Data Structures

Heaps

CS 225

October 10, 2023

Brad Solomon & G Carl Evans

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

Department of Computer Science

Learning Objectives

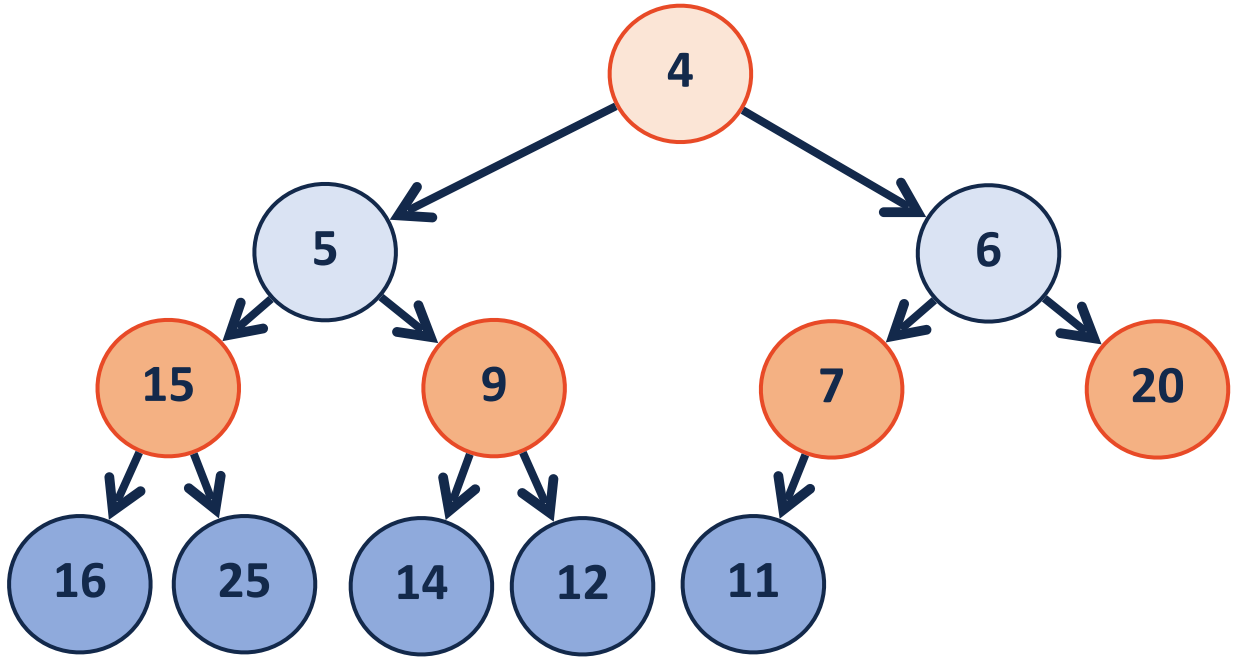
Review heap ADT

Analyze efficiency of minHeap implementations

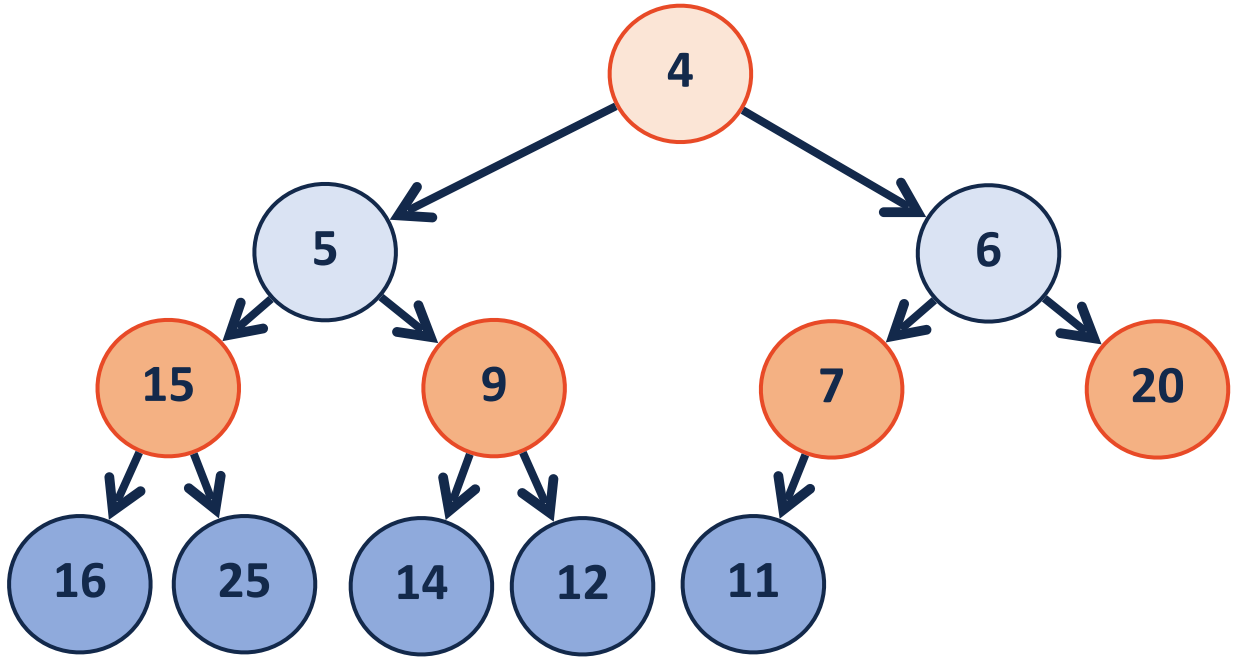
