The Heap Sort

The Heap Sort

- **Given an an array of randomly ordered entries:**
  - Create a heap
  - Repeat the following until the heap is of size two
    - Swap root and last entry
    - Decrease size of heap
    - Rebuild heap

Starting at eleme
call rehe
and repeat until

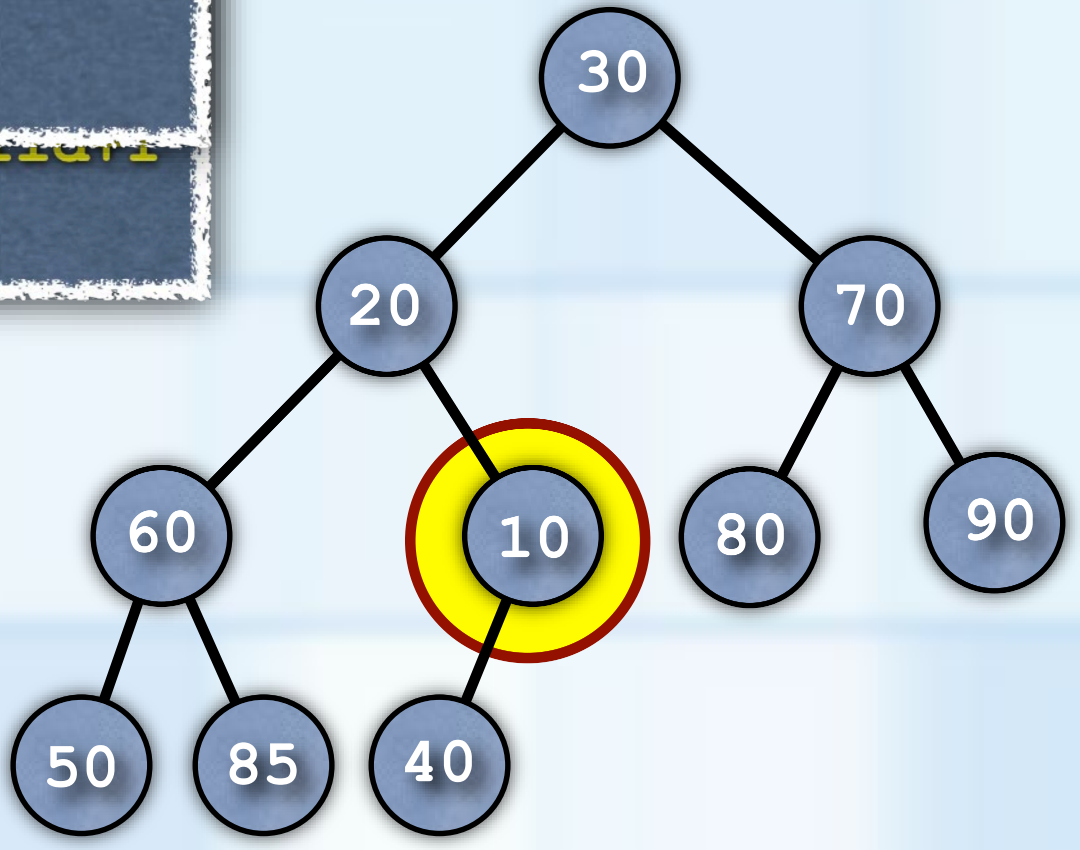
Note:

Some time later …

or 2*(i+1)

Heap

Tree View



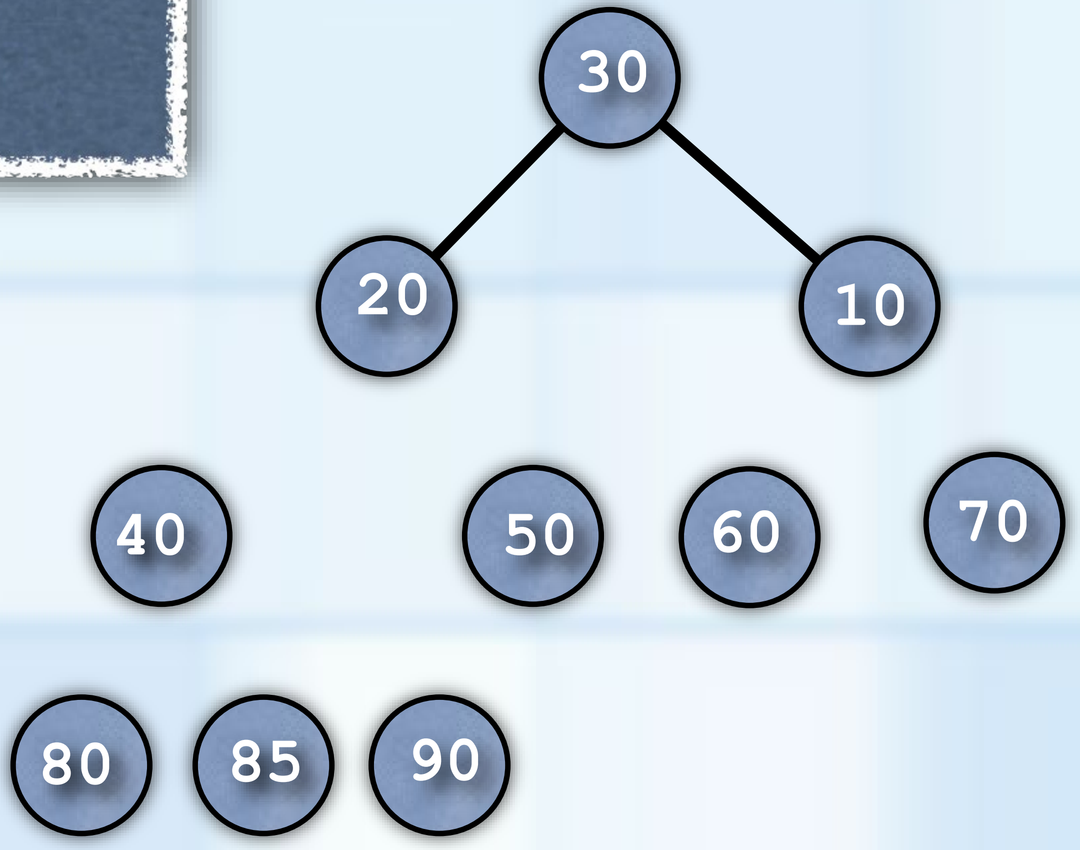
30	20	70	60	10	80	90	50	85	40		
0	1	2	3	4	5	6	7	8	9	10	11

The Heap Sort

- **Given an an array of randomly ordered entries:**
  - Create a heap
  - Repeat the following until the heap is of size two
    - Swap root and last entry
    - Decrease size of heap
    - Rebuild heap

Some time later ...

Tree View



30	20	10	40	50	60	70	80	85	90		
0	1	2	3	4	5	6	7	8	9	10	11

The Heap Sort

- **Given an a heap:**
  - Repeat the following until the heap is of size two
    - Save the root item (maximum value)
    - Swap root and last entry
    - Decrease size of heap
    - Rebuild heap
  - Return sorted values
  - Recreate a heap

```
template<class ItemType>
void ArrayHeap<ItemType>::heapSort(std::vector<ItemType>& sortedItems)
{
  sortedItems.clear();
  while(itemCount >0)
  {
    sortedItems.push_back(items[ROOTINDEX]);
    swap(items[ROOTINDEX], items[itemCount-1]);
    itemCount--;
    heapRebuild(ROOTINDEX);
  }
  itemCount = sortedItems.size(); //restore itemCount
  heapCreate(); // restore heap
}  // end toVector
```