

## Больше куч!

	<i>binaryheap</i>	<i>binomialheap</i>	<i>fibonacciheap</i>
<i>insert</i>	$O(\log n)$	$O(\log n)$	$O(1)$
<i>extract_min</i>	$O(\log n)$	$O(\log n)$	$\tilde{O}(\log n)$
<i>decrease_key</i>	$O(\log n)$	$O(\log n)$	$\tilde{O}(1)$
<i>increase_key</i>	$O(\log n)$	$O(\log n)$	$\tilde{O}(\log n)$
<i>merge</i>	$\tilde{O}(\log^2 n)$	$O(\log n)$	$O(1)$
<i>get_min</i>	$O(\log n)$ or $O(1)$	$O(1)$	

## Биномиальная куча

Храним биномиальные деревья. Каждому дереву сопоставим ранг. Ранг дерева полностью определяет его структуру. Дерево ранга 0 — одна вершина. Дерево ранга 1 — одно ребро. В общем случае, дерево ранга  $n$  содержит корень и полное двоичное дерево размера  $2^{n-1}$ . Дерево ранга  $n+1$  — это два слитых вместе дерева ранга  $n$  — корень второго указывает на корень первого, а корень первого теперь будет указывать на оба бинарных дерева.

Биномиальная куча — это набор из логарифма биномиальных куч.

Слияние двух деревьев мы научились делать за  $O(1)$  — меньшая вершина по ключу становится новым корнем, а дальше перекидываем указатели.

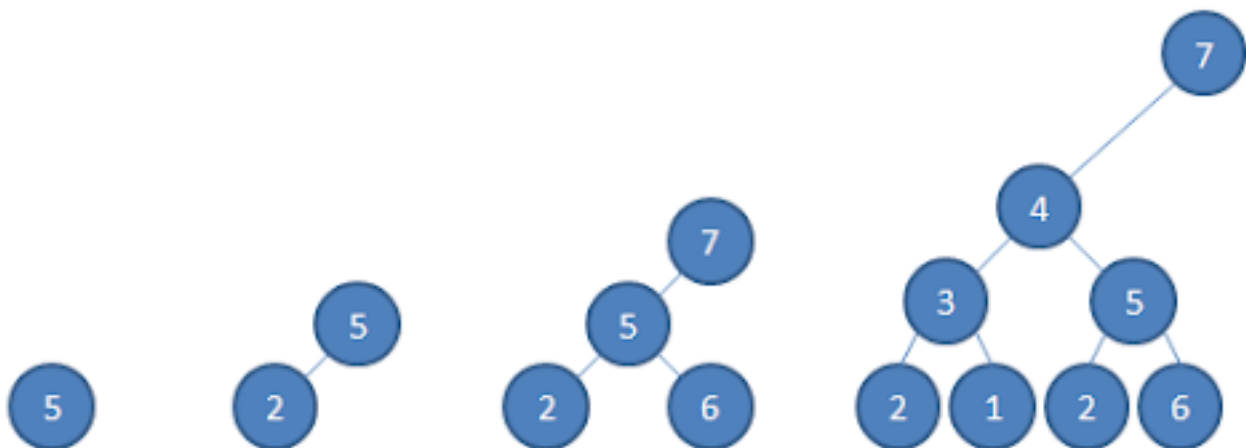
Слияние двух куч — это алгоритм сложения двоичных чисел — при сливании двух деревьев ранга  $n$  мы «переносим» прибавление кучи ранга  $n+1$

Как добавлять элемент? Создать кучу на 0 элементов и слить их вместе.

Уменьшение ключа — напомним *sift\_up* на нашей новой куче

Удаление минимума — Заметим, что если в правом дереве пройти по правым детям и обозначим их за корни, а их левых детей за полные бинарные деревья, то мы получим набор деревьев рангов  $0, 1, \dots, n-1$ . Обозначим их за новую кучу, и сольем все вместе.

