Review of Binary Trees

- What is a binary tree?
- What is the depth of the node?
- What is the height of a node?
- What is the height of the tree?
- What is a complete binary tree?
Facts about Perfect Binary Trees

- How many nodes at depth $d$?
- How many nodes $n$ in perfect tree of height $h$?
- If $n$ is number of nodes, what is corresponding height $h$?

Complete Binary Trees

- Nodes at depth $h$ (the lowest level) are as far left as possible
- What is the relationship between the height and the number of nodes?
Heaps

- A **heap** is a complete binary tree
- Extra nodes go from left to right at the lowest level
  - Where the value at each node is ≥ the values at its children (if any)
  - This is called the **heap property** for max-heaps
- Max or Min Heap

Example

```
       16
       /
      /  
     2   3
    /
   14
  /
  8
 / 
4  2
/ 
8 9
```

```
       10
       /
      /  
     6   7
    /
   9
  /
  3
```

```
       7
       /
      /  
     9   3
    /
   1
  /
  2
```
Storing Heaps

- As arrays!
- Root of tree is:
- Parent of $A[i]$ is:
- Left child of $A[i]$ is:
- Right child of $A[i]$ is:

Example
Example

- \( n = 13 \)
  
  92  85  73  81  44  59  64  13  23  36  32  18  54

Functions on Heaps

- MAX-HEAPIFY
- BUILD-MAX-HEAP
- HEAPSORT
- MAX-HEAP-INSERT
- HEAP-EXTRACT-MAX
- HEAP-INCREASE-KEY
- HEAP-MAXIMUM
**MAX-HEAPIFY, p 154**

Max_Heapify(A, i) // A: Array, i: int

```plaintext
1  int L = left(i)
2  int R = right(i)
4    largest = L
5  else Largest = i
6  if (Right <= A.heap_size and A[R] > A[largest])
7    largest = R
8  if largest != i
9    swap ( A[i], A[largest] )
10   Max_Heapify(A, largest)
```

**Example: (6.2)**

Issue: Doesn’t satisfy MAX-HEAP property, so call MAX-HEAPIFY(A,2)
Example: (6.2)

Consider indices of children: \( L = \text{left}(2) = 4 \), \( R = \text{right}(2) = 5 \)

Call MAX-HEAPIFY(A,largest) = MAX-HEAPIFY(A,4)
Example: (6.2)

Consider indices of children: L = left(4) = 8, R = right(4) = 9
Call MAX-HEAPIFY(A,largest) = MAX-HEAPIFY(A,9)

Example

\[ 15 \ 6 \ 4 \ 8 \ 5 \ 3 \ 1 \ 2 \ 7 \ i = 2 \]
Analysis: MAX-HEAPIFY(A,i)

- Space Complexity?

- Big-Oh?

---

Build_Max_Heap, p 157

Build_Max_Heap (A) // A: Array

1. A.heap_size = A.length
2. for i = floor ( A.length/2) to 1
3. Max_Heapify(A,i)
Example

- 4 3 7 13 1 20 12 16 2 18

---

HeapSort, p 160

HeapSort(A)  //  A: Array

1  Build_Max_Heap(A)
2  for i = A.length to 2
3  swap(A[1], A[i])
4  A.heap_size = A.heap_size - 1
5  Max_Heapify(A, 1)
Example

- 20 18 12 16 3 7 4 13 2 1

Next Time

- Priority Queue