, несколько билетов в одни руки кассир продаёт быстрее, чем когда эти билеты продаются по одному. Поэтому они предложили нескольким подряд стоящим людям отдавать деньги первому из них, чтобы он купил билеты на всех.

Однако для борьбы со спекулянтами кассир продавала не более 3-х билетов в одни руки, поэтому, договориться, таким образом, между собой могли лишь 2 или 3 подряд стоящих человека.

Известно, что на продажу *i-*му человеку из очереди одного билета кассир тратит *Ai* секунд, на продажу двух билетов – *Bi* секунд, трех билетов – *Ci* секунд. Напишите программу, которая подсчитывает минимальное время, за которое могли быть обслужены все покупатели.

Входные данные:

* Длина очереди *N*.
* Время обслуживания: *А* - время продажи одного билета, *В* – время продажи двух билетов, *C* – время продажи трех билетов.

Выходные данные: Минимальное время, за которое могли быть обслужены все покупатели.

1. Имеется калькулятор, который выполняет три операции:
2. Прибавить к числу *X* единицу.
3. Умножить число *X* на 2.
4. Умножить число *X* на 3.

Определите, какое наименьшее число операций необходимо для того, чтобы получить из числа 1 заданное число N. Также вывести последовательность операций, например 1,2,3.

Входные данные: Число *N*, не превосходящее 106.

Выходные данные: Наименьшее количество искомых операций.

Подпоследовательность

1. Даны две последовательности, требуется найти длину их наибольшей общей подпоследовательности. Найти и вывести найденную подпоследовательность.

Входные данные:

* Число N – длина первой последовательности (1 ≤ N ≤ 1000).
* Члены первой последовательности (через пробел) – целые числа, не превосходящие 10000 по модулю.
* Число M – длина второй последовательности (1 ≤ M ≤ 1000).
* Члены второй последовательности (через пробел) – целые числа, не превосходящие 10000 по модулю.

Выходные данные: Длина  наибольшей общей подпоследовательности двух данных последовательностей или 0, если такой подпоследовательности нет.

1. Дана текстовая строка. С ней можно выполнять следующие операции:

1. Заменить один символ строки на другой символ.

2. Удалить один произвольный символ.

3. Вставить произвольный символ в произвольное место строки.

Например, при помощи первой операции из строки "СОК" можно получить строку "СУК", при помощи второй операции - строку "ОК", при помощи третьей операции - строку "СТОК.

Минимальное количество таких операций, при помощи которых можно из одной строки получить другую, называется *стоимостью редактирования* или *расстоянием Левенштейна*.

Определите стоимость редактирования для двух данных строк.

Входные данные: Программа получает на вход две строки, длина каждой из которых не превосходит 1000 символов, строки состоят только из заглавных латинских букв.

Выходные данные: Стоимость редактирования для данных строк.

1. Шаблон с ? и \*

Шаблоном называется строка, состоящая из английских букв (a, ..., z, A, ..., Z) и символов ? и \*. Каждый из символов ? разрешается заменить на одну произвольную букву, а каждый из символов \* – на произвольную (возможно пустую) последовательность букв. Про любую строку из букв, которую можно получить из шаблона такими заменами, будем говорить, что она удовлетворяет этому шаблону.

Имеются два шаблона. Требуется найти строку минимальной длины, которая удовлетворяет обоим шаблонам, либо выдать сообщение, что такой строки не существует.

Входные данные**:** Два шаблона, длина каждого шаблона не превосходит 80 символов.   
Выходные данные: Выведите строку минимальной длины, удовлетворяющую обоим шаблонам, либо сообщение «No solution!»

1. Дана последовательность, требуется найти длину её наибольшей возрастающей подпоследовательности. Вывести найденную подпоследовательность.

Входные данные:

* Число *N* - длина последовательности (1 ≤ *N* ≤ 1000).
* Последовательность (разделитель -  пробел). Элементы последовательности - целые числа, не превосходящие 10000 по модулю.