(min)Heap

A complete binary tree T is a min-heap if:

- **T = {}** or
- **T = {r, T_L, T_R}**, where **r** is less than the roots of {T_L, T_R} and {T_L, T_R} are min-heaps.



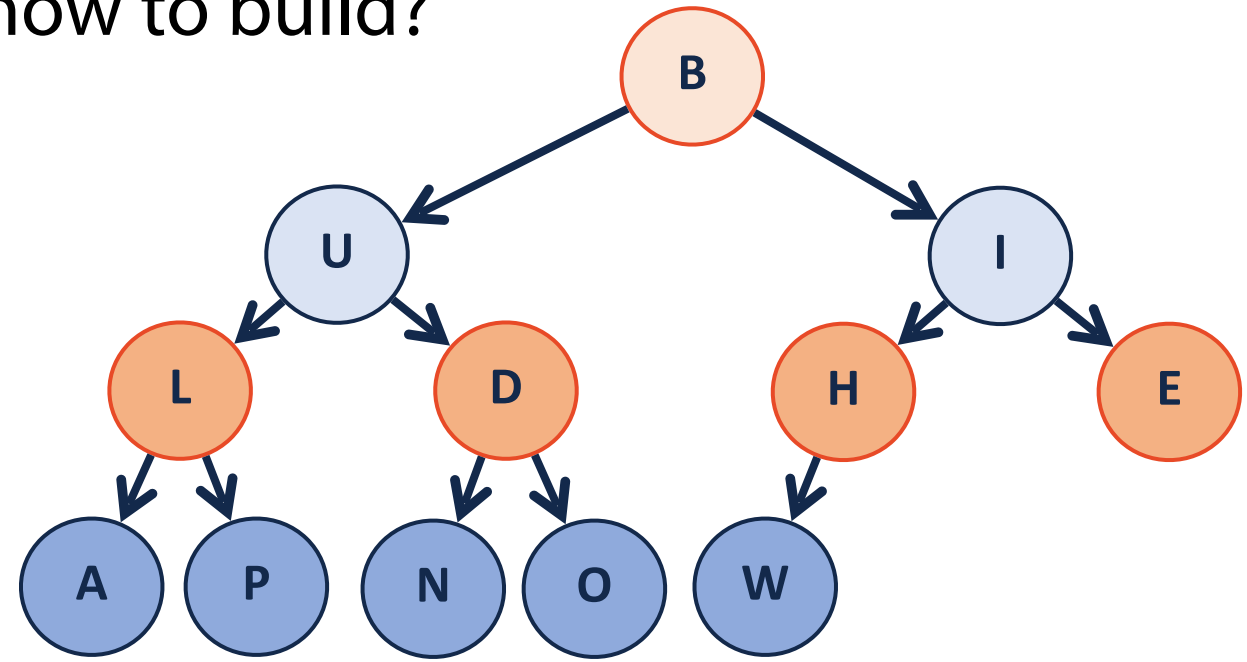
(min)Heap



	4	5	6	15	9	7	20	16	25	14	12	11			
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

