# Куча

**Куча** — это специализированная структура данных типа дерево, которая удовлетворяет *свойству кучи:* если *B* является узлом-потомком узла *A*, то ключ(*A*) ≥ ключ(*B*). Из этого следует, что элемент с наибольшим ключом всегда является корневым узлом кучи, поэтому иногда такие кучи называют *max-кучами* (в качестве альтернативы, если сравнение перевернуть, то наименьший элемент будет всегда корневым узлом, такие кучи называют *min-кучами*). Не существует никаких ограничений относительно того, сколько узлов-потомков имеет каждый узел кучи, хотя на практике (например, в двоичной куче) их число обычно не более двух.

Над кучами обычно проводятся следующие операции:

* *найти максимум* или *найти минимум*: найти максимальный элемент в max-куче или минимальный элемент в min-куче, соответственно
* *удалить максимум* или *удалить минимум*: удалить корневой узел в max- или min-куче, соответственно
* *увеличить ключ* или *уменьшить ключ*: обновить ключ в max- или min-куче, соответственно(меняем ключ у корня а дальше упорядочиваем кучу)
* *добавить*: добавление нового ключа в кучу.
* *слияние*: соединение двух куч с целью создания новой кучи, содержащей все элементы обеих исходных. (не будем рассматривать, там все делается через предыдущую операцию)



Как несложно догадаться, операция упорядочивания двоичной кучи выполняется за log n.

# Двоичная куча

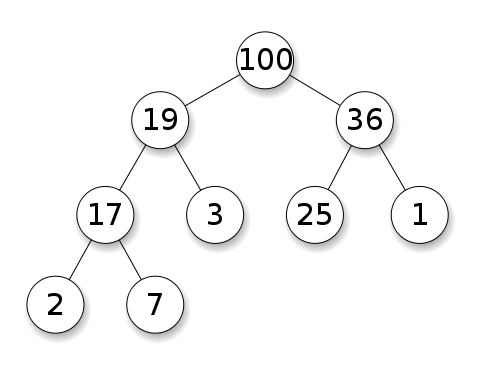
***Двоичная куча*** представляет собой полное бинарное дерево, для которого выполняется основное свойство кучи: **приоритет каждой вершины больше приоритетов её потомков**.

В простейшем случае приоритет каждой вершины можно считать равным её значению. В таком случае структура называется ***max-куча***, поскольку корень поддерева является максимумом из значений элементов поддерева.

В качестве альтернативы, если сравнение перевернуть, то наименьший элемент будет всегда корневым узлом, такие кучи называют ***min-кучами***.

Двоичную кучу удобно хранить в виде одномерного массива, причем

* левый потомок вершины с индексом i имеет индекс 2\*i+1,
* правый потомок вершины с индексом i имеет индекс 2\*i+2.



Корень дерева (кучи) – элемент с индексом 0.

То есть для примера выше:

H[0] = 100

H[1] = 19

H[2] = 36

H[3] = 17

H[4] = 3

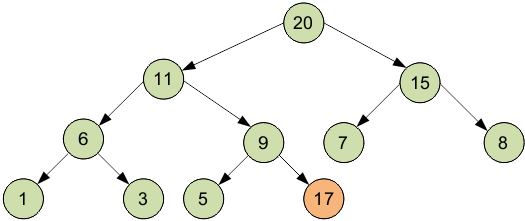
H[5] = 25

H[6] = 1

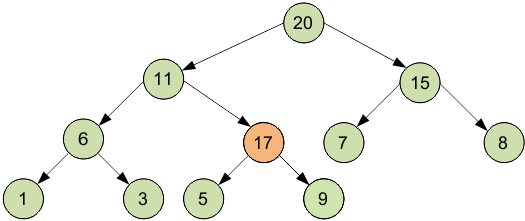
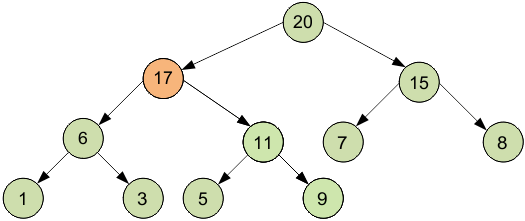
H[7] = 2

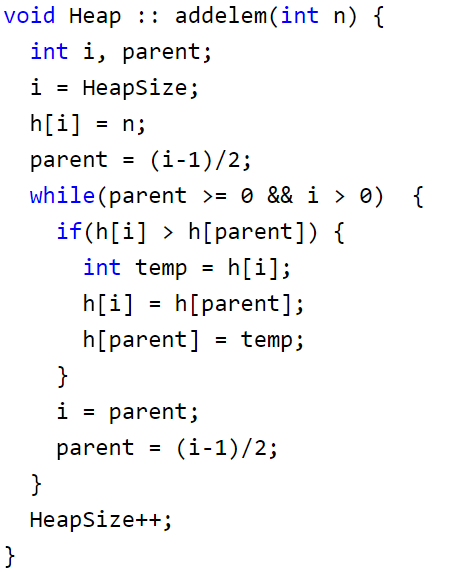
H[8] = 7

**Добавление элемента кучи**

Новый элемент добавляется на последнее место в массиве, то есть позицию с максимальным индексом.  


Возможно, что при этом будет нарушено основное свойство кучи, так как новый элемент может быть больше родителя. В таком случае новый элемент «поднимается» на один уровень (менять с вершиной-родителем) до тех пор, пока не будет соблюдено основное свойство кучи.

