University of Southern California

Viterbi School of Engineering

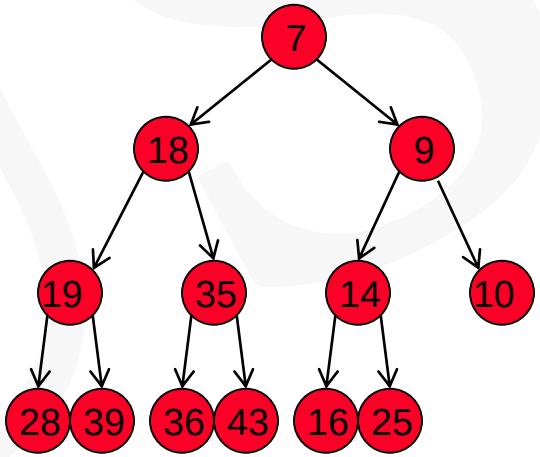
Software Design

Data Structures – Heaps

Reference: Professor Redekopp' EE355 slide units, online Resources

Heap Data Structure

- Can think of heap as a full binary tree (i.e., only the lowest-level contains empty locations and items added left to right) with the property that every parent is less-than (if min-heap) or greater-than (if max-heap) both children
  - But no ordering property between children
- Minimum/Maximum value is always the top element



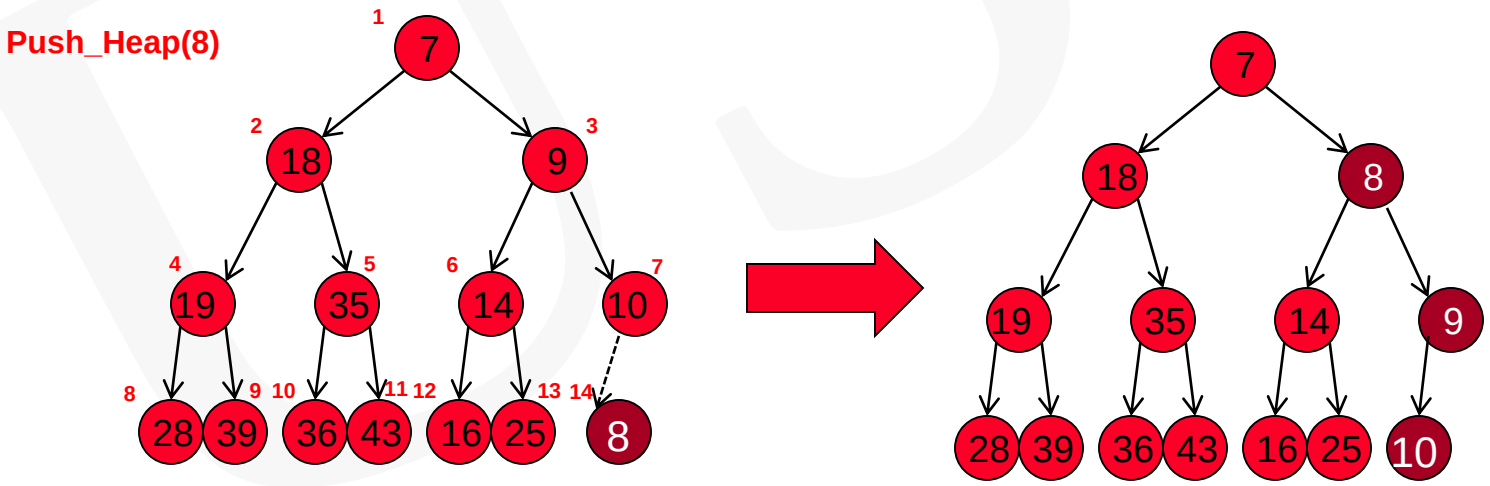
Min-Heap

Heap Operations

- Push: Add a new item to the heap and modify heap as necessary
- Pop: Remove min/max item and modify heap as necessary
- Top: Returns min/max
- To create a heap from an unordered array/vector takes $O(n*\log_2 n)$ time while push/pop takes $O(\log_2 n)$

