Heap and PQ

by Dr. Sethavidh Gertphol

Outline

* Priority Queue
* Array representation of binary tree
* Heap
* Heap Sort

Priority Q

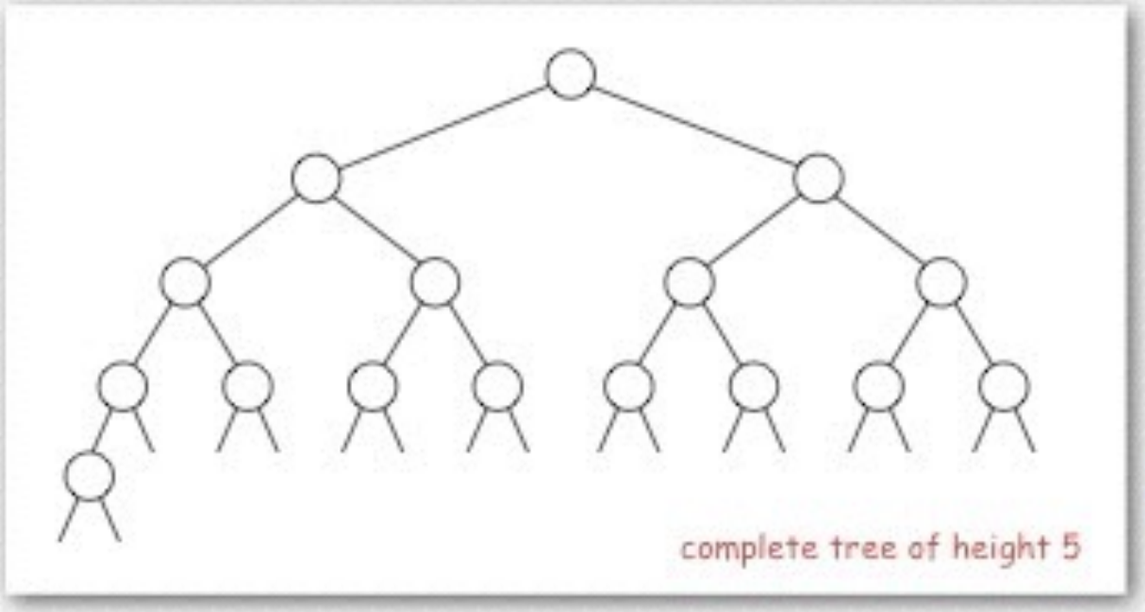
* Q: FIFO
  * using list or array, EnQ(x)/DeQ() ... both O(1)

* PQ: not FIFO
  * DeQ(): get maximum (or minimum) key out
  * using linked list or array
  * EnQ(x): O(1) / DeQ(): O(n) ... unordered

Complete Binary Tree

Binary tree. Empty or node with links to left and right binary trees.

Complete tree. Perfectly balanced, except for bottom level.



