Алгоритмы и структуры данных

Кучи

CS Center, Новосибирск

Как реализовать очередь с приоритетом?

Меньшее значение ключа – выше приоритет

Как реализовать очередь с приоритетом?

Меньшее значение ключа – выше приоритет

Абстрактная структура данных с методами:

- add(K)
- K getMin()

Как реализовать очередь с приоритетом?

Меньшее значение ключа – выше приоритет

Абстрактная структура данных с методами:

- add(K)
- K getMin() const
- K extractMin()

Как реализовать очередь с приоритетом?

Меньшее значение ключа – выше приоритет

Абстрактная структура данных с методами:

- add(K)
- K getMin() const
- K extractMin()
- delete(?)

Как реализовать очередь с приоритетом?

Меньшее значение ключа – выше приоритет

Абстрактная структура данных с методами:

- HANDLE add(K)
- K getMin() const
- K extractMin()
- delete(HANDLE)

Как реализовать очередь с приоритетом?

Меньшее значение ключа – выше приоритет

Абстрактная структура данных с методами:

- HANDLE add(K)
- K getMin() const
- K extractMin()
- delete(HANDLE)
- decreaseKey(HANDLE, K)

Куча | Heap

{

HANDLE add(K)

K getMin() const

K extractMin()

delete(HANDLE)

decreaseKey(HANDLE, K)

}

HeapSort

```
for ( k : a )
    heap.add(k)
while ( !heap.empty() )
    ans.add( heap.extractMin() )
```

HeapSort: временная оценка

```
for ( k : a )
    heap.add(k)
while ( !heap.empty() )
    ans.add( heap.extractMin() )
```

$T_{add} = O(?)$

$T_{extractMin} = O(?)$

HeapSort: временная оценка

```
for ( k : a )
    heap.add(k)
while ( !heap.empty() )
    ans.add( heap.extractMin() )
```

$T_{add} = O(?)$

$T_{extractMin} = O(?)$

$N*T_{add} + N*T_{extractMin} \geq T_{sort}(N)$

Дерево со свойством кучи

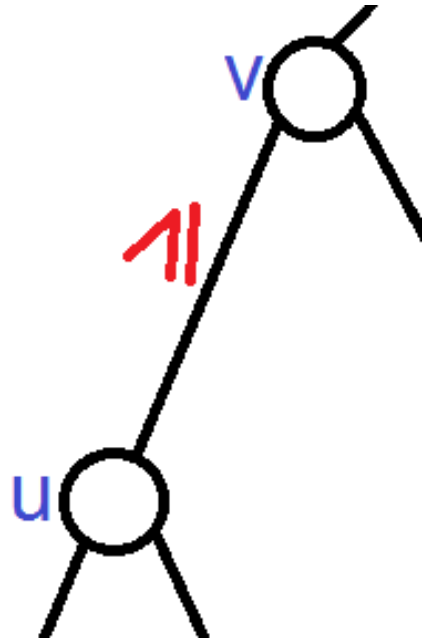
«Куча на минимум»

v = parent(u) => A[v] ≤ A[u]

Дерево со свойством кучи

«Куча на минимум»

v = parent(u) => A[v] ≤ A[u]



