Алгоритмы, язык С++. Домашнее задание № 1

Задача №1 (на однопроходные алгоритмы)

Последовательно передаются или считываются из файла натуральные числа от 1 до 10. Нужно найти длину самой длинной цепочки взаимно простых чисел.

Пример: 1 2 4 3 9 8 1 2 10 6 7 1. Ответ: 3 (например, {4, 3, 7})

Задача №2 (на O(n))

Даны два массива, первый A содержит, скажем, буквы, а второй P задаёт подстановку (перестановку). Надо расставить элементы первого массива по порядку в соответствии с P за O(n).

Пример: A = [a, b, c, d] P = [4, 1, 3, 2]. Ответ: [d, a, c, b]

Задача №3 (на однопроходные алгоритмы)

За один проход найти максимальное значение суммы интервала a[i] + a[i+1] + ... + a[j] по всем парам (i,j) в последовательности a[1], ..., a[n].

Пример: 1 -5 -2 3 2 -2 8 0 4 -5. Ответ: 12 = 8 + 0 + 4

Задача №4

В памяти дан массив A(i,j) - матрица смежности графа. A(i,j) = 1 <=> i и j соединены ребром. На вход программы поступают k1 и k2, две вершины. Надо за O(n) операций и O(1) памяти сказать, есть ли между ними путь длины ровно в 2 ребра.

Задача №5

Используя O(L) операций и O(1) памяти "нарисовать" отрезок / окружность на плоскости, то есть выдать последовательность точек-пикселей (x,y) на плоскости, аппроксимирующую отрезок / окружность. Координаты пикселя - целые числа. Здесь L - длина объекта, вычисляемая, как примерное количество пикселей.

Задача №6*

Есть множество сотрудников предприятия, всего N, и множество различных работ, которые они могут делать (тоже N).

Можно назначить работнику только одну работу. Требуется найти конфигурацию максимальной загрузки, когда максимум сотрудников заняты. Задача не такая простая, попробуйте решить и скажите, до какой оценки сложности Вы добрались. Эта задача, в частности, упоминается здесь:

http://www.cs.sunysb.edu/~algorith/major_section/1.4.shtml

Задача №7

Найти за линейное время в последовательности целых чисел два непересекающихся интервала значений, таких что сумма по одному чётная, а по другому - нечётная, причем суммарная длина интервалов максимальна.

Задача №8

В круглом заборе (ограничивающем круговую область) есть n дверей. На каждой написано направление «внутрь» или «наружу». Нужно проложить нитку в форме петли (то есть замкнутую) так чтобы нитка не пересекала себя и проходила ворота в соответствии с указанным направлением. Нитка ориентирована. Нужно найти максимальное количество дверей, которое можно обойти. Сколько потребуется вычислительных операций.

Задача №9

Дана карта дорог (как на рисунке), где каждый участок имеет определённое направление. Размер n x n. Мы хотим пройти из левого верхнего угла в правый нижний.

а) для начала нужно, применяя рекурсию, ответить на вопрос: есть ли "монотонный" путь, и найти его; таблица read-only, памяти O(1); монотонный означает идём только вправо или вниз

б) нужно ответить на вопрос существует ли (какой угодно) путь? таблица теперь с возможностью делать пометки, дополнительная память O(1)

