CSE214 Data Structures

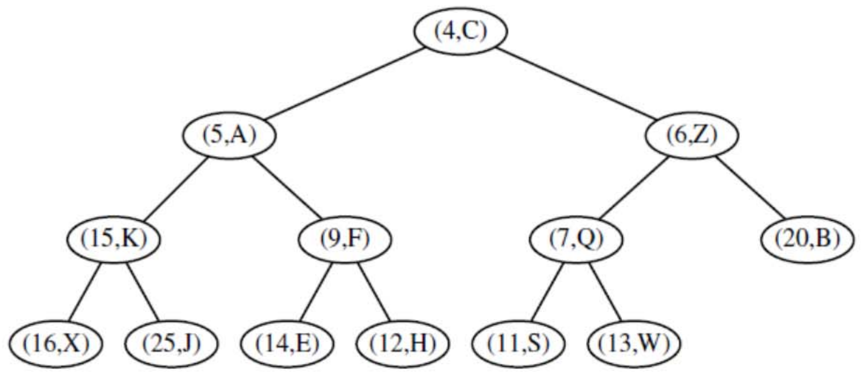
Heap Sort

YoungMin Kwon

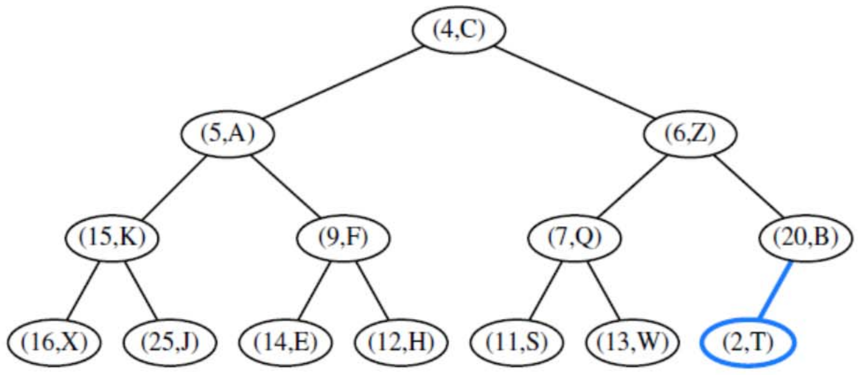
Heap Sort

- Heap sort
  - Add elements to a heap from an array
  - Remove elements from the heap and add them back to the array from the beginning

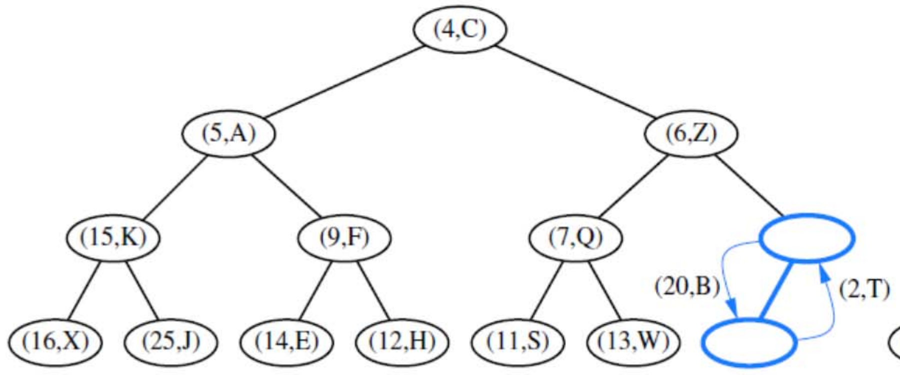
Adding to the Heap



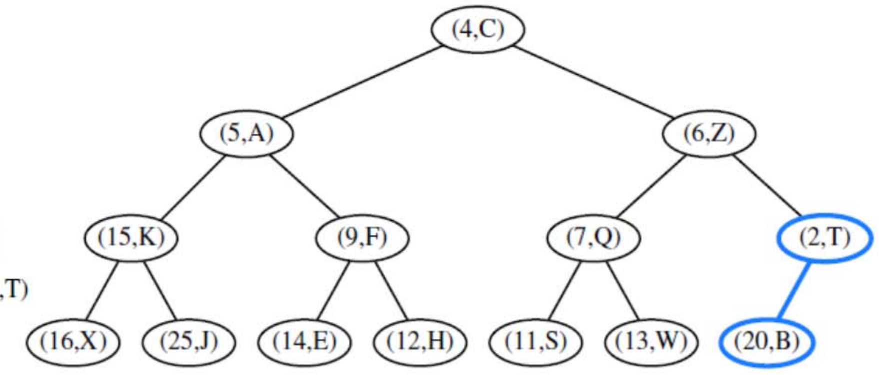
(a)