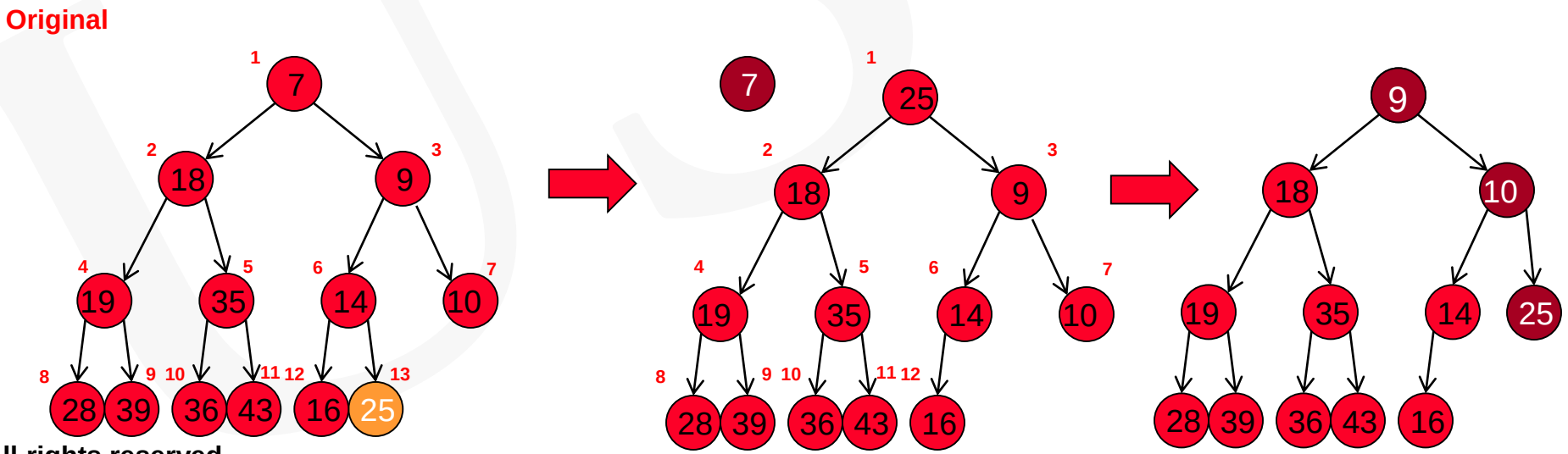
Push Heap

- Add item to first free location at bottom of tree
- Recursively promote it up until a parent is less than the current item



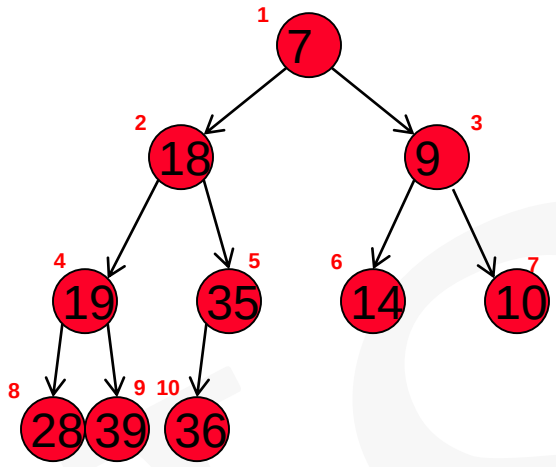
Pop Heap

- Takes last (greatest) node puts it in the top location and then recursively swaps it for the smallest child until it is in its right place

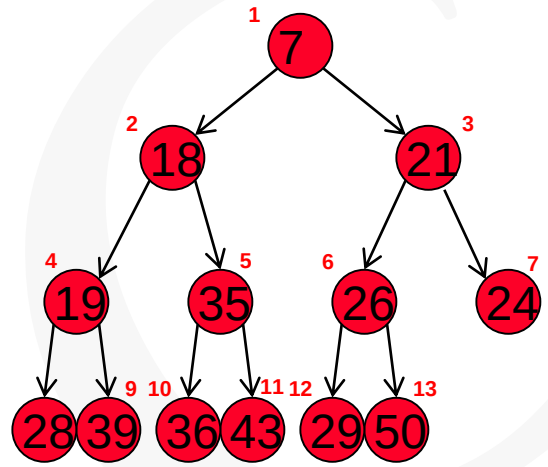


Practice

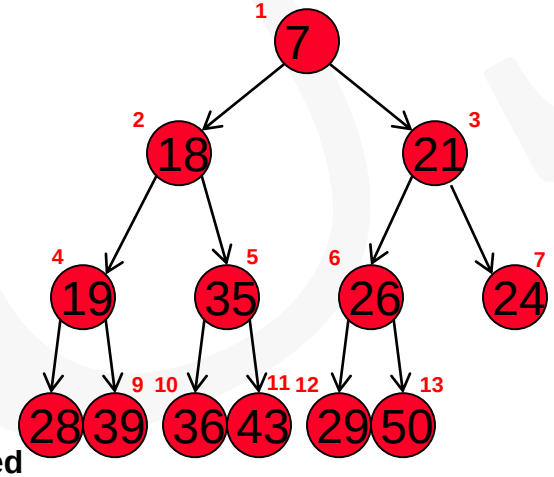
Push(11)



