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

HomeWork 1

18 сентября 2022 г.

Задача 1

а

Возьмем $C = 1, N = 2$.

Тогда $\forall n > N : \log(n) > \log(N) = \log(2) = 1$.

Следовательно $\exists C = 1, N = 2 : \forall n > N, n = n\log(N) \leq Cn\log(n)$.

Таким образом $n = O(n\log(n))$.

б

$\exists\varepsilon > 0 : n\log(n) = \Omega\left(n^{1+\varepsilon}\right) \Leftrightarrow$

$\exists\varepsilon > 0 : n^{1+\varepsilon} = O\left(n\log(n)\right) \Leftrightarrow$

$\exists\varepsilon, C, N > 0 : \forall n > N, n^{1+\varepsilon} \leq Cn\log(n)$

Пусть существуют такие ε, C, N для которых это верно.

Возьмем $\log$ от обоих частей неравенства $n^{1+\varepsilon} \leq Cn\log(n)$, тогда

$\log\left(n^{1+\varepsilon}\right) \leq \log\left(Cn\log(n)\right) \Leftrightarrow$

$(1+\varepsilon)\log(n) \leq \log(C) + \log(n) + \log(\log(n)) \Leftrightarrow$

$\varepsilon\log(n) \leq \log(C) + \log(\log(n)) \Leftrightarrow$

$\varepsilon\log(n) - \log(\log(n)) \leq \log(C)$

Возьмем $n = 2^{2^k}$. Тогда $\log(n) = 2^k, \log(\log(n)) = k$, следовательно

$\varepsilon 2^k - k \leq \log(C)$

Заметим что при увеличении k на 1 левая часть увеличивается на $(\varepsilon 2^{k+1} - (k+1)) - (\varepsilon 2^k - k) = \varepsilon 2^k - 1$. Если взять $k = \log(\frac{2}{\varepsilon})$, то после этого при увеличении k на 1, левая часть будет увеличиваться хотя бы на $\varepsilon\frac{2}{\varepsilon} - 1 = 1$. Таким образом при увеличении k на 1 в какой то момент неравенство перестанет быть верным так как справа стоит константа, противоречье.

Задача 2

1.а

Возьмем $f(n) = n\log(n), g(n) = 1$.

Тогда т.к.

$\forall n > 1, \log(n) < n$ следовательно $\forall n > 1, n\log(n) \le n^2$ следовательно $f(n) = n\log(n) = O(n^2)$.

$g(n) = 1 = \Omega(1)$ т.к. $\forall n : 1 \le 1$.

$g(n) = 1 = O(n)$ т.к. $\forall n : g(n) = 1 \le n$.

Тогда $h(n) = n\log(n)$. Очевидно что $h(n) = n\log(n) = \Theta(n\log(n))$.

1.б

Пусть возможно.

$g(n) = \Omega(1) \Leftrightarrow 1 = O(n) \Leftrightarrow \exists C_1, N_1 : \forall n > N_1, c_1 \le g(n)$.

$h(n) = \Theta(n^3) \Rightarrow \exists C_2, N_2 : \forall n > N_2, C_2 n^3 \le h(n)$

Тогда $\forall n > \max(N_1, N_2), C_1 C_2 n^3 \le g(n)h(n) = f(n) \Rightarrow f(n) = \Omega(n^3)$.

Противоречье, так как по условию $f(n) = O(n^2)$, С одной стороны $f(n) \le c_1 n^2$, с другой $f(n) \ge c_2 n^3$.

2.а

Возьмем $f(n) = 0, g(n) = 1$.

$\forall n, f(n) = 0 \le n^2 \Rightarrow f(n) = O(n^2)$.

$g(n) = 1 = \Omega(1)$ т.к. $\forall n : 1 \leq 1$.

$g(n) = 1 = O(n)$ т.к. $\forall n : g(n) = 1 \leq n$.

тогда $h(n) = \frac{f(n)}{g(n)} = 0$.

$\forall n, h(n) = 0 \geq 0 \Rightarrow h(n) = \Omega(0)$.

Более нижней нижней оценки чем $\Omega(0)$ не существует она достигается при приведенных f, g.

2.б

$f(n) = O(n^2) \Rightarrow \forall n > N_1, f(n) \leq c_1 n^2$

$g(n) = \Omega(1) \Rightarrow \forall n > N_2, g(n) \geq c_2$

Следовательно $\forall n > \max(N_1, N_2), h(n) = \frac{f(n)}{g(n)} \leq \frac{c_1 n^2}{g(n)} \leq \frac{c_1}{c_2} n^2 \Rightarrow h(n) = O(n^2)$.

$O(n^2)$ верхняя оценка $h(n)$ достигается при $f(n) = n^2, g(n) = 1$.

Задача 3

```
for (bound = 1; bound < n; bound *= 2 ) { Цикл 1
   for (i = 0; i < bound; i += 1) { Цикл 2
      for (j = 0; j < n; j += 2) Цикл 3
         печать (“алгоритм”)
      for (j = 1; j < n; j *= 2) Цикл 4
         печать (“алгоритм”)
   }
}
```

Цикл 3 имеет $\frac{n}{2} = \Theta(n)$ итераций, *Цикл 4* имеет $\log(n) = \Theta(\log(n))$ итераций.

