Домашняя работа №1

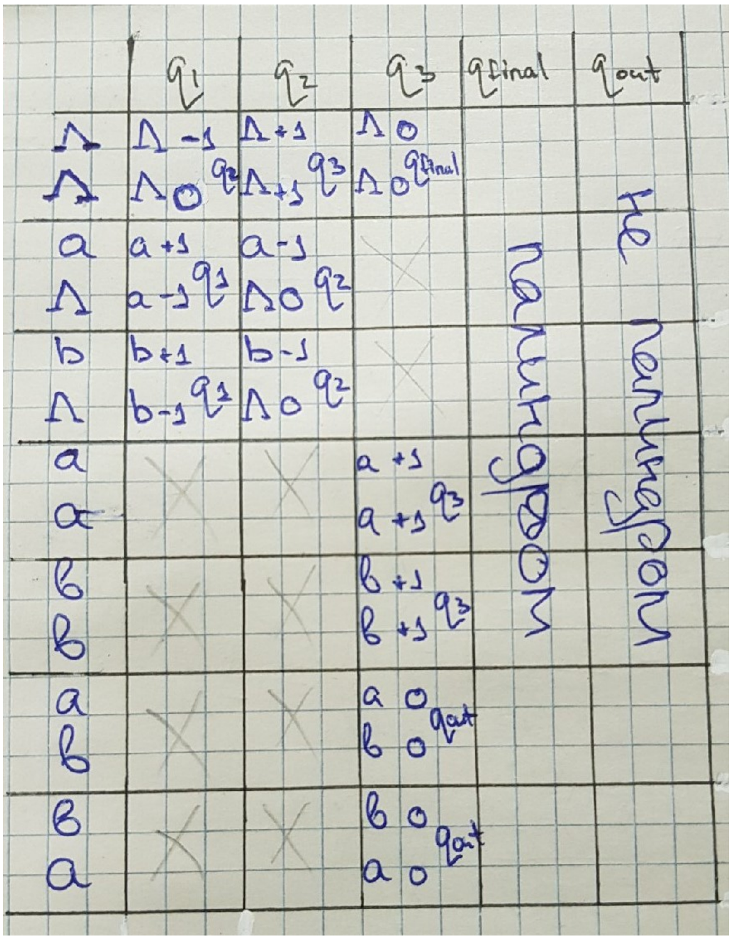
Бредихин Александр

17 февраля 2020 г.

Задача 1

Задача: построим МТ с двумя лентами, которая распознаёт является ли изначальная последовательность символов на первой ленте над алфавитом $A = \{a, b, \Lambda\}$ палиндром или нет.
Построим таблицу переходов такой МТ:
Крестик в таблице означает, что в такой ситуации МТ не окажется.

(не получилось построить таблицу в техе, так как ещё только знакомлюсь с ним)



	q_1	q_2	q_3	q_{final}	q_{out}
Λ Λ	Λ -1 Λ 0 q_2	Λ +1 Λ +1 q_3	Λ 0 Λ 0 q_{final}		
a Λ	a +1 a -1 q_1	a -1 Λ 0 q_2		палиндром	не палиндром
b Λ	b +1 b -1 q_1	b -1 Λ 0 q_2			
a a			a +1 a +1 q_3		
b b			b +1 b +1 q_3		
a b			a 0 b 0 q_{out}		
b a			b 0 a 0 q_{out}		

Алгоритм:

- Сначала, с помощью состояния q_1, смещаемся по первой ленте вправо и пишем на вторую ленту те же символы, что и на первой ленте. Когда на первой ленте встретим Λ, переходим в состояние q_2, перемещая головку на первой ленте на первый ненулевой символ (то есть делаем -1).
- Перемещаем головку на первой ленте в её начало. Когда встречаем Λ на первой ленте (что означает, что мы вернулись в её начало) сдвигаем головку верхней и нижней ленты вправо на 1, теперь они обе нахоядся в начале слов на ленте и на второй ленте символы написаны в перевёрнутом порядке по сравнению с первой. (см. работу в состояние q_2).
- С помощью состояния q_3 сверяем последовательности символов на первой и на второй лентах (на второй перевёрнутая последовательность). Если находим различие переходим в состояние q_{out}, которое

говорит, что последовательность не палиндром
Если дошли до Λ, то перевёрнутая последовательность сходится с прямой, значит, слово - палиндром (переходим в конечное состояние q_{final}.

