WANT MAX? ASK HEAP

Problem Solving with Computers-II

Make a copy of the handout for today's lecture

https://bit.ly/cs24-lect14-handout

What is **mystery** doing ? (2 min)

```
void mystery(vector<int>& v){
    int n = v.size();
    for (int i = 0; i < n; i++){
        int index = i;
        for (int j = i + 1; j < n; j++){
            if(v[j] > v[index]){
                index = j;
            }
        }
        if(index != i){
            int temp = v[index];
            v[index] =  v[i];
            v[i] = temp;
        }
    }
}
```

Example input:

20	5	7	1	3	2

What is the time and space complexity of mystery? (2 min)

```
void mystery(vector<int>& v){
    int n = v.size();
    for (int i = 0; i < n; i++){
        find max of vector: v[i:n]

        swap v[i] with max element

    }
}
```

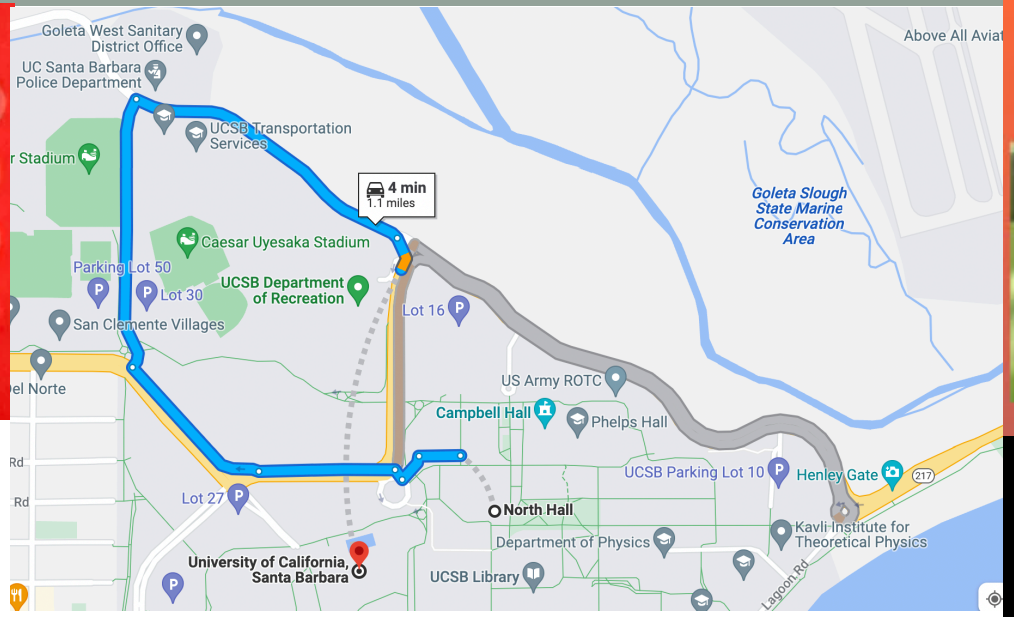
Brainstorm ideas to improve the running time. (3 min)

```
void mystery(vector<int>& v){
    int n = v.size();
    for (int i = 0; i < n; i++){
        int index = i;
        for (int j = i + 1; j < n; j++){
            if(v[j] > v[index]){
                index = j;
            }
        }
        if(index != i){
            int temp = v[index];
            v[index] =  v[i];
            v[i] = temp;
        }
    }
}
```