(b)

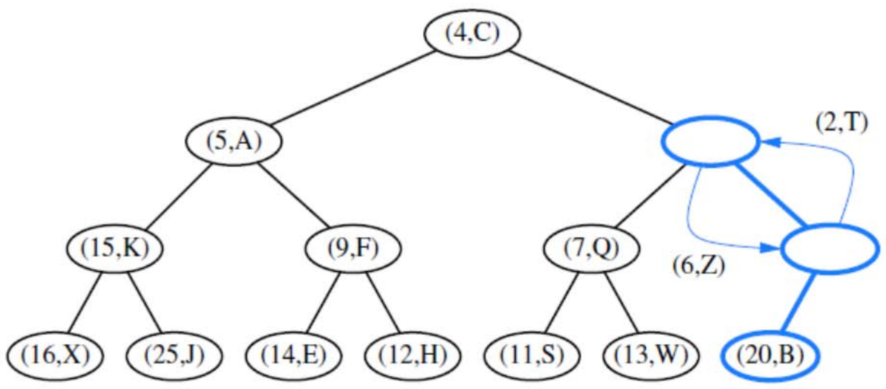


(c)

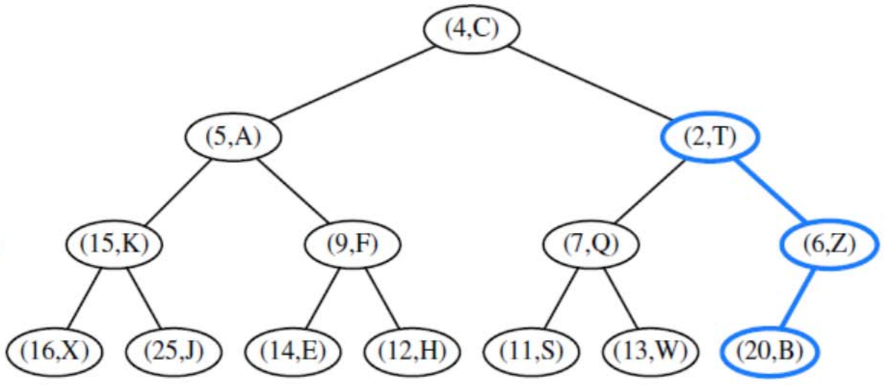


(d)