Выходные данные: Длина наибольшей возрастающей подпоследовательности.

Правильная скобочная последовательность (ПСП).

1. По данному натуральному *n* определите количество правильных скобочных последовательностей, составленных из *n* открывающихся и *n* закрывающихся круглых скобок.

Входные данные: Натуральное число *n*, не превосходящее 1000.

Выходные данные: Количество ПСП, составленных из *2n* круглых скобок.

1. По данному натуральному *n* определите количество правильных скобочных последовательностей длины 2*n*, составленных из круглых и квадратных скобок так, что внутри любой пары круглых скобок нет квадратных скобок.

Входные данные: Натуральное число *n*, не превосходящее 1000.

Выходные данные: Количество ПСП длины *2n*, составленных из круглых и квадратных скобок, причем внутри круглы скобок нет квадратных.

1. По данным числам *n* и *k* определите количество правильных скобочных последовательностей длины 2*n*, составленных из круглых скобок, максимальная вложенность скобок в которой составляет в точности *k*.

Входные данные:

* Натуральное число *n* , так что *2n* – длина искомой последовательности
* Натуральное число *k* - максимальная вложенность скобок (1 ≤ *k* ≤ *n* ≤ 50).

Выходные данные: Количество ПСП длины *2n*, составленных из круглых скобок.

1. Восстановление скобок.

Задан шаблон, состоящий из круглых скобок и знаков вопроса. Требуется определить, сколькими способами можно заменить знаки вопроса круглыми скобками так, чтобы получилось правильное скобочное выражение.

Входные данные: Шаблон, длина которого не превосходит 80 символов.

Выходные данные: Количество способов получения ПСП.

1. Уравнение с пропущенными скобками.

Задано уравнение вида *A*+*B*=*C*, где *A*, *B* и *C* – неотрицательные целые числа, в десятичной записи которых некоторые цифры заменены знаками вопроса (?). Примером такого уравнения является ?2+34=4?. Требуется так подставить вместо знаков вопроса цифры, чтобы это равенство стало верным, либо определить, что это невозможно.

Входные данные: Уравнение, длина которого не превосходит 15-и символов.

Выходные данные: Количество способов получения верного равенства.

1. Ход конем.

Шахматная ассоциация решила оснастить всех своих сотрудников такими телефонными номерами, которые бы набирались на кнопочном телефоне ходом коня. Например, ходом коня набирается телефон 340-49-27. При этом телефонный номер не может начинаться ни с цифры 0, ни с цифры 8.

|  |  |  |
| --- | --- | --- |
| 7 | 8 | 9 |
| 4 | 5 | 6 |
| 1 | 2 | 3 |
|  | 0 |  |

Напишите программу, определяющую количество телефонных номеров длины *N*, набираемых ходом коня.

Входные данные: Длина телефонного номера *N*.

Выходные данные: Количество телефонных номеров, длины *N*.

1. Дырокол.

Квадратный клетчатый лист бумаги *2N × 2N* клеток начинают складывать следующим образом. Сначала нижняя половина листа накладывается на верхнюю, затем правая половина листа накладывается на левую. Эту операцию повторяют *N-3* раза, в результате чего получается сложенный лист *8 × 8* клеток. Какие-то из клеток этого сложенного листа удаляются при помощи дырокола.

После развертывания исходный лист распадется на некоторое количество связных частей, т.е. таких множеств клеток, что из любой клетки одного множества можно пройти до любой другой, переходя каждый раз на соседнюю по вертикали или горизонтали клетку. Напишите программу, вычисляющую число частей, на которые распадется лист.   
  
Входные данные:

* Целое число *N (4 ≤ N ≤ 500)*.
* Матрица *8 × 8* из нулей и единиц, разделенных пробелом. Единицами отмечены клетки, выкалываемые дыроколом из сложенного листа *8×8*.

