Лекция по АиСД №2

Попов Никита

27 января 2016 г.

Двоичное дерево поиска

Так как курс всё же называется «Алгоритмы и *структуры данных*», рассмотрим такую структуру данных, как **двоичное дерево поиска**.

Во-первых, это двоичное дерево — есть узлы и связи между ними, при этом у каждого узла не более двух детей. В вершинах дерева — числа, при этом в левом поддереве узла все числа не больше, чем в самом узле; в правом же поддереве, наоборот, не меньше.

Введём обозначения: пусть x — некоторый узел. Тогда

- x.key — число в узле;
- x.left — левый потомок;
- x.right — правый потомок;
- x.p — родитель.

При этом считаем, что у каждого узла есть оба наследника, но некоторые могут узлами не являться и иметь значение NULL.

Указанные выше свойства двоичного дерева поиска можно записать так:

$y \in \mathrm{Tree}(x.\mathrm{left}) \implies y.\mathrm{key} \leqslant x.\mathrm{key}$

$y \in \mathrm{Tree}(x.\mathrm{right}) \implies y.\mathrm{key} \geqslant x.\mathrm{key}$

Что с этим деревом можно делать? Для начала, это дерево можно обойти так, чтобы перечислить элементы в порядке возрастания:

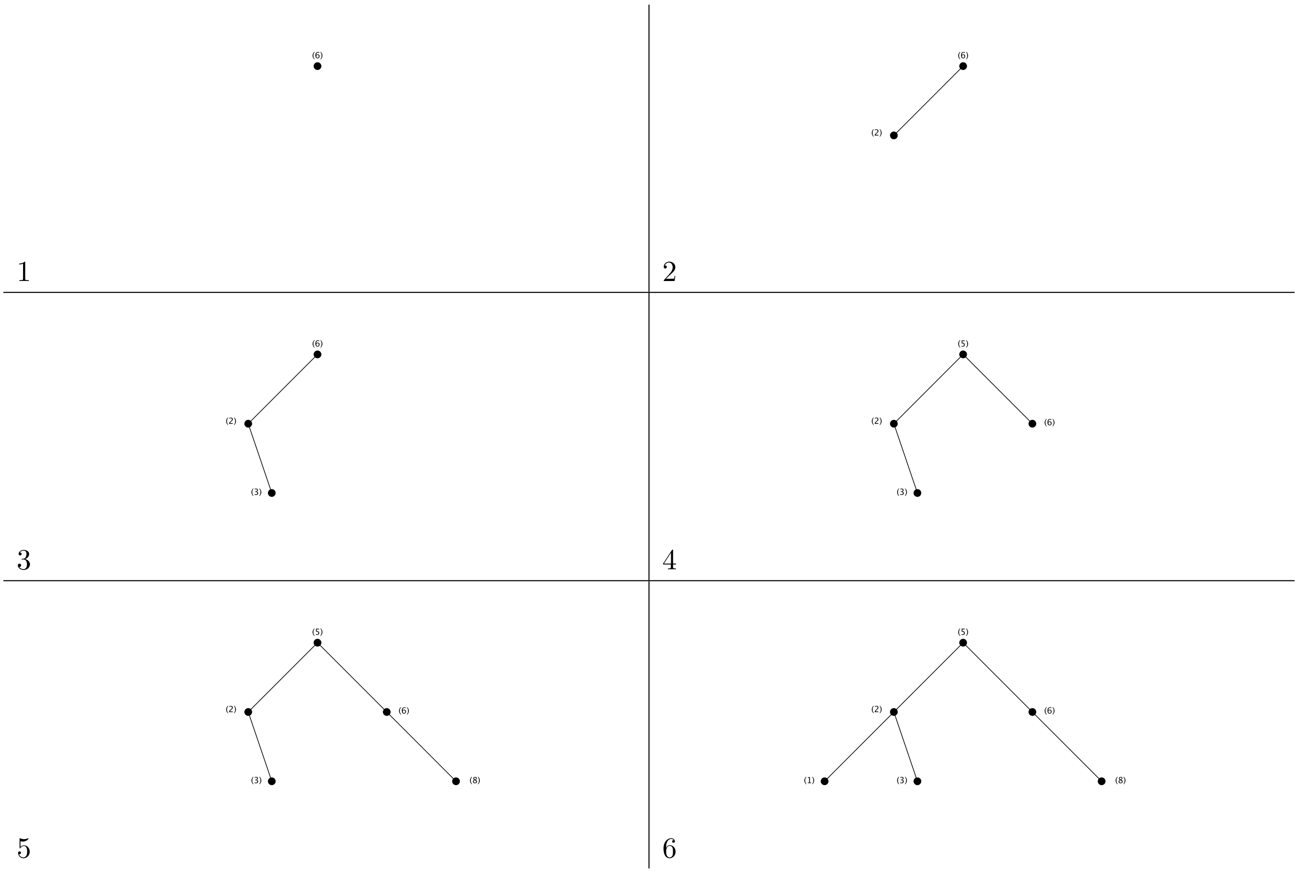
```
inorder_tree_walk(x)
    if x != NULL then
        inorder_tree_walk(x.left)
        output x.key
        inorder_tree_walk(x.right)
```

Сложность алгоритма — $\Theta(n)$; каждый узел мы посещаем не более одного раза, при этом операции внутри узла занимают константное время.