Задача 2

Доказать, что следующие определения перечислимого множества $X \subset N$ эквивалентны:

- Существует алгоритм, печатающий все элементы множества (в любом порядке и со сколь угодно большими паузами между элементами).
- Множество является областью определения некоторой вычислимой функции.
- Множество является областью значений некоторой вычислимой функции.

$\square$

(1 $\Rightarrow$ 3) Можно построить функцию $g(n)$, область значение которой – X: Для этого на входе n запускаюем алгоритм, печатающий элементы – X, и будем считать количество напечатанных элементов, когда будет напечатан nый элемент, он будет выдаваться в качестве значения функции $g(n)$ на входе n. Получается, множество значений функции g – X

(3 $\Rightarrow$ 2) Пусть $X = f(n)$ для некоторой функции f. Опишем алгоритм вычисления функции g, который получает на вход x: перебирает все числа и для каждого числа n вычисляет f(n) и сравнивает с x. Если $f(n) = x$, то $g(x) = 1$, иначе - $g(x)$ не определена в x (то есть алгоритм вычисления g не даёт никакого результата), получается, что X - область определения g.

(3 $\Rightarrow$ 1) Пусть функция $f(n)$ имеет область значений X. Для каждого натурального числа, если функция f(n) определена, то печатаем результат. Построили алгоритм, печатающий все элементы множества X и только их.

(2 $\Rightarrow$ 3) Пусть $g(x)$ - функция, область значений которой - X. Переопределим g(x) так, что для каждого x из X $g(x) = x$. Получили, что - область значений некоторой вычислимой функции.

■

Задача 3

Задача: найти элемент в массиве длиной n, который встречается более чем $n/2$ раз за 2 прохода по массиву (сложность по времени $\mathcal{O}(n)$) используя память $\mathcal{O}(n \log n)$

Алгоритм:
Создадим 2 переменные:
candidat - тут будет лежать элемент, кандитат на нужный нам
flag - счётчик

Проходимся по массиву циклом и делаем следующие:

- Если переменная flag равна 0, то в переменную candidat кладём текущий элемент, а flag увеличиваем на 1
- Иначе если текущий элемент равен тому, который в candidat, увеличиваем flag на 1
- во всех остальных случаях уменьшаем flag на 1

После этого прохождения по массиву в переменной condidat будет лежать элемент, который встречается более чем $n/2$ раз, если такой в массиве существует (строго док-ва не привожу, но алгоритм рассказывали в школе, а для интуитивного понимания привели такую ситуацию: n дамм приходят на балл в каких-то разных платьях(например, разного цвета). Если дамма встречает другую с другим платьем, то они садятся. Останутся стоять только даммы в одинаковых платьях.
Но в массиве может и не быть нужного элемента, но в переменной candidat все равно что-то будет. Поэтому вторым проходом по массиву проверям существует ли вообще такой элемент.
Получаем алгоритм сложность которого $\mathcal{O}(n)$, а требуемая дополнительная память — $\mathcal{O}(1)$
Пример кода алгоритма на python:

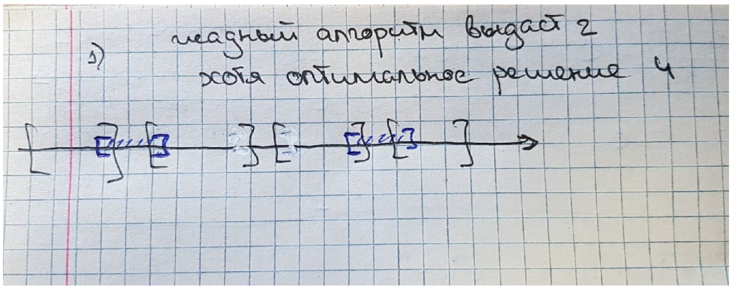
In [2]:
```
n = int(input())
s = list(map(int,input().split()))

candidat = 0
flag = 0
for i in range(n):
    if (flag == 0):
        candidat = s[i]
        flag += 1
    elif (candidat == s[i]):
        flag += 1
    else:
        flag -= 1

flag = 0
for i in range(n):
    if (s[i] == candidat):
        flag += 1

if (flag > n//2):
    print(candidat)
else:
    print("Нет такого элемента")
```

```
5
1 2 3 2 2
2
```