Увеличение ключа — удалим соответствующий элемент и добавим другой.



## Фибоначчиева куча

Хотим сделать биномиальную кучу с послаблениями — делать операции в самый последний момент, менее четкую структуру, etc

Есть деревья, их корни храним в двусвязном закольцованном списке.

Всех детей для всех вершин храним в двусвязном закольцованном списке.

Новый ранг — это количество вершин в списке детей.

На каждом дереве выполнена куча, а также поддерживаем глобальный минимум.

Улучшение ключа делается так — удаляем вершину из своего списка, вместе с поддеревом. Добавляем в корневой список. Если мы удалили уже вторую вершину в поддереве родителя, то делаем каскадное вырезание — прыгаем по предкам с  $mark = 1$ , и вырезаем их в корневой список, причем все вырезания делаются по очереди.

Удаление делается так — мы приписываем всех детей к корневому списку, а потом вызываем *compact*, которая должна спасти наше дерево и навести порядок.

Псевдокод тупых операций:

```
struct Node{
    Node *child;
    Node *left;
    Node *right;
    Node *parent;
    int rank;
    bool mark;
    int value;
}

list<Node *> roots;

void insert(int x) {
    Node *node = new Node(x);
    roots.insert(node);
}

void merge(list<Node *> a, list<Node *> b) {
    merge(a, b); // O(1) haha super easy
}

int getmin() {
    return argmin->value;
}
```

## compact

- Сбрасываем пометки корневого списка в 0
- Переводим дерево в состояние, где все ранги различны
- Храним ранги, мерджим одинаковые
- Как мерджим? Берем меньший корень, и записываем в его детей второй корень

Обозначим  $R = \max rank$ ,  $t(H) = \text{root list size}$ ,  $m(H) = \sum_v mark(v)$   
 $compact$  работает за  $O(R + t(H))$ .

**Анализ времени работы**  $extract\_min \& increase\_key - \tilde{O}(R)$ .

$$\Phi(H) = t(H) + 2m(H) < 3n$$

Пусть каскадное вырезание сделало  $t_i$  действий.  $m : 1 \rightarrow 0$ ,  $t : +1$ . Тогда  $\Phi'(H) = \Phi(H) - 1$  за каждое вырезание. Тогда амортизированно вырезание работает за  $O(1)$ .

Смраст:

$$t(H) \leq R, t'(H) = t(H) - R$$

Амортизированно работает за  $2R + 1$

Хотим показать  $R = O(\log n)$ .

$$\forall v \in H \ sz(v) \geq A^{rank(v)}, A > 1$$

Тогда

$$r(v) \leq \log_A s(v) \leq c \log n$$

Возьмем

$$s(v) \geq \phi^{rank(v)-2}$$

<Оффтоп про числа Фибоначчи>

$$1. F_n = F_{n-1} + F_{n-2}, F_0 = F_1 = 1$$

$$2. F_n \geq \phi^{n-2}$$

$$3. \forall i \geq 2: F_i = \sum_{j=0}^{i-2} F_j + 1$$

</Оффтоп про числа Фибоначчи>

Почему инвариант на размеры сохраняется? Возьмем вершину  $v$  с  $k$  детьми. Рассмотрим детей в том порядке, в котором их склеивал компакт. Тогда на момент добавления  $i$ -й вершины в структуру, ее ранг совпадал с рангом  $v$ . Тогда из условия на удаление не более чем одного сына ( $mark$ ) следует, что  $rank(i) \geq i - 1$ . Тогда мы доказываем по индукции, что у нас все размеры — хотя бы числа фибоначчи,

соответствующие рангу. Тогда  $s(v) = \sum_{u \in g(v)} s(u) + 1 \geq \sum_{u \in g(v)} F_{rank(u)} + 1 \geq \sum_{i=0}^{rank(v)-2} F_i + 1 \geq F_{rank(v)-2}$