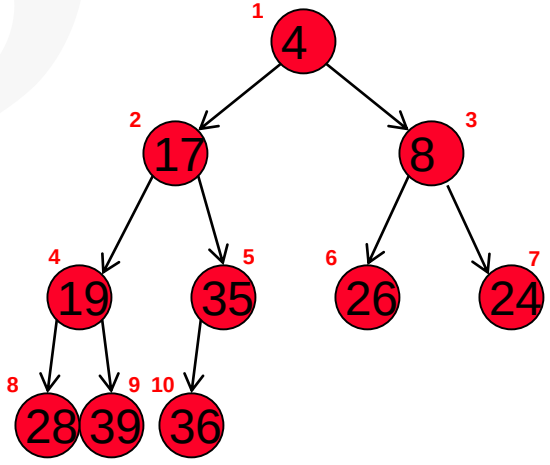
Push(23)



Pop()



Pop()

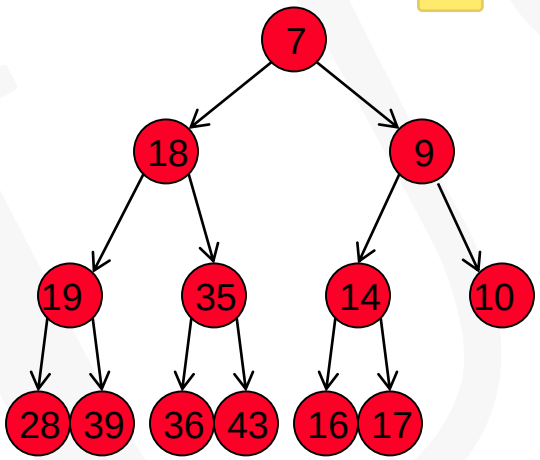


Array/Vector Storage for Heap

- Binary tree that is full can be modeled as an array (let's say it starts at index 1) where:
  - Parent(i) = i/2
  - Left_child(p) = 2*p
  - Right_child(p) = 2*p + 1

Min Heap:

$A[Parent(i)] \leq A[i]$



0	1	2	3	4	5	6	7	8	9	10	11	12	13
em	7	18	9	19	35	14	10	28	39	36	43	16	17

parent(5) = 5/2 = 2

Left_child(5) = 2*5 = 10

Right_child(5) = 2*5+1 = 11

STL Priority Queue

- std::priority_queue
- Defined in header <queue>

```
template <  class T,
    class Container = std::vector<T>,
    class Compare = std::less<typename Container::value_type>
> class priority_queue;
```

STL Priority Queue (cont.)

- Implements a max-heap by default
- Operations:
  - push(new_item)
  - pop(): removes but does not return top item
  - top() return top item (item at back/end of the container)
  - size()
  - empty()

```
// priority_queue::push/pop
#include <iostream>
#include <queue>

using namespace std;

int main ()
{
  priority_queue<int> mypq;
  mypq.push(30);
  mypq.push(100);
  mypq.push(25);
  mypq.push(40);
  cout << "Popping out elements...";
  while (!mypq.empty()) {
    cout<< " " << mypq.top();
    mypq.pop();
  }
  cout<< endl;
  return 0;
 }
```

Code here will print
100 40 30 25

Exercise

mypq
pqref

Question:
Why pass by C++ ref. to callee with a pop() does not reduce the size of pq in caller?

STL Priority Queue Template

- Template that allows type of element, container class, and comparison operation for ordering to be provided
- First template parameter should be type of element stored
- Second template parameter should be vector<type_of_elem>
- Third template parameters should be comparison function object/class that will define the order from first to last in the container

```
// priority_queue::push/pop
#include <iostream>
#include <queue>

using namespace std;

intmain ()
{
priority_queue<int, vector<int>, greater<int> > mypq;
mypq.push(30); mypq.push(100); mypq.push(25);
cout<< "Popping out elements...";
while (!mypq.empty()) {
  cout<< " " << mypq.top();
  mypq.pop();
  } }
```

'greater' will yield a min-heap
'less' will yield a max-heap

Code here will print
25, 30, 100

STL Priority Queue Template (cont.)

- For user defined classes, must implement

  operator<() for max-heap or
  operator>() for min-heap

Debug pqoperator.cpp