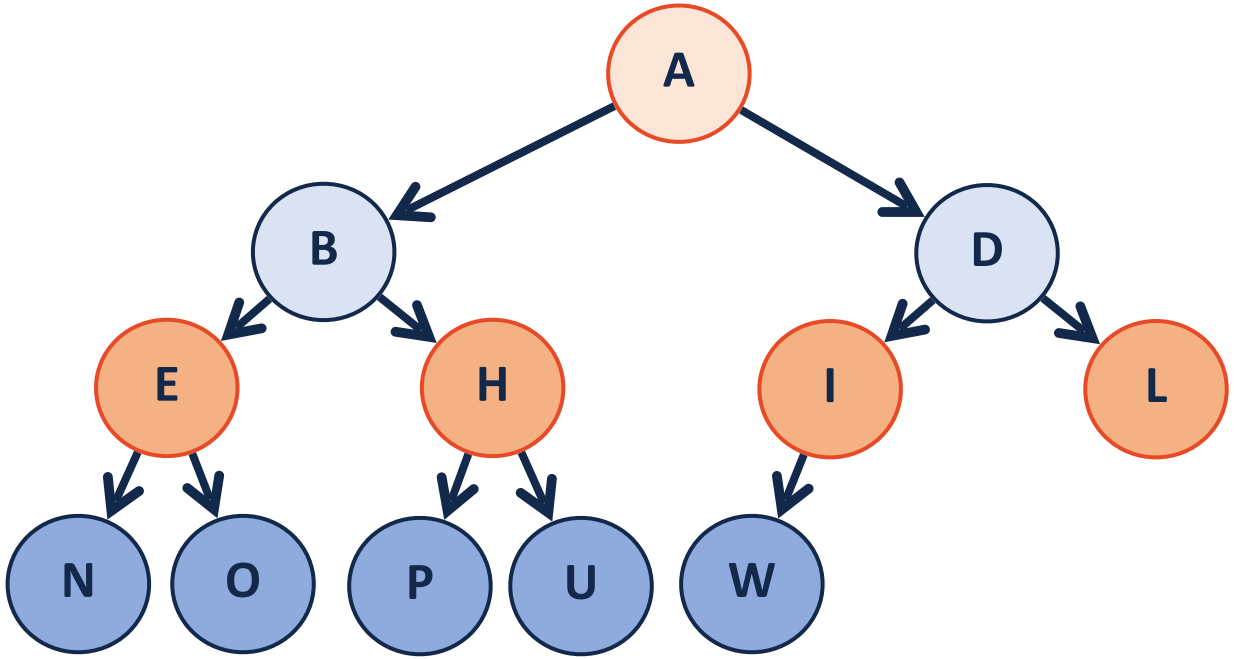
buildHeap (minHeap Constructor)

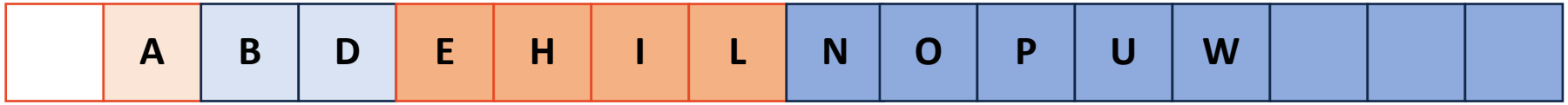
If you give me an array of data, how to build?