Дерево со свойством кучи

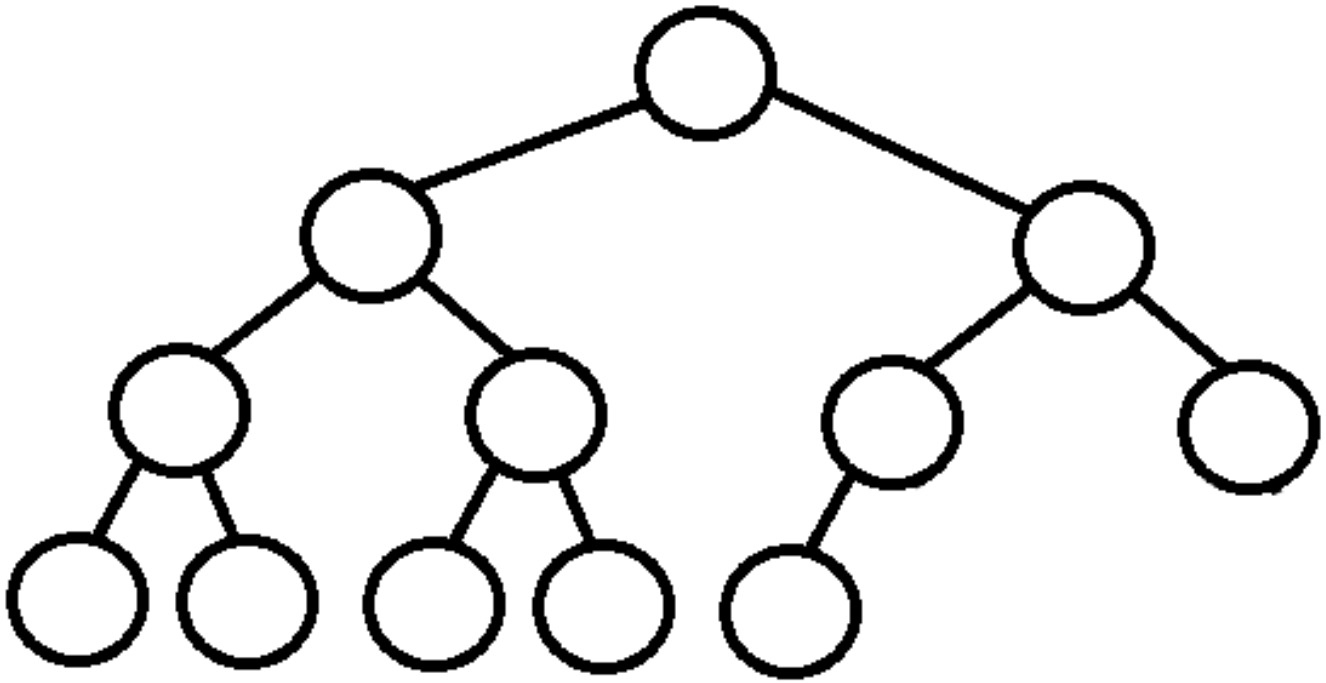
«Куча на минимум»

v = parent(u) => A[v] ≤ A[u]

