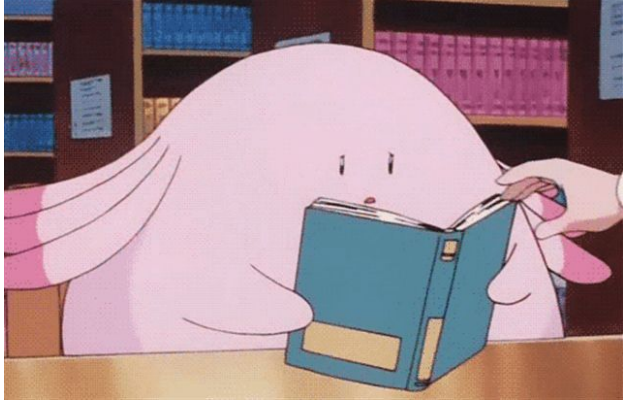


CSE 373 SP21 Sorting Study Guide

Keep this next to you when you work through Section 8 (or any sorting problems)!

Level 1: Sorting Cheat-Sheet

Sorting Cheat Sheet

	insertion	selection	merge	quick	heap
worst	n^2	n^2	n*logn	n^2	n*logn
In practice	n^2	n^2	n*logn	n*logn	n*logn
best	n	n^2	n*logn	n*logn	n
In place	yes	yes	no	yes	yes
stable	yes	no	yes	no	no

Sorting Buzz Words

- **Stable:**
  Any equal items remain in the same relative order before and after the sort.

- **In-Place:**
  Requires only O(1) extra space to perform the sort.

Level 2: Sorting Overview

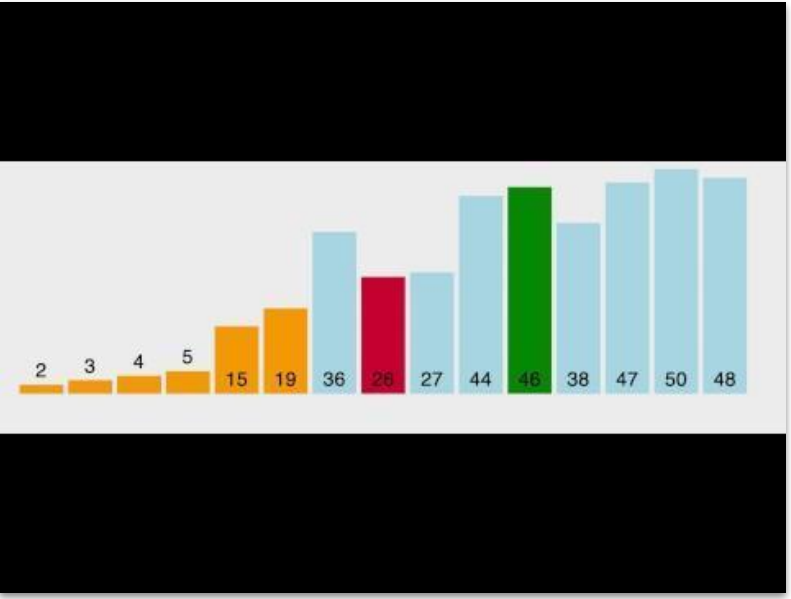
Selection Sort

Repeatedly select the **smallest remaining** item and swap it to its proper index.

1. Find the **smallest** item in the array, and swap it with the **first** item.
2. Find the **next smallest** item in the array, and swap it with the **next** item.
3. Continue until all items in the array are sorted.

We look through entire remaining array every time to find the minimum.

Selection Sort



If this video doesn't play try logging into UW Net or using Firefox. IDK why this happens on Chrome….

```
public void selectionSort(collection) {
    for (entire list)
        int newIndex = findNextMin(currentItem);
        swap(newIndex, currentItem);
}
public int findNextMin(currentItem) {
    min = currentItem
    for (unsorted list)
        if (item < min)
            min = currentItem
    return min
}
public int swap(newIndex, currentItem) {
    temp = currentItem
    currentItem = newIndex
    newIndex = currentItem
}
```

Selection Sort Stability

Question

Repeatedly select the smallest remaining item and swap it to its proper index.

1. Find the smallest item in the array, and swap it with the first item.
2. Find the next smallest item in the array, and swap it with the next item.
3. Continue until all items in the array are sorted.

Selection sort is not stable. Give an example.

Selection Sort Stability

Answer

Repeatedly select the smallest remaining item and swap it to its proper index.

1. Find the smallest item in the array, and swap it with the first item.
2. Find the next smallest item in the array, and swap it with the next item.
3. Continue until all items in the array are sorted.

Selection sort is not stable. Give an example.

