Ответы, указания и критерии проверки

Описание критериев:

Критерии

+1 Означает, что описанная в пункте часть решения стоит 1 балл.

-1 Означает, что за описанную в пункте ошибку снимается 1 балл.

2 По-умолчанию означает максимальный балл (2) за описанный случай. Иные трактовки поясняются.

Преамбула к контрольной

Необоснованные ответы не оцениваются! Если в задаче требуется построение алгоритма, то нужно построить оптимальный алгоритм (за неэффективность снижается оценка), доказать его корректность и оценить время работы (если не в условии не оговорено иное). Во всех задачах модель вычислений атомарная, т. е. арифметические операции стоят $O(1)$.

Задача 1 (2). На вход подаётся дерево с неотрицательными весами на рёбрах. Необходимо проверить, является ли это дерево деревом кратчайших путей в некотором графе. То есть, возможно ли выбрать одну из вершин дерева в качестве корня s и ориентировать рёбра от корня к листьям так, что бы получилось дерево кратчайших путей в некотором взвешенном графе G. Постройте эффективный алгоритм, решающий задачу.

Решение. Докажем, что для любого дерева ответ «да», предъявив алгоритм ориентации дерева и доказав существование графа G. Этот алгоритм служит для доказательства свойства, а не является требуемым в задаче алгоритмом.

Выберем произвольную вершину s в этом дереве в качестве корня, ориентируем все рёбра от неё (например, при помощи поиска в ширину). Получим ориентированное «дерево» с выделенным корнем.

Поскольку в дереве между каждыми двумя вершинами путь единственный, то он же будет и кратчайшим. Соответственно, получили такой граф, где поданное на вход дерево — дерево кратчайших путей.

Так мы доказали, что всегда можно найти такой граф G.

Тогда наш алгоритм — единственная команда «вывести «да»», работающая за $O(1)$.

Критерии

2 Верное решение за $O(1)$ со всем доказанным.

0 Любое решение с более чем константной асимптотикой.

-0,5 Не указана асимптотика ($O(1)$).

-0,5 Не указано, что берём вершину s и/или ориентируем рёбра.

+0,5 Указано, что ответ всегда «да».

+1 Рассказано, как можно выбрать корень и ориентировать рёбра, при отсутствии алгоритма с выводом.

Задача 2 (2). Верно ли для функций $f, g : \mathbb{N}_1 \to \mathbb{R}_{>0}$, что $f(n) = O(g(n))$ тогда и только тогда, когда выполняется условие: $\forall n\ f(n) \leqslant c_1 g(n) + c_2$ для некоторых констант $c_1, c_2 > 0$?

Критерии

0 Любое решение с ответом - «Да, верно».

0 Приведены какие-то рассуждения, не приводящие к ответу «да/нет» (скорее всего это означает, что ответ «да», так как иначе был бы контрпример)

0 Если f или g не удовлетворяют условию задачи.

0 Если есть какие-то утверждения вроде «в определении О с какого-то N выполняется, а во втором условии — для всех, поэтому неверно». В данном классе функций все подобные проблемы решаются достаточно большой константной c_2.

-0,5 Если решение верное, то не указаны в контрпримеры конкретные c_1, c_2

-0,5 Если приведены ошибочные утверждения, явно противоречащие математической логике, но не затрагивающие основное решение.

Решение: Утверждение неверно. Контрпример: $f(n) = \frac{1}{n}, g(n) = \frac{1}{n^2}$ (заметим, что эти функции удовлетворяют условию, хотя и редко встречались в курсе). Очевидно, что при $c_1 = 1$ и $c_2 = 1$ выполняется для любых натуральных n: $f(n) \leqslant 1 \cdot g(n) + 1$, но при этом $f(n) \neq O(g(n))$.

Задача 3 (3). Дома в городе на виде сверху имеют форму квадратов со стороной 1 и стоят вплотную друг к другу. Высота каждого дома — целое число от 1 до n^2, все дома составляют квадрат со стороной n. Стоимость соединения двух соседних домов проводом равна разности их высот. Постройте алгоритм, соединяющий все дома в городе в сеть с минимальной суммарной стоимостью.

