Алгоритмы и структуры данных. Домашнее задание №5

Выполнил студент Наседкин Дмитрий Сергеевич (группа 242)

Письменная часть

№ 1

Пункт а)

$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$

$f(x) = x^4 - a$, тогда $x_{i+1} = x_i - \frac{{x_i}^4 - a}{4{x_i}^3}$

То есть $\varphi(x) = x - \frac{x^4-a}{4x^3} = \frac{4x^4-x^4+a}{4x^3} = \frac{3x^4+a}{4x^3}$

Пункт б)

Докажем, что метод сойдется $\forall x_0 > 0$. (То есть для любого начального x из условия).

1. $x = \sqrt[4]{a}$, тогда $\varphi(x) = x$.
2. $x > \sqrt[4]{a}$, тогда $x > \frac{a}{x^3}$

$$\varphi(x) = \frac{1}{4}(3x + \frac{a}{x^3}) < \frac{1}{4}(3x + x) = x$$

$$\varphi(x) = \frac{1}{4}(3x + \frac{a}{x^3}) \geq \sqrt[4]{x \cdot x \cdot x \cdot \frac{a}{x^3}} = \sqrt[4]{a}$$

(Последнее неравенство выполняется так как среднее арифмитическое больше либо равно среднему геометрическому).
То есть если в какой-то последовательности x_i будет $\geq \sqrt[4]{a}$, то последовательность далее монотонно убывает к $\sqrt[4]{a}$. То есть последовательность сходится к корню. (Если $x_0 > \sqrt[4]{a}$, то тем более).

3. $0 < x < \sqrt[4]{a}$, тогда $x < \frac{a}{x^3}$

$$\varphi(x) = \frac{1}{4}(3x + \frac{a}{x^3}) > \frac{1}{4}(3x + x) = x$$

$$\varphi(x) = \frac{1}{4}(3x + \frac{a}{x^3}) \geq \sqrt[4]{x \cdot x \cdot x \cdot \frac{a}{x^3}} = \sqrt[4]{a}$$

То есть если $x_0 < \sqrt[4]{a}$, то $x_1 \geq \sqrt[4]{a}$, а дальше случай (2), то есть тоже сходится к корню.

Пункт в)

UPD: НЕ ЧЕКАТЬ, задание отменили(ну я считаю 0.01 дать стоит :3)

Разложим $\varphi(x) - \sqrt[4]{a}$ на множетели:

$$\varphi(x) - \sqrt[4]{a} = \frac{3x^4 + a}{4x^3} - \sqrt[4]{a} = \frac{3x^4 - 4x^3 \cdot \sqrt[4]{a} + a}{4x^3} = \frac{(x - \sqrt[4]{a})^2(3x^2 + 2\sqrt[4]{a} \cdot x + \sqrt{a})}{4x^3}$$

Т.о. условие из:

$$|\varphi(x) - \sqrt[4]{a}| \leq C \cdot |x - \sqrt[4]{a}|^2$$

превращается в:

$$\frac{|3x^2 + 2\sqrt[4]{a} \cdot x + \sqrt{a}|}{|4x^3|} \leq C$$

Так как $x > 0$ и $a > 0$, то можно снять модули:

$$\frac{3x^2 + 2\sqrt[4]{a} \cdot x + \sqrt{a}}{4x^3} \leq C$$

Выберем $C = \frac{3}{2}$, тогда $-6x^3 + 3x^2 + 2\sqrt[4]{a} \cdot x + \sqrt{a} \leq 0$ при $a > 1$ и $x \geq \sqrt[4]{a}$, то есть можно взять, например,отрезок $[\sqrt[4]{a}, max(\sqrt[4]{a} + 1, 2\sqrt[4]{a})]$, он будет ненулевым.

№ 2

P.S. В чате подтвердили, что n - степень двойки(хотя это и не важно как увидим ниже).

