

WANT MAX? ASK HEAP

Problem Solving with Computers-II



```
#include <iostream>
using namespace std;

int main(){
    cout<<"Hola Facebook\n";
    return 0;
}
```



Make a copy of the handout for today's lecture

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

How is PA 2 going?

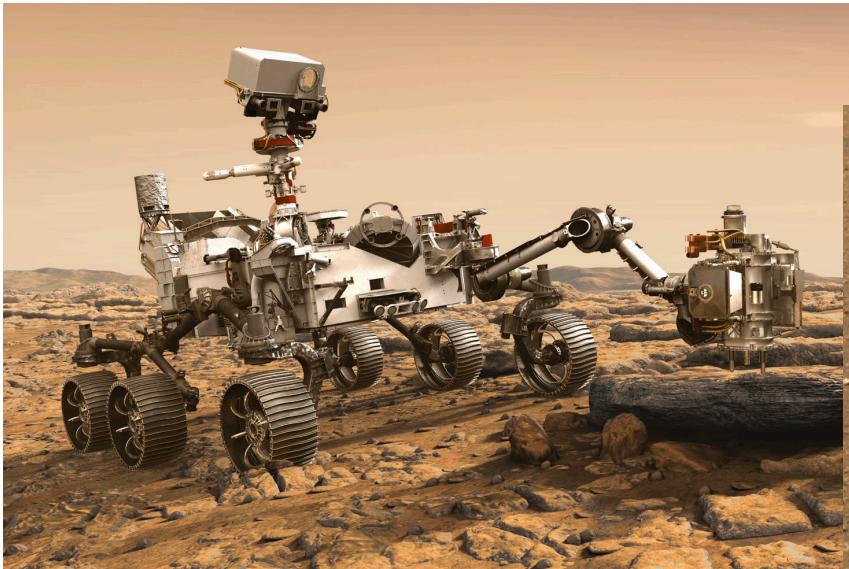
- A. Done!
- B. On track to finish and having fun!
- C. On track to finish but struggling (a bit).
- D. Falling behind and struggling a lot
- E. Haven't read the assignment.

Make a copy of the handout for today's lecture
<https://bit.ly/cs24-lect14-handout>

I can deal with pressure, and deadlines.



Interview Practice!



Perseverance

landed on Mars on Feb 18, 2021

Ingenuity

Flew over Mars
on April 19, 2021

Spirit and Opportunity

Mars Exploration: 2004

What is the running time of the **inner loop**? (1 min)

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

What is the **inner loop** doing (in plain English)?

What is **mystery** doing ?

What is its space and time complexity?

(2 min)

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

Example input:

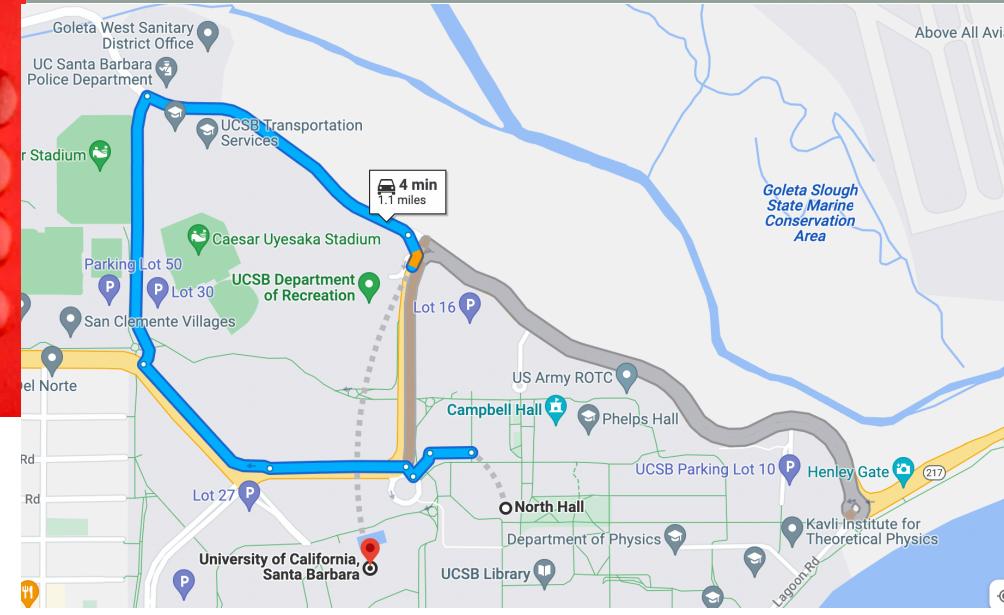
20, 5, 7, 1, 3, 2

Write your ideas to improve the running time of mystery (3 min)

