CS 61B Spring 2025

Sorting

Discussion 11: April 14, 2025

1 All Sorts Of Sorts

Show the steps taken by each sort on the following unordered list:

`0, 4, 2, 7, 6, 1, 3, 5`

(a) Insertion sort

① 0 4 2 7 6 1 3 5

② 0 4 2 7 6 1 3 5

③ 0 2 4 7 6 1 3 5

④ 0 2 4 7 6 1 3 5

⑤ 0 2 4 6 7 1 3 5

⑥ 0 1 2 4 6 7 3 5

⑦ 0 1 2 3 4 6 7 5

⑧ 0 1 2 3 4 5 6 7

(b) Selection sort

① 0 4 2 7 6 1 3 5

② 0 1 2 7 6 4 3 5

③ 0 1 2 7 6 4 3 5

④ 0 1 2 3 6 4 7 5

⑤ 0 1 2 3 4 6 7 5

⑥ 0 1 2 3 4 5 7 6

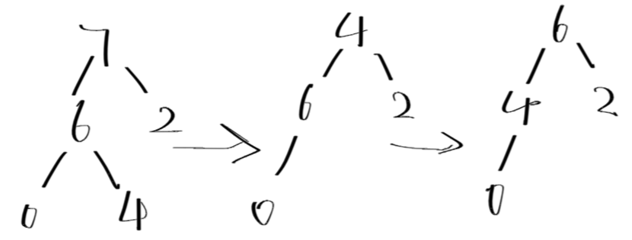
⑦ 0 1 2 3 4 5 6 7

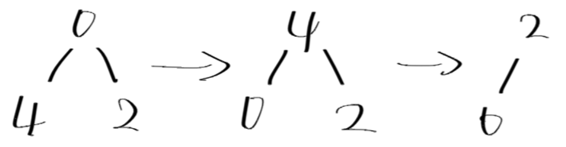
(c) Merge sort

① 04 27 16 35

② 0247 1356

③ 01234567





(d) Use heapsort to sort the following array (hint: draw out the heap).
Draw out the array at each step: `0, 6, 2, 7, 4`

Heapification: 7 6 2 0 4

① 6 4 2 0 7

② 4 0 2 6 7

③ 2 0 4 6 7

④ 0 2 4 6 7

2 Ethan Has Been Waiting For This

Ethan, an extremely tall 61B tutor, is trying to sort the TAs by height so he can snap a photo. Can you help him out?

```
public class TA {
  private String name;
  private int height;
  public TA(String name, int height) {
    this.name = name;
    this.height = height;
  }
}
```

(a) Implement a TAComparator below such that it compares two TAs' height. Recall that a Comparator's compare method returns a negative number when o1 is "less than" o2, positive number when o1 is "greater than" o2, and 0 when they are the same.

```
public class TAComparator implements Comparator<__> {
  @Override
  public int compare(TA o1, TA o2) {
       return o1.height - o2.height



  }
}
```

(b) Anniyat suggests that we use Quicksort with our comparator. Given the following list of TAs, who would make the worst pivot? What about the best pivot?

```
TA anniyat = new TA("Anniyat", 70000);
TA aditya = new TA("Aditya", 1);
TA elana = new TA("Elana", 5);
TA sree = new TA("Sree", 7);
TA kevin = new TA("Kevin", 25);
TA elaine = new TA("Elaine", 9);
TA daniel = new TA("Daniel", 4);
TA teresa = new TA("Teresa", 8);
TA diego = new TA("Diego", 8);
```

1 4 5 7 8 8 9 25 70000

best pivot : 8

worst pivot : 1 and 70000

70000 1 5 7 25 9 4 8_T 8_D

8_D 1 5 7 25 9 4 8_T 70000

(c) Diego points out that even though he got in line after Teresa, he ended up in front of Teresa in the sorted list produced by Quicksort (which he doesn't like because that makes it seem like he's shorter than Teresa)! How might we ensure that Diego ends up behind Teresa?

Merge Sort

(d) Our TAs have just been sorted by height, but suddenly Vika and Wilson come running in late! Which sort will do the most minimal work to get them in their correct spots, and what is the additional runtime it will take (ie. not including the runtime for sorting all the other TAs first)?

Insertion Sort

$\Theta(N)$

3 Zero One Two-Step

(a) Given an array that only contains 0's, 1's and 2's, write an algorithm to sort it in linear time without creating a new array. You may want to use the provided helper method, swap. Hint: Consider how Hoare partitioning rearranges elements in an array.

```
public static void specialSort(int[] arr) {
  int front = 0;
  int back = arr.length - 1;
  int curr = 0;
  while (~~front < back~~ curr<=back) {
      if (arr[curr] < 1) {
      swap(arr, front, curr);
      front++;
      curr++;
      } else if (arr[curr] > 1) {
      swap(arr, back, curr);
      back--;
      } else {
      curr++;
      }
    }
  }
}

private static void swap(int[] arr, int i, int j) {
  int temp = arr[i];
  arr[i] = arr[j];
  arr[j] = temp;
}
```

(b) We just wrote a linear time sort, how cool! Why can't we always use this sort, even though it has better runtime than Mergesort or Quicksort?

It has some limits that

(c) The sort we wrote above is also "in place". What does it mean to sort "in place", and why would we want this?

Space complexity will be less.