- Минимум в корне дерева

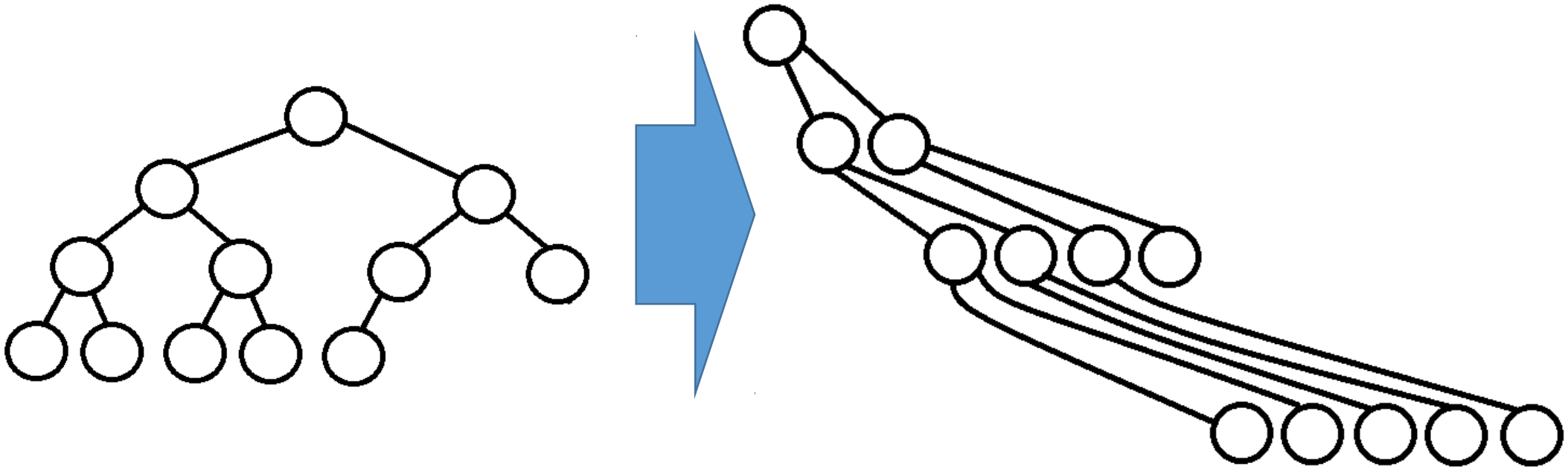
Квазиполное бинарное дерево

- Неполным может быть только нижний слой
- В нём могут отсутствовать несколько вершин «справа»



Квазиполное бинарное дерево на массиве

parent(v) = (v - 1) / 2

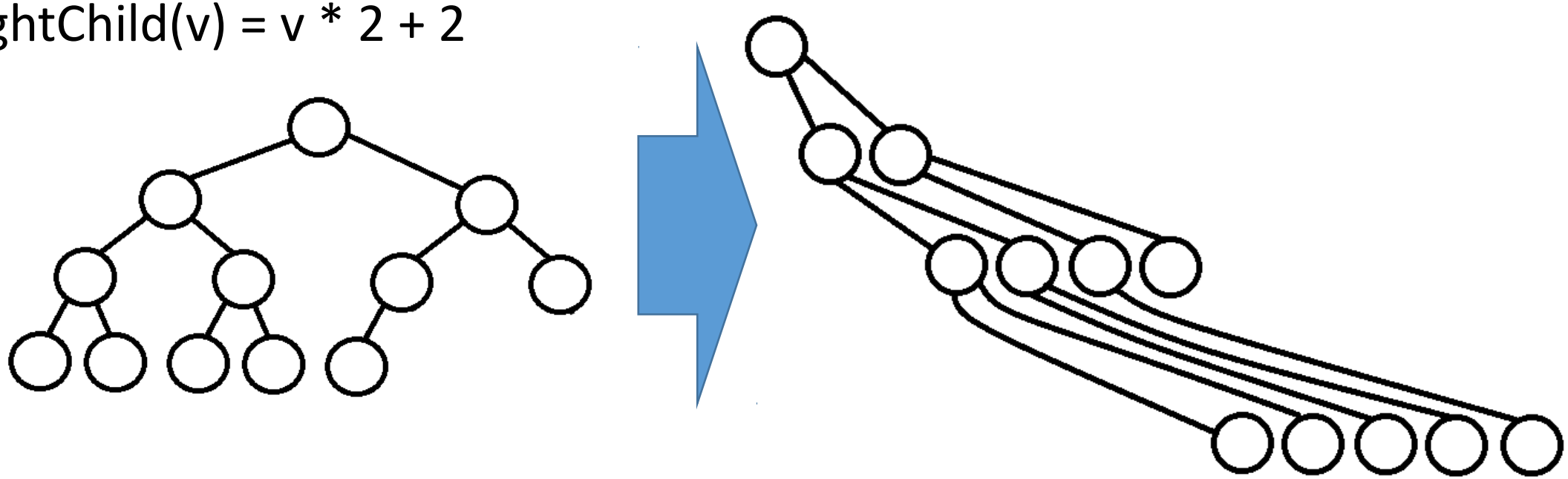


Квазиполное бинарное дерево на массиве

parent(v) = (v - 1) / 2

leftChild(v) = v * 2 + 1

rightChild(v) = v * 2 + 2



Двоичная куча | Binary heap

Квазиполное бинарное дерево со свойством кучи

Двоичная куча | Binary heap

Квазиполное бинарное дерево со свойством кучи

Как поддерживать это свойство, выполняя операции?

Двоичная куча | Binary heap

Квазиполное бинарное дерево со свойством кучи

Как поддерживать это свойство, выполняя операции?