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



Sorting



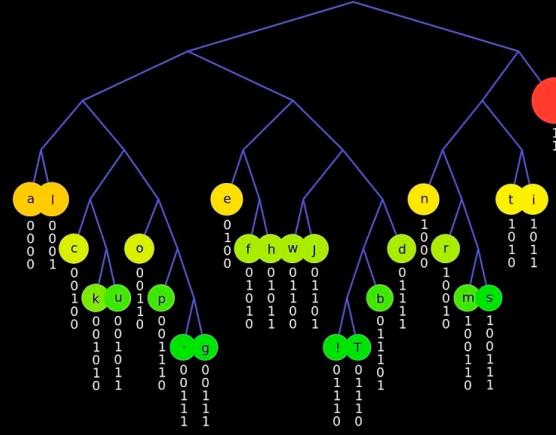
Shortest Path

Want min or max? Ask Heap!

⊖ Shrink Photo Size



Huffman Encoding

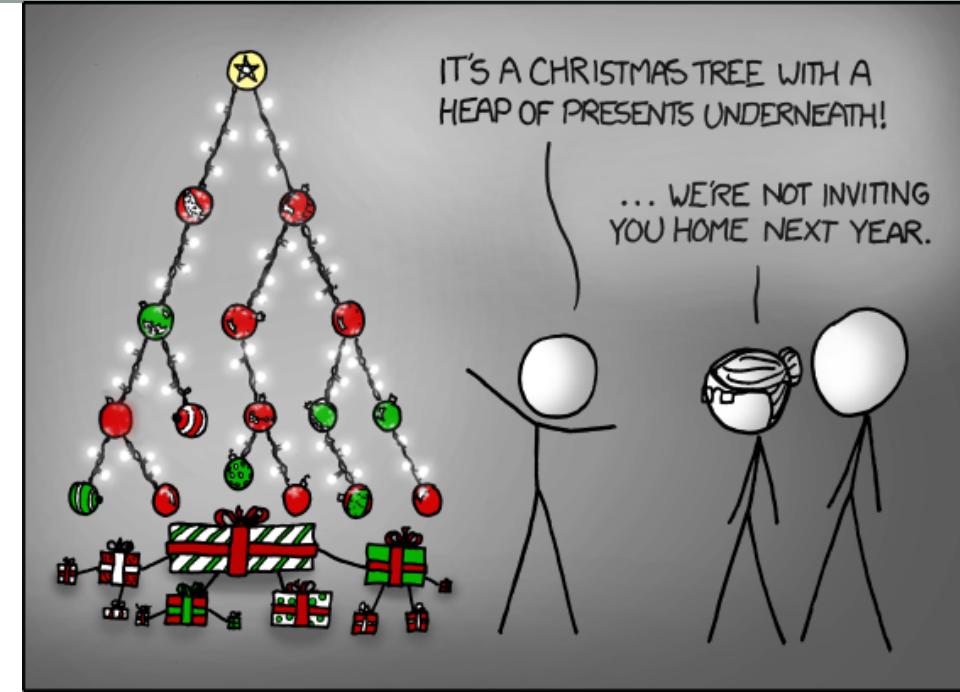


Data Compression

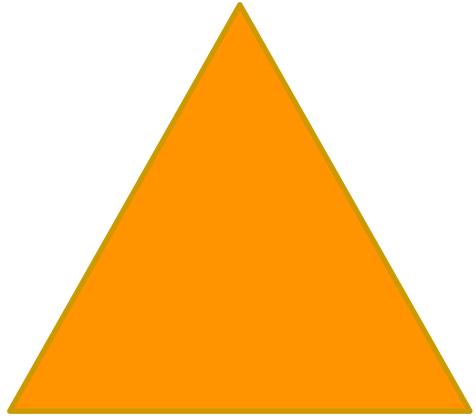
Many algorithms need to compute the min OR max repeatedly
Heap is used speed up the running time of such algorithms!

