AI Masters Algorithms **Name:** *Денис Грачев*

Homework 3

4 октября 2022 г.

Task 1

Запишем рекурентную формулу (для $n \le 2022$ считаем $f(n)$ константой не превышающей 2022)

$$f(n) = 3f(n/4) + c$$

$$d = \log_4(3) < \frac{1}{2}$$

Первый случай

$$c = O(n^{d-\varepsilon}) \text{ где } \varepsilon = d - \frac{1}{2}$$

Тогда из теоремы о рекурсии $a = 3, b = 4, f(n) = c$ следует что $f(n) = \Theta(n^{\log_4(3)})$

Task 2

a

$$T(n) = 36T\left(\left\lfloor \frac{n}{6} \right\rfloor\right) + n^2$$

Применим теорему о рекурсии, где $a = 36, b = 6, f(n) = n^2$,

тогда $d = \log_6(36) = 2,\ f(n) = \Theta(n^d)$.

Следовательно $T(n) = \Theta(n^2 \log(n))$

b

$$T(n) = 3T\left(\left\lfloor \frac{n}{3} \right\rfloor\right) + n^2$$

Применим теорему о рекурсии, где $a = 3, b = 3, f(n) = n^2$,

тогда $d = \log_3(3) = 1,\ f(n) = n^2 = \Omega(n^{d+0.1=1.1}),\ af\left(\frac{n}{b}\right) = 3\frac{n^2}{9} = \frac{n^2}{3} \leq n^2 = f(n)$

По третьему случаю $T(n) = \Theta(f(n)) = \Theta(n^2)$

c

$$T(n) = 4T\left(\left\lfloor\frac{n}{2}\right\rfloor\right) + \left\lfloor\frac{n}{\log(n)}\right\rfloor$$

Применим теорему о рекурсии, где $a = 4, b = 2, f(n) = \frac{n}{\log(n)}$,

тогда $d = \log_2(4) = 2,\ f(n) = \frac{n}{\log(n)} \leq n = O(n^{d-1=1})$.

По 1 случаю $T(n) = \Theta(n^d) = \Theta(n^2)$.

Task 3

1

$$T(n) = nT\left(\frac{n}{2}\right) + cn$$

Высота дерева будет $h = \log_2(n)$.

Посчитаем сколько операций на каждом уровне рекурсии.

глубина дерева	аргумент листа	количество листьев	операций в листе	операций всего
1	n	1	cn	cn
2	$\frac{n}{2}$	n	$c\frac{n}{2}$	$nc\frac{n}{2} = c\frac{n^2}{2}$
3	$\frac{n}{4}$	$\frac{n}{2}n = \frac{n^2}{2}$	$c\frac{n}{4}$	$\frac{n^2}{2}c\frac{n}{4} = c\frac{n^3}{8}$
4	$\frac{n}{8}$	$\frac{n}{4}\frac{n^2}{2} = \frac{n^3}{8}$	$c\frac{n}{8}$	$\frac{n^3}{8}c\frac{n}{8} = c\frac{n^4}{64}$
5	$\frac{n}{16}$	$\frac{n}{8}\frac{n^3}{8} = \frac{n^4}{64}$	$c\frac{n}{16}$	$\frac{n^4}{64}c\frac{n}{16} = c\frac{n^5}{1024}$
...	...	...	...	...
k + 1	$\frac{n}{2^k} = 2^{h-k}$	$\frac{n^k}{2^{\frac{k(k-1)}{2}}} = 2^{hk-\frac{k(k-1)}{2}}$	$c\frac{n}{2^k} = c2^{h-k}$	$c2^{hk-\frac{k(k-1)}{2}+h-k}$
...	...	...	...	...
h	1	$2^{h(h-1)-\frac{(h-1)(h-2)}{2}}$	c	$c2^{h(h-1)-\frac{(h-1)(h-2)}{2}}$

В формулах в табличке выше могли потеряться +- 1, в формуле ниже все работает

$$
\begin{aligned}
T(h) &= \sum_{k=1}^{h} c2^{h(k-1)-\frac{(k-1)(k-2)}{2}+h-k+1} \\
&= c\sum_{k=1}^{h} 2^{hk-\frac{k^2+k}{2}} \\
&\leq c\sum_{k=1}^{h} 2^{hk} \\
&= c\left(\frac{2^{h^2+h}-2^h}{2^h-1}\right) \\
&= c\left(\Theta\left(2^{h^2}\right)\right) \\
&= \Theta\left(2^{h^2}\right) = \Theta(2^{h^h}) = \Theta(n^{\log(n)})
\end{aligned}
$$

Таким образом $T(n) = O(n^{\log(n)})$.

