ДЗ по алгоритмам. Кривенко Андрей М3107

September 26, 2021

1.
```
for (int i = 0; i < 2n; ++i)
  for (int j = 0; j < n; ++j)
    for (int k = 0; k * k < 5n; ++k)
```

2. ,

```
i=0
while i < n:           //n + 1
  if i < n / 2:        //n
    j = 0              //n / 2
    while j * j < i:   //sum(from i = 0 to n / 2 of sqrt(i))
      j++              //sum(from i = 0 to n / 2 of sqrt(i))
  else:
    j = i / 3          //n / 2
    while j / 2 < i:   //17(n / 2), j / 2 < i <=> j < 2i => требуется
j += i / 10 20 раз, но j изначально j = i / 3
      j += i / 10      //17(n / 2)
  i += 1               //n
```

$$T(n) = (n+1)+n+(n/2)+\sum_{i=0}^{n/2}\sqrt{i}+\sum_{i=0}^{n/2}\sqrt{i}+(n/2)+17(n/2)+17(n/2)+n =$$

$$21n + 2\sum_{i=0}^{n/2}\sqrt{i} + 1$$

$O(n) = n\sqrt{n}$, так как внешний while запускается n раз, а вложенный while n раз

3. Докажите, что $\forall x > 0 : log_x n = \theta(\ln n)$

Proof.

$$\forall x > 0 : log_x n = \theta(\ln n) \iff \exists c_1, c_2, n_0 :$$

$$\begin{cases} c_1 \ln n \le \log_x n \le c_2 \ln n \\ n \ge n_0 \\ x > 0, x \ne 1 \end{cases}$$

(a) При $n = 1 : c_1 \ln n = \log_x n = c_2 \ln n = 0$

(b) Логарифмическая функция монотонно растет

(c) $(c_1 \ln n)' = \frac{c_1}{n},\ (c_2 \ln n)' = \frac{c_2}{n},\ (\log_x n)' = \frac{1}{n \ln x}$
Следовательно, мы можем выбирать $c_1,\ c_2$ так, чтобы скорость роста $c_1 \ln n$ была ниже скорости роста $\log_x n$, а скорость роста $c_2 \ln n$ была выше скорости роста $\log_x n$:
При $n_0 = 1$ и $n \in \mathbb{N} \land n >= n_0$: $\frac{c_1}{n} \le \frac{1}{n \ln x} \le \frac{c_2}{n} \iff c_1 \le \frac{1}{\ln x} \le c_2$
$\forall a \in \mathbb{R}\ \exists\, b, c \in \mathbb{R} : a < b, a > c \Longrightarrow \exists\, c_1, c_2 \in \mathbb{R} : c_1 \le \frac{1}{\ln x} \le c_2$

$\square$

4. Псевдокод алгоритма

```
int n, m;
int arr1[n];
int arr2[m];

int i = 0, j = 0;

bool isFound = false;

while (i < n && j < m) {
  if (arr1[i] < arr2[j]) {
    ++i;
  } else if (arr1[i] > arr2[j]) {
    ++j;
  } else {
    isFound = true;
    break;
  }
}
```

5. Инверсии

```
for (int i = 1; i < n; ++i) {
  int key = a[i];
  int j = i - 1;
  while (j > -1 && key < a[j]) {
    a[j + 1] = a[j];
    --j;
  }
  a[j + 1] = key;
}
```

for работает за n (строка 1).
Условие цикла while выполняется только тогда, когда элемент с большим индексом меньше элемента с меньшим индексом ($i < j \wedge a[j] < a[i]$).
То есть, оно выполняется столько раз, сколько существует пар (i, j) : $i < j \wedge a_i > a_j$, то есть k раз. Следовательно, сортировка вставками работает за O(n + k)

6. (a) Сортировка вставками устойчива, так как при ее использовании не изменяется относительный порядок элементов. Это происходит из-за того, что элементы добавляются в отсортированную часть массива последовательно справа налево (при сортировке в порядке неубывания), а также из-за того, что сравнение нового элемента и элемента отсортированной части массива строгое.

   (b) Сортировка пузырьком будет неустойчивой, если использовать нестрогое сравнение при сравнении элементов:

```
for (int i = 0; i < n - 1; ++i) {
  for (int j = 0; j < n - i - 1; ++j) {
    if (a[j] >= a[j + 1]) {
      swap(a[j], a[j + 1]);
    }
  }
}
```

7. $T(n) = \lfloor \log_3 n \rfloor + 2$, так как функция запускается столько раз, сколько n можно поделить на 3, а так же когда функция принимает n = 1 и n = 0
При n = 0 $T(n) = 1$
$\theta(n) = \log_3 n$, так как $\exists c_1, c_2 : c_1 \log_3 n \leq \log_3 n \leq c_2 \log_3 n$