Задача 4

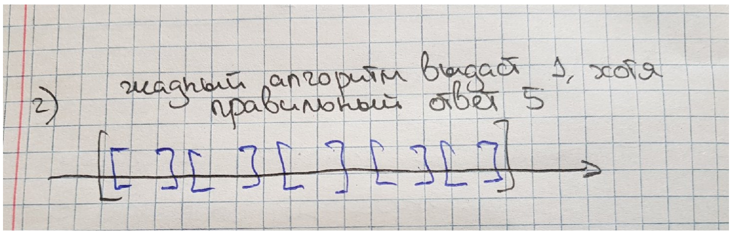
Задача: посетить как можно больше мероприятий при условии, что мероприятие можно посещать только полностью, то есть с начала и до конца

- Рассмотрим 1ый жадный алгоритм. Он не является оптимальным, так как существует контрпример:



В нём вместо каждого "короткого: отрезка , который выберет жадный алгоритм, оптимально выбирать 2 других непересекающихся длинных

- Рассмотрим 2ой жадный алгоритм. Он не является оптимальным, так как существует контрпример:



Вместо одного длинного события, которое начинается раньше всех, оптимально выбирать 5, которые идут последовательно друг за другом во время длинного.

- 3ий жадный алгоритм является оптимальным, докажем это:
От противного. Пусть существует пример, когда наш жадный алгоритм работает не оптимально. Выбирем из таких примеров самый минимальный. Например:
жадный алгоритм: $[1, 3)$, $[4, 6)$, $[6, 8)$ ...
оптимальный алгоритм: $[2, 4)$, $[4, 7)$ ...
Посмотрим на первую заявку в жадном решении, эта та заявка, которая раньше всех заканчивается, и пусть она не входит в оптимальное решение, значит, первая заявка в оптимальном решении заканчивается позже(не раньше) заявки в жадном решении. Тогда можно поменять первую заявку в жадном решении и оптимальном, то есть в нашем примере поменяем заявки $[1, 3)$ и $[2, 4)$. Теперь в жадном и оптимальном решениях первые заявки одинаковые (если мы так сделаем, ничего не изменится), поэтому мы можем убрать её и перейти к примеру с меньшим количеством заявок. Получаем

противоречие с тем, что изначально выбрали минимальный пример. Следовательно этот жажный алгоритм работает всегда оптимально.
Сложность алгоритма: $\mathcal{O}(n) + \mathcal{O}(n \log n)$
Предварительная сортировка по правому пределу ($\mathcal{O}(n \log n)$), проход по массиву($\mathcal{O}(n)$)

Задача 5

Задача: получить производящую функцию чисел BR_{4n+2} – правильные скобочные последовательности длины $4n+2$
Обозначим BR_{4n+2} за T_{2n+1}. И для чисел Каталана(правильных скобочных последовательностей длины n известно рекурентное соотношение (дальше будут сложные и не совсем логичные выкладки, берутся они из аналогии для вывода производящей функции для чисел Каталана длины $2n$):
$T_n = T_0T_{n-1} + T_1T_{n-2} + \ldots + T_{n-1}T_0$, где $T_0 = 1$
Производящая функция в нашем случае (для чисел Каталана, где открывающихся скобок $2n+1$) принимает вид:
$A(x) = T_1 + T_3x + T_5x^2 + \ldots + T_{2n+1}x^n + \ldots$
Только нечётные коэффициенты. Рассмотрим также и производящую функцию для чисел Каталана с чётным количеством открывающихся и закрывающихся скобок, то есть с чётными коэффициентами:
$B(x) = T_0 + T_2x + T_4x^2 + \ldots + T_{2n}x^n + \ldots$
Возьмём их произведение и преобразуем его:

$$AB = \left(T_1 + T_3x + T_5x^2 + \ldots\right)\left(T_0 + T_2x + T_4x^2 + \ldots\right)$$
$$AB = T_1T_0 + (T_3T_0 + T_1T_2)\,x + (T_1T_4 + T_2T_3 + T_5T_0)\,x^2 + \ldots \qquad (1)$$