Выходные данные: Требуется вывести  искомое число частей.

Маршруты на плоскости.

1. Дана прямоугольная доска размером *n×m* (*n* строк и *m* столбцов). В левом верхнем углу этой доски находится шахматный король, которого необходимо переместить в правый нижний угол. В клетке с координатами (*a,b*) располагается король противника, который ходит случайным образом на одну клетку вверх, вниз, вправо, влево (если это возможно). Противники не могут приближаться друг к другу ближе, чем на одну клетку. Вывести маршруты шахматных королей, причем первый движется из верхней левой клетки в правую нижнюю, а второй случайным образом.

Входные данные:

* Размер доски – натуральные числа *n, m*, меньшие 20.
* Координаты клетки (*a,b*) начальной позиции противника.

Выходные данные: Маршруты шахматных королей.

1. Дана прямоугольная доска размером *n×m* (*n* строк и *m* столбцов). В левом верхнем углу этой доски находится шахматный король, которого необходимо переместить в правый нижний угол. Пусть каждой клетке (*a*, *b*) доски приписано некоторое число *P*(*a*, *b*) -- стоимость данной клетки. Проходя через клетку, мы получаем сумму, равную ее стоимости. Требуется определить максимально возможную сумму, которую можно собрать по всему маршруту, если разрешается передвигаться только вниз или вправо.

Входные данные:

* Размер доски – натуральные числа *n, m*, меньшие 20.
* Матрица стоимости каждой клетки *P* размером n×m.

Выходные данные: Максимально возможную сумму, которую можно собрать двигаясь только вниз или вправо, начиная с верхней левой клетки в правую нижнюю.

1. Прямоугольники.

Дан прямоугольник *N×M*, состоящий из квадратиков *1×1*. Некоторые квадратики вырезали. Надо найти наименьшее количество прямоугольников, на которое можно разрезать оставшуюся фигуру.

Входные данные:

* Числа *N* и *M*, (*1 ≤  N, M ≤ 10*).
* Матрицу из 0 и 1. Элемент матрицы [ i, j] равен 1, если клетку на i-й строке и в j-м столбце вырезали из доски, 0 – если оставили.

Выходные данные: Минимальное количество прямоугольников, на которое можно разрезать полученную фигуру.

Оптимизация.

1. В некотором государстве в обращении находятся банкноты определенных номиналов. Национальный банк хочет, чтобы банкомат выдавал любую запрошенную сумму при помощи минимального числа банкнот, считая, что запас банкнот каждого номинала неограничен. Помогите Национальному банку решить эту задачу.

Входные данные:

* Количество номиналов в государстве.
* Запрашиваемая сумма.

Выходные данные: Набрать запрашиваемую сумму при помощи минимального числа банкнот.

1. Дано *N* золотых слитков массой *m*1, …, *mN*. Ими наполняют рюкзак, который выдерживает вес не более *M*. Какую наибольшую массу золота можно унести в таком рюкзаке?

Входные данные:

* Количество золотых слитков - натуральное число *N*, не превышающее 100.
* Максимальный вес рюкзака - натуральное число *M*, не превышающее 10000.
* Масса золотых слитков - числа *mi*, не превышающих 100.

Выходные данные: Наибольшая возможная масса золота, которую можно унести в данном рюкзаке.

1. Дано N предметов массой m1, …, mN и стоимостью c1, …, cN соответственно. Ими наполняют рюкзак, который выдерживает вес не более M. Определите набор предметов, который можно унести в рюкзаке, имеющий наибольшую стоимость.

Входные данные:

* Количество предметов - натуральное число *N*, не превышающее 100.
* Максимальный вес рюкзака - натуральное число *M*, не превышающее 10000.
* Масса предметов - *N* натуральных чисел *mi* , не превышающая 100.
* Стоимость предметов – *N* натуральных чисел *сi* , не превышающая 100.