Рассмотрим $hk - \frac{k^2+k}{2}$ как параболу относительно k.

Так как ее корни $0, 2h-1$, то максимальное минимальное значение достигается при $k = h$.

$$
\begin{aligned}
T(h) &= \sum_{k=1}^{h} c2^{h(k-1)-\frac{(k-1)(k-2)}{2}+h-k+1} \\
&= c\sum_{k=1}^{h} 2^{hk-\frac{k^2+k}{2}} \\
&\geq c\sum_{k=1}^{h} 2^{h^2-\frac{h^2+h}{2}} \\
&= ch2^{\frac{h^2-h}{2}} \\
&= \Theta(\log(n)n^{\frac{\log(n)-1}{2}})
\end{aligned}
$$

Таким образом $T(n) = \Omega(\log(n)n^{\frac{\log(n)-1}{2}})$.

Точнее оценить пока не удалось.

Task 4

Пусть битовая длина максимального из чисел n. Тогда с помощью алгоритма умножения Карацубы можно посчитать ab за $O(n^{log_2(3)})$. Опишем алгоритм деления в двоичной системе.

- умножим делитель на такую степень 2, чтобы длины делимого и делителя совпали
- вычтем из делимого полученное число, запомним результат

Повторять пока делимое больше делителя. Таким образом мы каждый раз отрезаем от делителя 1 бит а вычитание стоит $O(n)$. Итоговая сложность деления $O(n^2)$.

Чтобы посчитать НОД нам необходимо сделать $O(n)$ делений. Итоговая сложность $O(n^3)$.

$\operatorname{lcm}(a, b) = \frac{ab}{\gcd(a,b)}$. Битовая длина $ab < 2n$, соответственно сложность деления $O((2n)^2) = O(4n^2) = O(n^2)$.

Итоговая сложность $O(n^{\log_2(3)} + n^3 + n^2) = O(n^3)$.

Task 5

Для того чтобы посчитать количество инверсий, заведем счетчик который будет обновлять во время слияния.

При слиянии подмассивов a, b (a - левый, b - правый), если $a_i > b_j$ это значит, что $\forall k \leq j : a_i > b_k$ так как они отсортированы, при этом изначально b_k были правее a_i, соотвественно у нас было j инверсий, добавим к счетчику j.

Почему все инверсии посчитаются и ровно 1 раз. Когда мы переставляем a_i элемент перед b_j то устраняем ровно j инверсий.

В отсортированном массиве инверсий нет. Значит все инверсии посчитаны.

Task 6

$ab = \frac{(a+b)^2 - a^2 - b^2}{2}$.

Для вычисления этого числа необходимо произвести

- Сложение $O(n)$
- Возведение в квадрат $O(n)$
- Возведение в квадрат $O(n)$
- Возведение в квадрат $O(n)$
- Вычитание $O(n)$
- Вычитание $O(n)$
- Деление на 2 $O(1)$

Итоговая сложность $O(n)$.

Task 7

a

$T(n) = T(\alpha n) + T((1-\alpha)n) + \Theta(n)$

Количество операций на каждом уровне $\Theta(n)$.

Минимальная глубина дерева $-\log_\alpha(n)$, максимальная $-\log_{1-\alpha}(n)$.

Значит количество операций зажато между $-\log_\alpha(n)n \le T(n) \le -\log_{1-\alpha}(n)n$.

Значит $T(n) = \Theta(n\log(n))$.

b

$T(n) = T(\frac{n}{2}) + 2T(\frac{n}{4}) + \Theta(n)$

Количество оперций на каждом уровне рекурсии n.

