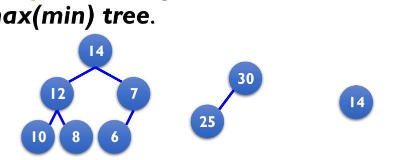
7.6

Heap Sort

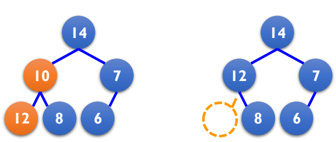
Max Heap (Priority Queue)

Definition: A ***max (min) tree*** is a tree in which the key value in each node is ***no smaller*** (***larger***) than the key values in its children (if any). A ***max(min)*** ***heap*** is a ***complete*** ***binary tree*** that is also a ***max(min) tree***.



Max Heap Max Heap Max/Min Heap

Examples: not max heap



Not a heap (12 > 10)

Not a heap (Not a complete binary tree)

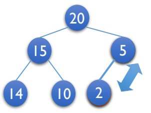
Max Heap: Representation

- Since the heap is a complete binary tree, we could adopt "**Array Representation**" as we mentioned before!
- Let node i be in position i (array[0] is empty)
  - $\boldsymbol{Parent(i) = \lfloor i/2 \rfloor}$ if $i \neq 1$. If $i = 1$, i is the root and has no parent.
  - $\boldsymbol{leftChild(i) = 2i}$ if $2i \leq n$. If $2i > n$, then i has no left child.
  - $\boldsymbol{rightChild(i) = 2i + 1}$ if $2i + 1 \leq n$, if $2i + 1 > n$, then i has no right child.

52

Max Heap: Insert

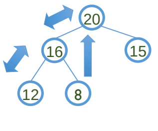
- Make sure it is a complete binary tree
- Insert a new node
- Check if the new node is greater than its parent
- If so, swap two nodes



53

Max Heap: Delete

1. Always delete the root
2. Move the last element to the root (maintain a complete binary tree)
3. Swap with larger and largest child (if any)
4. Continue step 3 until the max heap is maintained (trickle down)



54

7.6

Heap Sort

- Utilize the max-heap structure
- The insertion and deletion could be done in O(logn)
- Build a max-heap using n records, insert each record one by one (O(nlogn))
- Iteratively remove the largest record (the root) from the max-heap (O(nlogn))
- Not a stable sort

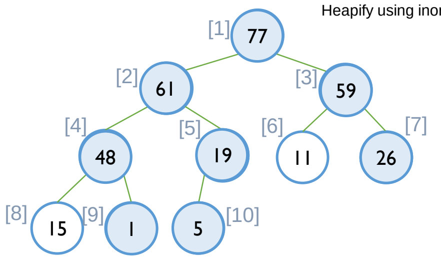
Heap Sort (code)

```
template <class T>
void HeapSort(T *a, const int n)
{
  Heapify(a, n);
  for (i = n-1; i >= 1; i--)  // Sorting
  {
      swap(a[1], a[i+1]);     // swap the root with last node
      Heapify(a, i);          // rebuild the heap (a[1:i])
  }
}
```


Heap Sort Example

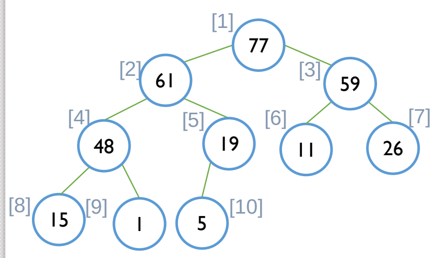
26 5 77 1 61 11 59 15 48 19

Heapify using inorder (LVR)



