АиСД 2023. Первый семестр

Задания на практики M3138-M3139

⟨Версия от 1 октября 2023 г.⟩

Темы

Неделя 1. Асимптотики

Определения, которые были на лекции:

- $f(n) = \mathcal{O}(g(n))$, если существует константа c, что, начиная с некоторого места, f ограничена сверху $c \cdot g$, или же

$$\exists c > 0, n_0 > 0 : \ \forall n > n_0 : \ f(n) \leqslant c \cdot g(n)$$

- $f(n) = o(g(n))$, если для любой константы c, начиная с некоторого места, f ограничена сверху $c \cdot g$, или же

$$\forall c > 0 : \ \exists n_0 > 0 : \ \forall n > n_0 : \ f(n) < c \cdot g(n)$$

  Альтернативное определение:

$$\frac{f(n)}{g(n)} \underset{n \to +\infty}{\longrightarrow} 0$$

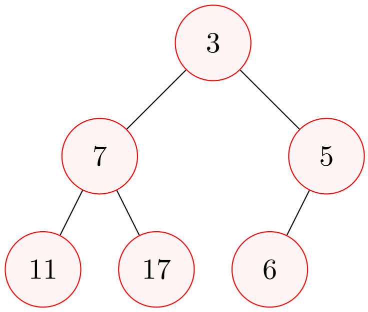
Стоит отметить, что знак '=' в данном случае не очень корректен. Формально, $\mathcal{O}(n)$ – это *множество функций*, ограниченных сверху линейной. Поэтому корректно было бы писать $f(n) \in \mathcal{O}(g(n))$, но мы будем пользоваться традиционно принятым '=', подразумевая это.

Неделя 2. Стек, очередь, дек

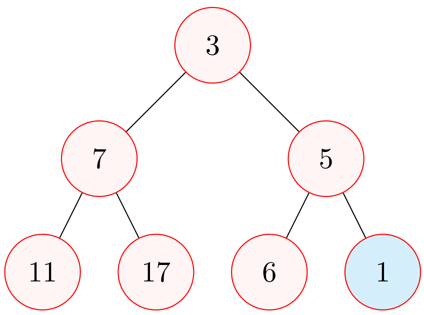
Неделя 3. Куча. Сортировка слиянием

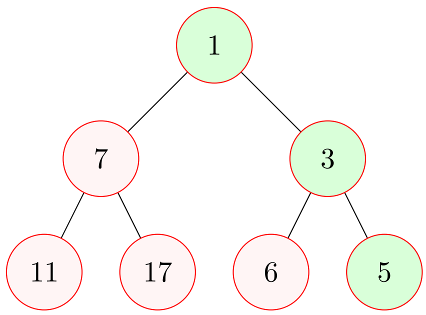
Двоичная куча

- Свойства двоичной кучи:
  - Подвешенное двоичное дерево
  - На i-м слое 2^i вершин (кроме последнего слоя)
  - Последний слой заполнен слева-направо «без пробелов»
  - Верно одно из двух:
    - ⋆ на всех ребрах написано отношение '$\leqslant$' (*куча по минимуму*)
    - ⋆ на всех ребрах написано отношение '$\geqslant$' (*куча по максимуму*)

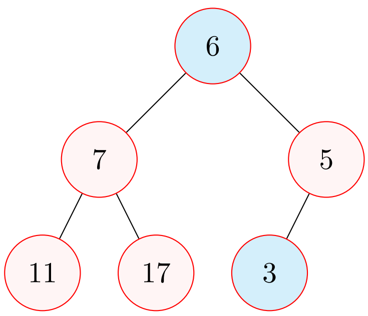


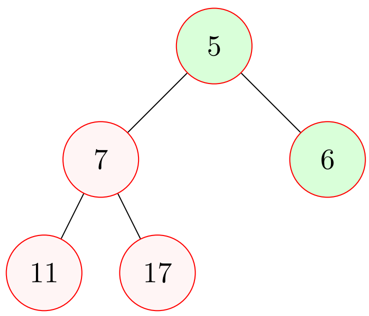
- Операции с двоичной кучей (*по минимуму*):
  - `siftUp`(x) — просеять элемент x вверх по дереву
  - `siftDown`(x) — просеять элемент x вниз по дереву
  - `add`(x) — добавить элемент x в кучу (добавляем x последней вершиной на последнем слое, а затем вызываем `siftUp`(x))