Из рекурентной формулы заметим, что

$$T_2 = T_1T_0 + T_0T_1 => T_1T_0 = T_2/2$$
$$T_3T_0 + T_1T_2 = T_4/2$$
$$T_1T_4 + T_2T_3 + T_5T_0 = T_6/2$$

И так далее . . . Подставляем эти соотношения в формулу (1), получается

$$AB = \frac{T_2}{2} + \frac{T_4}{2}x + \frac{T_6}{2}x^2 + \ldots + \frac{T_{2n}}{2}x^{n-1+\ldots} \qquad (2)$$

Домножаем обе части уравнения (2) на $2x$, в правой части получается $B(x)$ без первого члена $T_0 = 1$, поэтому уравнение (2) приобретает такой вид:

$$2xAB = T_2x + T_4x + \ldots + T_{2n}x^n + \ldots$$

$$2xAB = B - 1 \tag{3}$$

Нужно получить ещё одно уравнение на производящие функции A и B, для этого возведём B и A в квадрат и снова пользуясь соотношениями из рекурентной формулы для чисел Каталана получим уравнение:

$$B^2 = \left(T_0 + T_2x + T_4x^2 + \ldots\right)\left(T_0 + T_2x + T_4x^2 + \ldots\right)$$

$$B^2 = T_0^2 + \left(T_0T_2 + T_2T_0\right)x + \left(T_0T_4 + T_2T_2 + T_4T_0\right)x^2 + \ldots$$

Из рекупентной формулы:

$$T_0^2 = T_1$$

$$T_0T_2 + T_2T_0 = T_3 - T_1^2$$

$$T_0T_4 + T_2T_2 + T_4T_0 = T_5 - \left(T_1T_3 + T_3T_1\right)$$

Получается:

$$B^2 = T_1 + T_3x - T_1^2x + T_5x^2 - \left(T_1T_3 + T_3T_1\right)x^2 + \ldots$$

Заметим, то что с минусом в выражении выше немного похоже на A^2, распишем A^2 и получим второе уравнение:

$$A^2 = \left(T_1 + T_3x + T_5x^2 + \ldots\right)\left(T_1 + T_3x + T_5x^2 + \ldots\right)$$

$$A^2 = T_1^2 + \left(T_1T_3 + T_3T_1\right)x + \left(T_1T_5 + T_3T_3 + T_5T_1\right)x^2 + \ldots$$

Домножаем A^2 на x и получаем выражение:

$$B^2 = A - A^2x \tag{4}$$

Получаем систему из двух уравнений: (3) и (4), где A – производящая функция, которую нужно найти:

$$\begin{cases} 2xAB = B - 1, \\ B^2 = A - A^2x \end{cases}$$

Из уравнения (3): $A = \frac{B-1}{B \cdot 2x}$ подставляем в уравнение (4), получаем:

$$B^2 = \frac{B-1}{2xB} - \frac{(B-1)^2}{4B^2x^2}x$$

$$4xB^4 = (B-1)(B+1)$$
$$4xB^4 = B^2 - 1$$

Это квадратное уравнение относительно B^2, его решения:

$$B_{12}^2 = \frac{1 \pm \sqrt{1 - 16 \cdot x}}{8x}$$

Выбираем знак из условия, что если в характеристическую функцию поставить 0, то получится $T_0 = 1$, тогда из $8xB^2 = 1 \pm \sqrt{1-16x}$ понятно, что нужно выбирать знак «-». В итоге:

$$B^2 = \frac{1 - \sqrt{1-16x}}{8x} \tag{5}$$

Решаем (4) как квадратное уравнение относительно A, знак корня выбираем из тех же соображений, что и раньше, получаем:

$$A = \frac{1 - \sqrt{1 - 4xB^2}}{2x}$$

Подставляем в это выражение значение для B^2 из (5), получаем ответ для производящей функции A:

$$A = \frac{1 - \sqrt{\frac{1}{2} + \frac{\sqrt{1-16x}}{2}}}{2x}$$

Ответ: $A = \frac{1 - \sqrt{\frac{1}{2} + \frac{\sqrt{1-16x}}{2}}}{2x}$

Задача 6

По условию задачи сразу можно составить рекуренту:

$$T(n) = 3T\left(\left[\frac{n}{\sqrt{3}}\right] - 5\right) + \theta\left(10\frac{n^3}{\log n}\right)$$