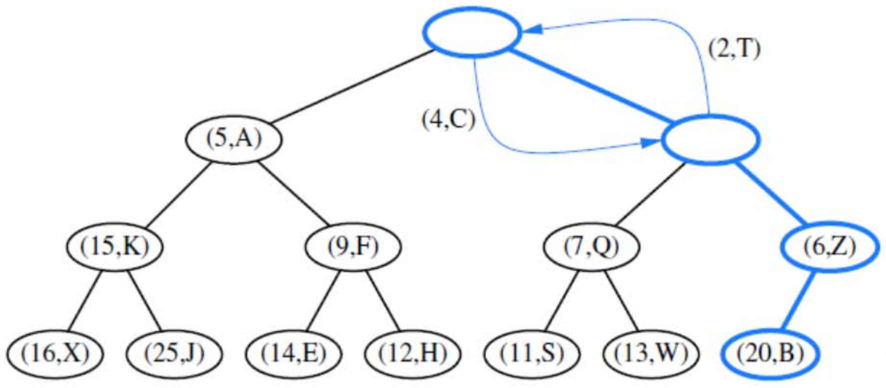
Adding to the Heap



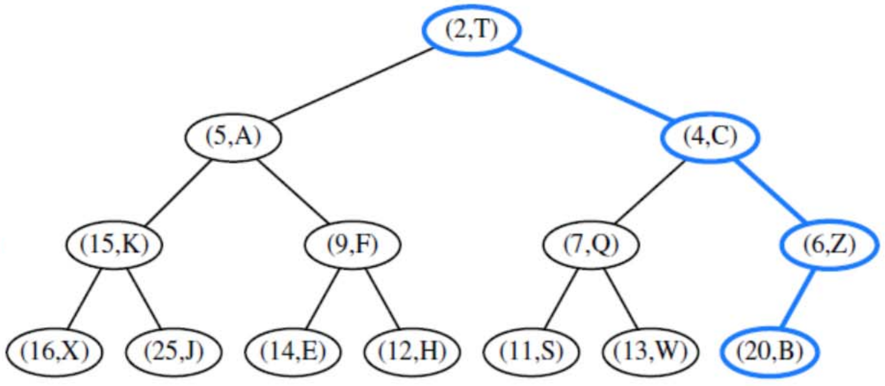
(e)



(f)

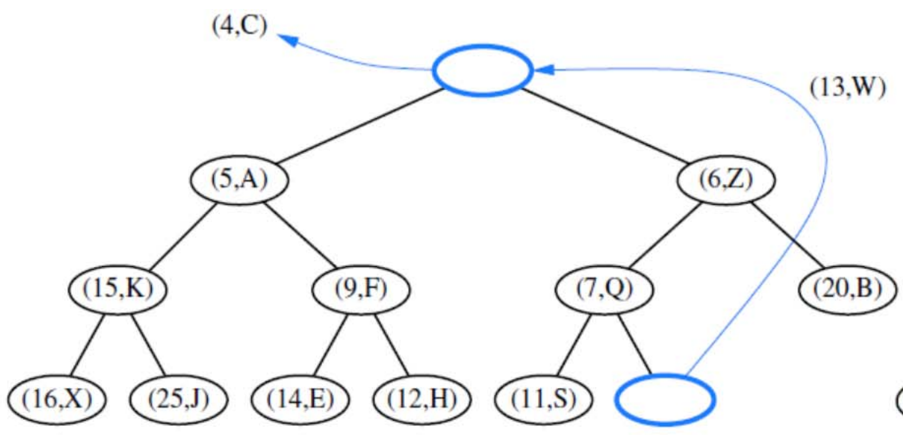


(g)

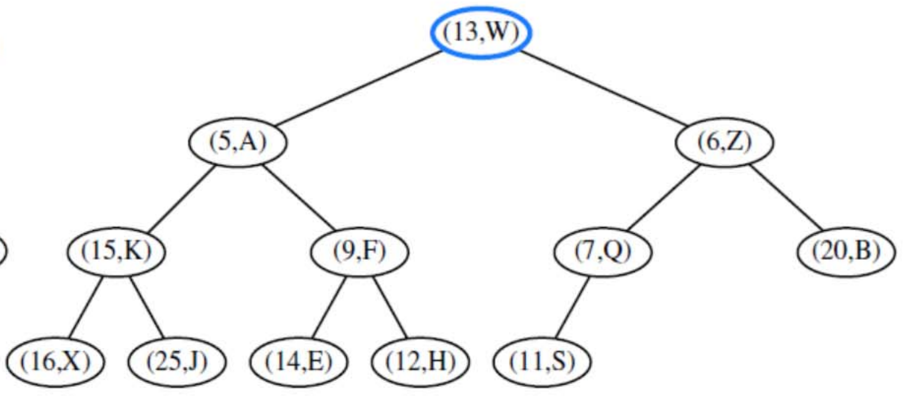


(h)