Решение. Задача объединения всех домов в сеть с минимальной стоимостью может быть рассмотрена как задача построения минимального остовного дерева.

Используем алгоритм Краскала с небольшой модификацией. Уменьшим все веса на единицу, получив ограничения в 0 и $n^2 - 1$, и представим их в системе счисления с основанием n. Это можно сделать с использованием двух разрядов. Поскольку число рёбер оценивается как $O(n^2)$, поразрядная сортировка (radix sort) будет выполнена за $O(n^2)$. Также можно использовать сортировку подсчётом (counting sort) с той же асимптотикой.

Далее используем систему непересекающихся множеств (disjoint set) со временем работы $O(n^2\alpha(|V|))$ или $O(n^2 \log^*(|V|))$ в зависимости от реализации.

Таким образом, суммарное время работы составляет $O(n^2\alpha(|V|))$ или $O(n^2 \log^*(|V|))$. ■

Критерии

+3 - полностью правильное решение

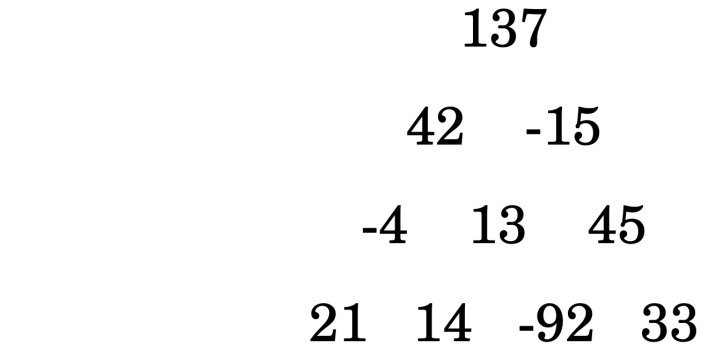
-1 - корректный алгоритм, работающий за время, асимптотически большее приведённого выше, но не больше $O(n^2 \log n)$

-1 - корректный алгоритм, но неправильная оценка сложности или обоснование корректности

1 - корректный алгоритм со временем работы больше $O(n^2 \log n)$

0 - не обоснована корректность или асимптотика

Задача 4 (3). На вход задачи подаётся пирамида из чисел (см. картинку); вход имеет вид n, после чего идут n элементов пирамиды $a_1, \ldots, a_n$ начиная с верхнего (первого) уровня (на i-ом уровне i элементов, гарантируется, что последовательность $a_1, \ldots, a_n$ образует пирамиду). Спуском в пирамиде называется последовательность элементов, начиная с a_1, такая что каждый следующий элемент находится на следующем уровне, причём либо отступает на шаг по диагонали влево от текущего элемента, либо на шаг по диагонали вправо; путь заканчивается на последнем уровне. Стоимостью пути называется сумма элементов вдоль пути. Постройте алгоритм, который находит путь максимальной стоимости.



Решение Для каждого элемента a_i пирамиды подсчитаем максимальную стоимость пути m_i, заканчивающегося в нем и начинающегося в одном из элементов нижнего уровня, а также запомним предпоследний элемент $\pi(i)$ этого пути. Инициализация: для всех элементов нижнего уровня, $a_{n-h+1}, \cdots a_n$, где $h = \frac{\sqrt{1+8n}}{2}$, значение $m_i = a_i$, предшествующий отсутствует. Далее, для всех элементов, начиная от a_{n-h} и до a_1 вычисляем $m_i = max(m_{i+t}, mi + t + 1) + a_i$ (t - номер уровня), $\pi(i) = argmax((m_{i+t}, mi + t + 1))$. На каждом шаге стоимости максимальных путей будут известны для всех элементов, находящихся уровнем ниже рассматриваемого, соответственно, можно вычислить значение для текущего элемента. По итогу стоимостью максимального пути будет значение m_1, сам путь восстанавливается по массиву π.