- add
- extractMin
- decreaseKey
- delete

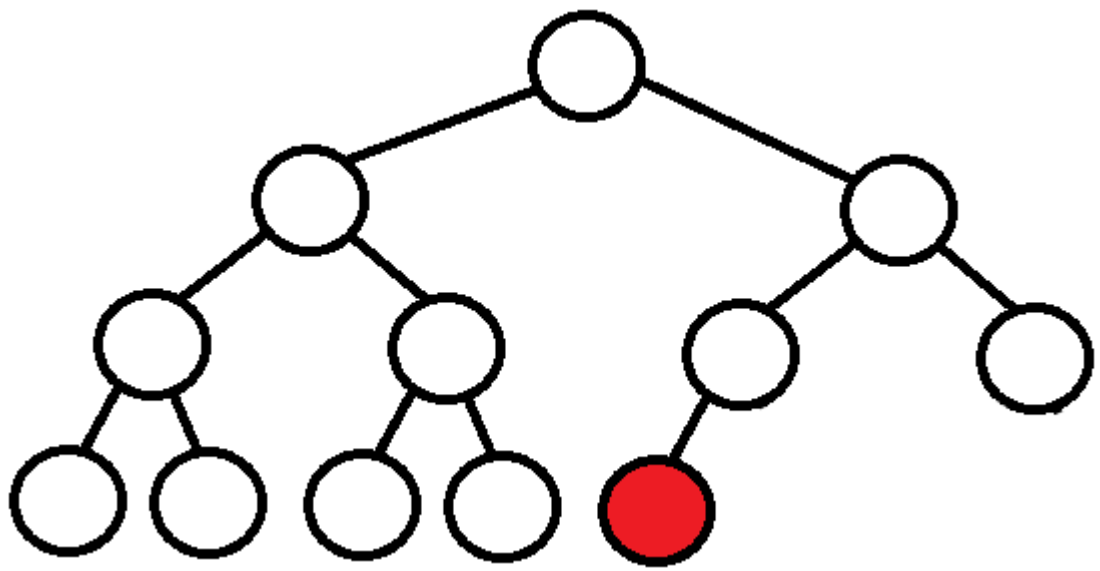
Поддержание свойств кучи

Вспомогательные операции:

- Sift-up
- Sift-down

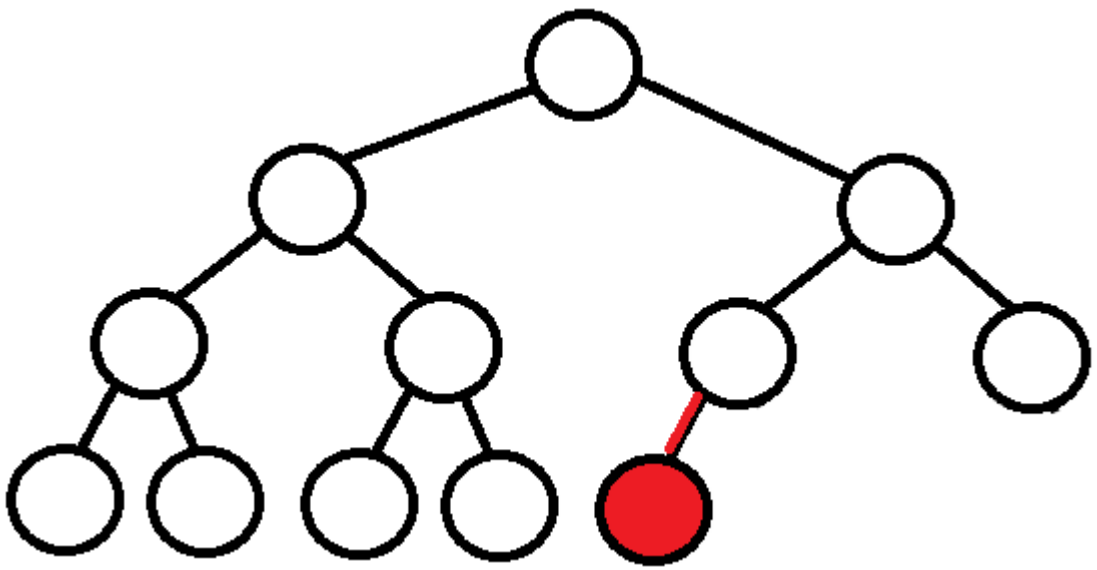
Sift-up

- Один элемент внизу нарушает свойство кучи



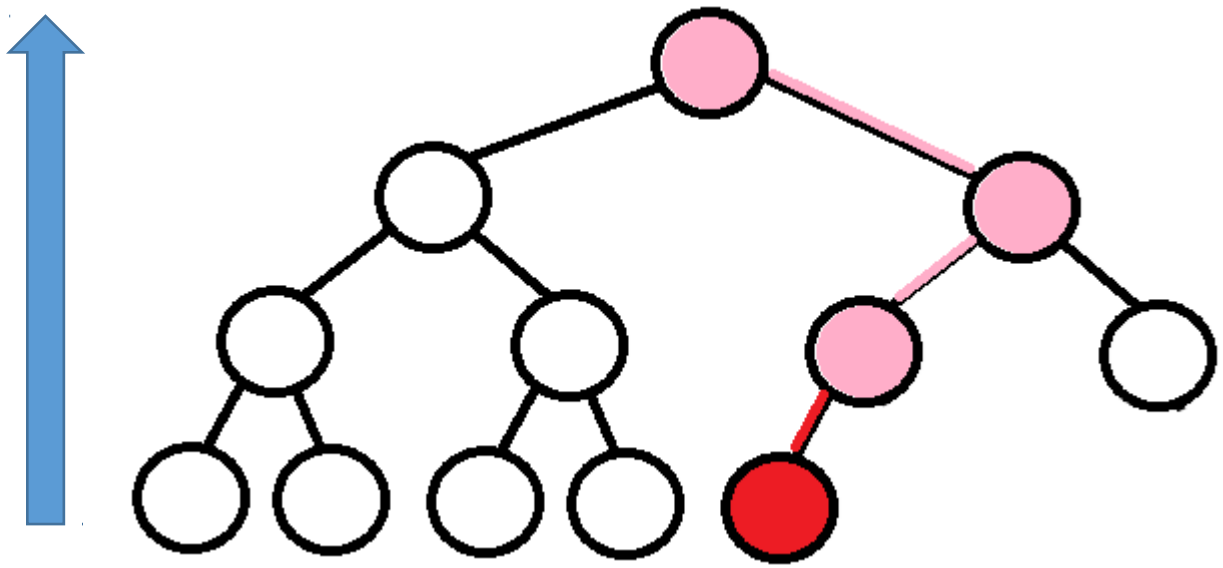
Sift-up

- Один элемент внизу нарушает свойство кучи
  - «конфликтует с родителем»



