

Today's announcements:

MP6 available, due 11/15, 11:59p. EC due 11/8, 11:59p.

Which collision resolution strategy is better?

- Big records –
- Structure speed –

What structures do hash tables replace for us?

There is a constraint on Keyspaces for BST that does not affect hashing...

Why do we talk about balanced BST if hashing is so great?

Applications of hashing?

Area of active research in mathematics to develop general purpose hash functions

Secret Mystery Data Structure

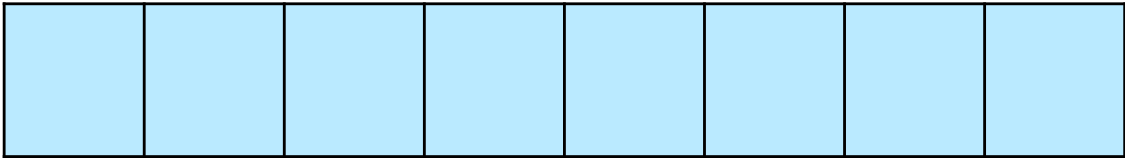
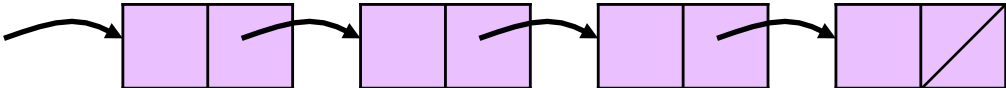
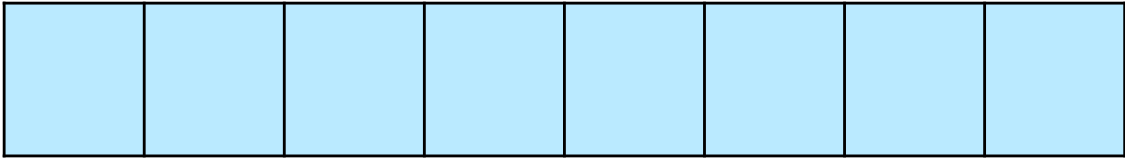
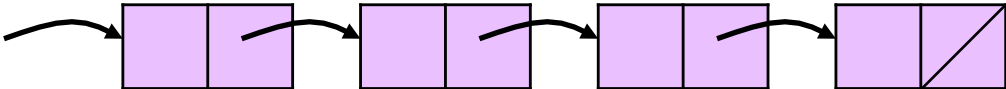
ADT - _____