б*) то же самое, но таблица read-only и есть O(n) памяти

в) в дополнение к предыдущему заданию найти кратчайший путь

Задача №10

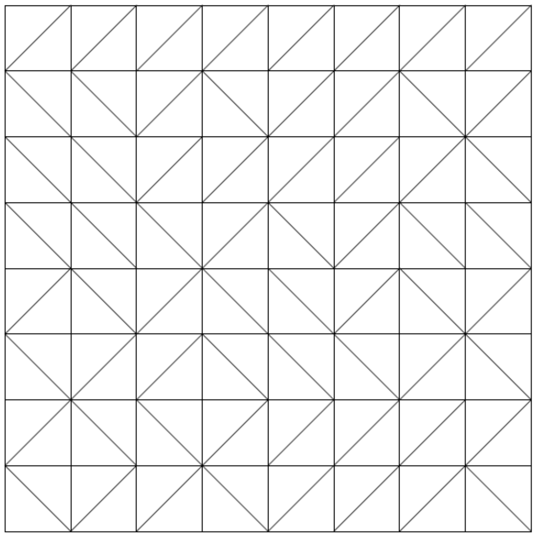
Теперь можно двигаться в любом направлении, но задана стоимость участка.

а) нужно найти оптимальный монотонный путь (когда машина едет либо на юг, либо на восток) б) нужно найти самый оптимальный путь из всех возможных

в) Верно ли, что оптимальный путь всегда будет без самопересечений?

Задача №11

Пусть дан граф, как на рисунке.



а) нужно найти оптимальный путь и оценить число операций.

Задача №12

Пусть дан неориентированный граф с вершинами 1,2,...,n, такой, что i -- i+1 и n -- 1 всегда соединены, но возможны и другие ребра. На рёбрах указаны веса. Будем называть путь локально минимальным, если никакое элементарное перестраивание пути не улучшает общую "стоимость" пути. Элементарными считаются:

a--b => a--c—b, a--c--b => a--b

Сколько нужно операций, чтобы найти локально минимальный путь между двумя вершинами?

Задача №13

Теперь разрешается оценивать не максимальное число операций, а среднее, считая, что алгоритм использует случайность. Пусть дан произвольный неориентированный граф с вершинами 1,2,...,n, и пусть известно, что из любой вершины можно попасть в любую за K шагов. Нужно придумать наиболее оптимальный алгоритм (можно использовать случайность) поиска локально минимального пути между заданными вершинами и оценить среднее время работы.

Задача №14

В городе есть одна длинная улица с перекрёстками 0,1,2,...,n-1, где $n = 2^m$. Между соседними перекрёстками проложена дорога, но на некоторых участках идёт ремонт.

Это случайный набор. Доля ремонтируемых дорог = p. Кроме того, в городе есть быстрый транспорт, который соединяет перекрёстки

$j\ 2^k$ -- $(j+1)\ 2^k$

Например, в городе с 16 перекрёстками дополнительно соединены

0 -- 2 -- 4 -- 6 -- 8 -- 10 -- 12 -- 14

0 -- 4 -- 8 -- 12

0 -- 8

Эти линии работают исправно.

Детектив ищёт в городе подозреваемого, личность которого ему известна, и который находится в кафе на углу некоторого перекрёстка. Процесс поиска состоит в том, что агент передвигается по городу используя тот или иной транспорт. Находясь на перекрёстке он, конечно, видит, в каком направлении может двигаться (но другие перекрёстки не видны). Нужно предложить по возможности самый быстрый алгоритм поиска подозреваемого и оценить среднее время поиска (снова допускается использование случайности), оно же длина пути. Агент стартует в точке 0. Подозреваемый на случайном перекрёстке (равномерное распределение). Сложность в том, что нельзя пройти всё улицу подряд. Кстати, в этом случае мы получили бы очевидно линейный алгоритм, но, используя, быстрый транспорт, вероятно добраться можно было бы быстрее? Скажите, какова могла бы быть средняя длина пути при p = 0 ? И как изменится оценка в общем случае?

Задача №15
Точно так же, как в линейной задаче, даны две квадратные таблицы. Вторая указывает, как расставить элементы первой. Сколько нужно операций? Докажите, что нельзя быстрее.

Задача №16
Дана последовательность весов камней. Идея в том, что мы хотим разложить камни на две кучи, чтобы веса куч отличались не более, чем в 2 раза. Нужно вычислить результирующее отношение весов куч.Как обычно: O(n), O(1)

Задача №17*.
Как добиться отношения не хуже 1.5?