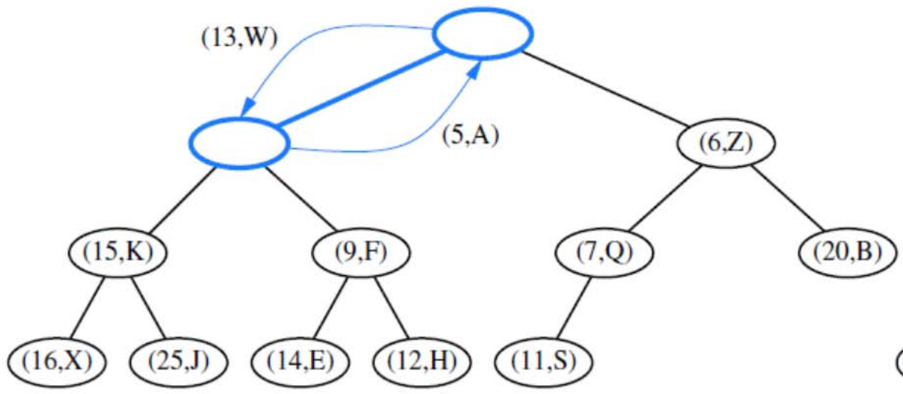
Removing from the Heap



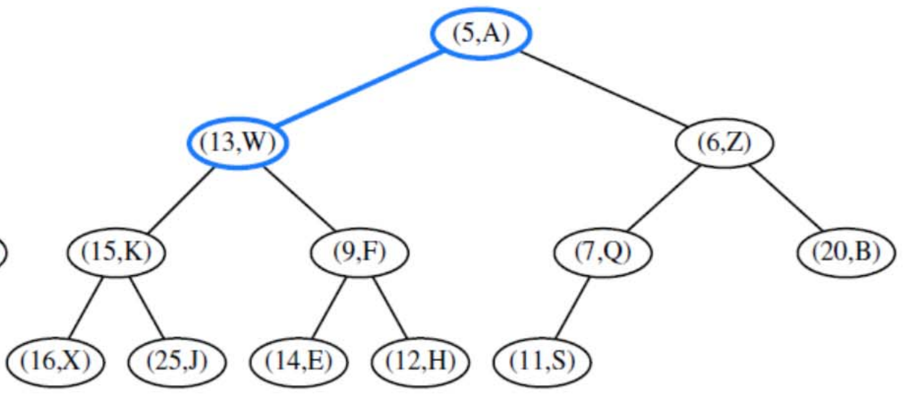
(a)



(b)

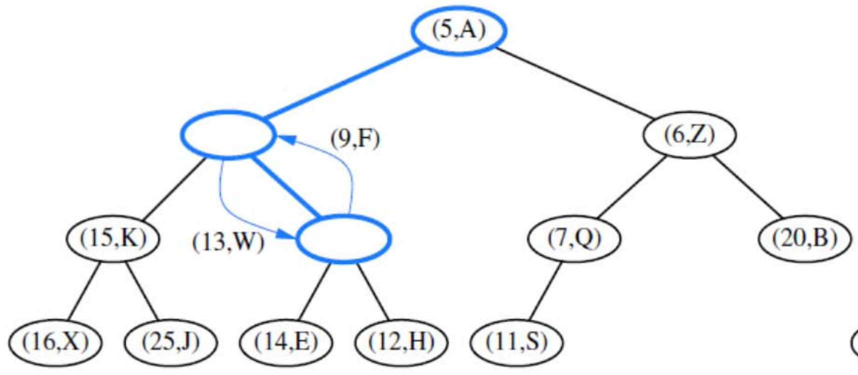


(c)

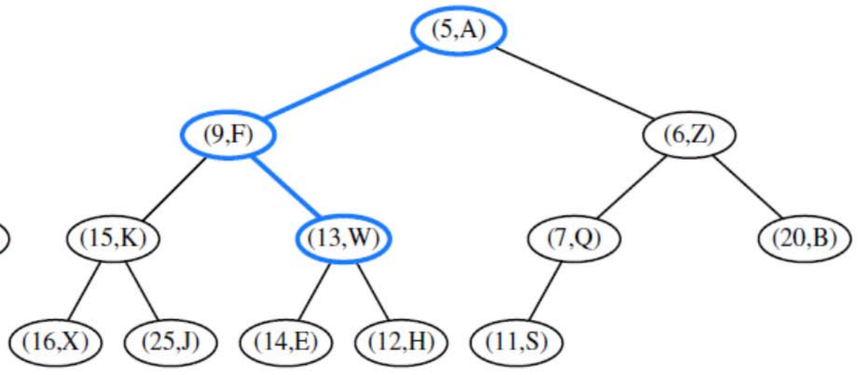


(d)