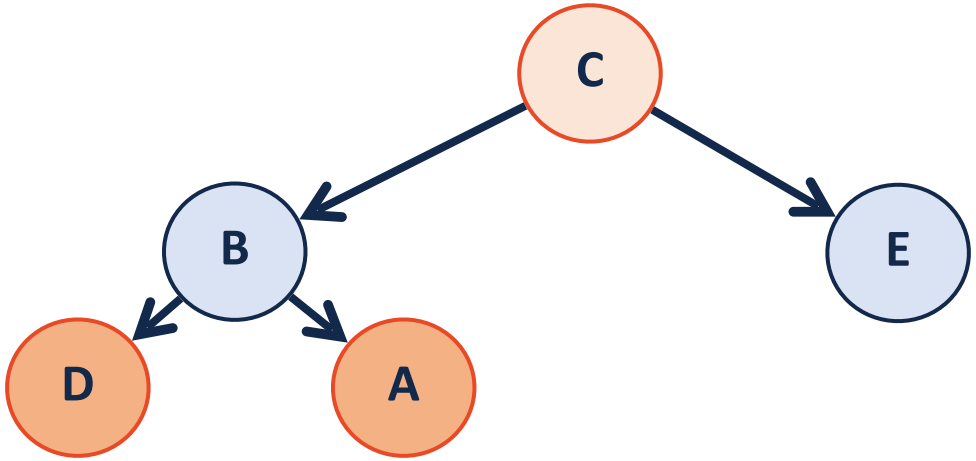
	B	U	I	L	D	H	E	A	P	N	O	W			

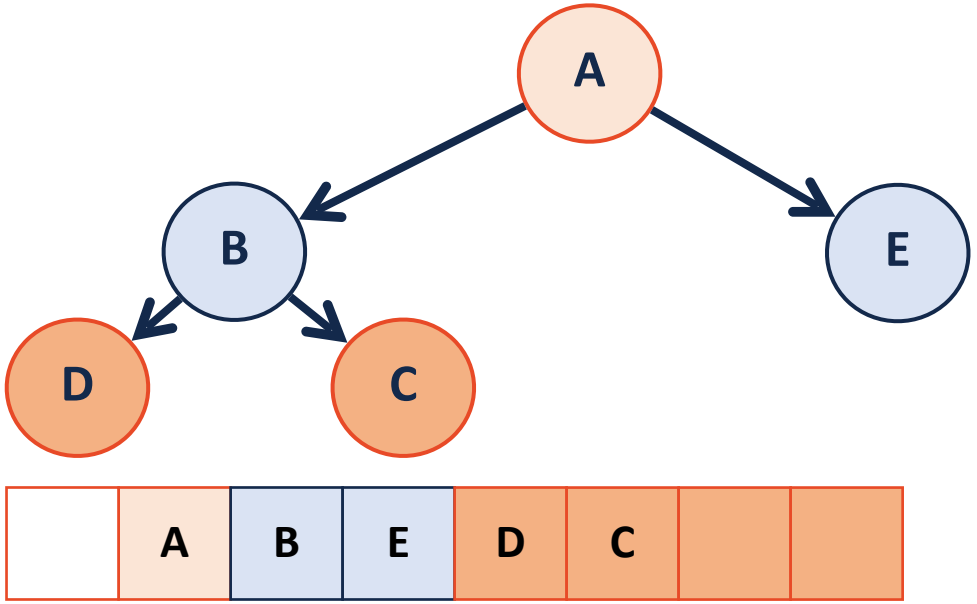
buildHeap – sorted array



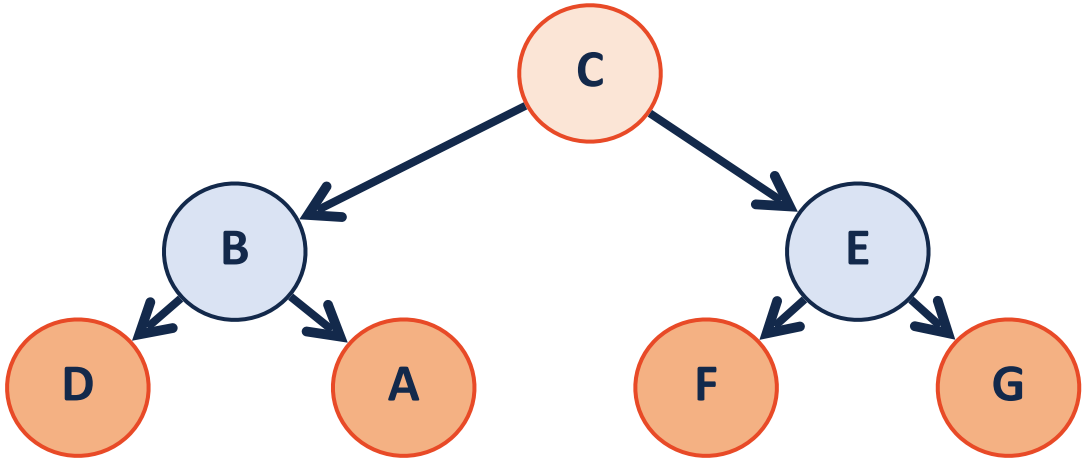


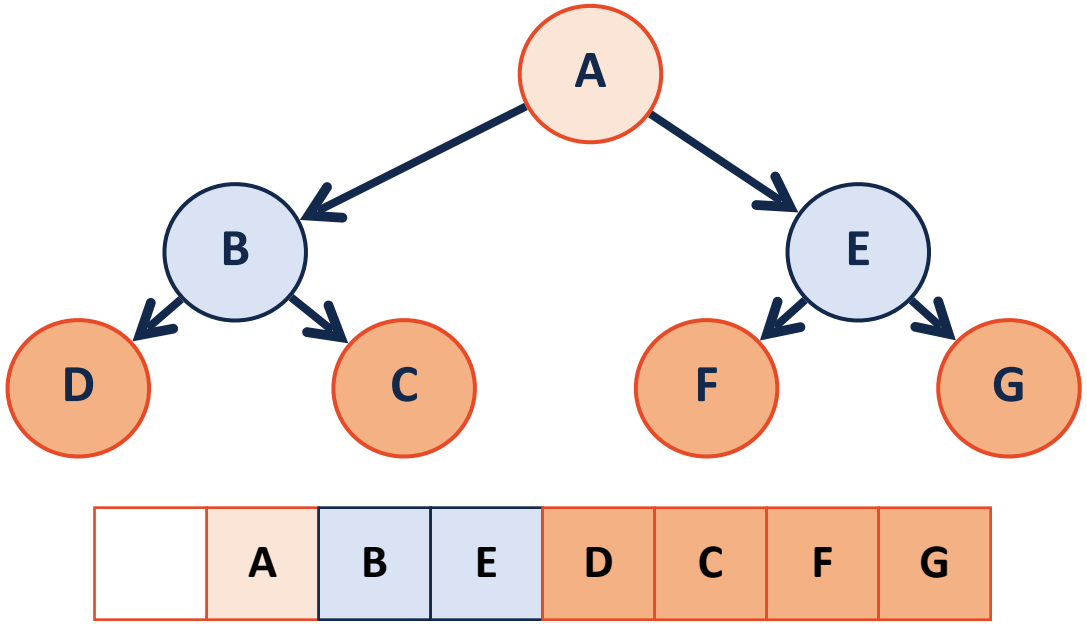
buildHeap - heapifyUp