Также давайте обозначим $e^{\frac{2\pi}{n}i}$ как w - главный комплексный корень n-ой степени. Также заранее очев, что $w^n = w^0 = 1$.

Помним, что дискретным преобразованием Фурье называется вычисления значений многочлена в комплексных корнях из 1, т.е.:

$$y_j = \sum_{k=0}^{n-1} a_k \cdot e^{ij\frac{2\pi k}{n}} = \sum_{k=0}^{n-1} a_k \cdot w^{jk}$$

Нам дана последовательность $1, 2, \ldots, n$, тогда

$$y_j = \sum_{k=0}^{n-1} (k+1) \cdot w^{jk}$$

$$y_j \cdot w^j = \sum_{k=0}^{n-1} (k+1) \cdot w^{j(k+1)}$$

Т.О.:

$$y_j(1 - w^j) = \sum_{k=0}^{n-1} w^{kj} - nw^{jn}$$

Заметим, что слева у нас сумма геометрической прогрессии, то есть получим:

$$y_j(1 - w^j) = \frac{1 - w^{jn}}{1 - w^j} - nw^{jn} = \frac{1 - w^{jn} - nw^{jn} + nw^{j(n+1)}}{1 - w^j}$$

$$y_j = \frac{1 - w^{jn}(1+n) + nw^{j(n+1)}}{(1 - w^j)^2} = \frac{1 - (1+n) + nw^j}{(w_j - 1)^2} = \frac{n(w^j - 1)}{(w_j - 1)^2} = \frac{n}{w_j - 1}$$

Если $j = 0$, то давайте(так как иначе в знаменателе 0) явно посчитаем y_0 за $O(n)$, это будет просто $\sum_{k=1}^{n} k$.

Если $j \neq 0$, то есть $j \in [1, n-1]$, тогда воспользуемся посчитанной формулой для y_j и за $O(1)$ посчитаем $\frac{n}{w_j - 1}$, после чего возьмем от полученного комплексного числа вещественную часть(и округлим). Итоговое время работы - $O(n)$.

Устная и письменная часть

№ 3

Пункт а

Для начала посчитаем сумму весов всех гирек, если она не делится на 4, то ответ очевидно, что нельзя распределить на 4 равные кучки, тогда назовем S - нужная сумма для каждой кучки(четверь суммы всех гирек).

Давайте воспользуемся meet-in-the-middle, а именно разобьем массив на 2 равные части(если n нечетно, то почти равного размера). Для левой части массива переберем все $4^{\frac{n}{2}} = 2^n$(для каждого предмета 4 варианта) вариантов и для каждого за $\frac{n}{2}$ посчитаем сколько попало веса в каждую кучку. Так получим 4 числа, которые сохраним в хеш-таблицу.

Теперь переберем все $4^{\frac{n}{2}} = 2^n$ вариантов для правой части и точно также за $O(n)$ насчитаем вес каждой кучки. Тогда чтобы получить вес (S, S, S, S), если у нас сейчас $(w1, w2, w3, w4)$ нужно лишь проверить, есть ли в хеш-таблице набор $(S - w1, S - w2, S - w3, S - w4)$. Тогда если хотя бы для одного из 2^n вариантов был ответ да, то распределить можно, иначе нельзя.

Время работы - $O(2^n \cdot n)$.

Пункт б

P.S. Веса гирек неотрицательные.

Точно также как в пунке а, давайте для начала проверим, делится ни сумма весов гирек на k, если нет, то решения не существует, иначе скажем, что нужно разделить гирки на k групп, сумма каждой должна быть S
.

Заметим, что одно из наивных решений, это перебрать все перестановки, и тогда как проверить что перестановка задает k групп массой каждой S? Будем поддерживать текущую сумму весов слева направо, и как только при добавлении очередной гирки она стала S, то нужно организовать новую группу, а если стала больше S, то данная перестановка нам не подходит. Так как сумма всех весов делится на k, то если мы дошли до конца перестановки, то значит решение существует.

