Given the String, "OUT OF ORDER", show the steps/swaps that would occur during selection sort if we turned the string into a char array. Assume we compare by alphabetical order and that spaces are the smallest.

Insertion Sort	Walking the halls of his abbey late at night, master Bawan came upon a monk in distress. A morning deadline was fast approaching, and after three all-nighters the exhausted monk had dropped his only remaining coins into the soda machine without first noticing the paper sign taped to it. The sign read OUT OF ORDER in large red capitals. "Can you fix it?" asked the monk, who knew of Bawan's skill with primitive machines. Bawan studied the machine for a moment, then crossed out the letters of the sign and under them wrote DEFOOORRTU. Immediately a soda can dropped into the chute. Satisfied, Bawan continued on his way.
OUT_OF_ORDER OUT_OF_ORDER OTU_OF_ORDER _OTUOF_ORDER _OOTUF_ORDER _FOOTU_ORDER __FOOTUORDER __FOOOTURDER __FOOORTUDER __DFOOORTUER __DEFOOORTUR __DEFOOORRTU	

Selection Sort
OUT_OF_ORDER
_UTOOF_ORDER
__TOOFUORDER
__DOOFUORTER
__DEOFUORTOR
__DEFOUORTOR
__DEFOUORTOR #nothing changes, O is the best
__DEFOOURTOR
__DEFOOORTUR
__DEFOOORTUR #nothing changes, R is the best
__DEFOOORRUT
__DEFOOORRTU
__DEFOOORRTU #last step, nothing left

What type of arrays would insertion sort be fast at?

Nearly sorted arrays (best case O(n))

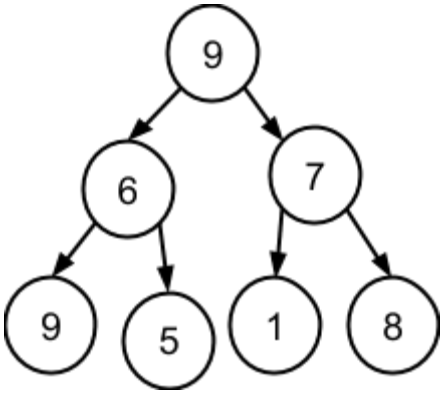
```
/* You are the proud owner of Berkeley Bytes Buffet and business is doing
well! You currently have the policy that children under 8 eat for free
and senior citizens eat for 50 percent off. Since you're the savvy business
owner that you are, you keep track of all the ages of your customers.
Summer is coming around and you want to see how much of each age which is
an integer is coming to your buffet to plan out your marketing.
   a. Write an algorithm that takes in a int array of all customers and
      returns a histogram (how much of each age) as an array of size 150
      which we assume is the max age.
   b. Now for tax reasons, you need to submit a list of the ages of all
      your customers in sorted order. Write an algorithm that takes an int
      array
of all customers and returns a sorted array. HINT: This can be done in
linear time
*/
class CountSort {
      static int maxAge = 150;
      // Part A
      public static int[] histOfAges(int[] ages) {
            int[] countOfAges = new int[maxAge];
            for (int age : ages) {
                  countOfAges[age] += 1;
            }
            return countOfAges;
      }
      // Part B
      public static int[] ageSort(int[] ages) {
            int[] countOfAges = histOfAges(ages);
            int[] sortedAges = new int[ages.length];
            int i = 0; // current index of sortedAges
            for (int age = 0; i < maxAge; i++) {
                  for (int count = countOfAges[i]; count > 0; count--) {
                        sortedAges[i] = age;
                        i++;
                  }
            }
            return sortedAges;
      }
      public static void main(String[] args) {
            int[] test1 = {10,9,8,5,1,4,5,6,7,7,7,40,13,21,64,94,33};
            int[] rtn = ageSort(test1);
            System.out.println(Arrays.toString(rtn));
      }
```

```
}
```