Сложность по времени: $O(n)$.

Критерии

0.5 Построение ориентированного графа и запуск Беллмана-Форда.

0 Построение ориентированного графа и запуск алгоритма Дейкстры без объяснения корректности его использования.

-1 Использование алгоритма Дейкстры с объяснением.

-0.5 Алгоритм выдает в ответ максимальную стоимость пути, но не сам путь.

-0.5 Отсутствует оценка сложности

Задача 5 (3+4). Назовём M-маршрутом маршрут, рёберная длина которого делится на M. Постройте алгоритм, который получив на вход граф G, положительное целое число M и вершины s и t, находит кратчайший M-маршрут из вершины s в вершину t (или сообщает, что таких нет) в графе G, в случае когда

а) G — ориентированный ациклический граф с единичными весами на рёбрах;

б) G — произвольный ориентированный взвешенный граф с положительными целыми весами на рёбрах (длина маршрута — сумма весов вдоль маршрута).

Решение. Построим новый граф, в нем каждая вершина v исходного графа будет присутствовать в M копиях $v_0, v_1, \ldots, v_{M-1}$. Если в исходном графе было ребро из u в v, то в новом графе будут ребра такого же веса из u_i в $v_{(i+1) \bmod M}$ для всех $i = 0, 1, \ldots M-1$. Таким образом если в старом графе было V вершин и E ребер, то в новом графе будет $M \cdot V$ вершин и $M \cdot E$ ребер. Задача поиска кратчайшего M-маршрута из s в t в исходном графе эквивалентна поиску кратчайшего пути из s_0 в t_0 в новом графе.

a) Все веса ребер единичные, поэтому можно найти кратчайший путь поиском в ширину, итоговая сложность $O(M \cdot (V + E))$

b) Все веса ребер неотрицательные, поэтому можно найти кратчайший путь с помощью Дейкстры, итоговая сложность $O(M \cdot E \cdot \log V)$

Критерии В обоих пунктах

0 Экспоненциальное решение

1 Любое решение с асимптотикой хуже авторского

-1 Не указана асимптотика алгоритма

-0.5 В решении про возведение матрицы в степень не доказана (или неверная) верхняя граница на максимальную степень, в которую надо будет возводить

Задача 6 (3). На вход подаётся число $n > 0$ и последовательность $a_1, a_2, \ldots, a_n$. Постройте алгоритм со сложностью $O(n^2)$, который находит её возрастающую подпоследовательность с максимальной суммой.

Решение. **Алгоритм.** Решение заключается в том чтобы модифицировать алгоритм решения задачи о наибольшей возрастающей подпоследовательности (рассказывался на лекции).

Для этого заполним массив S где на i-м месте будет стоять максимальная сумма возрастающей подпоследовательности оканчивающейся в a_i. Положим $S[i] = a_i$, а затем пересчитываем его по формуле вида

$$S[i] = \max_{j<i : a_j < a_i} \{S[j] + a[i],\ S[i]\}.$$

Помимо этого будем поддерживать массив предков $\pi[i]$ (изначально заполним его как $\pi[i] = -1$), обновляя его в случае когда для некоторого j происходит обновление по правилу выше как $\pi[i] = j$.

После этого сумма подпоследовательности восстанавлливается как $\max_i\{S[i]\}$, а положив $end = \arg\max_i\{S[i]\}$ мы сможем восстановить подпоследовательность как

$$end \to \pi[end] \to \ldots \to \pi[\pi[\ldots]].$$

Корректность. Корректность доказывается по индукции. База индукции $S[1] = a_1$ – очевидна (последовательность из одного числа). Докажем переход. Рассмотрим все возрастающие подпоследовательности которая оканчивается в a_k. Чтобы найти максимум по суммам

в этом множестве надо максимизировать сумму подпоследовательности без учета a_k (т.к. он входит в каждую) – а это у нас записано в $S[i], i < k$ (либо, если a_k меньше всех предыдущих у нас уже записано оно само в $S[k]$), таким образом переход корректен.