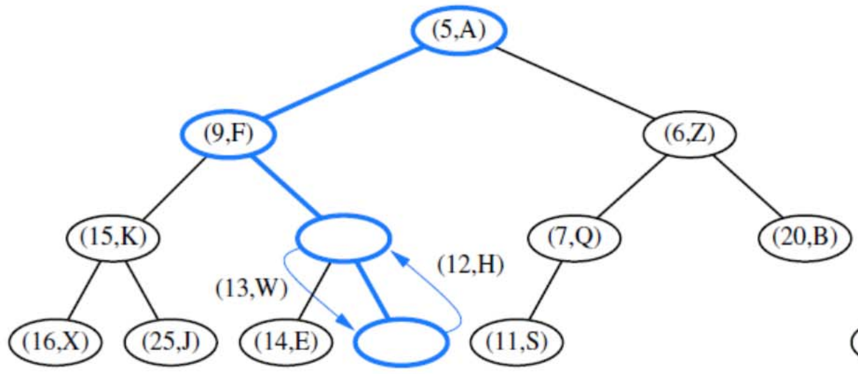
Removing from the Heap



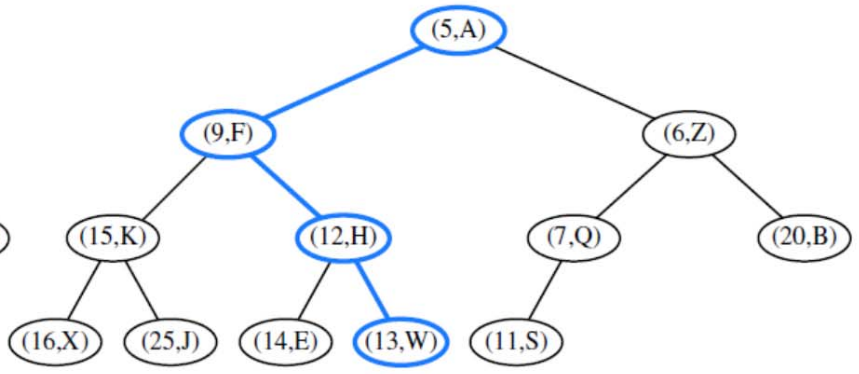
(e)



(f)



(g)



(h)

Array Heap

```
public class HeapSort<E extends Comparable<E>> {
    protected E[] arr;
    protected int size;

    public HeapSort() {
        arr = (E[]) new Comparable[16];
        size = 0;
    }
    public int size()        { return size; }
    public boolean isEmpty() { return size == 0; }

    public E min() {
        if(isEmpty())
            throw new IndexOutOfBoundsException("Empty heap");
        return arr[0];
    }
```

```java
public void add(E e) {
    //dynamic array
    if(size + 1 == arr.length)
        resize(arr.length * 2);

    //TODO: add e at arr[size] and increase size

    //TODO: call upheap

}

public E remove() {
    if(isEmpty())
        throw new IndexOutOfBoundsException("Empty heap");

    E ret = arr[0];
    //TODO: copy the last element to the root and reduce size

    //TODO: call downheap

    return ret;
}
```

```java
protected void upheap(int i) {
    //TODO: if i is the root, return

    int p = parent(i);
    //TODO: if parent is less than or equal to arr[i], return

    //TODO: swap

    //TODO: recursivelyu call upheap
}
protected void downheap(int i) {
    //Find the smaller child
    //TODO: if i does not have left child, return

    int c = left(i);
    //TODO: if i has right child and it is smaller than arr[c],
    //      c = right child

    //TODO: if arr[c] is larger than or equal to arr[i], return

    //TODO: swap i and c

    //TODO: recursively call downheap
}
```

```java
public static <E extends Comparable<E>> void sort(E[] arr) {
    HeapSort_Sol<E> heap = new HeapSort_Sol<E>();

    //TODO: add arr elements to heap

    //TODO: remove elements from heap and add them to arr

}

public static void main(String[] args) {
    Integer[] arr = new Integer[] {3, 5, 2, 4, 1, 8, 7, 6, 0, 9 };

    System.out.println("Before sorting");
    for(int i : arr)
        System.out.print(i + ", ");
    System.out.println();

    sort(arr);

    System.out.println("After sorting");
    for(int i : arr)
        System.out.print(i + ", ");
    System.out.println();
}
```