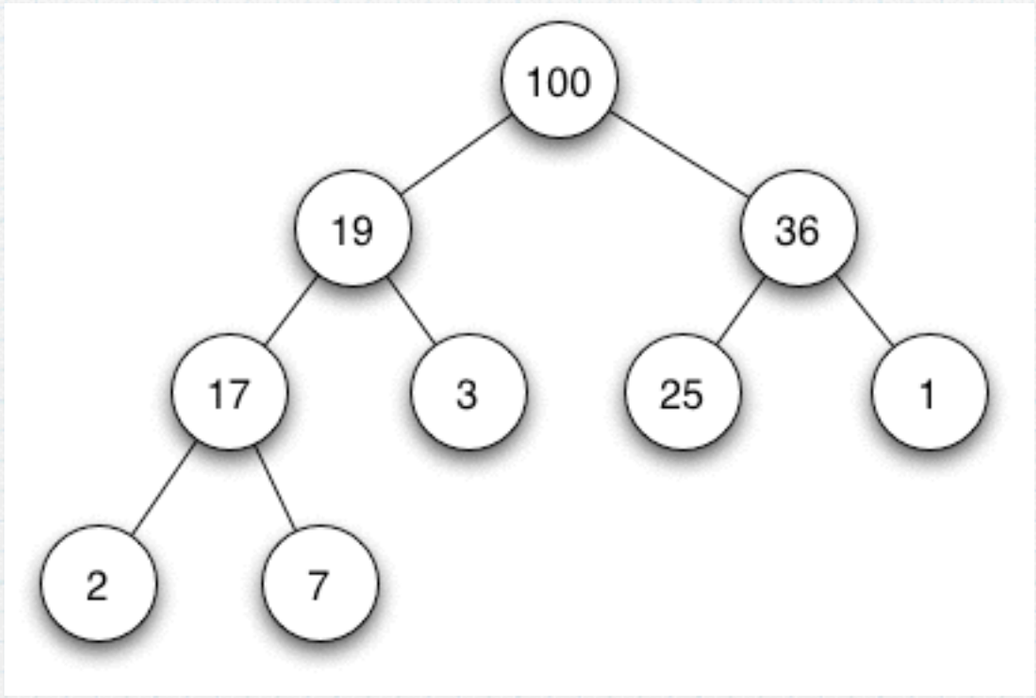
N = 16
⌊lg N⌋ = 4
height = 5

Property. Height of complete tree with N nodes is 1 + ⌊lg N⌋.

Pf. Height only increases when N is exactly a power of 2.

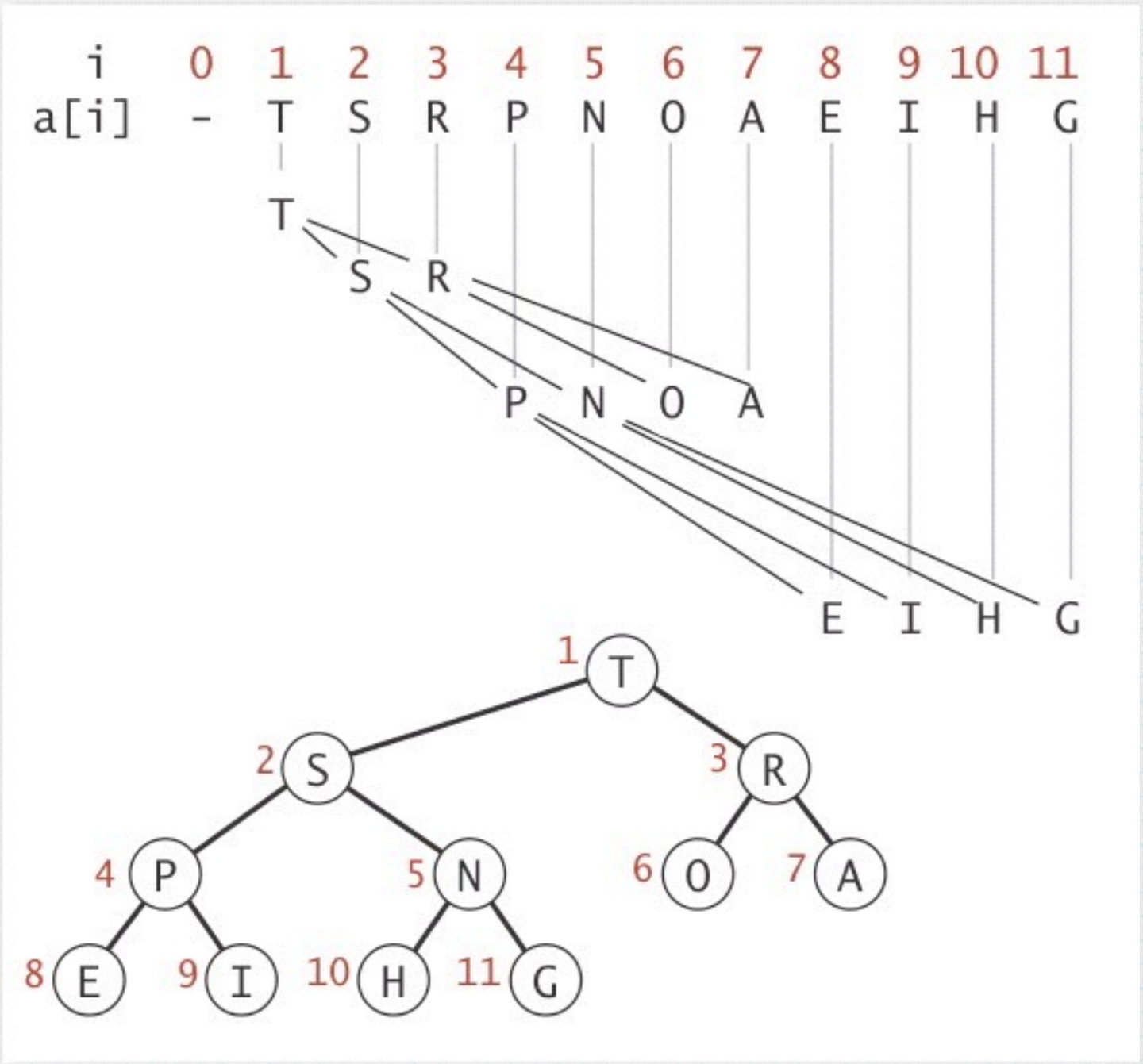
Binary Heap

* a complete binary tree with structure
* parent's key is always greater than children's (root is max)