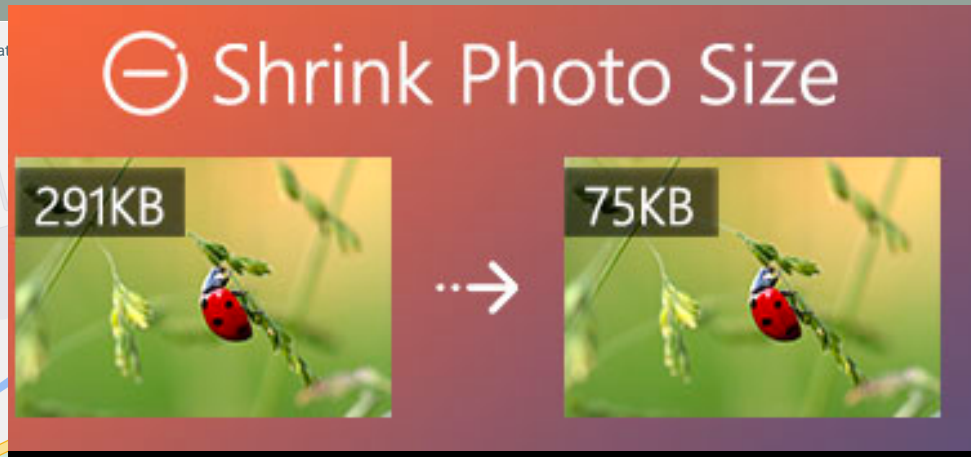
Notice that we are repeatedly finding the max!

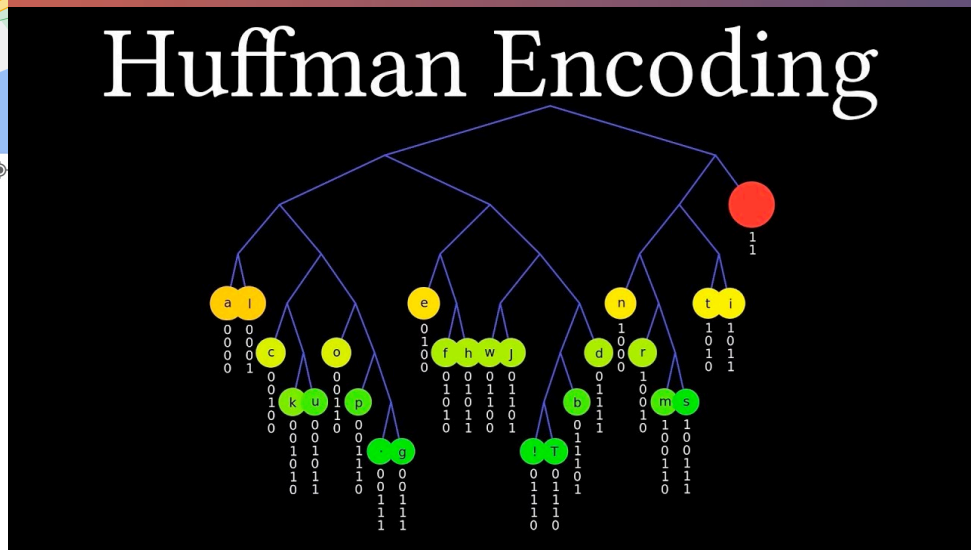
```
void mystery(vector<int>& v){
    int n = v.size();
    for (int i = 0; i < n; i++){
        find max of vector: v[i:n]

        swap v[i] with max element

    }
}
```