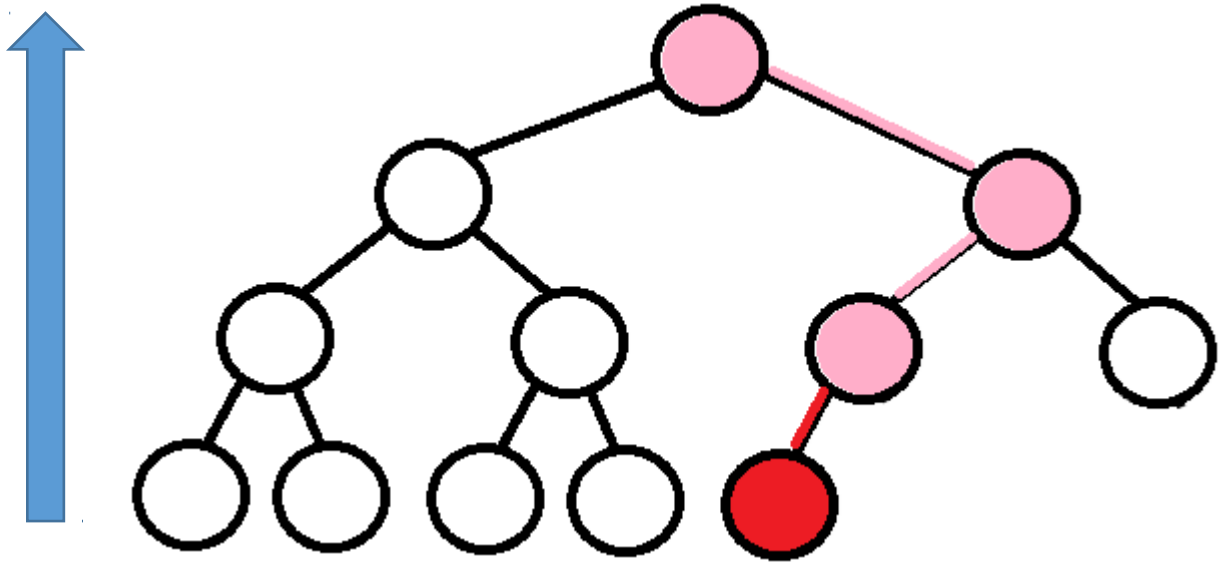
Sift-up

- Один элемент внизу нарушает свойство кучи
  - «конфликтует с родителем»
- За O(H) обменами поднимем этот элемент на его место



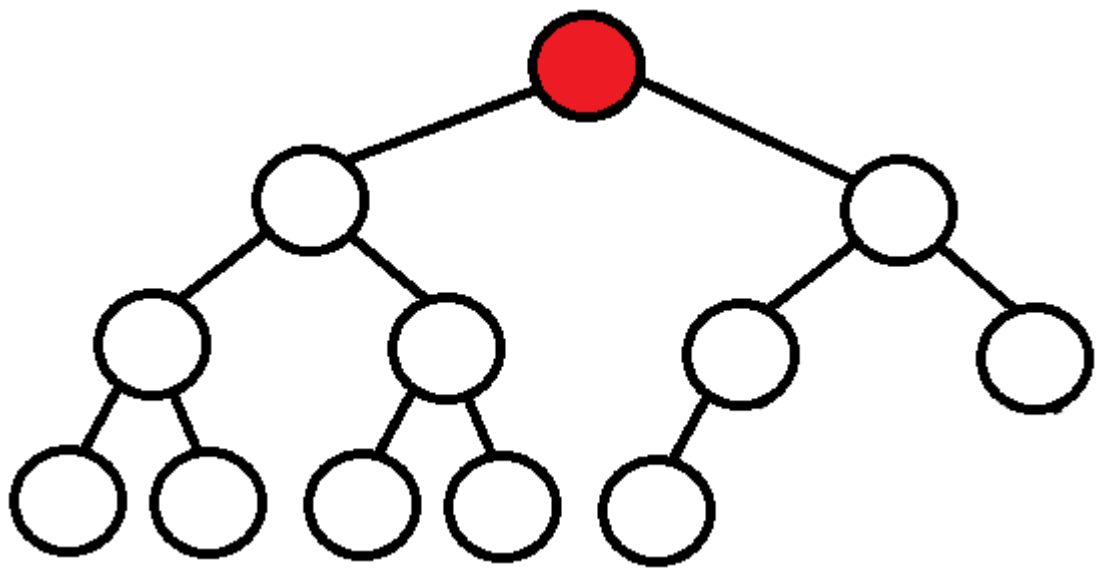
Sift-up

- Один элемент внизу нарушает свойство кучи
  - «конфликтует с родителем»
- За O(H) обменами поднимем этот элемент на его место
  - Теперь свойство выполняется