$\Theta(n) + \Theta(\log(n)) = \Theta(n)$, соответсвенно то что внутри *Цикл 2* имеет сложность $\Theta(n)$.

Рассмотрим *Цикл 1* и *Цикл 2*, количество итераций в *Цикл 2* - bound, bound - удваивается кажлый раз, пусть в было k итераций в *Цикл 1*.

Тогда количетсво итераций *Цикл 1* и *Цикл 2* это $1 + 2 + \ldots 2^k = 2^{k+1} - 1$. Из условия цикла очевидно что $\frac{n}{2} < 2^k \leq n$, тогда $n \leq 2^{k+1} - 1 \leq 2n$.

Следовательно количество итераций *Цикл 1* и *Цикл 2* это $\Theta(n)$.

Итого получается что количество слов алгоритм $\Theta(n)\Theta(n) = \Theta(n^2)$.

Задача

Обозначим массивы x_1, x_2, x_3 и их длины n_1, n_2, n_3 соответсвенно.

Алгоритм.

Заведем $c = i_1 = i_2 = i_3 = 0$.

Пока $(i_1 < n_1 | i_2 < n_2 | i_3 < n_3)$ {
    $c+ = 1$
    $m = min(x_k[i_k] | k \in (1, 2, 3), i_k < n_k)$
    Если $i_1 < n_1$ & $x_1[i_1] == m$
        $i_1+ = 1$
    Если $i_2 < n_2$ & $x_2[i_2] == m$
        $i_2+ = 1$
    Если $i_3 < n_3$ & $x_3[i_3] == m$
        $i_3+ = 1$

}
Печать c

Доказательство корректности.

m увеличивается на каждом шаге цикла.

Так как массивы отсортированы, то элементы массивов которые были равны m на предыдущем шаге увеличатся, элементы массивов которые не были равны m на предыдущем шаге больше m, таким образом m точно увеличится. Таким образом никакое значение m не примет дважды.

m примет все возможные значения из данных массивов.

Легко заметить, что i_k увеличивается и проходит все значения от 0 до n_k. Итерация на которой происходит увеличение i_k соответсвует ситуации когда $m = x_k[i_k]$. Значит m проходит все значения массивов. Таким образом m проходит все значения массивов и никакое значение не проходит дважды. Значит c - счетчик уникальных значений m считает количество различных чисел. Так как на каждой итерации хотя бы один индекс увеличивается, то количество итераций $O(n_1 + n_2 + n_3)$.

Задача 6

Обозначим

$A(n) = \sum_{i=1}^{n} a_i$

$B(n) = \sum_{i=1}^{n} b_i$

$F(n) = \sum_{i,j \leq n, i \neq j} a_i b_j$

Тогда

$F(n+1) = \sum_{i,j \leq n+1, i \neq j} a_i b_j = \sum_{i,j \leq n+1} a_i b_j - \sum_{i,j \leq n+1, i=j} a_i b_j = \sum_{i,j \leq n} a_i b_j - \sum_{i,j \leq, i=j} a_i b_j + \sum_{i \leq n} a_i b_{n+1} + \sum_{j \leq n} a_{n+1} b_j - a_{n+1} b_{n+1} = F(n) + A(n) b_{n+1} + B(n) a_{n+1} - a_{n+1} b_{n+1}$.

Алгоритм.

$F(0) = A(0) = B(0) = 0$

$A(n+1) = A(n) + a_{n+1}$

$B(n+1) = B(n) + b_{n+1}$

$F(n+1) = F(n) + A(n) b_{n+1} + B(n) a_{n+1} - a_{n+1} b_{n+1}$

Будем хранить $F(n), A(n), B(n)$ и обновлять их при поступлении новых a_{n+1}, b_{n+1}.

Задача 7

заведем массив l размера n (индексация с 1).

$l[i]$ - длина наидленнейщей возрастающей подпоследовательности среди подпоследовательности $a_i, a_{i+1}, \ldots a_n$, начинающаяся с a_i.

Тогда $l[n] = 1$.

$l[i] = max(1, l[j] + 1 | a_j > a_i)$. *занимает O(n).* Таким образом заполняя справа налево мы получим длины наиболее длинных возрастающих подпоследовательностей начинающихся с i-ого элемента за $O(n) * O(n) = O(n^2)$.

Корректность этого заполнения: Так как мы ищем наиболее длинную возрастающую подпоследовательность начиная с a_i, то нам подходят элементы стоящие после a_i такие что $a_i < a_j$. Так как для них задача уже решена, то длина получившейся последовательности будет $l[j] + 1$ и мы перебираем все подходяшие подпоследовательности поэтому найдем длину наидленнейшей.

После этого найдем $i : l[i] = \max(l)$. Это по определению индекс первого элемента строго возрастающей подпоследовательности которая имеет максимальную длину. После этого найдем если $l[i] = 1$, то это последний элемент подпоследовательности, иначе найдем $j : j > i, l[i] = l[j] + 1, a_j > a_i$. Такой элемент найдется так как мы так строили $l[i]$ и это будет следующий элемент последовательности. Приравняем $i = j$ и повторим поиск следующего элемента. Поиск элемента занимает $O(n)$, всего элементов $O(n)$, значит сложность $O(n^2)$. На каждой итерации мы добавляем 1 корректный элемент к искомой последовательности, элемент всегда находится из построения. Итоговая сложность $O(n^2)$.