insert

remove

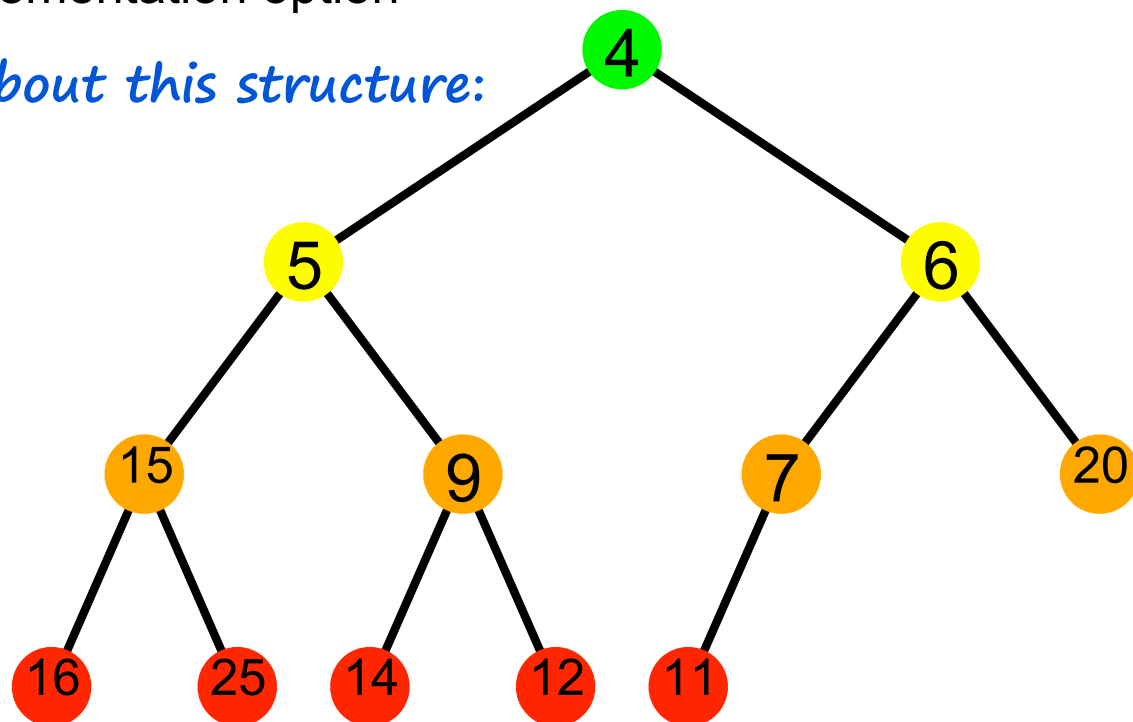
getSize

Priority Queue ADT:

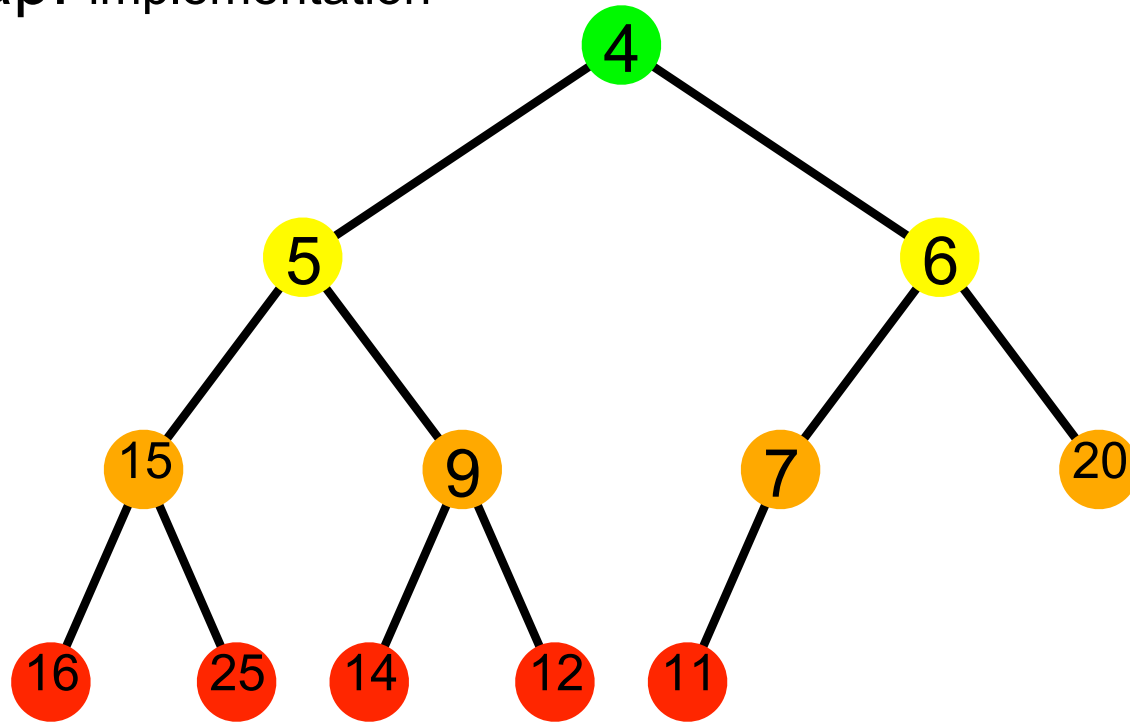
insert	removeMin	implementation
$O(n)$	$O(n)$	 unsorted
$O(1)$	$O(n)$	
$O(\log n)$	$O(1)$	 sorted
$O(\log n)$	$O(1)$	