New data structure: Heap

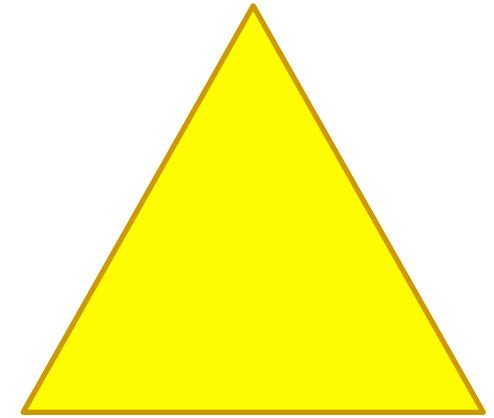
- Clarification
 - *heap*, the data structure is not related to *heap*, the region of memory
- What are the operations supported?
- What are the running times?



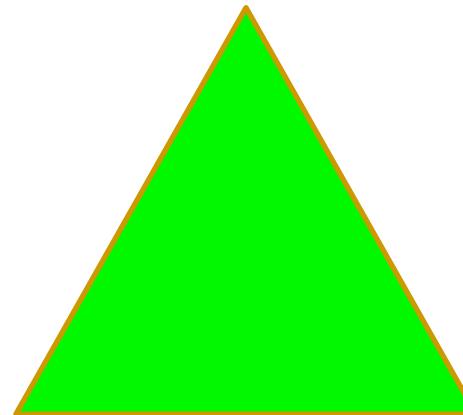
`20, 5, 7, 1, 3, 2`



min-heap



max-heap



`std::priority_queue`

Coding demo

(3 min) Now consider the interview problem again. Provide a faster algorithm to mystery using priority_queue (aka heap data structure). Analyze its space and run-time complexity.

Page 3 of handout

Congrats, you got the job!



Fun fact: Ingenuity's flight control software has C++ framework that provides core capabilities.

Open-Source Flight Software Framework, F-Prime: <https://github.com/nasa/fprime>

Reeves, Glenn E., and Joseph F. Snyder. "An overview of the Mars exploration rovers' flight software." 2005 IEEE International Conference on Systems, Man and Cybernetics. Vol. 1. IEEE, 2005.

20, 5, 7, 1, 3, 2

min-heap

max-heap

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

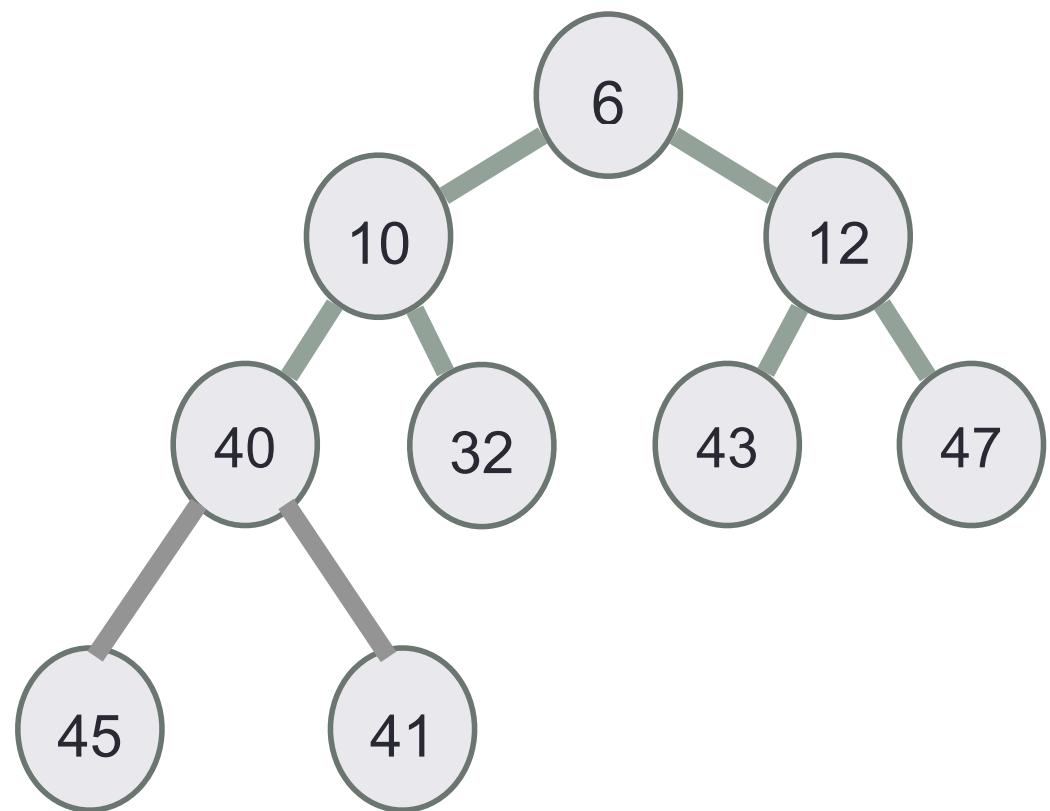
In a **min-heap**, for each node (x):
 $\text{key}(x) \leq \text{key}(\text{children of } x)$