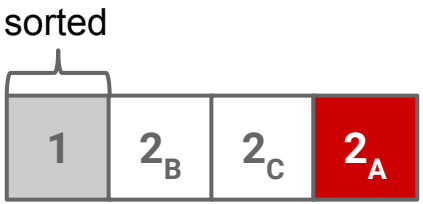
Selection Sort Stability

Answer

Repeatedly select the smallest remaining item and swap it to its proper index.

1. Find the smallest item in the array, and swap it with the first item.
2. Find the next smallest item in the array, and swap it with the next item.
3. Continue until all items in the array are sorted.

Selection sort is not stable. Give an example.



2A and 2B are not in the same relative order!

Insertion Sort

Build a sorted subarray (like selection sort) by using left-neighbor swaps for stability.

Scan from left to right…

1. If an item is out of order with respect to its left-neighbor, swap left.
2. Keep on swapping left until the item is in order with respect to its left-neighbor.

Insertion Sort


```
public void insertionSort(collection) {
   for (entire list)
      if(currentItem is smaller than largestSorted)
         int newIndex = findSpot(currentItem);
         shift(newIndex, currentItem);
}
public int findSpot(currentItem) {
   for (sorted list)
      if (spot found) return
}
public void shift(newIndex, currentItem) {
   for (i = currentItem > newIndex)
      item[i+1] = item[i]
   item[newIndex] = currentItem
}
```

Insertion Sort Stability

Question

Build a sorted subarray (like selection sort) by using left-neighbor swaps for stability.

Scan from left to right…

1. If an item is out of order with respect to its left-neighbor, swap left.
2. Keep on swapping left until the item is in order with respect to its left-neighbor.

Insertion sort is stable. Give an example.

Insertion Sort Stability

Answer

Build a sorted subarray (like selection sort) by using left-neighbor swaps for stability.

Scan from left to right…

1. If an item is out of order with respect to its left-neighbor, swap left.
2. Keep on swapping left until the item is in order with respect to its left-neighbor.

Insertion sort is stable. Give an example.

Insertion Sort Stability

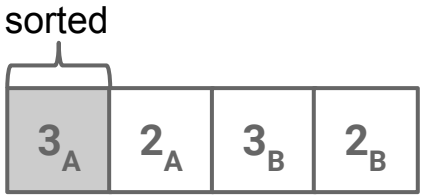
Answer

Build a sorted subarray (like selection sort) by using left-neighbor swaps for stability.

Scan from left to right…

1. If an item is out of order with respect to its left-neighbor, swap left.
2. Keep on swapping left until the item is in order with respect to its left-neighbor.

Insertion sort is stable. Give an example.



Insertion Sort Stability

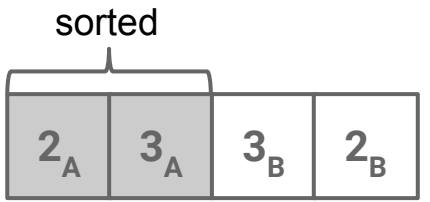
Answer

Build a sorted subarray (like selection sort) by using left-neighbor swaps for stability.

Scan from left to right…

1. If an item is out of order with respect to its left-neighbor, swap left.
2. Keep on swapping left until the item is in order with respect to its left-neighbor.

Insertion sort is stable. Give an example.



Insertion Sort Stability

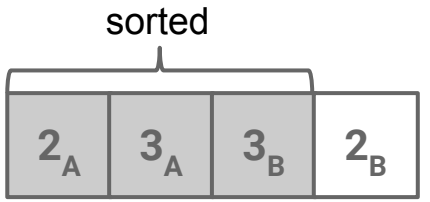
Answer

Build a sorted subarray (like selection sort) by using left-neighbor swaps for stability.

Scan from left to right…

1. If an item is out of order with respect to its left-neighbor, swap left.
2. Keep on swapping left until the item is in order with respect to its left-neighbor.

Insertion sort is stable. Give an example.



Insertion Sort Stability

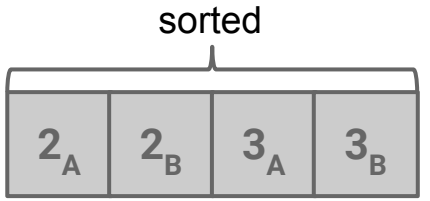
Answer

Build a sorted subarray (like selection sort) by using left-neighbor swaps for stability.