Запишем алгоритм сортировки с помощью дерева:

```
tree_sort(a)
    t := Tree()
    for x in a do
        tree_insert(x, t)
    inorder_tree_walk(t)
```

Пусть $a = [6, 2, 3, 5, 8, 1]$. Тогда построение дерева может выглядеть так (может и по-другому, дерево далеко не всегда строится однозначно):



Заметим, что в общем и целом, это довольно похоже на алгоритм быстрой сортировки (особенно, если не двигать корневой элемент при вставке новых), а корень (под-)дерева — опорный элемент.

Время работы влгоритма: $T(n) = 2T(\frac{n}{2}) + cn \implies \Theta(n \log n)$

Предположим, что слияние дорогое:

$T(n) = 2T(\frac{n}{2}) + cn^2 \implies O(n^2 \log n)$

HERE BE KARTINKA

i-ый уровень:

- Число подзадач $= 2^i$
- Размер подзадачи $= \frac{n}{2^i}$
- Время на решение подзадачи $= c\left(\frac{n}{2^i}\right)^2$
- Всего работы $= \frac{cn^2}{2^{2i}} \cdot 2^i = \frac{cn^2}{2^i}$

$T(n) \leqslant \sum\limits_{i=0}^{\log_2 n-1} \frac{cn^2}{2^i} = cn^2 \sum \frac{1}{2^i} \leqslant 2cn^2 = O(n^2)$

$T(n) = kn^d$

Удобно начинать индукцию с шага:

Пусть $T(m) \leqslant km^d$ для $m < d$.

$T(n) \leqslant 2T(\frac{n}{2}) + cn^2 \leqslant 2k\left(\frac{n}{2}\right)^d + cn^2 = 2k\frac{n^d}{2^d} + cn^2 = \{d = 2\} = 2k\frac{n^2}{4} + cn^2 = \frac{k}{2}n^2 + cn^2 = \{k = 2c\} = kn^2$

База: $T(2) \leqslant c \leqslant 2c \cdot 2^2$

я перестал понимать происходящее и переписываю формулы

$T(n) = aT(\frac{n}{2}) + cn \implies O(n^{\log_2 a})$

$T(n) \leqslant kn^d$
$T(n) = aT(\frac{n}{2}) + cn \leqslant ak\frac{n^d}{2^d} + cn = \{d = \log_2 a\} = kn^d + cn$
$T(n) \leqslant kn^d - ln$
$T(n) = aT(\frac{n}{2}) + cn \leqslant a\left(k\frac{n^d}{2^d} - l\frac{n}{2}\right) + cn = kn^d - al\frac{n}{2} + cn = kn^d - \left(\frac{al}{2} - c\right)n = \left\{l = \frac{al}{2} - c\right\} = kn^d - ln$
База: $T(2) \leqslant c \leqslant k \cdot 2^d - 2l$
$T(n) = T\left(\frac{n}{2}\right) + cn$
$T(n) = \sum\limits_{i=0}^{\log_2 n} c\frac{n}{2^i} = cn\sum\frac{1}{2^i} \leqslant 2cn$
Или с помощью метода частичной подстановки:
$T(n) \leqslant kn^d$
$T(n) = T\left(\frac{n}{2}\right) + cn \leqslant \frac{kn}{2} + cn = kn^d$ ааааааааааа

$T(n) = aT\left(\frac{n}{b}\right) + cn^d$

HERE BE KARTINKA

i-ый уровень:

- Число подзадач $= a^i$
- Размер подзадачи $= \frac{n}{b^i}$
- Время на решение подзадачи $= c\left(\frac{n}{b^i}\right)^b$
- Всего работы $= c\left(\frac{n}{b^i}\right)^d \cdot a^i = cn^d\left(\frac{a}{b^d}\right)^i$

Всего $\leqslant cn^d \sum\limits_{i=0}^{log_b n}\left(\frac{a}{b^d}\right)^i$

1. $a = b^d$: $O(n^d \log n)$
2. $a < b^d$: $O(n^d)$
3. $a > b^d$: $\sum\left(\frac{a}{b^d}\right)^i = O\left(\left(\frac{a}{b^d}\right)^{log_b n}\right)$ $\ldots\ldots\ldots O\left(n^{\log_b n}\right)$