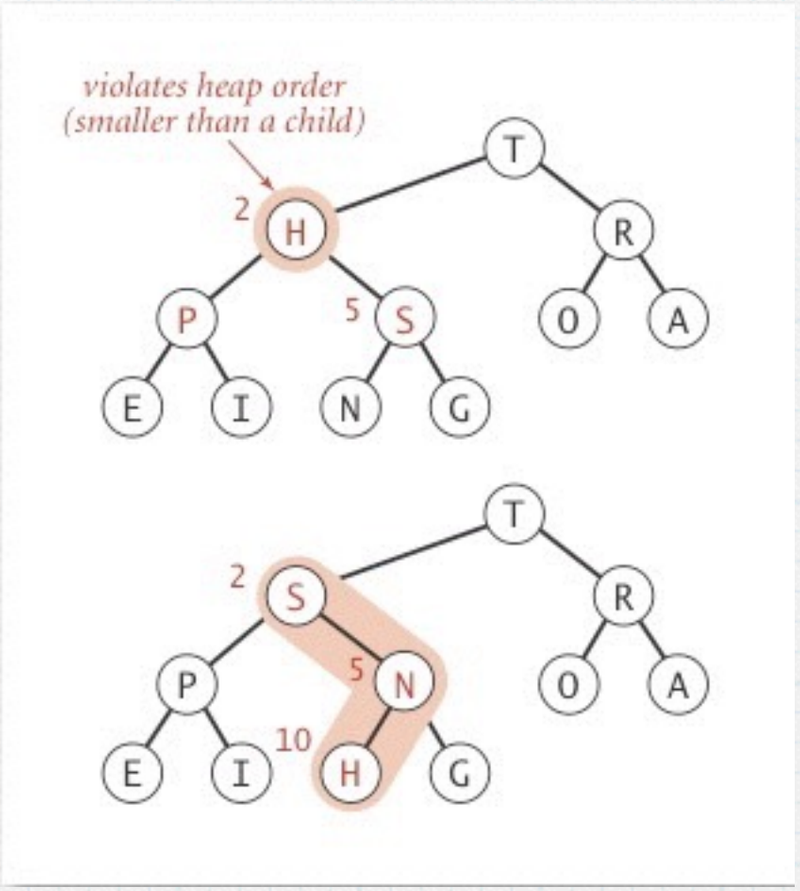
Array representation

* parent at x
  * left(x) = 2x
  * right(x) = 2x+1
* child at i
  * parent(i) = i/2 (rounded down)
  * A[parent(i)] > A[



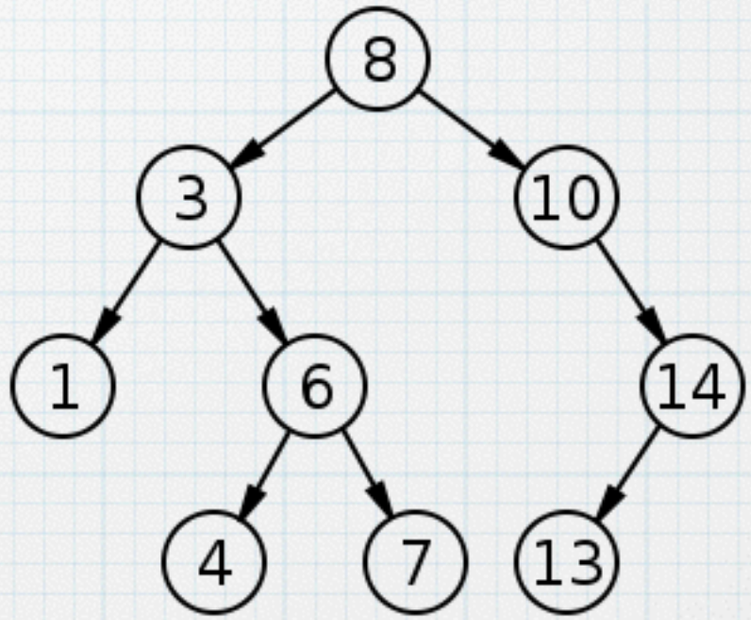
Heap operations

* sink: create heap from 2 sub-heaps
* top-of-heap may not be largest yet
  * exchange with larger child
  * recursively call heapify