Сложность. Мы заполняем $S[i]$ и на каждом шаге просматриваем весь его префикс – это два вложенных цикла, $O(n^2)$. Восстановление ответа затем за $O(n)$, итоговое – $O(n^2)$.

По памяти мы дополнительно потребляем $O(n)$. ■

Решение. **Алгоритм.** Еще один вариант решения (скорее записи решения) это решение через граф следующего вида: в качестве вершин берем a_i, ребро проводим между вершинами $i < j$ если $a_i < a_j$, и в качестве веса ребра берем a_j. Помимо этого добавляем фиктивную вершину s, из которой проводим ребра (s, a_i) с весом a_i для всех i. Тогда задача переформулируется как поиск пути с наибольшей реберной стоимостью в этом графе от вершины s.

В данном графе ребра направлены только от вершин с меньшими номерами в вершины с большими, значит это DAG, таким образом эта задача решается за $O(|V| + |E|)$ (мы инвертируем стоимости ребер, и сводим задачу к задаче поиска пути мин. стоимости в DAG). В данном случае $|V| = n$, $|E| = n^2$ и сложность, таким образом, $O(n^2)$ по времени и по памяти.

Корректность. В данном случае следует из того как мы свели задачу к поиску реберного пути наименьшей стоимости в DAG.

Сложность. Выше описана, $O(n^2)$ по времени и п памяти. ■

Критерии

+3 - полностью правильное решение

-0.5 - Не выписана сложность по времени (надо проговорить почему получается именно $O(n^2)$)

-0.5 - Сложность по памяти $O(n^2)$ в случае решения через ДП (двумерные динамики)

-0.5 - Проблемы с обоснованием корректности

-0.5 - Нет восстановления путей (не снижать если в решении строится массив предков. Если его нет то надо привести алгоритм как восстанавливать пути)

-0.5 - Мелкие ошибки в формуле перехода (не те индексы, не так максимум берется, но при этом из решения понятно что идея динамики корректна)

0.5 - Если сказано что надо свести задачу к задаче поиска наиб. возр. подпосл. но сама сводимость построена некорректно

0.5 - Сложность решения больше $O(n^2)$ по времени (к примеру Форд-Беллман в варианте с графами, если не применять что граф - DAG)

0 Решение для подслов вместо подпоследовательностей (обычно что-то с индексами левой и правой границы подотрезка)

Задача 7(3). В ориентированном взвешенном графе G (возможно с отрицательными весами на рёбрах, но без циклов отрицательного веса) на n вершинах вычислена матрица кратчайших путей между всеми парами вершин. После чего в G добавляют новую вершину и не более чем $2n$ смежных с ней рёбер. Требуется вычислить новую матрицу кратчайших путей между всеми парами вершин за $O(n^2)$. Гарантируется, что в результате добавления рёбер не появится цикл отрицательного веса.

Решение. Обозначим матрицу размера $n \times n$ кратчайших путей графа G как D, новую вершину как v_{n+1}, множество новых ребер как E_{new}, т.ч. $|E_{new}| \leq 2n$. Соответственно, новый граф обозначим за $G' = (V(G) \cup \{v_{n+1}\},\ E(G) \cup E_{new},\ w : E(G) \cup E_{new} \to \mathbb{Z} \cup \{+\infty\})$, его матрицу расстояний как D'.

Определим $d(v_{n+1}, v_i)$ как

$$d(v_{n+1}, v_i) = \min_{j \in [1 \ldots n]\ :\ (v_{n+1}, v_j) \in E_{new}} [w(v_{n+1}, v_j) + D[j][i]].$$

Покажем, что $d(v_{n+1}, v_i) = D'[n+1][i]$. Действительно, кратчайший путь из вершины v_{n+1} в вершину v_i имеет вид $v_{n+1} \to v_{j_1} \to \ldots \to v_{j_k} = v_i$. При этом $(v_{n+1}, v_{j_1}) \in E_{new}$. Заметим, что т.к. условие задачи гарантирует отсутствие циклов отрицательного веса, то для всякого k вершины v_{j_k} лежат во множестве $V(G)$. Также отметим, что путь $v_{j_1} \to \ldots \to v_{j_k}$ будет кратчайшим для пары вершин (v_{j_1}, v_{j_k}) в графе G' (проверяется методом от противного). А это значит, что длина это кратчайшего пути равна $w(v_{n+1}, v_{j_1}) + D[j_1][j_k]$.