  - `extractMin()` — извлечь текущий минимальный элемент из кучи (свапаем корень с последней вершиной на последнем слое, удаляем последнюю вершину на последнем слое, а затем вызываем `siftDown(root)`)





Неделя 4. Быстрая сортировка, k-я порядковая статистика

Матчасть

Def. Пусть X — некоторая дискретная *случайная величина*, которая принимает **одно** из не более чем счётного множества значений $\{x_1, x_2, x_3, \ldots\}$.

Def. Вероятность, с которой случайная величина принимает значение x_i обозначим за p_i. Так как случайная величина гарантированно примет одно из своих значений, то:

$$\sum_{i=1}^{\infty} p_i = 1, \quad p_i = \mathbb{P}(X == x_i)$$

Def. *Математическим ожиданием* $\mathbb{E}[X]$ случайной величины X называется взвешенное среднее значение:

$$\mathbb{E}[X] = \sum_{i=1}^{\infty} p_i \cdot x_i$$

Def. Математическое ожидание *линейно*, то есть:

$$\mathbb{E}[a \cdot X + b \cdot Y] = a \cdot \mathbb{E}[X] + b \cdot \mathbb{E}[Y]$$

где X, Y — случайные величины; $a, b \in \mathbb{R}$ — произвольные константы.

Быстрая сортировка

```python
def quickSort(arr):
    n = len(arr)
    if n <= 1:
        return arr
    pivot = arr[random.randint(0, n - 1)] # inclusively
    left = [x for x in arr if x < pivot]
    right = [x for x in arr if x >= pivot]
    return quickSort(left) + quickSort(right)
```

Listing 1: **Simple QuickSort**

Так как время работы рандомизированного алгоритма — случайная величина, определим его математическое ожидание:

$$T^*(n) = \mathbb{E}[T(n)]$$

Пусть нам необходимо отсортировать некоторый массив, все элементы которого различны. На листинге 1 приведён алгоритм быстрой сортировки с лекции: массив длины n делится на две части, с длинами k и $(n-k)$ соответственно. Длина левой части k зависит от значения опорного элемента, который был выбран случайным образом. По определению, математическое ожидание времени работы такого алгоритма равно:

$$T^*(n) = \sum_{k=0}^{n-1} \frac{1}{n} \cdot (T^*(k) + T^*(n-k) + n)$$

> На лекции мы ограничили математическое ожидание времени работы алгоритма быстрой сортировки:
>
> $$T^*(n) \leqslant \frac{1}{3} \cdot \left(T^*\left(\frac{n}{3}\right) + T^*\left(\frac{2n}{3}\right) + n\right) + \frac{2}{3} \cdot \left(T^*(0) + T^*(n) + n\right)$$
>
> полагая $T^*(0) = 0$, упростили:
>
> $$T^*(n) \leqslant T^*\left(\frac{n}{3}\right) + T^*\left(\frac{2n}{3}\right) + 3n \tag{1}$$
>
> Аналогично, мы ограничили математическое ожидание времени работы алгоритма поиска k-й порядковой статистики:
>
> $$T^*(n) \leqslant \frac{1}{3} \cdot \left(T^*\left(\frac{2n}{3}\right) + n\right) + \frac{2}{3} \cdot \left(T^*(n) + n\right)$$
>
> и упростили:
>
> $$T^*(n) \leqslant T^*\left(\frac{2n}{3}\right) + 3n \tag{2}$$

Быстрая сортировка in-place

```
fun quickSort(arr: Int[], l: Int, r: Int) {
    if (l >= r)
        return
    val pivot: Int = arr[random(l, r)] // inclusively
    var i: Int = l
    var j: Int = r
    while (i <= j) {
        while (arr[i] < pivot)
            i++
        while (pivot < arr[j])
            j--
        if (i <= j) {
            swap(arr[i], arr[j])
            i++
            j--
        }
    }
    quickSort(arr, l, j)
    quickSort(arr, i, r)
}
```

Listing 2: **In-place QuickSort**