глубина рекурсии между $\log_4(n)$ и $\log_2(n)$.

Таким образом $T(n) = \Theta(n\log(n))$.

c

$T(n) = 27T(\frac{n}{3}) + \frac{n^3}{\log^2(n)}$

рассмотрим количество операций на следующем уровне рекурсии $27\frac{n^3/3^3}{\log^2(n/3)} = \frac{n^3}{\log^2(n/3)}$.

$$\begin{aligned}
T(n) &= \sum_{k=0}^{\log_3(n)} \frac{n^3}{\log^2(n/3^k)} \\
&= n^3 \sum_{k=0}^{\log_3(n)} \frac{1}{\log^2(n/3^k)} \\
&= n^3 \sum_{k=0}^{\log_3(n)} \frac{1}{(\log(n) - k\log(3))^2} = \Theta\left(\frac{n^3}{\log^2(n)}\right)
\end{aligned}$$

Task 8

Пусть n нечетное. Существует три варианта расположения $[a_i, a_{i+1} \dots a_j]$:

- $j < \frac{n+1}{2}$ - полностью лежит до середины.
- $i > \frac{n+1}{2}$ - полностью лежит после середины.
- $i \leq \frac{n+1}{2} \leq j$ - пересекает середину.

Найдем максимумы 1 и 2 случая с помощью рекурсивного вызова функции для левой и правой половинок.

Решим задачу за $\Theta(n)$ для 3 случая.

Заведем два индекса $\text{lhs} = \frac{n+1}{2} = \text{rhs}$ и два минимума $\min_e = a_{\frac{n+1}{2}} = \max_s$. Будем изменять их по следующему принципу:

Если $a_{\text{lhs}-1} > a_{\text{rhs}+1}$:

$\text{lhs} -= 1$

$\min_e = \min(\min_e, a_{\text{lhs}})$

Иначе:

$\text{rhs} += 1$

$\min_e = \min(\min_e, a_{\text{rhs}})$

$\max_s = \max(\max_s, (rhs - lhs + 1)\min_e)$

Если какой то индекс доходит до края, перестаем его изменять.

Такое действие занимает $\Theta(n)$ по сложности, так как каждый раз какой то из индексов отъезжает от середины и расстояние между ними не может быть больше n. Докажем, что в конце переменная $\max_s$ будет хранить максимальное возможное искомое значение среди подмассивов проходящих через середину. Предположим противное, тогда есть подмассив края которого $\text{lhs}_w, \text{rhs}_w$ и минимальный элемент $\min_w$ для которого искомое значение больше того что мы получили.

Рассмотрим последний момент когда наш расширяющийся подмассив был внутри $\text{lhs}_w, \text{rhs}_w$.

Они не могли идеально совпадать, так как тогда наилучшее значение было бы записано в $\max_s$ и никогда более бы не изменилось.

Значит один из краев у них совпадает и в следующий момент этот край выйдет за пределы наилучшего подмассива.

Без ограничения общности можем считать что это левый край, то есть $\text{lhs} = \text{lhs}_w$, $\text{rhs} < \text{rhs}_w$ и в следующий момент мы сделаем $\text{lhs} -= 1$, следовательно $a_{\text{lhs}-1} > a_{\text{rhs}+1}$.

$\min_w \leq a_{\text{rhs}+1}$ так как $a_{\text{rhs}+1}$ находится внутри $\text{lhs}_w, \text{rhs}_2$.

$\min_w \leq a_{\text{rhs}+1} \leq a_{\text{lhs}-1}$.

Значит размер нашего идеального подмассива можно увеличить и значение не наилучшее, противоречье.

Пусть сложность это $T(n) = 2T(n/2) + \Theta(n)$.

Аналогично сортировке из лекции $T(n) = \Theta(n \log(n))$

Для того чтобы найти не только наилучшее значение но и индексы подмассива, будем вместе с наилучшим значением хранить науилчшие lhs и rhs. Случай когда размер переданного отрекзка равен 1 тривиален.