Priority Queue: another implementation option

Tell me everything you can about this structure:

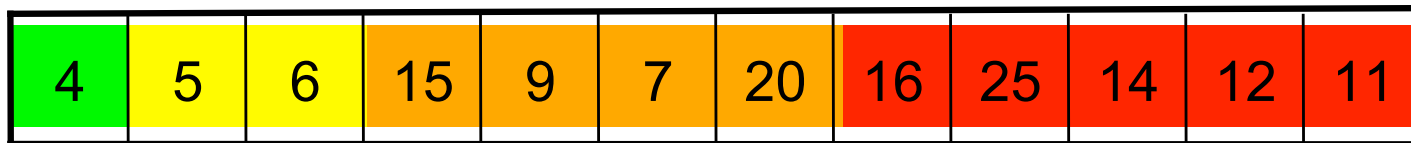
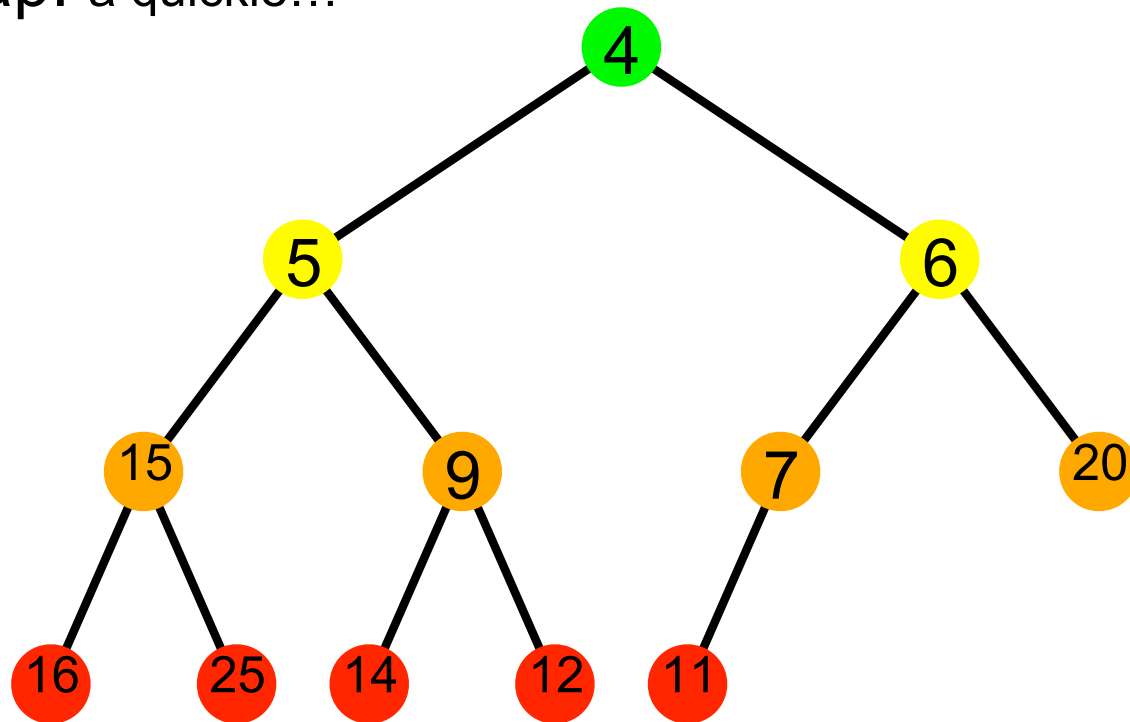