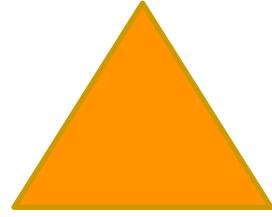
In a **max-heap**, for each node (x):
 $\text{key}(x) \geq \text{key}(\text{children of } x)$

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

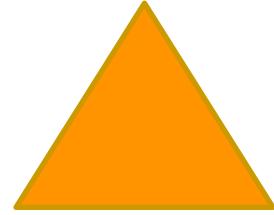




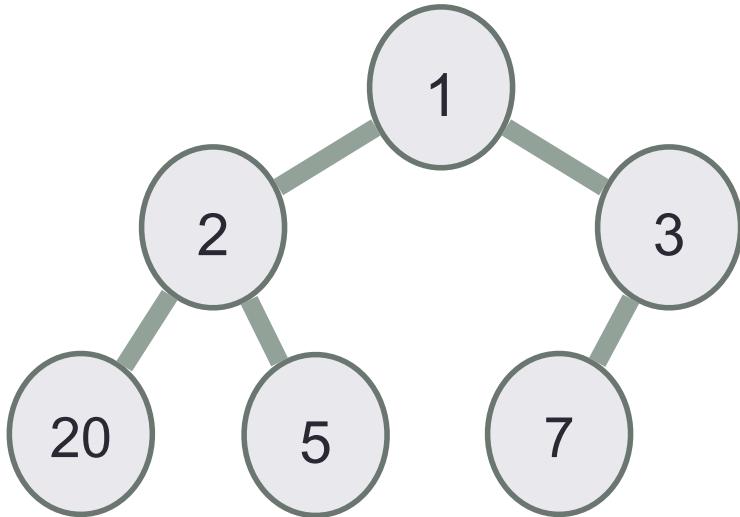
min-Heap

20, 5, 7, 1, 3, 2
push:0(log n)

```
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 {x was inserted into a min-heap}
```

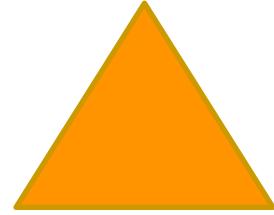


min-Heap

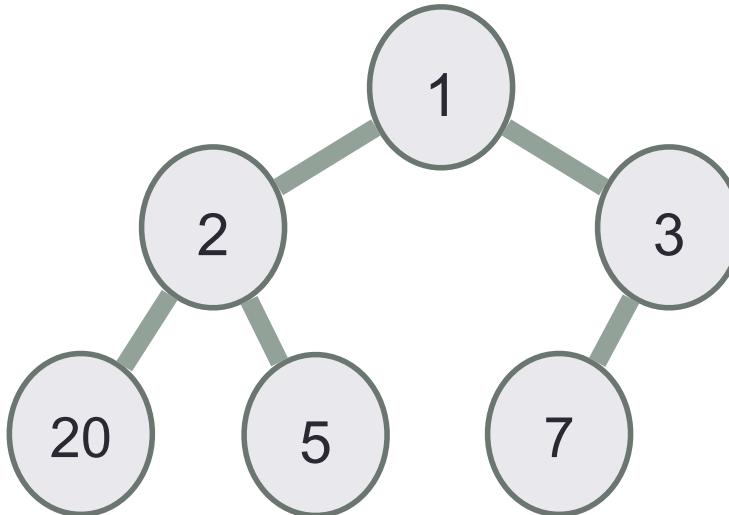


20, 5, 7, 1, 3, 2
top:0(1)

