Домашнее задание

1. Как модифицировать алгоритм Флойда-Уоршелла, чтобы он находил не только длины кратчайших путей между всеми парами вершин, но и сами пути?

2. Как используя выходные данные алгоритма Флойда-Уоршелла проверить, что в графе есть цикл отрицательного веса?

3. В Главе 2 [ДПВ] (раздел 2.5) приведён алгоритм Штрассена для умножения матриц сложностью $O(n^{\log_2 7})$.

1. Объясните почему с его помощью нельзя ускорить алгоритмы поиска транзитивного замыкания (поиска вершин, достижимых из каждой вершины) и поиска кратчайших путей, основанных на быстром возведении в степень до $O(n^{\log_2 7} \log n)$?

Указание: булева алгебра $\{\vee, \wedge\}$ *и тропическая алгебра* $\{\min, +\}$ *не являются кольцами.*

2. Постройте алгоритм, который на основе алгоритма Штрассена находит матрицу транзитивного замыкания ($a_{ij} = 1$ тогда и только тогда, когда j достижима из i) оптимальнее, чем за $O(n^3 \log n)$ и оцените его сложность.

4. На вход задачи подаётся число n и последовательность целых чисел $a_1, \ldots, a_n$. Необходимо найти номера i и j ($1 \leqslant i < j \leqslant n$), такие что сумма $\sum\limits_{k=i}^{j} a_k$ максимальна.

1. Постройте линейный алгоритм, решающий задачу.

2. Постройте онлайн-алгоритм (достаточно выполнить только этот пункт).

5. Назовём последовательность $x_1, x_2, \ldots, x_n$ строго унимодальной, если существует такой индекс k, что $x_1 < x_2 < \cdots < x_k$, а $x_k > x_{k+1} > \cdots > x_n$, т. е. до индекса k последовательность строго возрастает, после чего строго убывает. Постройте алгоритм, который, получив на вход конечную последовательность натуральных чисел, находит её самую длинную строго унимодальную подпоследовательность.

6. Постройте алгоритм, который, получив на вход две строки x и y, находит их самую длинную общую подстроку (непрерывную подпоследовательность $x[i]x[i+1]\ldots x[i+m] = y[j]y[j+1]\ldots y[j+m]$).

7. В одном языке программирования операция разрезания строки на две части реализовано через копирование. Разрезать строку $w = uv$ на две части u и v будет стоить длину строки $O(|w|)$, где $|w|$ — длина строки w, вне зависимости от длин строк u и v. Постройте алгоритм, который получив на вход строку $w = u_1 u_2 \ldots u_k$ и номера позиций $i_1 = |u_1|, i_2 = |u|_1 + |u_2|, \ldots, i_k = |u_1| + \ldots + |u_k|$ строит последовательность разрезов строки w, которая приводит к разрезанию w на подстроки $u_1, \ldots, u_k$ и использует для этого минимальное число операций.

Заметьте, что в зависимости от порядка разрезов меняется общее число операций: если строку длины 20 нужно разрезать на строки с номерами позиций 3, 10, то отрезав сначала строку в позиции 3 будет затрачено $20 + 17 = 37$ операций, а отрезав сначала строку в позиции 10, будет потрачено $20 + 10 = 30$ операций.

8 [Шень 8.1.4]. Дан выпуклый n-угольник (заданный координатами своих вершин в порядке обхода). Его разрезают на треугольники диагоналями, для чего необходимо $n - 2$ диагонали (это можно доказать индукцией по n). Стоимостью разрезания назовём сумму длин всех

использованных диагоналей. Постройте полиномиальный алгоритм, который находит минимальную стоимость разрезания. (Перебор не подходит, так как число вариантов не ограничено многочленом.)

9*. Всюду далее предполагаем граф ориентированным и взвешенным. Наша задача в том, чтоб найти кратчайшие пути между всеми парами вершин. Мы уже рассмотрели алгоритм Флойда–Уоршелла, однако сейчас попробуем решить ту же задачу, используя известные алгоритмы поиска кратчайших путей от заданной вершины до всех остальных.

1. Если веса рёбер неотрицательны, мы можем использовать асимптотически более быстрый алгоритм Дейкстры

а) какова сложность выполнения алгоритма "запустить алгоритм Дейкстры из каждой вершины"?

Если есть отрицательные ребра мы можем воспользоваться алгоритмом Беллмана–Форда, но запуск его на каждой вершине даёт очень медленный алгоритм. Попробуем усовершенствовать алгоритм Дейкстры.

Напрямую нельзя ни прибавить ко всем рёбрам одно число, ни класть в очередь с приоритетом вершину после релаксации ребра

б) почему?

2. Пусть наш граф имеет не только взвешенные рёбра, но и взвешенные вершины, каждой вершине v сопоставлено действительное число $c(v)$, которое мы будем называть стоимостью вершины. Пусть вес ребра (u, v) это $w(u, v)$. Зададим новую функцию весов рёбер

$$w'(u, v) = c(u) + w(u, v) - c(v)$$

Тогда при выходе из вершины мы платим $c(u)$, при входе в вершину мы получаем обратно $c(v)$.

а) Доказать, что для любых двух вершин u, v кратчайший путь между ними при весовой функции w совпадает с кратчайшим путём при весовой функции w' при любом выборе функции c.

Итак, если мы выберем функцию c так, чтобы все веса w' были неотрицательны (и сделаем это достаточно быстро), мы сможем использовать алгоритм Дейкстры и получить эффективный алгоритм поиска всех кратчайших путей.

Пусть вершина s такова, что все вершины графа достижимы из неё.

б) Доказать, что если $c(v) = d(s, v)$, где $d(s, v)$ — кратчайшее расстояние между вершинами s и v в исходном графе, то w' неотрицательна для всех рёбер графа.

Что делать, если в графе нет такой вершины s?

в) Модифицируйте исходный граф так, чтобы для некоторой известной вершины все остальные вершины были достижимы из неё.

Осталось лишь свести все шаги вместе:

г) Полностью опишите алгоритм поиска кратчайших путей между всеми парами вершин графа, докажите его корректность и оцените время его работы. Алгоритм должен работать корректно на графах с отрицательными рёбрами и должен корректно сообщать о невозможности решения задачи в случае наличия цикла отрицательной длины.

д) Модифицируйте алгоритм так, чтобы он корректно искал все кратчайшие расстояния без ограничений на наличие циклов, а именно: если пути из s в t не существует, алгоритм выдает

на этой паре $+\infty$, если существует путь из s до некоторого цикла отрицательной длины и путь из этого цикла до t, алгоритм выдает на этой паре $-\infty$, иначе алгоритм выдаёт корректное значение кратчайшего расстояния.