Sorting



Shortest Path





Data Compression

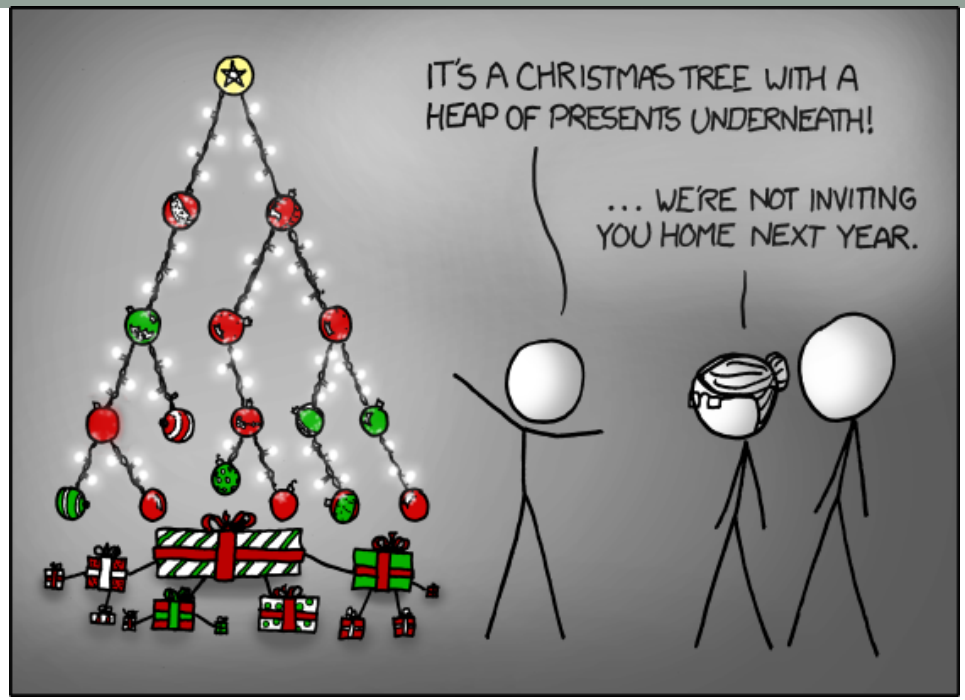
Want min or max? Ask Heap!

Many algorithms need to compute the min OR max repeatedly

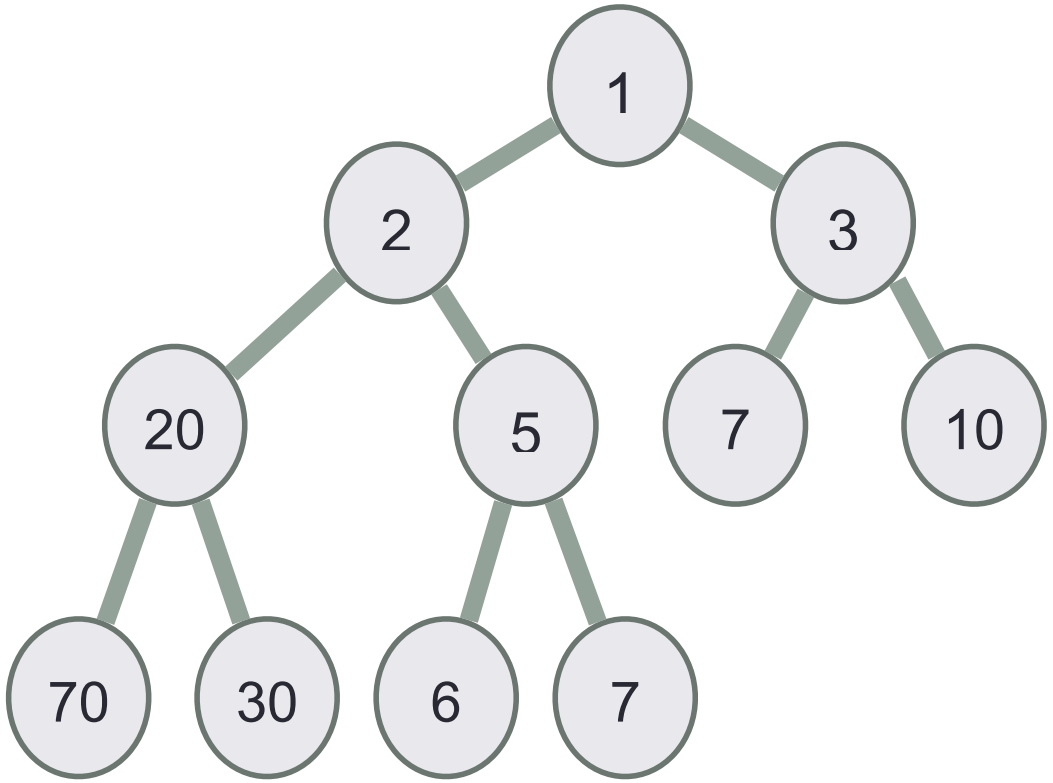
Heap is used speed up the running time of such algorithms!

New data structure: Heap

- Clarification
  - *heap*, the data structure is not related to *heap*, the region of memory
- What are the operations supported?
- What are the running times?