Теперь ускорим наше наивное решение с помощью dp по подмаскам, введем $dp(mask)$ - сумма в текущей(незаконченной) группе, если мы взяли подмножество $mask$, и -1, если такой конфигурации не существует. Очев, что $dp(0) = 0$. Тогда переберем маски по возрастанию, и для каждой пройдемся по очередному предмету, который хотим добавить в маску, тогда если $i \notin mask$ и $dp[mask] \neq -1$ и $dp[mask] + w[i] <= S$, то $dp[mask|(1 << i)] = (dp[mask] + w[i]) \mod S$. Другими словами, мы добавляем предмет к какой-то "хорошей" недостроенной перестановке, и добавляем новую группу, если текущая сумма добралась до значения S. В конце остается лишь проверить, что $dp(2^n - 1) \neq -1$, тогда решение существует, иначе нет. (Оно либо -1, либо 0, так как мы уже знаем что сумма всех гирек делится на k).

№ 4

Если в графе нет ребер($m = 0$), то очевидно, что размер минимального вершинного покрытия - 0.

Давайте перебирать d - размер вершинного покрытия от 1 до n, и как только получим ответ "да, вершинное покрытие такого размера существует" - остановимся, это и будет ответ на задачу. Заметим, что если мы докажем, что проверка условия "Правда ли, что размер вершинного покрытия равен k" работает за $O(2^k \cdot nm)$, то суммарное время работы будет
$O(2nm) + O(4nm) + O(8nm) + \dots + O(2^d \cdot nm) = O(2^{d+1} \cdot nm) = O(2^d \cdot nm)$. Теперь научимся проверять нужное условие.

Будем искать существование вершинного покрытия размера k рекурсивно:

Пусть мы уже взяли s вершин:

- Если покрыли все ребра, то ответ "да".
- Если $s = k$, то ответ "нет".

- Иначе берем любое непокрытое ребро (v, u), и запустимся рекурсивно, попробовав взять в множество v, и если ответ "нет" то взять u. Если взяв v и u(не вместе, а попробовав каждую по отдельности) получили ответ "нет", то ответ - "нет", иначе - "да"

Корректность очевидна. Давайте найдем время работы:

- Всего в дереве рекурсии не более $1 + 2 + 4 + \dots + 2^k = 2^{k+1} - 1 = O(2^k)$ вершин.
- В каждой вершине дерева рекурсии мы за $O(nm)$ проверяем, покрыты ли все ребра, и возвращаем любое, если покрыты не все. Для этого, например, будем поддерживать текущий список взятых вершин в векторе(добавление\удаление очередной взятой вершины делается за $O(1)$), и для каждого ребра будем за $O(n)$ проверять покрыто ли оно.

Итого $O(2^k \cdot nm)$ для проверки условия, а значит и суммарное время работы составит $O(2^d \cdot nm)$.

№ 5

Заметим, что наш минимальный подграф ответ - это обязательно дерево. Док-во: пусть это не так, и минимальный ответ это не дерево, то есть в нем есть цикл, тогда уберем из цикла любое ребро, заметим что связность не потерялась, а ответ точно стал меньше(все ребра положительны), значит это не минимальный подграф, пр-ие.

Назовем выбраные k вершин терминальными. (А также перенумеруем граф таким образом, чтобы терминальные вершины были первыми, так удобнее будет работать с масками)

Давайте введем $dp[mask][v]$ - дерево минимального веса, который связывает $mask$(подмножество) терминальных вершин, и подвешено за вершину v(то есть обязательно содержит вершину v).

Заметим следующий факт: пусть у нас есть оптимальное дерево, которое содержит $mask$ терминалов, и оно подвешено за вершину v, тогда это дерево можно однозначно разбить на 2 дерева, которые содержат какой-то поднабор(непересекающихся) терминалов, и они будут связаны каким-то ребром(назовем его ключевым ребром). Ключевое ребро соединит либо 2 непустых поднабора, либо полный поднабор и пустой(но в этом случае в оптимальном поддереве v будет простой путь от v до единственной доступной терминальной вершины, это важно - так как иначе пришлось бы запускать Дейкстру от всех вершин).

