Филиппова Анастасия

Домашнее задание 1

(1) (a) $n = O(n \cdot \log n)$

По определению верхней грани:

$$\exists\ c > 0, n_0 \in N : \forall\ n \geqslant n_0 \hookrightarrow n \leqslant c \cdot \log n$$

Возьмем $c = 1 : 1 \leqslant \log n$

Очевидно, что при достаточно больших n неравенство верно.

(b) $\exists\ \varepsilon > 0 : n \cdot \log n = \Omega(n^{1+\varepsilon})$

По определению верхней грани:

$$\exists\ c > 0,\ n_0 \in N : \forall\ n \geqslant n_0 \hookrightarrow n \cdot \log n \geqslant c \cdot n^{1+\varepsilon}$$

$\exists\ c > 0,\ n_0 \in N : \forall\ n \geqslant n_0 \hookrightarrow \log n \geqslant c \cdot n^{\varepsilon}$

Найдем предел отношения функций по правилу Лопяталя:

$\lim_{n\to\infty} \left(\frac{\log n}{n^{\varepsilon}}\right) = \lim_{n\to\infty} \left(\frac{1}{n^{\varepsilon} \cdot \ln a \cdot \varepsilon}\right) = 0$

Следовательно, $\log n = o(n^{\varepsilon})$

Тогда по опредению:

$$\forall\ c > 0 : \exists\ n_0 \in N : \forall\ n \geqslant n_0 \hookrightarrow \log n \leqslant c \cdot n^{\varepsilon}$$

Очевидно, что мы получили противоречие.

Ответ: утверждение неверно.

(2) $f(n) = O(n^2)$, $g(n) = O(n)$, $g(n) = \Omega(1)$

(a) Возможно ли: $h(n) = \Theta(n \cdot \log n)$?

Пример: $f(n) = n \cdot \log n$, $g(n) = 1$

$h(n) = n \cdot \log n$, тогда очевидно, что $h(n) = \Theta(n \cdot \log n)$

Ответ: да, возможно.

(b) Возможно ли: $h(n) = \Theta(n^3)$?

По определениям верхней и нижней граней:

$$\exists\ c_1 > 0,\ n_1 \in N : \forall\ n \geqslant n_1 \hookrightarrow f(n) \leqslant c_1 \cdot n^2$$

$$\exists\ c_2 > 0,\ n_2 \in N : \forall\ n \geqslant n_2 \hookrightarrow g(n) \leqslant c_2 \cdot n$$

$$\exists\ c_3 > 0,\ n_3 \in N : \forall\ n \geqslant n_3 \hookrightarrow g(n) \geqslant c_3$$

Тогда $h(n) \leqslant \frac{c_1 \cdot n^2}{c_3}$, т.е:

$$\exists\ c_4 = \frac{c_1}{c_3},\ n_4 = max(n_1, n_3) : \forall\ n \geqslant n_4 \hookrightarrow h(n) \leqslant c_4 \cdot n^2$$

Из этого следует, что $h(n) = O(n^2)$

Но т.к. $h(n) = \Theta(n^3)$, то:

$$h(n) = \Omega(n^3)$$

Получаем противоречие.

Ответ: нет.

(3) (a) Псевдо - код:

s - текущая сумма переменных
n - длинна последовательности
i - счетчик цикла
a - текущая переменная

```
считываем n
for i := 0 to n − 1 do
    считываем a;
    s := s + a;
end
answer: k;
```

(b) Псевдо - код:

n - длинна последовательности
k - счетчик
a - текущая переменная
max - текущий максимальный элемент последовательности

```
считываем n;
k := 1;
считываем a;
max := a;
for i := 1 to n − 1 do
    считываем a;
    if (a = max) then
        k := k + 1;
    end
    if (a > max) then
        max := a;
        k := 1;
    end
end
answer: k;
```

(c) Псевдо - код:
n - длинна последовательности
a' - последний считанный элемент последовательности
a - текущая переменная
max - максимальное число идущих подряд элементов последовательности
l - число идущих подряд элементов последовательности в конце уже считанной подпоследовательности

```
считываем n;
считываем a';
max := 1;
l := 1;
for i := 1 to n − 1 do
|   считываем a;
|   if (a' = a) then
|   |   l := l + 1;
|   end
|   else
|   |   max := max(max, l);
|   |   l := 1;
|   end
|   a' := a
end
max := max(max, l);
answer: max;
```

(4) Пусть даны 3 массива a, b, c размера n, k, l соответственно.