(min)Heap: implementation



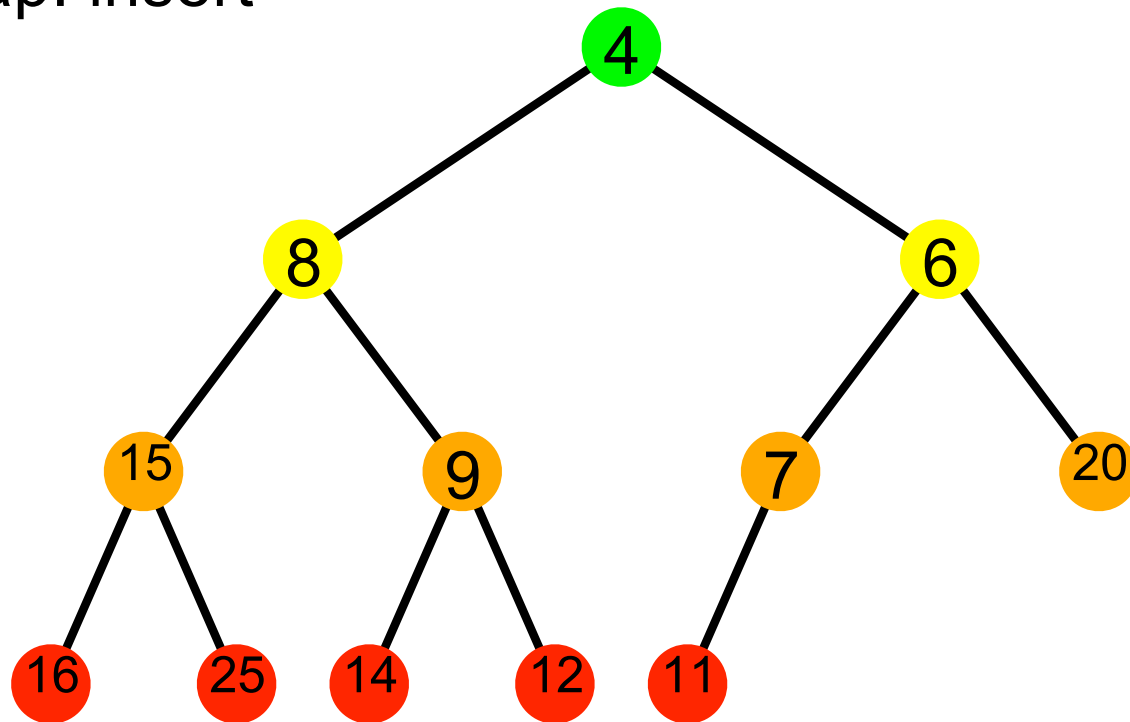
4	5	6	15	9	7	20	16	25	14	12	11
---	---	---	----	---	---	----	----	----	----	----	----

(min)Heap: a quickie...



What is the max height of a complete tree containing n nodes?

(min)Heap: insert



4	8	6	15	9	7	20	16	25	14	12	11			
---	---	---	----	---	---	----	----	----	----	----	----	--	--	--

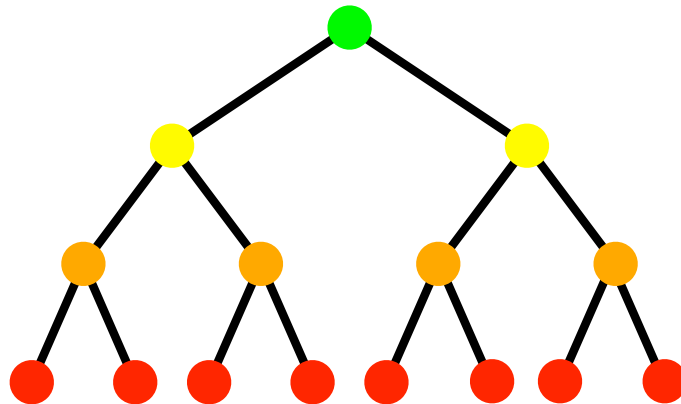
Code:

```
template <class T>
void Heap<T>::insert(const T & key){

    if (size==capacity) growArray();
    size++;
    items[size] = key;
    heapifyUp(size);

}
```

growArray()