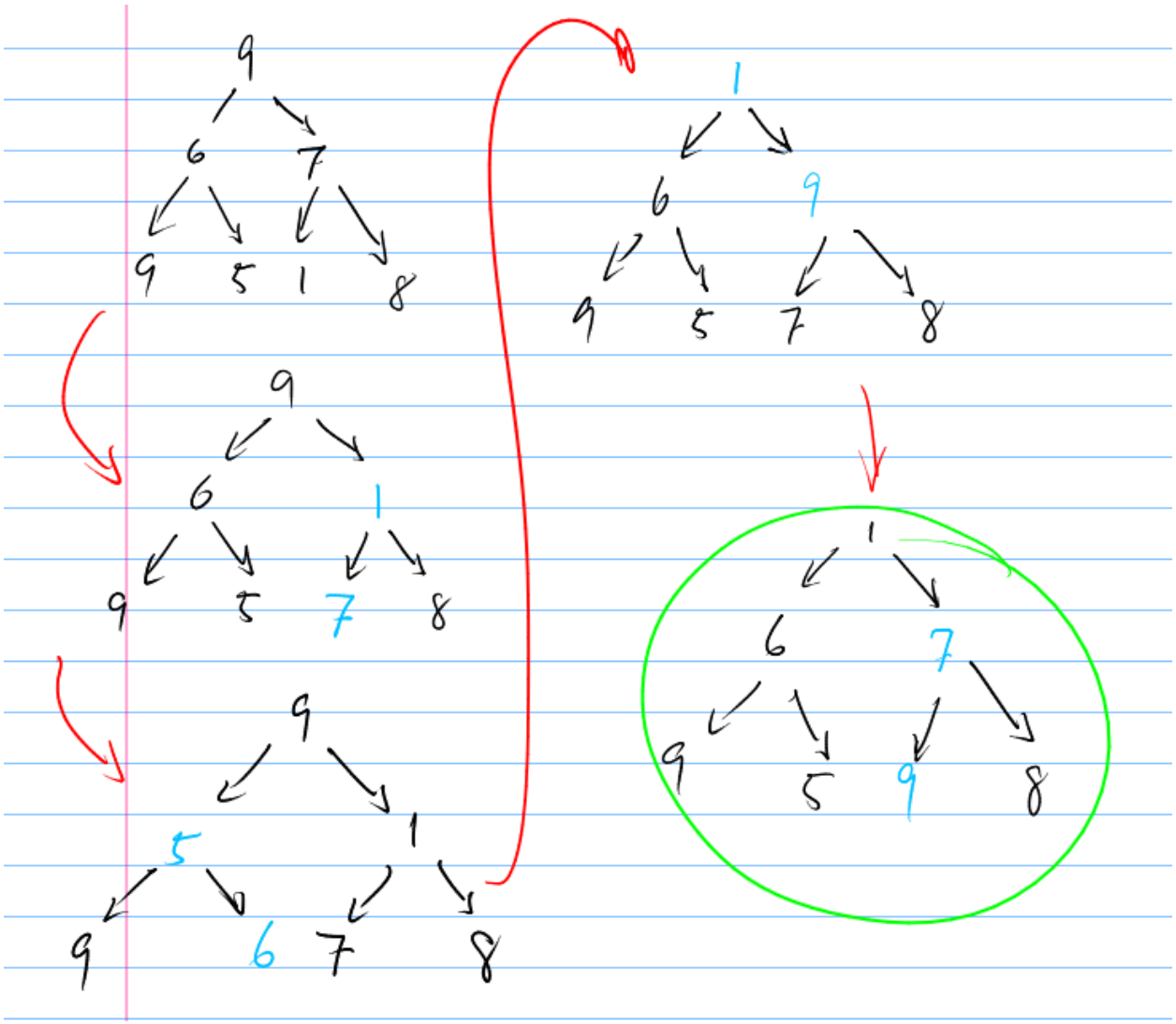
When would we be able to use this type of sort? (Linear time sorting is great!)

