[TAP:SZAPB] Sort

- Which of the following does not run in O(n^2)?
  A. Bubble Sort (best)
  B. Bubble Sort (worst)
  C. Selection Sort (best)
  D. Selection Sort (worst)
  E. Whatever

Administrative Details

- Lab 5 Posted
  - Sorting with Comparators
- Midterm Wednesday March 14
  - Held in your scheduled Lab (same time and place)
  - Study guide and sample exam
  - Review session

Today's Outline

- Sort
  - Insertion Sort
  - Merge Sort

Sorting a Deck of Cards

Time Complexity:

A. O(n) ← best

B. O(n log n)

C. $O(n^2)$ ← worst, ave

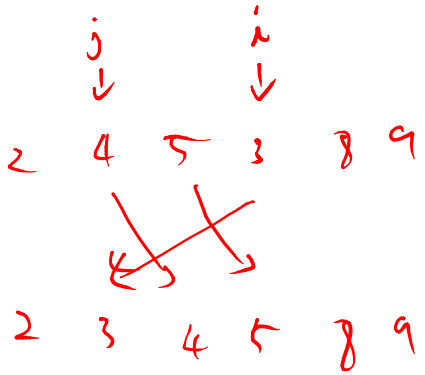
D. $O(n^3)$

E. Not sure

Insertion Sort

- [5 7 0 3 4 2 6 1]
- [5 7 0 3 4 2 6 1]
- [0 5 7 3 4 2 6 1]
- [0 3 5 7 4 2 6 1]
- [0 3 4 5 7 2 6 1]
- [0 2 3 4 5 7 6 1]
- [0 2 3 4 5 6 7 1]
- [0 1 2 3 4 5 6 7]

Insertion Sort




```
public static void insertionSort(int[] data) {
    for(int i=1; i<data.length; i++){
        int temp = data[i];
        int j;
        for(j=i; j>0; j--){
            if(temp < data[j-1])
                data[j] = data[j-1];
            else
                break;
        }
        data[j] = temp;
    }
}
```

Insertion Sort Summary

- Overview
  - After *ith* iteration, at least *i* items are sorted.
  - During *ith* iteration, take the first item in the unsorted portion of the list and **insert** it to the "correct" location in the sorted portion.
- Time complexity:
  - Best case: O(n)
  - Worst case: $O(n^2)$
  - Average case: $O(n^2)$

Today's Outline

- Sort
  - Insertion Sort
  - Merge Sort

Sorting a Deck of Cards

Time Complexity:

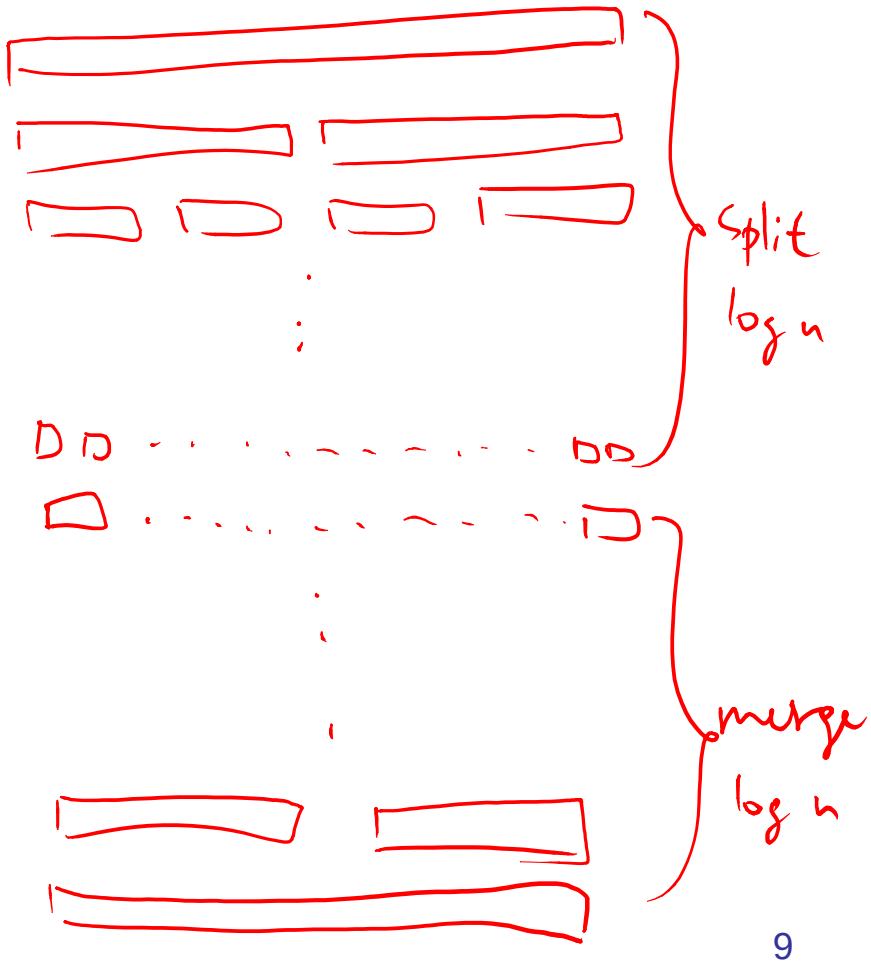
A. O(n)

B. O(n log n)

C. $O(n^2)$

D. $O(n^3)$

E. Not sure



Merge Sort

- [8 14 29 1 17 39 16 9]
- [8 14 29 1] [17 39 16 9]
- [8 14] [29 1] [17 39] [16 9]
- [8] [14] [29] [1] [17] [39] [16] [9]
- [8 14] [1 29] [17 39] [9 16]
- [1 8 14 29] [9 16 17 39]
- [1 8 9 14 16 17 29 39]

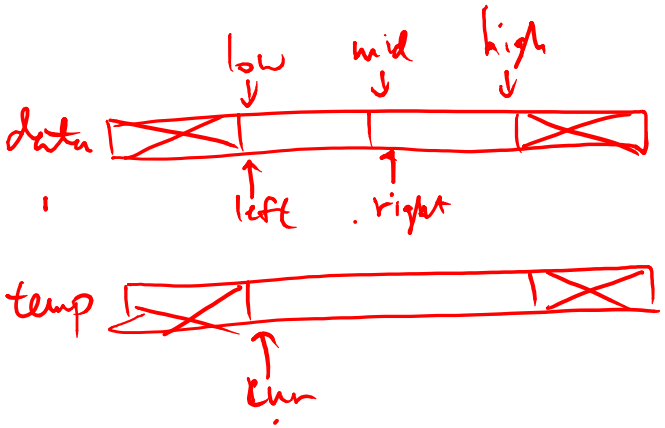
Merge Sort

```
public static void mergeSort(int[ ] data) {

    mergeSortHelper(data, 0, data.length-1);
}
public static void mergeSortHelper(int[ ] data, int low, int high) {
    // base case
    if(low >= high)
        return;

    //recursive case
    int mid = (low + high)/2;
    mergeSortHelper(data, low, mid);
    mergeSortHelper(data, mid+1, high);
    merge(data, low, mid+1, high);
}
```

Aside: merge() method

public static void merge(int[ ] data, int low, int mid, int high)

```
int[] temp = new int[data.length];
int left = low;
int right = mid;
int cur = low;

while (left <= mid-1 && right <= high){
    if(data[left] < data[right])
        temp[cur++] = data[left++];
    else
        temp[cur++] = data[right++];
}
while (left <= mid-1)
    temp[cur++] = data[left++];
while (right <= high)
    temp[cur++] = data[right++];
```




```
for(int i=low; i<=high; i++)
    data[i] = temp[i];
```