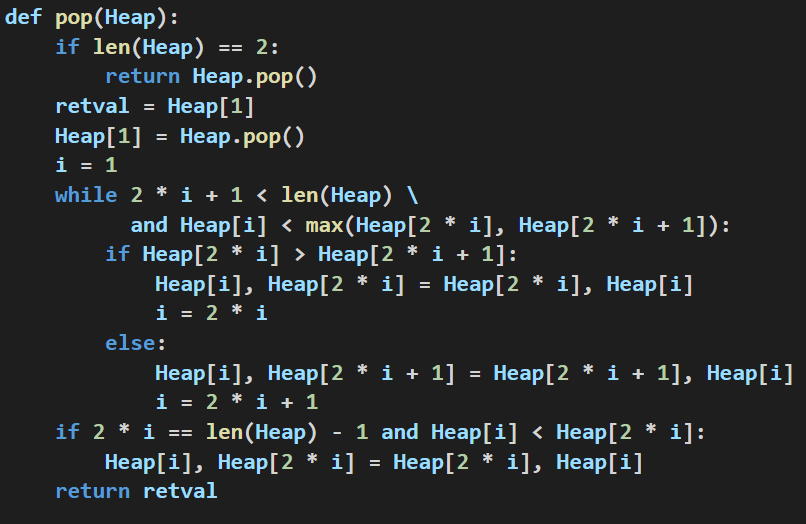
  
  


Сложность алгоритма не превышает высоты двоичной кучи (так как количество «подъемов» не больше высоты дерева), то есть равна log2 N.

**Удаление элемента из кучи**

Из кучи можно удалить наибольший элемент, то есть тот, который хранится в вершине куче. На его место нужно поставить какой-нибудь элемент кучи. Поставим последний элемент кучи, удалив его из конца. Теперь в вершине кучи может нарушиться свойство кучи, значит, верхний элемент нужно сместить вниз, обменяв его с одним из своих потомков. При этом из двух потомков нужно выбрать наибольший и если этот наибольший потомок больше стоящего в вершине кучи, обменяем их местами.

Тем самым элемент, который был взят снизу кучи, спустится на один уровень вниз. Будем дальше опускать этот элемент до тех пор, пока оба его потомка не станут меньше его (или у него не будет потомков, также необходимо аккуратно обработать случай одного потомка.

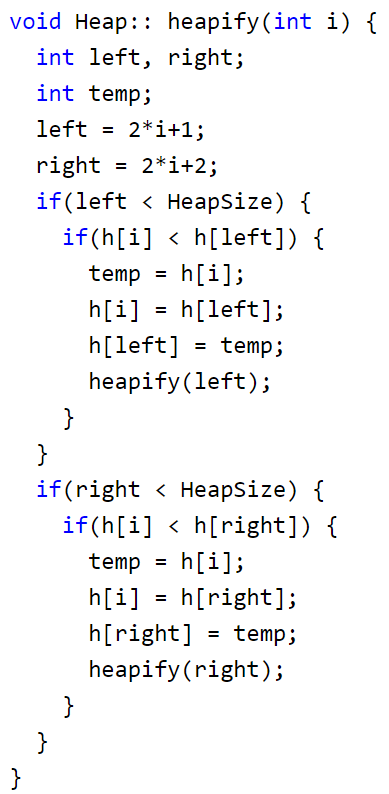


В этом примере сохраняется значение на вершине кучи в переменной retval, затем последний элемент удаляется из кучи и ставится на вершину кучи. Отдельно обрабатывается случай, когда куча состояла ровно из одного элемента, т. е. после удаления она становится пустой. Далее в основном цикле элемент опускается. Цикл продолжается пока у элемента два потомка и хотя бы один из потомков больше текущего элемента. В этом случае элемент меняется местами с наибольшим из потомков и цикл повторяется заново.

После окончания цикла отдельно обрабатывается случай, когда у элемента ровно один потомок (нет правого потомка) и единственный левый потомок больше данного элемента, в этом случае необходимо провести еще один обмен.

**Упорядочивание кучи**

Вообще он тут уже фигурировал, но вот вам отдельной функцией.



В упорядоченном max-heap максимальный элемент всегда хранится в корне. Восстановить упорядоченность двоичной кучи после удаления максимального элемента можно, поставив на его место последний элемент и вызвав метод упорядочения для корня, то есть упорядочив все дерево.

**Применения кучи**

Одно из наиболее известных применений кучи — сортировка при помощи кучи или пирамидальная сортировка (англ. heapsort). В данной сортировке из элементов списка снача строится куча, потом элементы по одному удаляются из кучи — сначала наибольший элемент, потом - наибольший из оставшихся и т. д. При этом кучу можно хранить там же, где хранятся элементы самого списка, тем самым пирамидальная сортировка имеет сложность O(), но при этом не требует дополнительной памяти (как сортировка слиянием) и не является вероятностной (как быстрая сортировка Хоара).

Также при помощи кучи можно организовать структуру данных «очередь с приоритетами». В очереди каждому элементу сопоставляется приоритет — некоторое целое число. При удалении элемента из очереди удаляется не тот элемент, который был добавлен раньше (как в обычной очереди), а элемент с наибольшим приоритетом. То есть элементы в очереди с приоритетами можно хранить в куче, сравнивая их при этом по приоритету.

В очереди с приоритетами также есть операция изменения приоритета элемента. Для этого реализованы две функции — повышения и понижения приоритета. При повышении приоритета элемент поднимается, поэтому эта функция реализована аналогично операции добавления элемента. При понижении приоритета элемент спускается, как в операции удаления элемента**.**