Аналогично доказывается, что $D'[i][n+1] = \min\limits_{j \in [1 \ldots n]\ :\ (v_j, v_{n+1}) \in E_{new}} [D[i][j] + w(v_j, v_{n+1})]$.

Рассмотрим теперь произвольную пару вершин $v_i, v_j \in V(G')$. Кратчайший путь из v_i в v_j либо проходит через вершину v_{n+1}, либо нет. В первом случае кратчайший путь содержит вершины только из множества $V(G)$. Но тогда длина кратчайшего из таких путей будет равна $D[i][j]$ по определению матрицы D. Во втором случае кратчайший путь имеет вид суммы двух путей: из v_i в v_{n+1} и из v_{n+1} в v_j. А длину каждого из этих путей мы посчитали выше. Поэтому длина кратчайшего пути v_i в v_j равна $D'[i][j] = \min[D[i][j], d[i][n+1] + d[n+1][j]]$.

Примечание. На последнем шаге можно было использовать одну итерацию алгоритма Флойда-Уоршелла для матрицы D', в которой крайние столбец и строка заполнены в соответствии с функцией d.

Подсчёт $d(v_{n+1}, v_i)$, $d(v_i, v_{n+1})$ занимает $O(n^2)$ арифметических операций, т.к. перебирается $2n$ пар, обсчёт каждой из которой требует $O(n)$ арифметических операций (min пробегает по n вершинам, а минимизируемая функция вычисляется за $O(1)$). Подсчёт $D'[i][j]$ также стоит $O(n^2)$ времени, т.к. обсчитывается n^2 пар вершин, а минимум вычисляется за $O(1)$. ■

Критерии

0 Решение не за $O(n^2)$.

0 В решении не указывается *что* релаксируется.

0 Решение основано на алгоритме Беллмана-Форда и/или нет *полного математического* доказательства почему это верно и *сколько* релаксаций будет сделано.

0.5 Решение состоит из одной итерации Флойда-Уоршелла.

0.5 В решении говорится, что при применении итерации Флойда-Уоршелла старые пути не обновляются.

0.5-1 Если решение сначала пересчитывает кратчайшие расстояния для пар вершин $v, u \in [1 \ldots n]$ и только потом кратчайшие расстояния от или до новой вершины.

-0.5 Если нет обоснования корректности формулы для подсчёта расстояний от или до новой вершины.

-0.5 Если нет обоснования корректности формулы для пересчёта расстояний между вершинами $v, u \in [1 \ldots n]$.

-0.5 Если нет обоснования асимптотики.

до -1.5 Штраф за ошибки в зависимости от грубости.

Задача 8 (5). Вы находитесь в одной из вершин графа, состоящего из квадратных ячеек, и смотрите в одном из направлений - вправо ($0°$), влево($180°$), вверх($90°$), вниз($270°$). За один шаг Вы можете пройти по ребру в прямом направлении, это стоит $w_{ij} > 0$ для прохода из i-й вершины в j-ю, или повернуть по или против часовой стрелки на $90°$, это стоит $r_{ijk} > 0$ для поворота из ориентации j в ориентацию k при нахождении в вершине i. Сведите задачу нахождения кратчайшего пути в таком графе из точки s с ориентацией d_s в точку t с ориентацией d_t к алгоритму Дейкстры с сохранением асимптотики.