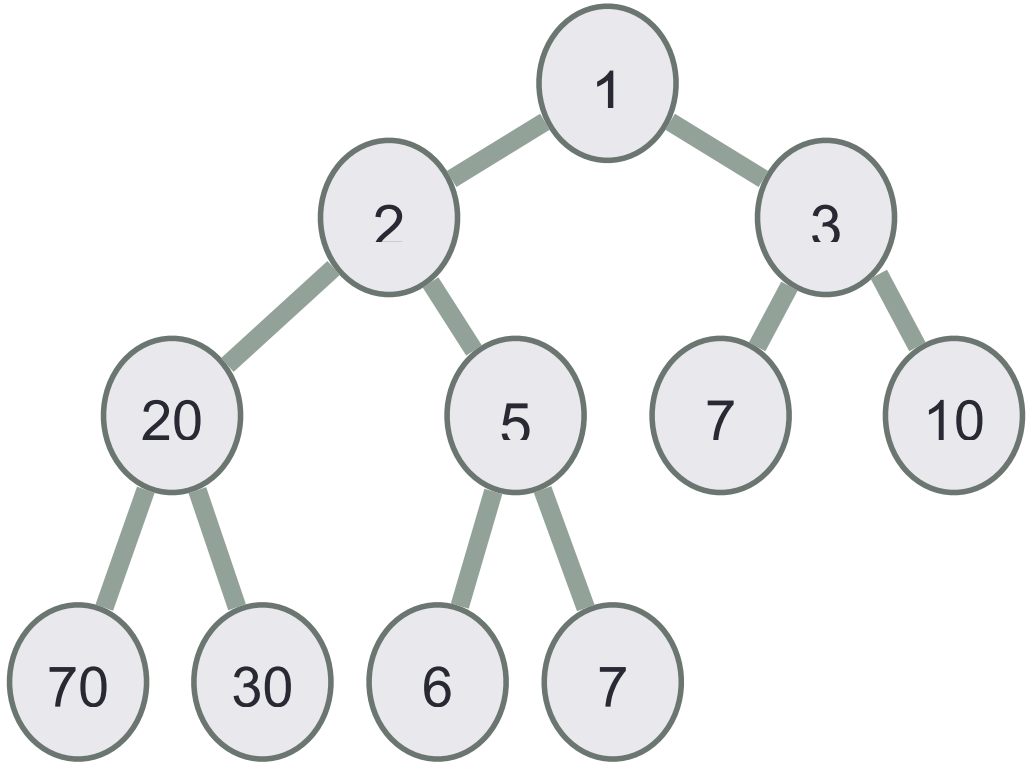


Two important properties of a heap



Shape property:

Heap property :



Shape property:

Internally, a heap is a **complete binary tree,** where each node satisfies the heap property

Heap property :

In a min-heap, for each node (x):

key(x) <= key(children of x)