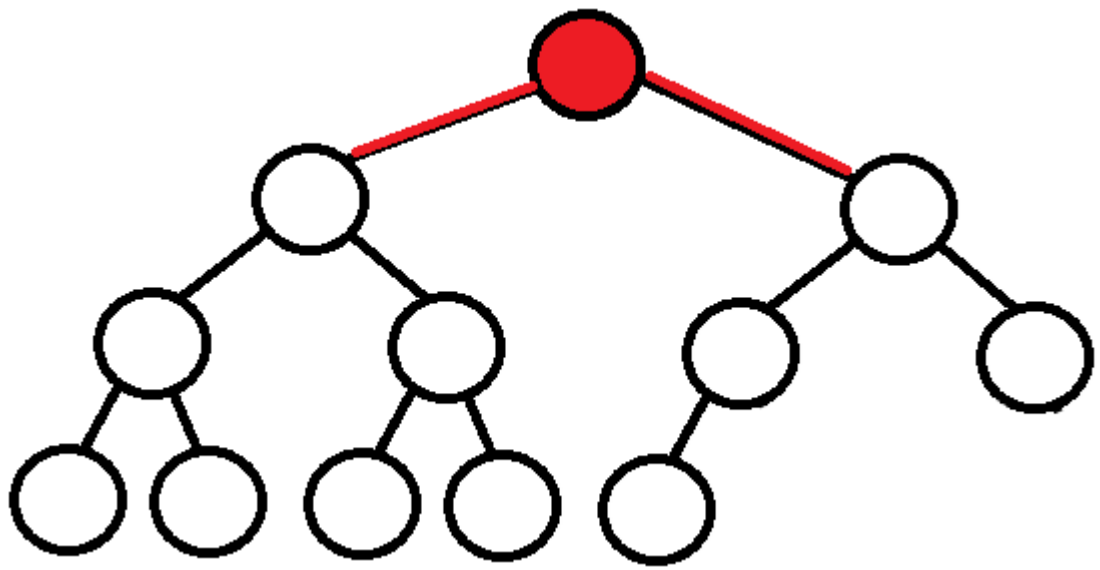
Sift-down

- Один элемент наверху нарушает свойство кучи



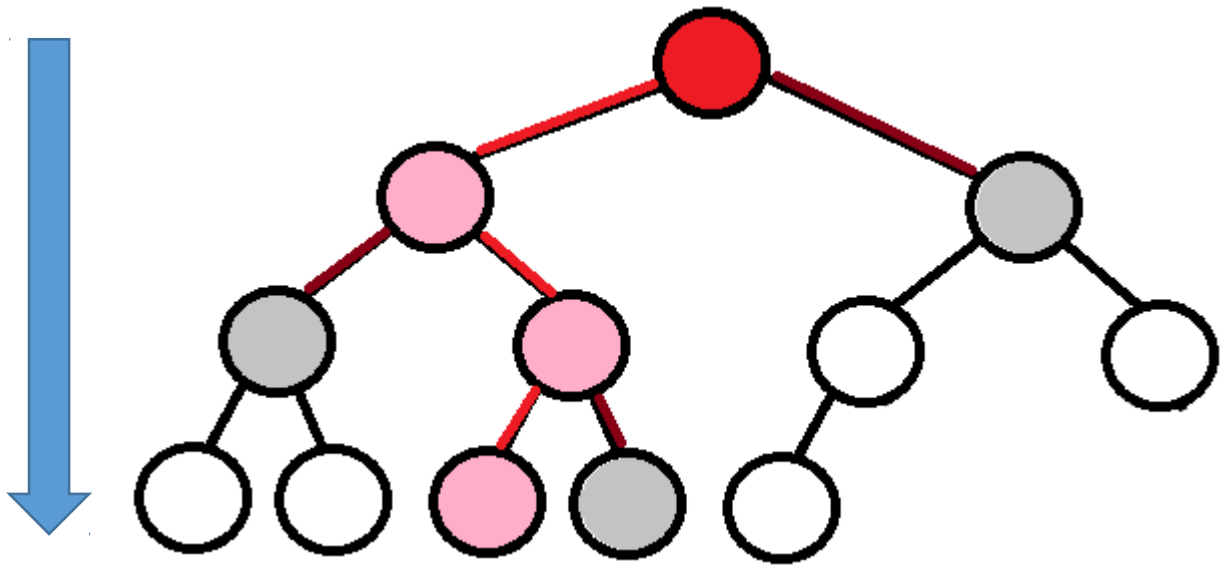
Sift-down

- Один элемент наверху нарушает свойство кучи
  - «конфликтует с детьми»