Решение. **Вариант 1**

Преобразуем исходный клетчатый граф в граф G следующим образом: каждую клетку i представим в виде четырех вершин, соответствующих возможным ориентациям i_1, i_2, i_3, i_4 (влево, вверх, вправо, вниз), эти вершины соединены ребрами с весами r_{ijk}, например, из i_1 в i_4 ведет ребро r_{i14}, а обратно - r_{i41} (ребер между i_1 и i_3, как и между i_2 и i_4 не будет). Далее добавим ребра с весами w_{ij}, соответствующие переходу из клетки i в смежную клетку j. При этом, если клетка j стоит сверху от i, то в новом графе ребро соединяет i_2 и j_2.

Пусть в исходном графе было n клеток. Очевидно, что в построенном графе G вершин будет $4n$, а ребер не более (граничные клетки порождают меньше) чем $16n$ - столько ребер будет у каждой клетки, имеющей четырех соседей. Само число ребер увеличится не более чем в 5 раз - наибольшее относительное увеличение дают клетки, смежные только с одной, если не учитывать вырожденный случай, когда на вход граф из одной клетки.

Получили ориентированный граф G с неотрицательными весами, на котором запустим алгоритм Дейкстры для вершин d_s и d_t.

Число вершин и ребер увеличилось в $O(1)$ раз, поэтому асимптотика Дейкстры сохранится.

■

Решение. **Вариант 2 - спойлер: неверный**

Запустим Дейкстру на исходном графе со следующей модификацией операции релаксации между u и v. Пусть в вершине u мы находились в ориентации d_u, а переход в вершину v возможен в направлении d_v (они могут быть одинаковыми). Тогда релаксация выполняется по условию $d[v] > d[u] + w_{uv} + R(d_u, d_v)$, где $R(d_u, d_v)$ - минимальный по весу поворот из d_u в d_v, равный сумме необходимых поворотов r_{ijk} (можем в нужное направление попасть последовательностью поворотов по часовой или против часовой стрелки, находим за $O(1)$).

В конечную клетку t мы можем попасть не в нужном направлении, поэтому добавим терминальную вершину t', в которую ведут не более четырех ребер (так как клетка могла иметь меньше смежных с ней) с весами, соответствующими минимальным стоимостям поворотов до нужного направления, что также происходит за $O(1)$.

Это решение неверное, так как мы нарушаем инвариант, что подпуть кратчайшего пути — кратчайший. Для понимания этого факта стоит обратить внимание на первое решение, нарисовать граф G и посмотреть, что происходит при подобном поведении, или на такой пример: возьмем квадрат из четырех клеток, начинаем в нижней левой 1, смотря вправо. Попасть нужно в нижнюю правую 4, смотря вниз. Верхняя левая под номером 2. Переход $1 - 4$, как и $3 - 4$ стоит 10, остальные стоят по 1, поворот в четвертой клетке справа вниз в любом направлении стоит 1000, повороты в остальных клетках стоят по 1.

■

Критерии

$\leqslant 1$ За идею с 4 вершинами, но с невнятным построением (например, не показано, как преобразуются ребра-переходы между клетками)

-2 4 вершины, но с неправильным соединением (например, ромб, который можно пройти по диагонали или который не позволяет проходить исходные клетки в одном направлении)

-1 Неправильный подсчет ребер (дана не верхняя оценка, а точное значение)

-2 Отсутствие подсчета ребер или вершин

-1 Не показано, что асимптотика сохраняется

-1 Не учтено, что переходы или повороты возможны в обе стороны

+2 За использование второго варианта, далее критерии для него

-1 Не сказано, что поворот нужен минимальный

-0,5 Не учтен необходимый поворот в последней вершине

0 За идею второго решения без каких бы то ни было верных пояснений

Задача 9 (6). На вход подаётся слово w в английском алфавите; требуется разрезать его на минимальное число палиндромов, т. е. разбить его в конкатенацию подслов $w = v_1 v_2 \dots v_n$, такую что каждое слово v_i читается слева направо и справа налево одинаково (слово из одной буквы годится) и n минимальное из возможных. Сложность $O(n^2)$.