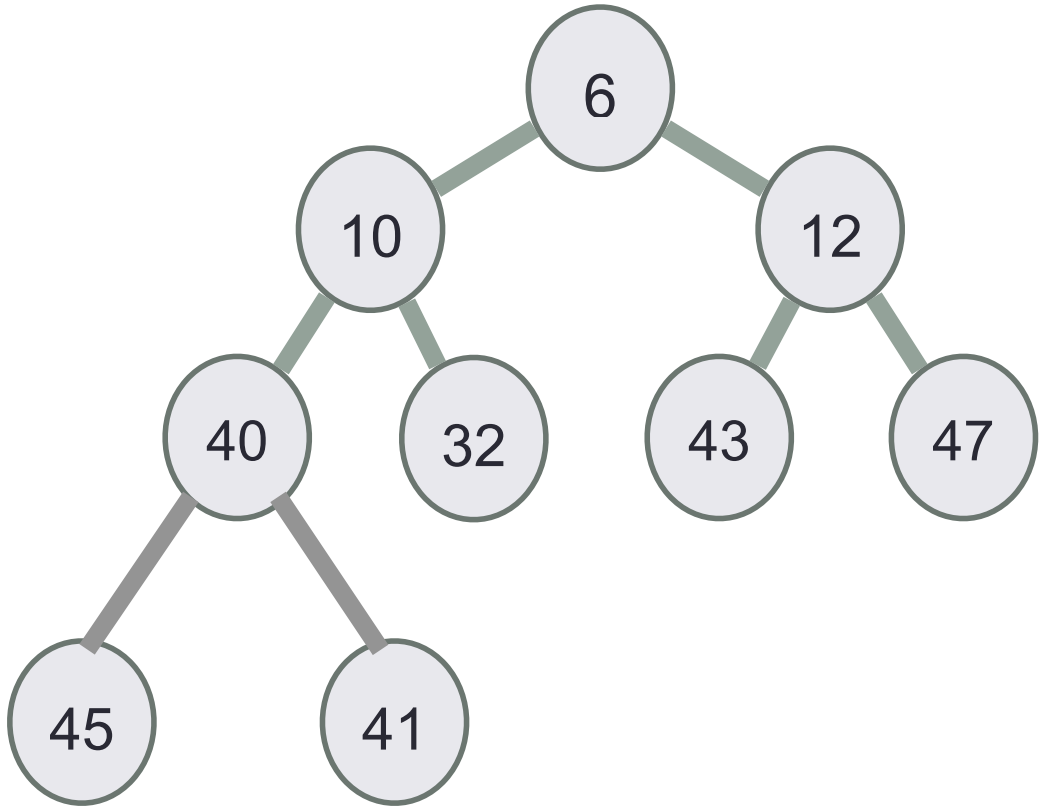
In a max-heap, for each node (x):

key(x) >= key(children of x)

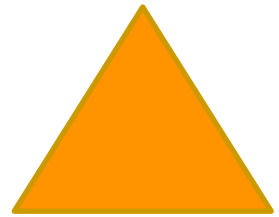
Identifying heaps

Starting with the following min-Heap which of the following operations will result in something that is NOT a min Heap

A. Swap the keys 40 and 32
B. Swap the keys 32 and 43
C. Swap the keys 43 and 40
D. Insert 50 as the left child of 45
E. C&D

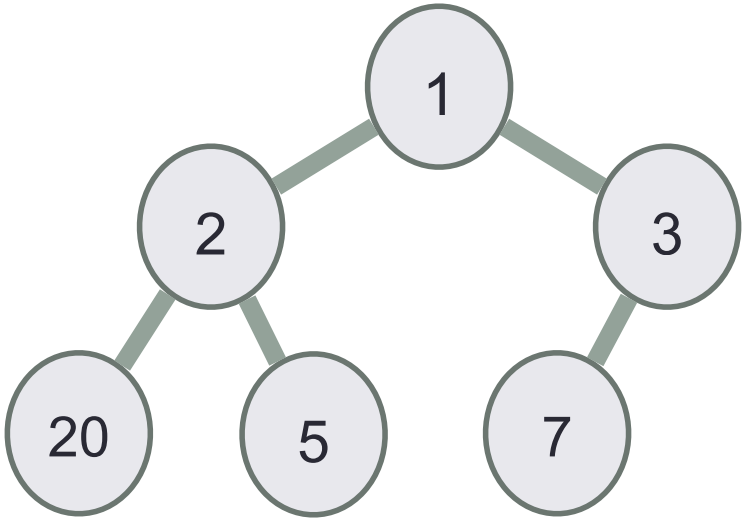


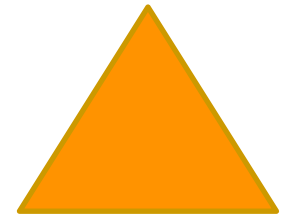
20, 5, 7, 1, 3, 2
push(x)