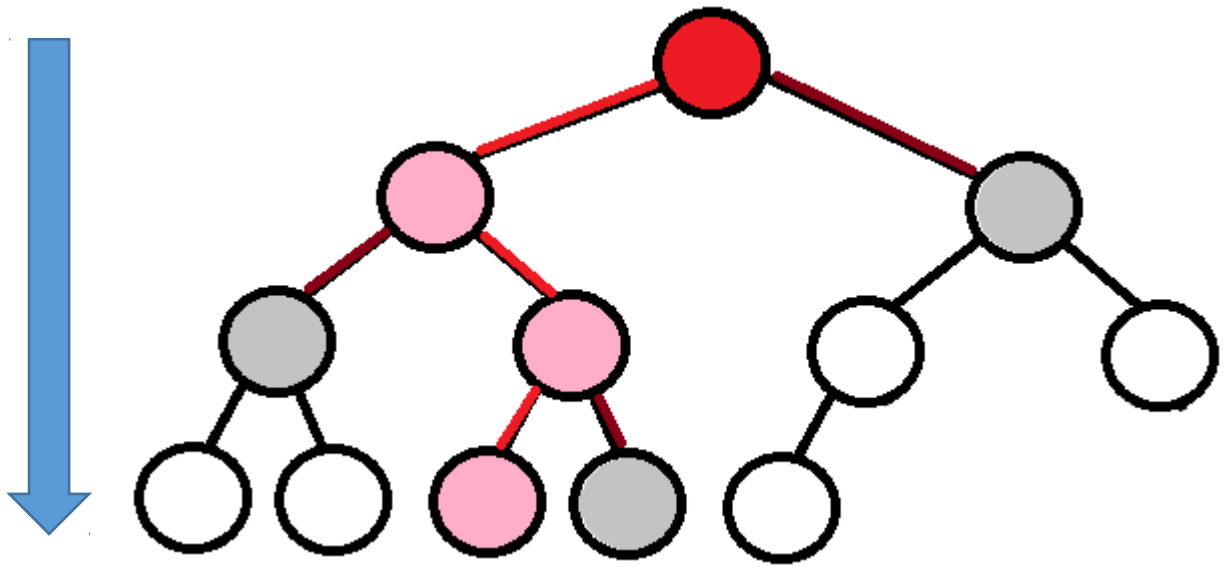
Sift-down

- Один элемент наверху нарушает свойство кучи
  - «конфликтует с детьми»
- За O(H) обменами опустим его туда, где он должен быть



Sift-down

- Один элемент наверху нарушает свойство кучи
  - «конфликтует с детьми»
- За O(H) обменами опустим его туда, где он должен быть
  - Теперь свойство выполняется



Сложность sift-up и sift-down

- Квазиполное бинарное дерево, H = log(N)

Сложность sift-up и sift-down

- Квазиполное бинарное дерево, H = log(N)
  - T = O(logN)

Реализация операций

- add
- extractMin
- decreaseKey
- delete

Реализация операций

- add : добавление и siftUp
- extractMin
- decreaseKey
- delete

Реализация операций

- add : добавление и siftUp
- extractMin : замена ключа в корне и siftDown
- decreaseKey
- delete

Реализация операций

- add : добавление и siftUp
- extractMin : замена ключа в корне и siftDown
- decreaseKey : уменьшение ключа и siftUp
- delete

Реализация операций

- add : добавление и siftUp
- extractMin : замена ключа в корне и siftDown
- decreaseKey : уменьшение ключа и siftUp
- delete : dectreaseKey и extractMin

Оценка сложности операций

- add : добавление и siftUp
- extractMin : замена ключа в корне и siftDown
- decreaseKey : уменьшение ключа и siftUp
- delete : dectreaseKey и extractMin

$T_{add} = O(?)$

$T_{extractMin} = O(?)$

$T_{decreaseKey} = O(?)$

$T_{delete} = O(?)$

Оценка сложности операций

- add : добавление и siftUp
- extractMin : замена ключа в корне и siftDown
- decreaseKey : уменьшение ключа и siftUp
- delete : dectreaseKey и extractMin

$T_{add} = O(logN)$

$T_{extractMin} = O(logN)$

$T_{decreaseKey} = O(logN)$

$T_{delete} = O(logN)$

Преобразование массив-> куча

Кучу из N элементов можно построить за O(NlogN)

- Можно ли быстрее?

k-ичная куча

Биномиальная куча

Левацкая куча | Leftist heap

Косая куча | Skew heap