Решение. Сначала заполним двумерный булевый массив P (размера $n \times n$), такой что $P[i][j] = 1$ тогда и только тогда, когда слово $w[i] \dots w[j]$ является палиндромом. Динамическое программирование:

- $P[i][i] = 1$
- $P[i][i+1] = 1$, если и только если $w[i] = w[i+1]$
- $P[i][j] = P[i+1][j-1] \wedge (w_i = w_j)$ при $j - i > 1$
- Иначе $P[i][j] = 0$

Массив P очевидно заполняется за $O(n^2)$, например динамическим программированием сверху (перебираем все пары i, j и вызываем рекурсию с запоминанием).

После заполнения массива P заполняем массив d, такой что $d[i]$ — ответ на вопрос задачи для префикса длины i. $d[k] = 1$ при $P[1][k] = 1$, иначе

$$d[k] = \min_{\substack{P[i+1][k]=1, \\ 1 \leqslant i < k}} \{d[i] + 1\}$$

Решение восстанавливаем стандартно через массив предков.

Корректность. В оптимальном разрезании префикса $w_1 \dots w_k$ в конкатенацию палиндромов есть последний. Переберём все возможные последние палиндромы $w_{i+1} \dots w_k$, стоимость оптимального разрезания $w_1 \dots w_k$ при условии отрезания последнего палиндрома $w_{i+1} \dots w_k$ таким образом равна 1 плюс стоимость оптимального разрезания $w_1 \dots w_i$, которая уже вычислена в $d[i]$. Взяв минимум по всем возможным последним палиндромам получим ответ (среди них встретится входящий в оптимальное разрезание). Таким образом в ячейке $d[n]$ записана стоимость оптимального разрезания, а само разрезание стандартно восстанавливается по релаксациям или обратным проходом по d.

Сложность. Вычисление массива P производится на препроцессинге и стоит $O(n^2)$. Вычисление ячейки $d[k]$ стоит $O(k)$: перебираем i от 1 до $k-1$ и проверяем условие $P[i+1][k]$ за $O(1)$. Итого, заполнение массива d стоит $O(n^2)$, что и является итоговой сложностью.

■

Критерии

+0,5 за продвижение в задаче. Типичные продвижения: выписано что-то близкое к нужному рекуррентному соотношению, но как проверять на палиндромы и какие i рассматривать (все, одни, подходящие) — не сказано; общая формула «перебираем все возможные разрезы слова на два, а дальше рекурсивно» без объяснения подробностей. Были также нестандартные идеи, которые можно было довести до правильного решения.

1 Если вычислен только массив P и дальше нет продвижения в решении самой задачи. То же самое в случае поиска всех подслов-палиндромов «от центров» (кроме только лозунгов «идём от центров, найдём палиндромы»)

3 Решение за $O(n^3)$

0 Жадные решения: отрезаем самый длинный первый палиндром или отрезаем первый палиндром, который можем. Полный перебор. (Возможны баллы за продвижение)

0 Нет внятного описания алгоритма (после прочтения текста, алгоритм невозможно исполнить). (Возможны баллы за продвижение)

Типовые случаи. Заполнялся массив P (или какой-то другой двумерный булевый массив) после чего было описание типа «идём вниз пока не встретим первую 1». Ни в одном таком решении я не смог восстановить подразумеваемый алгоритм (описан он не был). Если читать буквально, то первая единица встречается сразу же. Неясно как первая единица влияет на итоговое разбиение. Ставился 0, поскольку сам алгоритм описан не был, и доказательство корректности тоже приведено не было.

Обозначим через a^n слово $\underbrace{aa\ldots a}_{n}$. Жадное разрезание (отрезаем самый длинный палиндром сначала) даёт неверный ответ на слове $a^{2n}ba^n = a^{2n}\cdot b\cdot a^n$, в то время как верный ответ $a^n\cdot a^n b a^n$.

В нескольких решениях «использовались хеш-функции для проверки подслова на палиндром за $O(1)$». Это невозможно из-за коллизий.