min-Heap

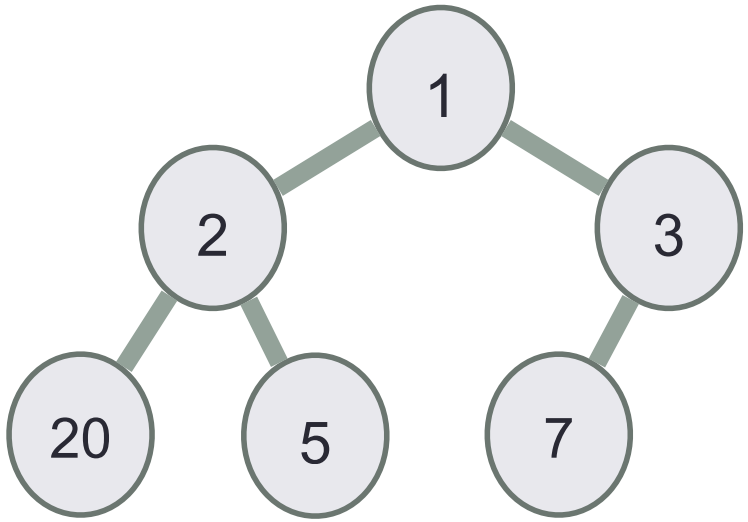
```
procedure push(x: key value)
  insert x in the first open spot in the tree
  while(x has a parent && x < parent(x)):
    swap(x, parent(x))
  return {x was inserted into a min-heap}
```





min-Heap

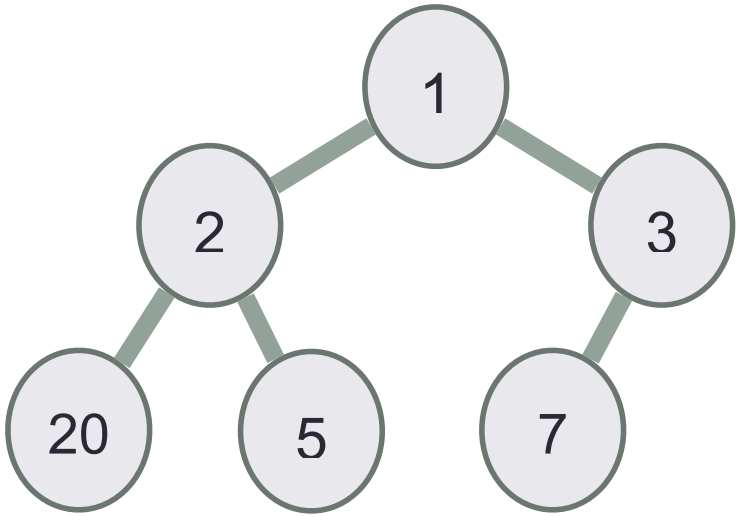
```
procedure top()
   return key of root node {top element is returned}
```

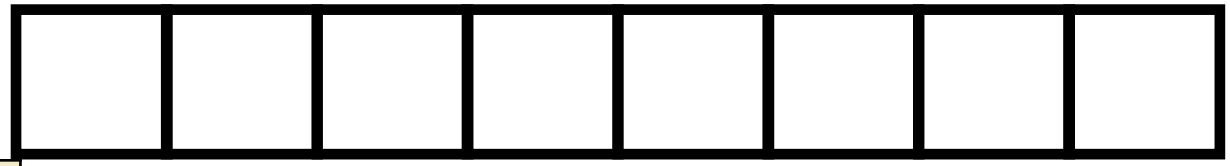


20, 5, 7, 1, 3, 2
pop()

```
procedure pop()


  return {key on top of the heap is deleted}
```

Internally the “heap binary tree” is really just a vector!





Index of key
Index of parent
Index of left child
Index of right child

Work to complete the table on page 5 on your handout

Repeat the exercise on page 4 of your handout to insert the values 20, 5, 7, 1, 3, 2 into an initially empty min-heap. But instead of drawing the results as a tree, draw the resulting vector

```
procedure push(x: key value)
  insert x in the first open spot in the tree
  while(x has a parent && parent(x) > x):
    swap(x, parent(x))
  return
```

Next lecture

STL implementation of heap : priority_queue

Configuring priority_queue in different ways