Heap Sort Example

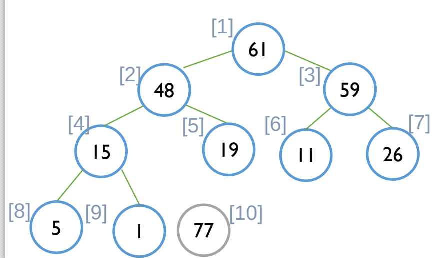
77 61 59 43 19 11 26 15 1 5



Yi-Shin Chen -- Data Structures 58

Heap Sort Example

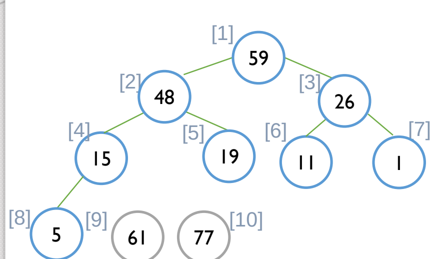
61 48 59 15 19 11 26 5 1 77



Yi-Shin Chen -- Data Structures 59

Heap Sort Example

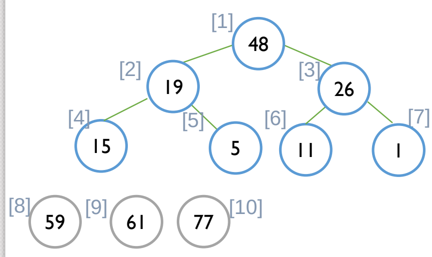
59 48 26 15 19 11 1 5 61 77



Yi-Shin Chen -- Data Structures 60

Heap Sort Example

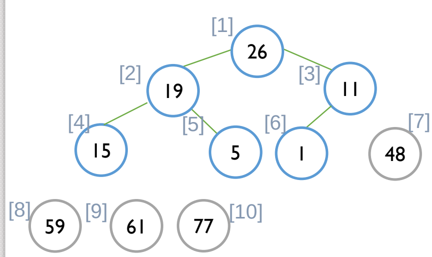
48 19 26 15 5 11 1 | 59 61 77



Yi-Shin Chen -- Data Structures 61

Heap Sort Example

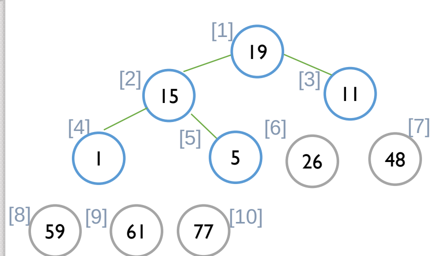
26 19 11 15 5 1 | 48 59 61 77



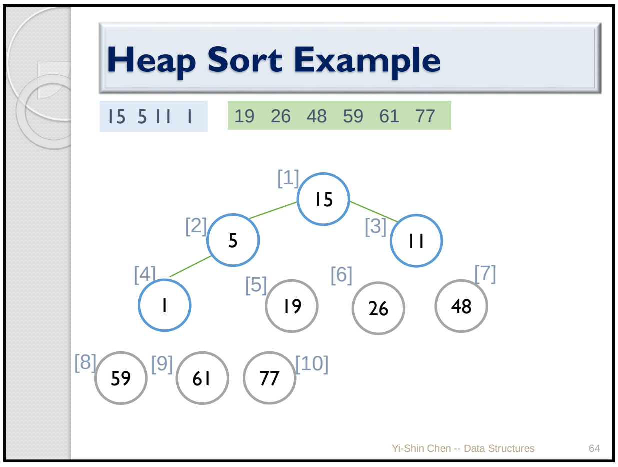
Yi-Shin Chen -- Data Structures 62

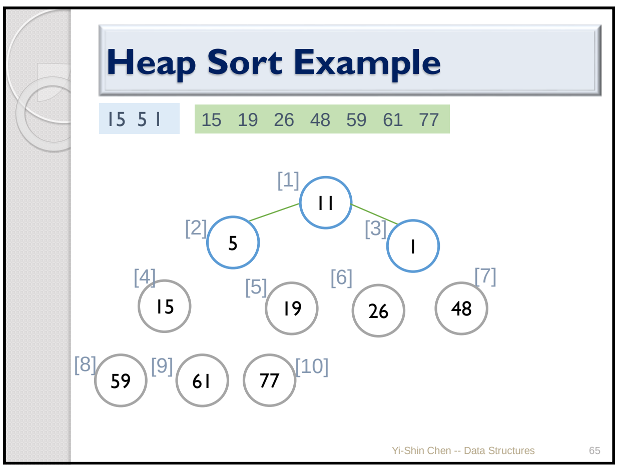
Heap Sort Example

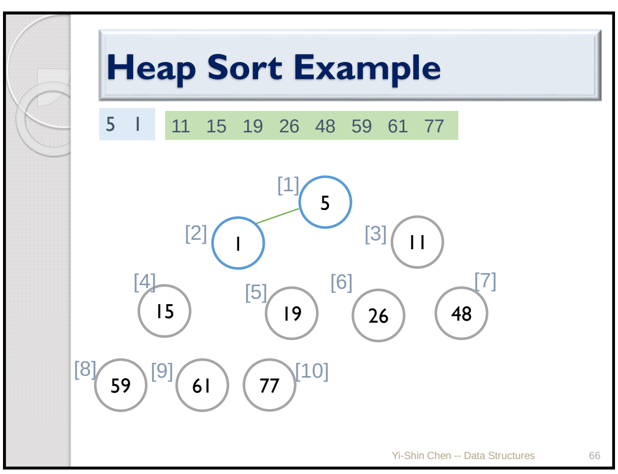
19 15 11 1 5 | 26 48 59 61 77

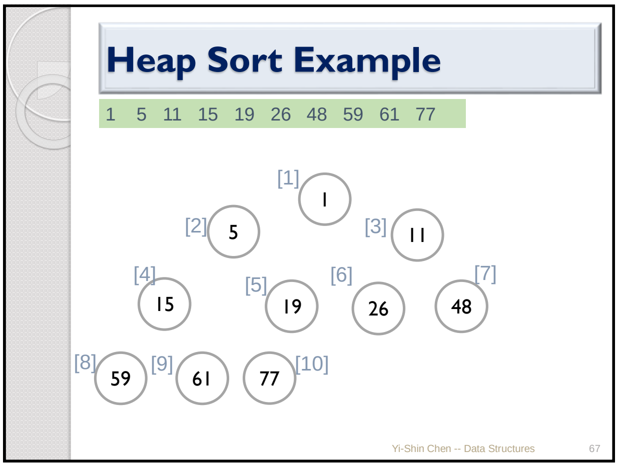


Yi-Shin Chen -- Data Structures 63


Heap Sort Example
15 5 11 1
19 26 48 59 61 77
[1] 15
[2] 5
[3] 11
[4] 1
[5] 19
[6] 26
[7] 48
[8] 59
[9] 61
[10] 77
Yi-Shin Chen -- Data Structures 64


Heap Sort Example
15 5 1
15 19 26 48 59 61 77
[1] 11
[2] 5
[3] 1
[4] 15
[5] 19
[6] 26
[7] 48
[8] 59
[9] 61
[10] 77
Yi-Shin Chen -- Data Structures 65


Heap Sort Example
5 1
11 15 19 26 48 59 61 77
[1] 5
[2] 1
[3] 11
[4] 15
[5] 19
[6] 26
[7] 48
[8] 59
[9] 61
[10] 77
Yi-Shin Chen -- Data Structures 66


Heap Sort Example
1 5 11 15 19 26 48 59 61 77
[1] 1
[2] 5
[3] 11
[4] 15
[5] 19
[6] 26
[7] 48
[8] 59
[9] 61
[10] 77
Yi-Shin Chen -- Data Structures
67