buildHeap - heapifyDown





buildHeap

1. Sort the array — its a heap!

2. heapifyUp()

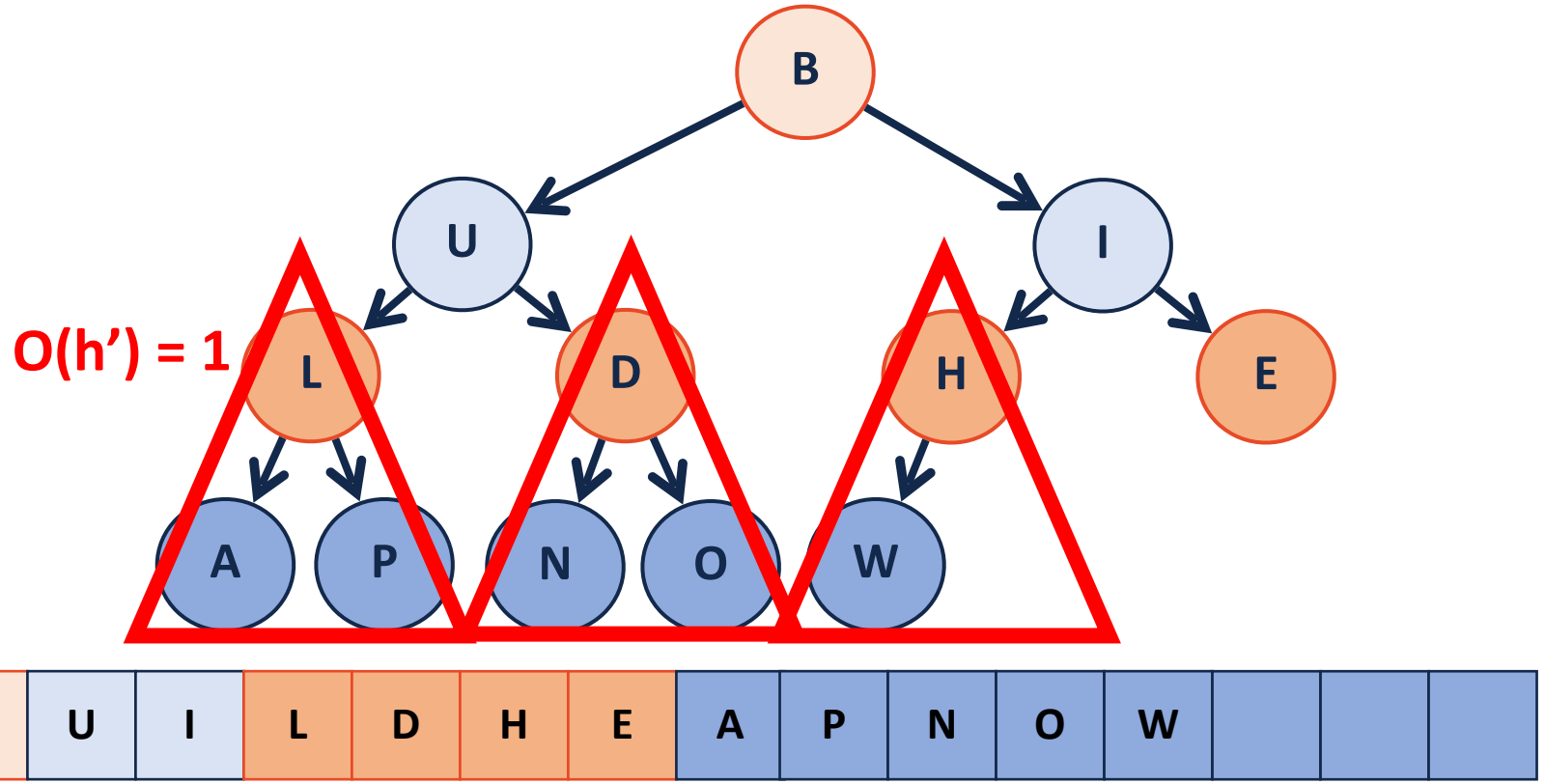
```
1  template <class T>
2  void Heap<T>::buildHeap() {
3    for (unsigned i = 2; i <= size_; i++) {
4      heapifyUp(i);
5    }
6  }
```