То есть получим следующий план пересчета: сначала насчитать $dp[mask][v]$ для каждой v, так мы найдем все оптимальные разбиения, где ключевое ребро точно будет между двумя непустыми поднаборами терминалов, после чего запустить от каждой терминальной вершины в $mask$ алгоритм дейкстры и насчитать кратчайшие расстояния от t_i до всех, попробовать обновить $dp[mask][u]$ с помощью $dp[mask][t_i] + d(t_i, u)$, где $d(v, u)$ - кратчайшее расстояние от v до u. Так найдем оптимальные значения и для тех вершин, где ключевое ребро между полным поднабором и пустым.

Начальные значения: $dp[2^{t_i}][v] = d(v, t_i)$, где t_i - i-ный терминал, а $d(v, u)$ кратчайший путь от v до u.

А также $dp[0][v] = 0$, для всех остальных поставим очень большое значение.

Формула пересчета:

$$dp[mask][v] = \min_{u \in V,\, submask \in mask} dp[submask][v] + dp[submask \oplus mask][u] + w(v, u),$$

где $w(v, u)$ - длина ребра между (v, u).

Ответ очевидно это $\min\limits_{v \in V} dp[2^k - 1][v]$.

Оценим время работы: всего подмасок у всех масок - $O(3^k)$, тогда пересчет dp по формуле пересчета суммарно работает за $O(3^k \cdot n^2)$. Всего будет раз запущено Дейкстра $2^k \cdot k \leq 3^k$, то есть $O(3^k)$. Каждый работает за $O(n^2)$, то есть суммарно $O(3^k \cdot n^2)$.

А значит и весь алгоритм работает за $O(3^k \cdot n^2)$.

№ 6

P.S. Нумерация всегда с 0.

Давайте обозначим $e^{\frac{2\pi}{n}i}$ как w - главный комплексный корень n-ой степени. Также заранее очев, что $w^n = w^0 = 1$.

Помним, что дискретным преобразованием Фурье называется вычисления значений многочлена в комплексных корнях из 1, т.е.:

$$y_j = \sum_{k=0}^{n-1} a_k \cdot e^{ij\frac{2\pi k}{n}} = \sum_{k=0}^{n-1} a_k \cdot w^{jk}$$

Заметим, что комплексное число y_j имеет только вещественную часть, если и только если $y_j = \overline{y_j}$.

$$\overline{y_j} = \sum_{k=0}^{n-1} a_k \cdot \overline{w^{jk}} = \sum_{k=0}^{n-1} a_k \cdot w^{-jk}$$

Таким образом, чтобы y_j имело только вещественную часть необходимо и достаточно:

$$\sum_{k=0}^{n-1} a_k \cdot w^{jk} = \sum_{k=0}^{n-1} a_k \cdot w^{-jk}$$

Распишем подробнее правую часть:

$$\sum_{k=0}^{n-1} a_k \cdot w^{-jk} = a_0 + \sum_{k=1}^{n-1} a_{n-k} \cdot w^{jk}$$

Таким образом получим уравнение $\sum_{k=1}^{n-1} (a_k - a_{n-k}) w^{jk} = 0 \quad \text{для всех } j$. Очевидно, что тогда единственное решение - $a_k - a_{n-k} = 0 \ \forall k \in [1, n-1]$. То есть другими словами, единственное условие: массив должен быть симметричным относительно $\frac{n}{2}$. (То есть a_0 и $a_{\frac{n}{2}}$ могут быть любыми).

То есть чтобы вычислить нужно пройтись по $i = 1, 2, \ldots \frac{n}{2} - 1$ и проверить, что $a_i = a_{n-i}$, иначе заменить одно из чисел на "зеркальное". Очевидно что это делается за $O(n)$.