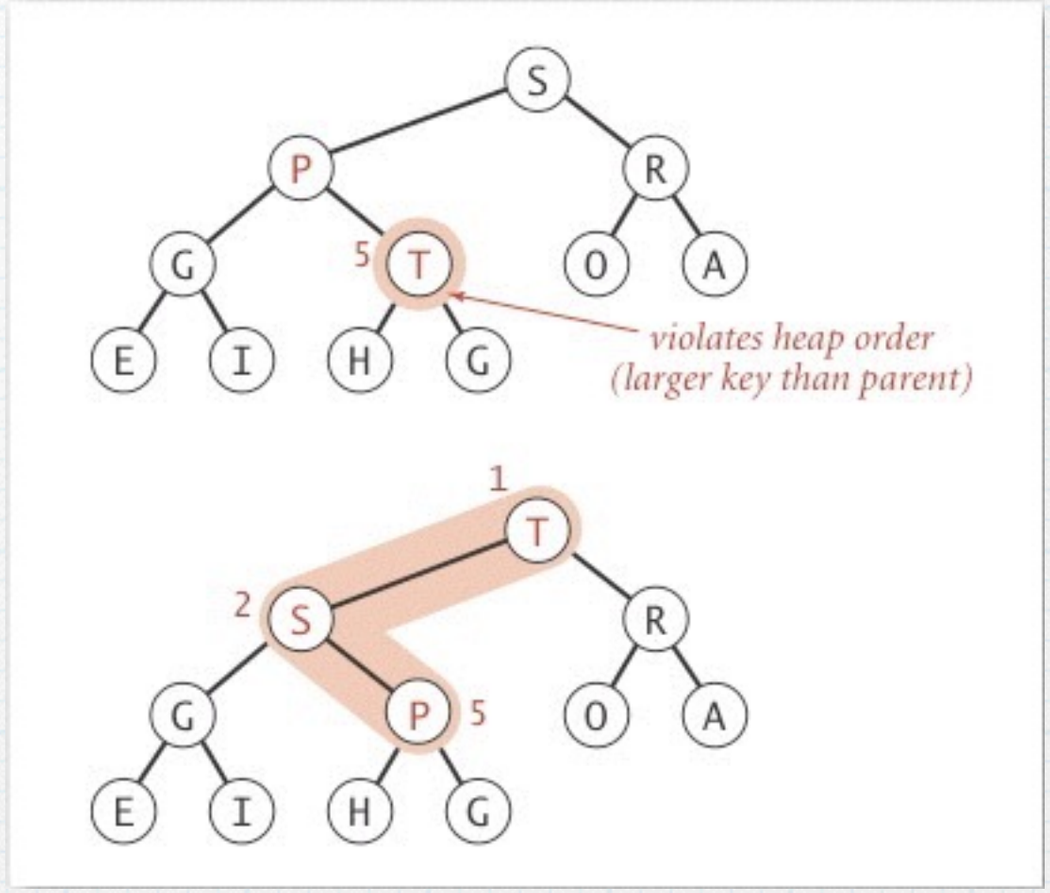
Build-heap

* give unordered array, create heap
* sink nodes recursively
  * start from next-to-last depth is enough
  * sink from A[|A|/2] to A[1]
* O(n)
  * sink uses O(lg n), n/2 times
  * but height of heap not lg n from start



Float heap

* reverse of sink
  * one child is larger than parent
* swap with parent, recursively called



Heap insert/extract

* insert: O(lg n)
  * at last free position in array
  * then float up
* extract: O(lg n)
  * remove top of heap
  * move last element to top of heap
  * sink it down

PQ implementation

methods\ops	EnQ	Max	ExtractMax
unordered array	O(1)	O(n)	O(n)
ordered array	O(n)	O(1)	O(1)
BST	O(lg n)	O(lg n)	O(lg n)
Heap	O(lg n)	O(1)	O(lg n)

Heap Sort

* sorting elements in heap "in place"
  * switch A[1] with A[n]
  * now Max value is in place, n--
  * sink(A[1])
  * repeat
* O(n lg n)

PQ in C++ STL

* priority_queue container (heap inside)
* methods:
  * push/pop
  * top
  * size/empty

Example

```
#include <iostream>          // std::cout
#include <queue>             // std::priority_queue
int main ()
{
  std::priority_queue<int> mypq;
  mypq.push(30);
  mypq.push(100);
  mypq.push(25);
  mypq.push(40);

  std::cout << "Popping out elements...";
  while (!mypq.empty())
  {
     std::cout << ' ' << mypq.top();
     mypq.pop();
  }
  std::cout << '\n';

  return 0;
}
```

output

```
Popping out elements... 100 40 30 25
```