Псевдо - код:

```
считываем n; считываем k; считываем l;
a - последняя считанная переменная из массива a;
b - последняя считанная переменная из массива b; c - последняя считанная переменная из массива c;
n1 := n; k1 := k; l1 := l;
h - счетчик;
считываем a, b, c;
h := 0;
while (n1 > 0) или (k1 > 0) или (l1 > 0) do
    if (a < b) then
        if (a < c) then
            h := h + 1;
            if (n1 > 0) then
                считываем a;
                n1 := n1 − 1;
            end
        end
        if (a > c) then
            h := h + 1;
            if (l1 > 0) then
                считываем c;
                l1 := l1 − 1;
            end
        end
        if (a = c) then
            if (n1 > 0) then
                считываем a;
                n1 := n1 − 1;
            end
        end
        if (l1 > 0) then
            считываем c;
            l1 := l1 − 1;
        end
    end
    if (a > b) then
        if (a > c) then
            h := h + 1;
            if (b > c) then
            end
            if (l1 > 0) then
                считываем c;
                l1 := l1 − 1;
            end
            if (b < c) then
                if (l1 > 0) then
                    считываем b;
                end
                k1 := k1 − 1;
            end
            if (b = c) then
                if (k1 > 0) then
                    считываем b;
                    k1 := k1 − 1;
                end
                if (l1 > 0) then
                    считываем c;
                    l1 := l1 − 1;
                end
            end
        end
        if (a < c) then
            h := h + 1;
            if (k1 > 0) then
                считываем b;
                k1 := k1 − 1;
            end
        end
        if (a = c) then
            if (n1 > 0) then
                считываем a;
                n1 := n1 − 1;
            end
            if (l1 > 0) then
                считываем c;
                l1 := l1 − 1;
            end
        end
    end
    if (a = b) then
        if (a > c) then
            if (l1 > 0) then
                считываем c;
                l1 := l1 − 1;
            end
            if (a < c) then
                if (n1 > 0) then
                    считываем a;
                    n1 := n1 − 1;
                end
                if (k1 > 0) then
                    считываем b;
                    k1 := k1 − 1;
                end
            end
        end
    end
end
answer: h;
```

Данный алгоритм корректен, так как, проходясь таким образом по массиву, мы не пропустим ни одного элемента и очевидно, что полученное h и есть искомое количество элементов.
Асимптотика: Очевидно, что для реализации данного алгоритма требуется $O(1)$ битов памяти, так как достаточно всего 7 переменных.
Время работы алгоритма есть $O(n+k+l)$, то есть алгоритм линейный.

(5) Создадим динамический массив l, в который на i место будем записывать максимальную длину возрастающей подпоследовательности, заканчивающейся на $a[i]$ и массив b, в который на i место будем записывать то j, на котором достигается максимум.
Псевдо - код:

```
считываем n;
max - длина максимальной возрастающей подпоследовательности
for i := 0 to n − 1 do
    считываем a[i];
    l[i] := 1;
    for j := i − 1 down to 0 do
        считываем a[j];
        if (a[i] > a[j]) then
            if (l[j] + 1 > l[i]) then
                l[i] := l[j] + 1;
                b[i] := j;
            end
        end
    end
    считываем l[0]
    max := l[0];
    k := 1;
    for i := 1 to n − 1 do
        считываем l[i];
        if (l[i] > max) then
            max := l[i];
            k := i;
        end
    end
    зная элемент a[k], на котором заканчивается самая длиная возрастающая подпоследовательность, и
      зная элементы массива нетрудно востановить нужную подпоследовательность.
end
```

Данный алгоритм корректен, так как позваляет найти длину самой длинной возрастающей подпоследовательности (если несколько подпоследовательностей - одну из них).
И при помощи массива b восстанавливает подпоследовательность.
Асимптотика: Очевидно, что для реализации данного алгоритма требуется $O(n)$ битов памяти.
Время работы алгоритма в худшем случае на массиве размера n есть $O(n^2)$, так как есть вложенный цикл.

(6) Кандидатом будем называть элемент, который может быть искомым. Будем рассматривать 3 возможных случая: считанный элемент равен кандидату, не равен и случай, при котором кандидат будет не определен.

k - кандидат

h - счетчик

a - текущая переменная

Псевдо - код:

считываем n;
считываем первую переменную a;
$k := a$;
$h := 1$;
считываем a;
if *($k = a$)* **then**
    $h := h + 1$;
end
if *(k - не определен)* **then**
    $h := 1$;
    $k := a$
end
if *($k <> a$)* **then**
    if *($h = 1$)* **then**
        k не определен;
        $h := 1$;
    end
    else
        $h := h - 1$;
    end
end
answer : k .

Докажем корректность данного алгоритма:

Условие того, что в массиве больше, чем $\frac{n}{2}$ одинаковых элементов, гарантирует, что последним останется именно искомый элемент.

Действительно: представим, что элементы массива можно разделить на два множества: множество искомых элементов, мощность которого больше $\frac{n}{2}$ и множество остальных.

Данный алгоритм "вычеркивает"элементы массива попарно:один из первого множество, второй - из второго.

В итоге, так как мощность первого больше $\frac{n}{2}$, кандидат и будет искомым элементом.

Найдем асимптотику:

Для реализации данного алгоритма требуется $O(1)$ битов памяти, так как не зависит от длины последовательности. Алгоритм выполняется за $O(n)$ времени, так как мы один раз проходимся по последовательности.

Задача с семинара:
n - длина первой последовательности
k - длина второй последовательности
Псевдо - код:

```
считываем n;
считываем k;
n_1 := n;
k_1 := k;
while (n_1 > 0) and (k_1 > 0) do
    if (a[n_1] = b[k_1]) then
        k_1 := k_1 - 1;
        n_1 := n_1 - 1;
    end
    else
        n_1 := n_1 - 1;
    end
end
if (k_1 = 0) then
    answer : да;
end
else
    answer : нет;
end
```

Асимптотика: Очевидно, что для реализации данного алгоритма требуется $O(1)$ битов памяти.
Алгоритм выполняется за $O(n + k)$ времени.
Корректность:
Очевидно, что мы дойдём до конца последовательности b только в том случае, если все элементы последовательности содержаться в последовательности a, т.е., что b - подпоследовательность a.