Выходные данные: Номера предметов (числа от *1* до *N*), которые войдут в рюкзак наибольшей стоимости.

1. Дано *N* предметов массой *m1, …, mN*. Ими наполняют рюкзак, который выдерживает вес не более *M*. Как набрать вес в точности *M*, используя как можно меньше предметов?

Входные данные:

* + Натуральное число *N*, не превышающее 100
  + Натуральное число *M*, не превышающее 10000.
  + *N* натуральных чисел *mi*, не превышающих 100.

Выходные данные: Наименьшее необходимое число предметов или 0, если набрать данный вес невозможно.

Заказываем пиццу.

1. Вы решили заказать пиццу с доставкой на дом. В пиццерии продаются пиццы стоимостью *c1, …, cN* , а вы желаете приобрести *M* пицц. Каждый товар в заказ попадает любое количество раз. Известно, что для клиентов, сделавших заказ на сумму более C рублей, доставка является бесплатной, при заказе на C рублей и меньше доставка стоит B рублей. Определить номера пицц, которые войдут в ваш заказ, так, чтобы стоимость была минимальной.

Входные данные:

* + Количество пицц, согласно ассортименту - натуральное число *N*, не превышающее 20.
  + Количество приобретаемых пицц - натуральное число *M*, не превышающее 10000.
  + Стоимость товара – натуральные числа сi, не превышающие 500.
  + Сумма выше, которой доставка бесплатная - натуральное число *С (1000 ≤  С  ≤ 10000).*
  + Стоимость доставки – натуральное число *B (100 ≤  B ≤ 500).*

Выходные данные:

* Суммарное количество денег, которое придется потратить.
* Номера товаров в возрастающем порядке.
* Способ доставка: платный или бесплатный.

1. Вы решили заказать пиццу с доставкой на дом. Известно, что для клиентов, сделавших заказ на сумму более C рублей, доставка является бесплатной, при заказе на C рублей и меньше доставка стоит B рублей. Вы уже выбрали товар, стоимостью A рублей. В наличии имеются еще N товаров стоимостью d1, ..., dN рублей, каждый в единственном экземпляре. Их также можно включить в заказ. Как потратить меньше всего денег и получить на дом уже выбранный товар в A рублей?

Входные данные:

* Числа A, B, C, N. Все числа целые, 1 ≤ A ≤ 1000, 1 ≤ B ≤ 1000, 1 ≤ C ≤ 1000, 0 ≤ N ≤ 1000.
* N чисел d1, ..., dN , 1 ≤ di ≤ 1 000 000.

Выходные данные:

* + Cуммарное количество денег, которое придется потратить.
  + Если при этом вы планируете сделать дополнительный заказ c расчетом на бесплатную доставку, то далее выведите количество этих товаров и их номера в возрастающем порядке.
  + Если же Вы будете оплачивать доставку сами, то далее выведите одно число –1 (минус один).

Гирьки.

1. Дан набор гирек массой m1, …, mN. Можно ли их разложить на две чаши весов, чтобы они оказались в равновесии?

**Входные данные:**

* Натуральное число N, не превышающее 100.
* N натуральных чисел mi, не превышающих 100.

**Выходные данные:** Программа должна вывести YES, если гирьки можно разложить на две кучки равной массы или NO в противном случае.

1. Дан набор гирек массой *m*1, …, *mN*. Разделите этот набор на две кучки равной массы, содержащие равное число гирек.

Входные данные:

* Натуральное число *N*, не превышающее 100.
* *N* натуральных чисел *mi*, не превышающих 100.

Выходные данные:

* Номера гирек (числа от *1* до *N*), входящие в первую кучку,
* Номера гирек во второй кучке.
* Если задача не имеет решения, выведите строку No solution.

Размен денег.

1. Покупатель хочет приобрести товар стоимостью S рублей. У него есть N банкнот номиналом P1, P2, ..., PN рублей. У продавца есть M банкнот номиналом Q1, Q2, ..., QM. рублей. Определите, смогут ли они рассчитаться.