Заметим, что при больших n константа -5 очень мала по сравнению с тем, что мы делим на $\sqrt{3}$, следовательно ей и целым округлением можно пренебречь, получим:

$$T(n) = 3T\left(\frac{n}{\sqrt{3}}\right) + \theta\left(\frac{n^3}{\log n}\right)$$

Распишем, как меняется сумма уровня дерева от глубины и найдём закономерность

$$\frac{n^3}{\log n} \to 3 \cdot \frac{\frac{n^3}{(\sqrt{3})^3}}{\log\left(\frac{n}{\sqrt{3}}\right)} = \frac{\frac{n^3}{\sqrt{3}}}{\log\left(\frac{n}{\sqrt{3}}\right)} \to 9 \cdot \frac{\frac{n^3}{(\sqrt{3})^3}}{\log\left(\frac{n}{(\sqrt{3})^2}\right)} = \frac{\frac{n^3}{\sqrt{3}}}{\log\left(\frac{n}{(\sqrt{3})^2}\right)}$$

И так далее, в итоге можем записать kый элемент: $\frac{\frac{n^3}{(\sqrt{3})^k}}{\log\left(\frac{n}{(\sqrt{3})^k}\right)}$

Глубина рекурсии будет равна $\log n$, так как каждый раз мы делим на $\sqrt{3}$, поэтому кол-во операций, которое сделает наш алгоритм эта следующая сумма:

$$T(n) = \sum_{k=0}^{\log(n)} \frac{\frac{n^3}{(\sqrt{3})^k}}{\log\left(\frac{n}{(\sqrt{3})^k}\right)}$$

Снизу эту сумму можно оценить первым членом при $k = 0$ (понятно, что вся сумма больше первого члена суммы, так как все числа положительные), то есть

$$T(n) = \Omega(\frac{n^3}{\log n}) \tag{6}$$

Пусть сверху сумма также оценивается первым членом, докажем это по индукции по глубине рекурсии, то есть $\exists C > 0, \forall k < \log(n) \longrightarrow T(k) < C\frac{k^3}{\log k}$

База индукции: $k = 0$, тогда сумма представляет собой первое слагаемое и равна $\frac{n^3}{\log n}$. Доказано

Шаг индукции: Пусть доказано для $k < m$, докажем для $k = m$:

По предположению индукции верно:

$$\exists C > 0 : \exists N > 0 : \forall n \geq N \mapsto \sum_{k=0}^{m-1} \frac{\frac{n^3}{(\sqrt{3})^k}}{\log\left(\frac{n}{(\sqrt{3})^k}\right)} \leq C\frac{n^3}{\log n}$$

Преобразуем слагаемое с номером m и оценим его:

$$\frac{\frac{1}{(\sqrt{3})^m}}{\log\left(\frac{n}{(\sqrt{3})^m}\right)} = \frac{1}{(\sqrt{3})^m(\log n - m \cdot \log(\sqrt{3}))}$$

$$\frac{1}{(\sqrt{3})^m(\log n - m \cdot \log(\sqrt{3}))} \leq C\frac{1}{\log n}$$

$$\log n \leq C(\sqrt{3})^m(\log n - m\log\sqrt{3})$$
$$m\log\sqrt{3} \leq \Big(C\cdot(\sqrt{3})^m - 1\Big)\log n$$
$$m \leq \frac{\big(C(\sqrt{3})^m - 1\big)}{\log\sqrt{3}}\log n$$

Всегда найдутся C и N такие, что неравенство выполняется, для $\forall m < \log(n)$. Тогда слагаемое с номером m можно оценить сверху $C\frac{1}{\log n}$. Сложим это неравенство с неравенством из предположения индукции: для $_2 > C_1 + C$ и $N_2 = max(N_1, N)$ выполняется:

$$\exists C_2 > 0 : \exists N_2 > 0 : \forall n \geq N_2 \mapsto \sum_{k=0}^{m} \frac{\frac{n^3}{(\sqrt{3})^k}}{\log\left(\frac{n}{(\sqrt{3})^k}\right)} \leq C_2\frac{n^3}{\log n}$$

Следовательно, шаг индукции доказан.

Таким образом, $O(\frac{n^3}{\log n})$ оценка сверху, следовательно

Ответ: $T(n) = \Theta\left(\frac{n^3}{\log n}\right)$