-Finite Range of (countable) integers

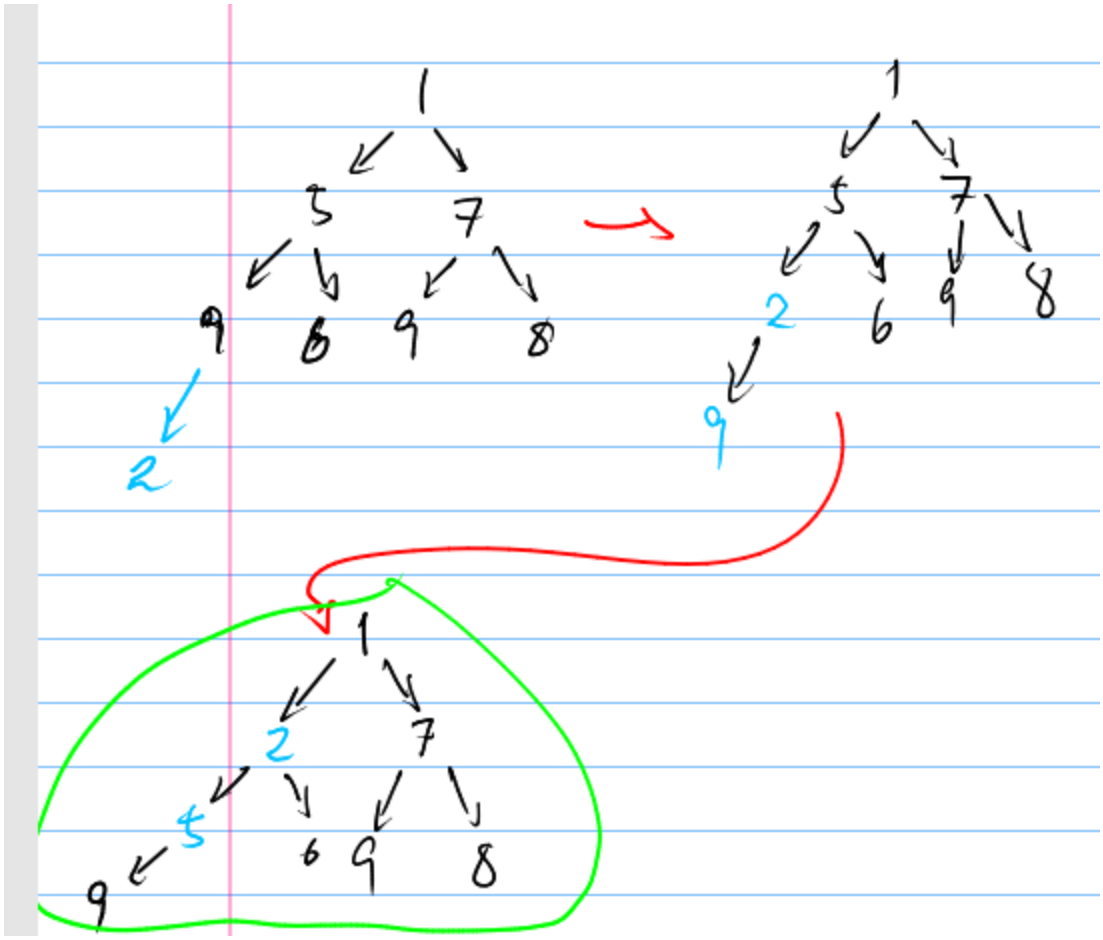
(min)heapify this!