3. heapifyDown()

```
1  template <class T>
2  void Heap<T>::buildHeap() {
3    for (unsigned i = parent(size); i > 0; i--) {
4      heapifyDown(i);
5    }
6  }
```

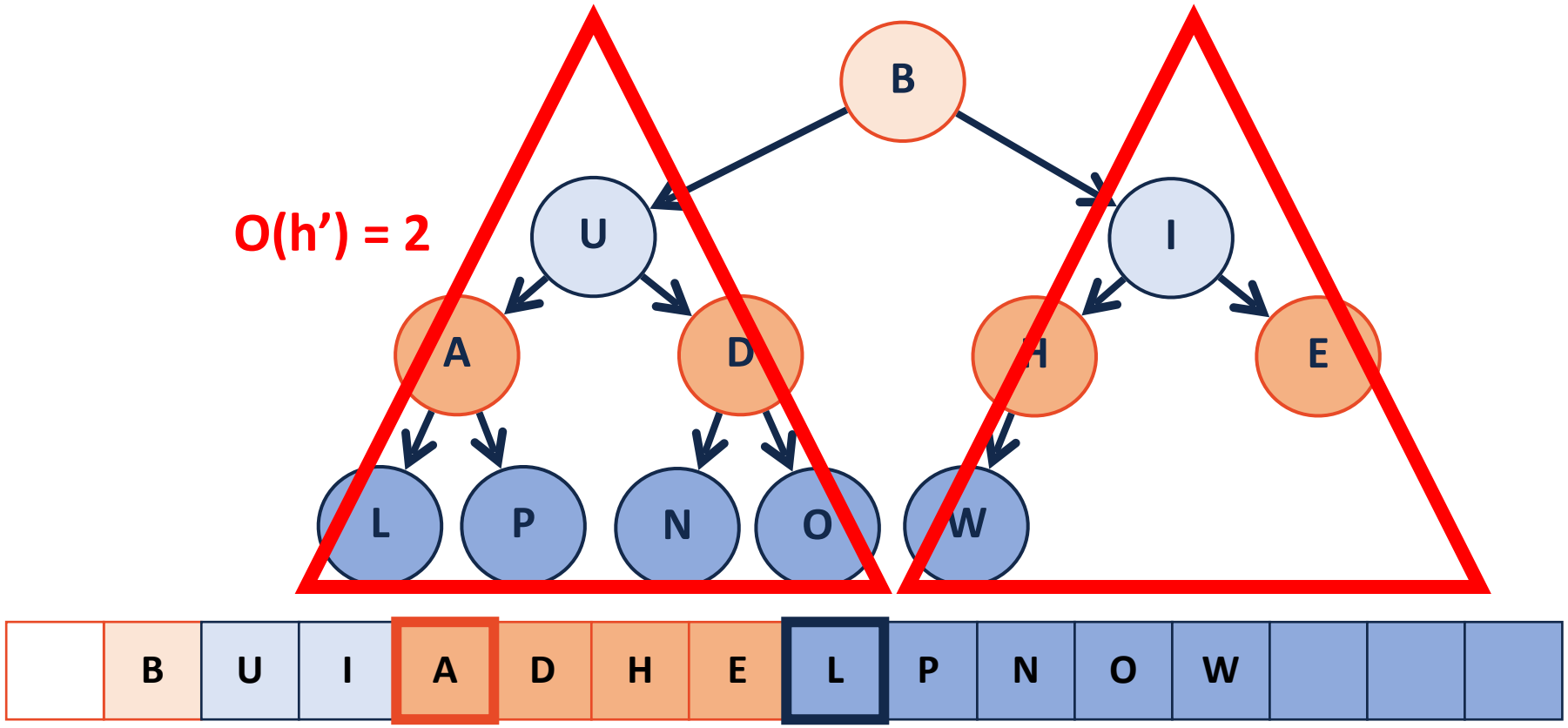
buildHeap - heapifyDown

Lets break down the total ‘amount’ of work:



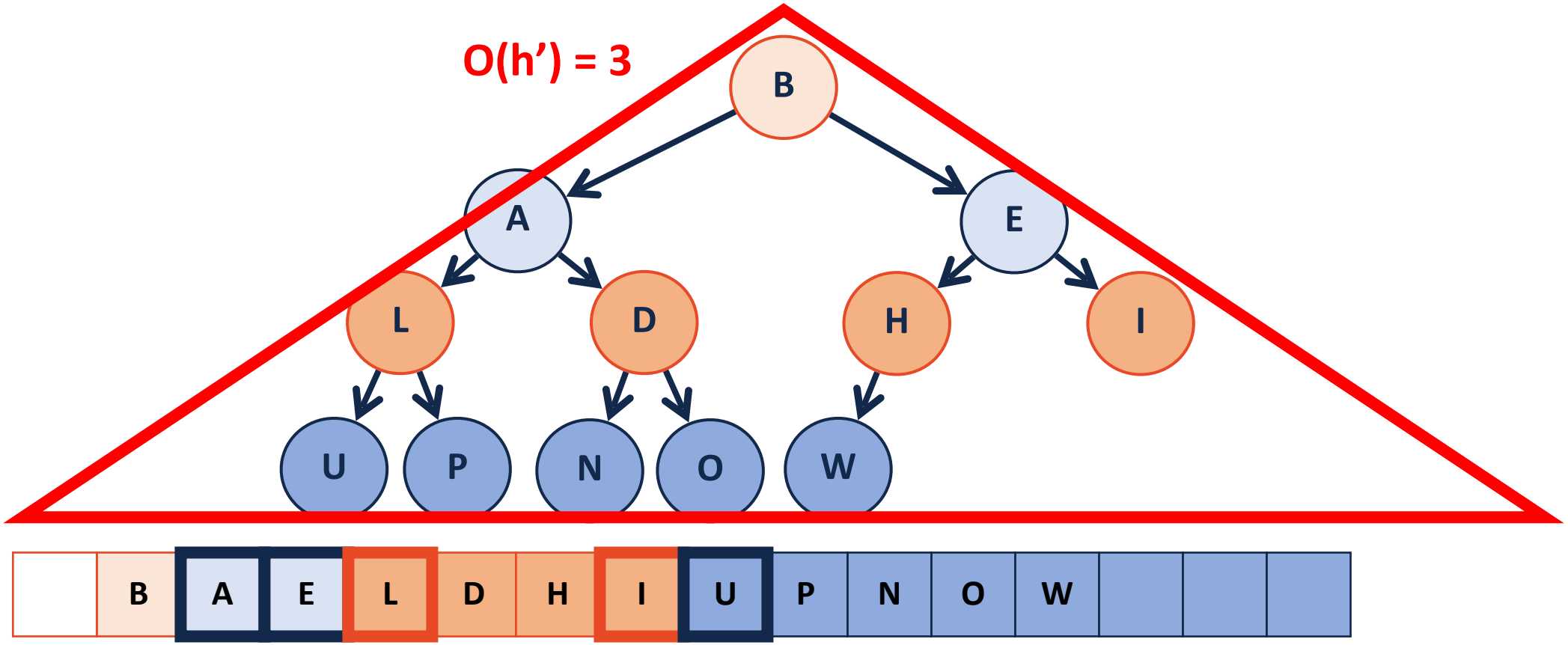
buildHeap - heapifyDown

Lets break down the total ‘amount’ of work:



buildHeap - heapifyDown

Lets break down the total 'amount' of work:



Proving buildHeap Running Time

Theorem: The running time of buildHeap on array of size **n** is:

Strategy:

Proving buildHeap Running Time

S(h): Sum of the heights of all nodes in a **perfect** tree of height **h**.

S(0) =

S(1) =

S(2) =

S(h) =

Proving buildHeap Running Time

Claim: Sum of heights of all nodes in a perfect tree: $S(h) = s^{h+1} - 2 - h$

Base Case:

Proving buildHeap Running Time

Claim: Sum of heights of all nodes in a perfect tree: $S(h) = s^{h+1} - 2 - h$

Induction Step:

Proving buildHeap Running Time

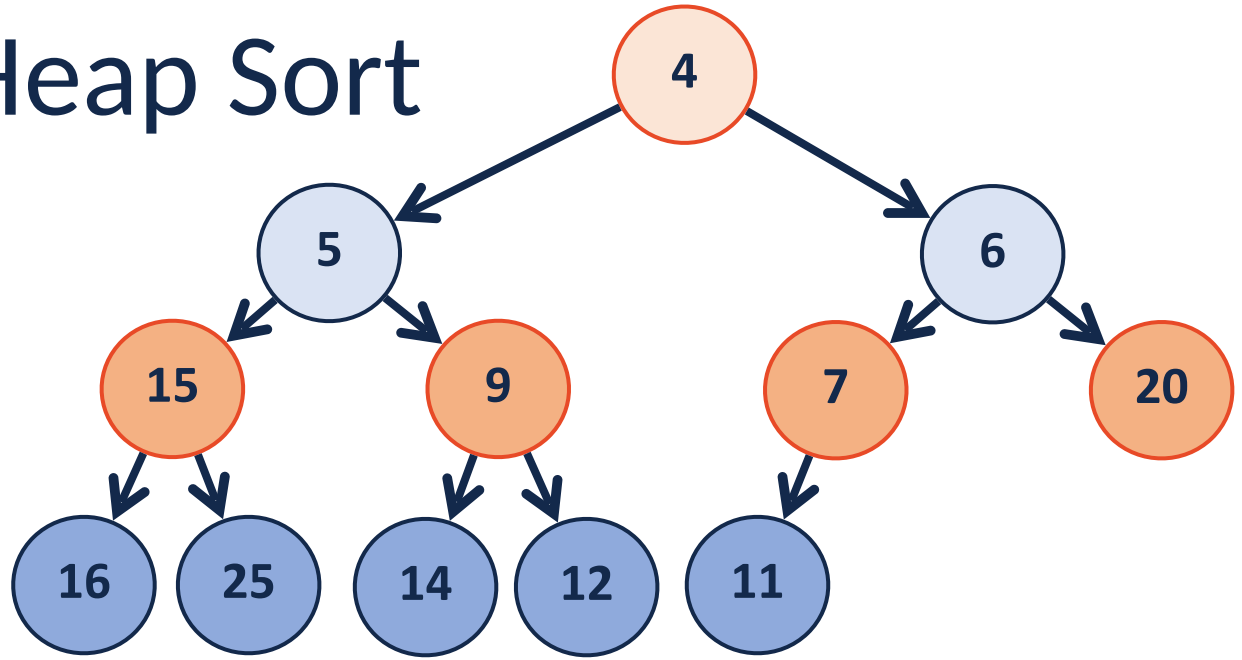
Theorem: The running time of buildHeap on array of size **n** is O(n)

$$S(h) = s^{h+1} - 2 - h$$

How can we relate **h** and **n**?

How can we estimate running time?

Heap Sort



1.

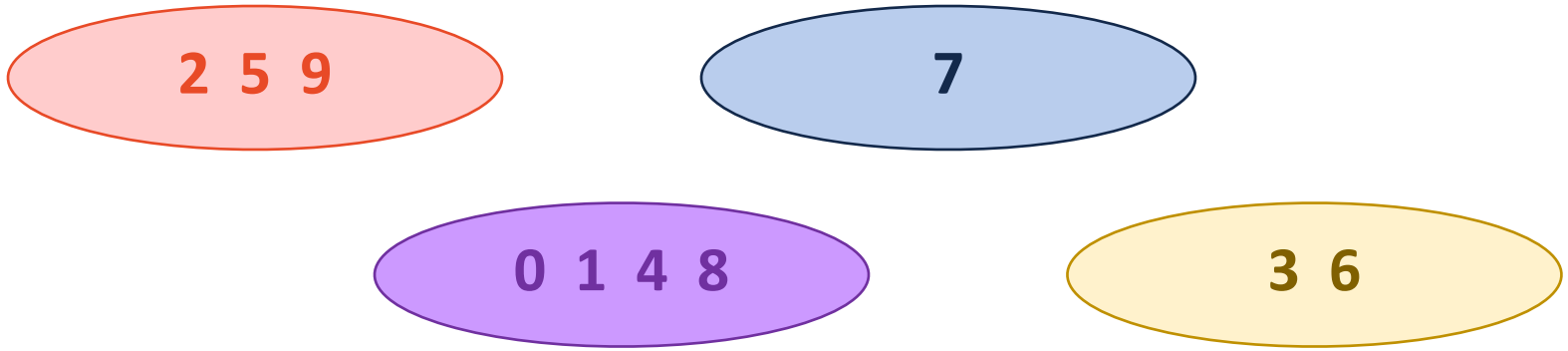
2.

3.

Running time?

minHeap is a good example of tradeoffs:

Disjoint Sets



Key Ideas:

- Each element exists in exactly one set.
- Every item in each set has the same representation
  - In other words: find(4) == find(8) == find(0) …
- Each set has a different representation
  - In other words: find(7) != find(4)

Disjoint Sets ADT

Constructor

InsertSet

Find

Union