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



min-Heap

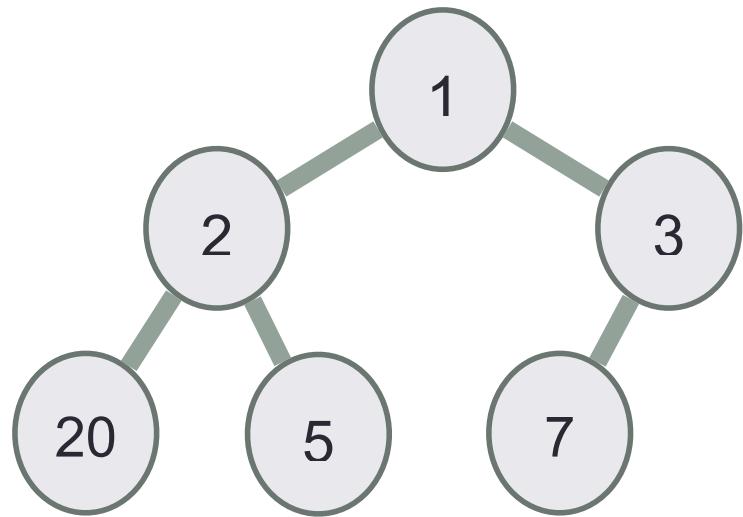


20, 5, 7, 1, 3, 2
pop:0(log n)

```
procedure pop()
```

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

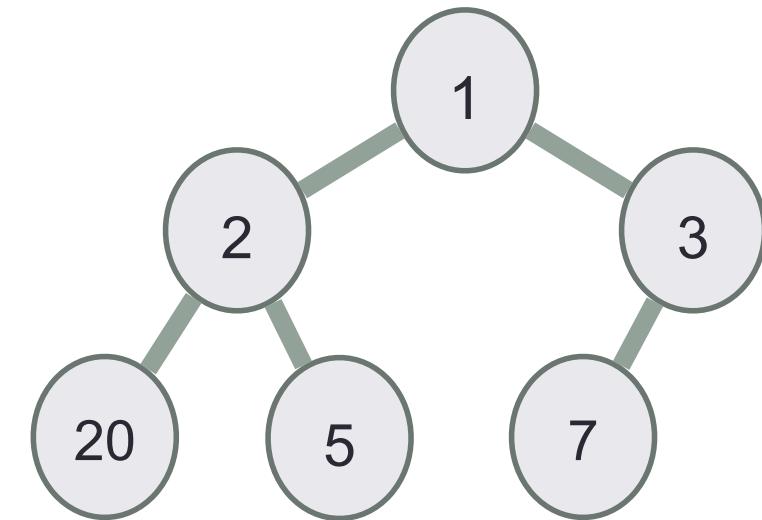
Internally the “heap binary tree” is really just a vector!



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

Work to complete the table on page 6 on your handout

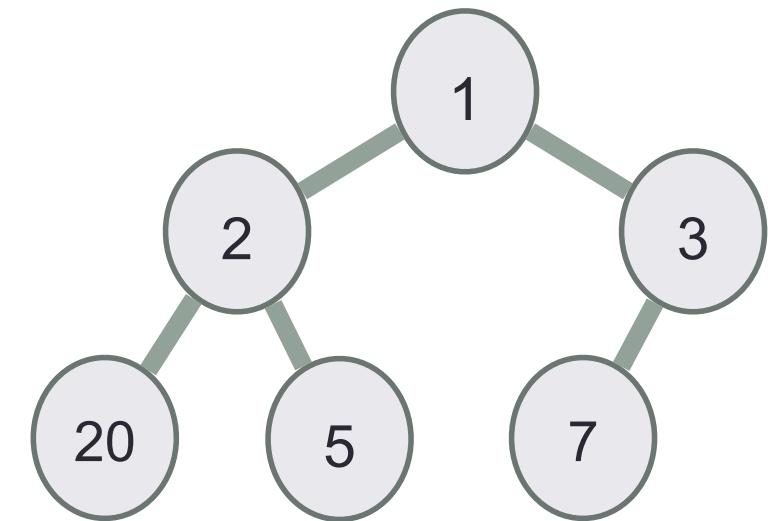
Internally the “heap binary tree” is really just a vector!



Given the index of a key in the vector, how can we find its parent?

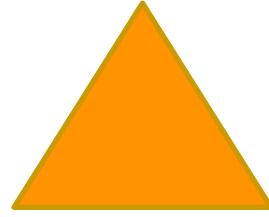
- A. Calculate the index of parent
- B. Start at the root (first node in the vector), and traverse down, checking left and right children of each node until you find the parent
- C. Can't be done: A vector cannot capture parent-child relationships
- D. I am not sure

Under the hood the “binary tree” is really just a vector!



For a key at index i , what is the index of the left and right children?

- A. $(2i, 2i+1)$
- B. $(2i+1, 2i+2)$
- C. $(\log(i), \log(i)+1)$
- D. None of the above



min-Heap

20, 5, 7, 1, 3, 2

The final push!

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

Configuring priority_queue in different ways

Queues and their applications

Heap sort: <https://en.wikipedia.org/wiki/Heapsort>