Code:

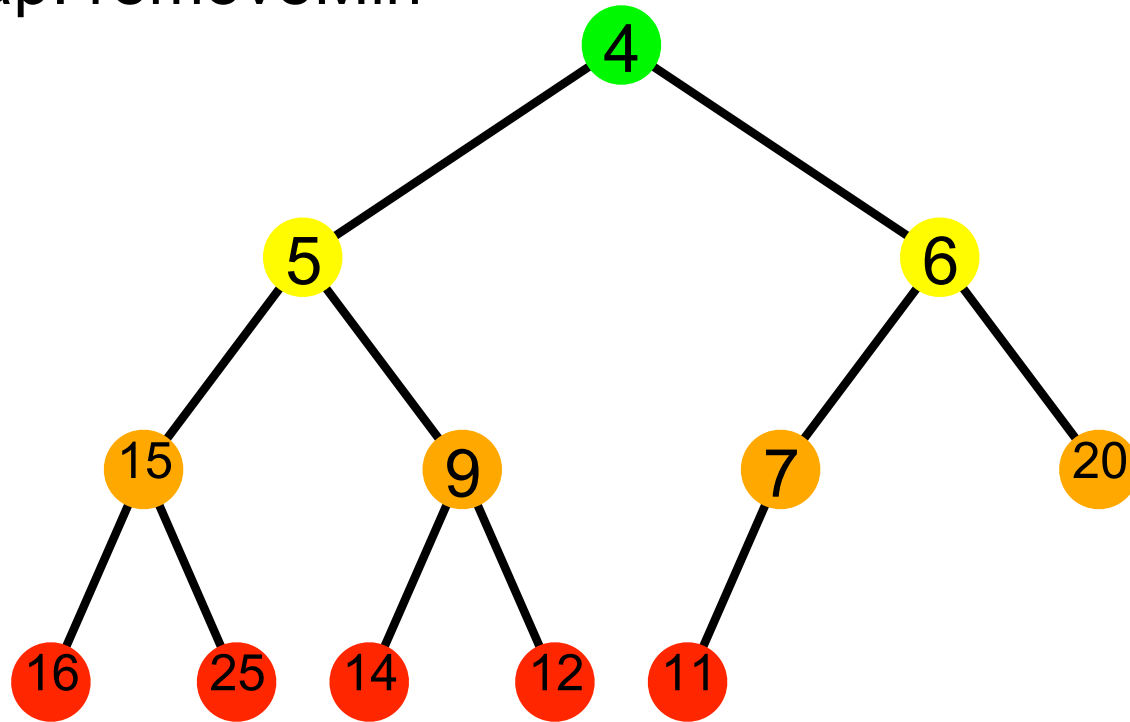
```
template <class T>
void Heap<T>::insert(const T & key){

    if (size==capacity) growArray();
    size++;
    items[size] = key;
    heapifyUp(size);

}
```

```
template <class T>
void Heap<T>::heapifyUp(int cIndex){
    if (cIndex > ____){
        if (items[cIndex] ____ items[parent(cIndex)]){
            swap(____, ____);
            heapifyUp(____);
        }
    }
}
```

(min)Heap: removeMin



4	5	6	15	9	7	20	16	25	14	12	11
---	---	---	----	---	---	----	----	----	----	----	----

Code:

```
template <class T>
T Heap<T>::removeMin() {
    T minVal = items[1];
    items[1] = items[size];
    size--;
    heapifyDown(1);
    return minVal;
}
```

```
template <class T>
void Heap<T>::heapifyDown(int cIndex) {
    if (hasAChild(cIndex)) {
        minChildIndex = minChild(cIndex);
        if (items[cIndex] ____ items[minChildIndex]) {
            swap(____, ____);
            ____;
        }
    }
}
```

What have we done?