Входные данные:

* Программа получает на вход сумму S.
* Количество банкнот каждого номинала у покупателя – натуральное число *N ( N < 100 ).*
* Номиналы покупателя P1, P2, ..., PN.
* Количество банкнот каждого номинала у продавца – натуральное число *M ( M < 100 ).*
* Номиналы продавца Q1, Q2, ..., QM.

Выходные данные:

* Если продавец сможет рассчитаться с покупателем, выведите номиналы банкнот, которые покупатель отдает продавцу и которые он получает в качестве сдачи.
* Выводите число со знаком “+”, если банкноту соответствующего номинала покупатель отдает продавцу и со знаком “-”, если покупатель получает эту банкноту на сдачу.
* Номиналы банкнот разделяйте пробелом.
* Если они не могут рассчитаться, выведите строку Impossible.

Палиндром.

1. Палиндромом называется строка, которая одинаково читается как слева направо, так и справа налево. Подпалиндромом данной строки называется последовательность символов из данной строки (в том же порядке, но не обязательно идущих подряд), являющаяся палиндромом. Например, HELOLEH является подпалиндромом строки HTEOLFEOLEH. Напишите программу, находящую в данной строке подпалиндром максимальной длины.

Входные данные: Cтрока длиной не более 100 символов, состоящая из заглавных букв латинского алфавита.

Выходные данные: Длина максимального подпалиндрома.

1. Вам нужно распилить деревянный брус на несколько кусков в заданных местах. Распилочная компания берет  *k* рублей за распил одного бруска длиной k метров на две части.

Понятно, что различные способы распила приводят к различной суммарной стоимости заказа. Например, рассмотрим брус длиной 10 метров, который нужно распилить на расстоянии 2, 4 и 7 м, считая от одного конца. Это можно сделать несколькими способами. Можно распилить сначала на отметке 2 м, потом 4 и, наконец, 7 м. Это приведет к стоимости 10+8+6=24, потому что сначала длина бруса, который пилили, была 10 м, затем она стала 8 м, и, наконец, 6 м. А можно распилить иначе: сначала на отметке 4 м, затем 2, затем 7м. Это приведет к стоимости 10+4+6=20, что лучше.

Определите минимальную стоимость распила бруса на заданные части.

Входные данные:

* Целое число *L (2 ≤ L ≤ 106)* - длина бруса.
* Целое число *N (1 ≤ N ≤ 100)* - количество распилов.
* N целых чисел *Сi (0 < Ci < L)* в строго возрастающем порядке - места, в которых нужно сделать распилы.

Выходные данные: Натуральное число - минимальную стоимость распила.

Игры.

1. На столе лежит кучка из N спичек. Двое играют в такую игру. За один ход разрешается взять из кучки одну, две или три спички, так чтобы оставшееся количество спичек не было простым числом (Например, можно оставить в кучке 1 или 4 спички, но нельзя оставить 2 или 3). Выигрывает тот, кто забирает последнюю спичку. Требуется определить, кто из игроков имеет выигрышную стратегию.

**Входные данные: Ч**исло N (1 ≤ N ≤ 10000).

**Выходные данные:**

* Число 1, если выигрышную стратегию имеет начинающий игрок.
* Число 2, если выигрышную стратегию имеет второй игрок.

1. На столе лежат *N* камней. За ход игрок может взять

* 1 или 2 камня, если *N* делится на 3;
* 1 или 3, если *N* при делении на 3 дает остаток один;
* 1, 2 или 3, если *N* при делении на 3 дает остаток два.

Каждый ход можно сделать при наличии достаточного количества камней. Проигрывает тот, кто хода сделать не может.

Входные данные: Целое число *N ( 0 < N ≤ 100).*

Выходные данные: Номер игрока, который выиграет при правильной игре.