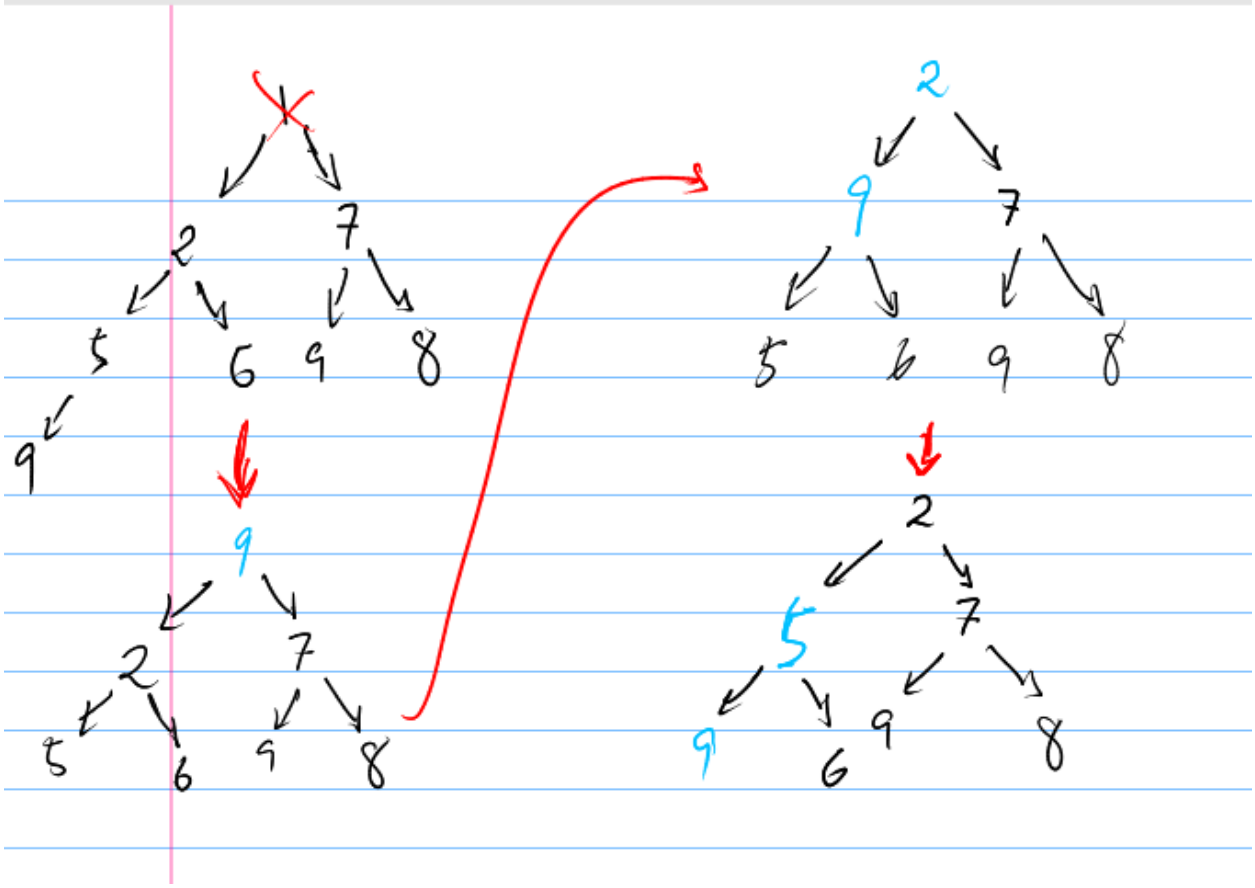
For reference: https://www.cs.usfca.edu/~galles/visualization/Heap.html



Now Insert a 2



Remove the 1



Write HeapSort, which takes in an unsorted list and uses a Heap to solve this problem. How long will this run?

```
public interface Heap<T> {
      Heap(T[] arr); // Heapify!
      void add(T item);
      T removeSmallest();
      boolean isEmpty();
}
```

```
LinkedList<Integer> ints = new LinkedList<Integer>();
You can use “Heap<Integer> h = new Heap<Integer>(ints);” to create a new
min heap out of the new list.
```

```
public static List<T> heapSort(List<T> unsorted) {
    Heap<T> heap = new Heap<T>();
    for (T item : unsorted) heap.add(item);
    // or Heap<T> heap = heapify(unsorted)

    List<T> sorted = new ArrayList<T>;
    while (!heap.isEmpty()) sorted.add(heap.removeSmallest());
    return sorted;
}
```