Scan from left to right…

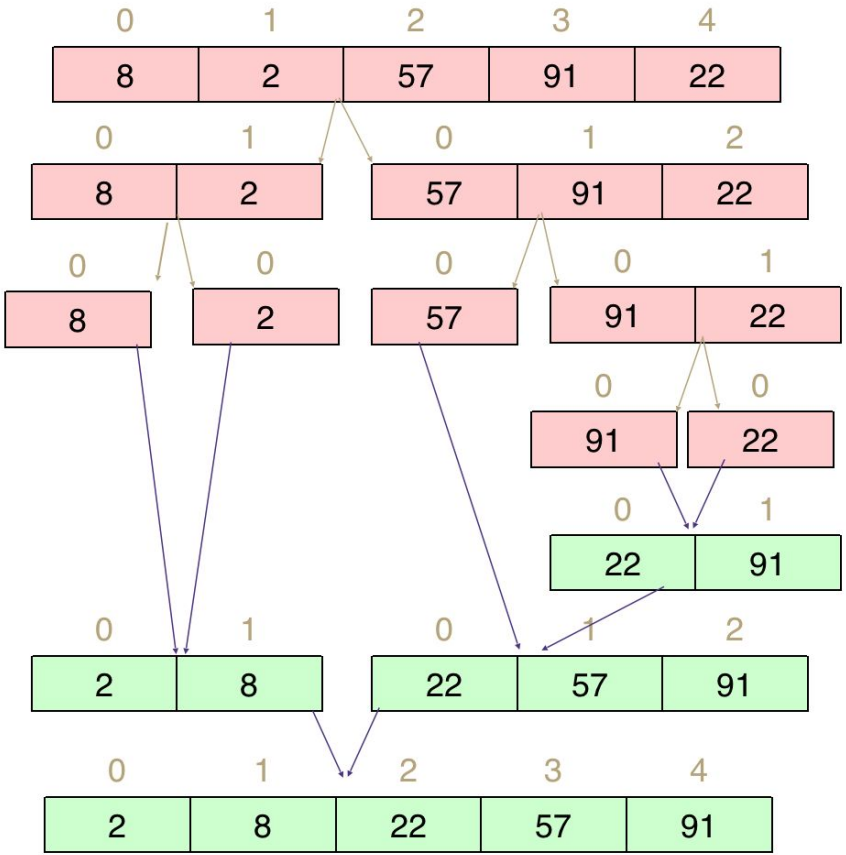
1. If an item is out of order with respect to its left-neighbor, swap left.
2. Keep on swapping left until the item is in order with respect to its left-neighbor.

Insertion sort is stable. Give an example.



Relative orders never broken!

Merge Sort

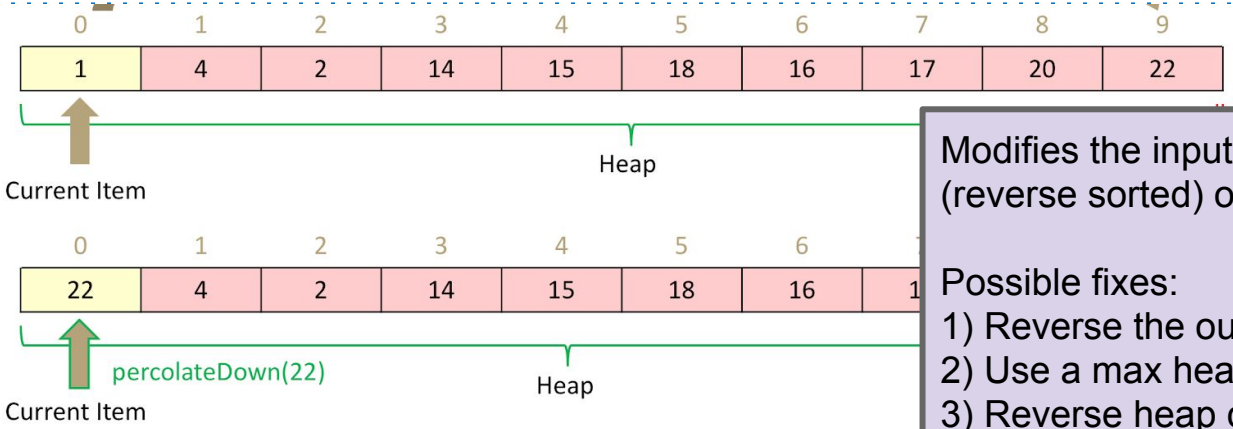



```
mergeSort(input) {
    if (input.length == 1)
        return
    else
        smallerHalf = mergeSort(new [0, ..., mid])
        largerHalf = mergeSort(new [mid + 1, ...])
        return merge(smallerHalf, largerHalf)
}
```

1. If array is of size 1, return.
2. Merge sort the left half.
3. Merge sort the right half.
4. Merge the two sorted halves.

Stable! → uses the fact that left-half items come before right-half items.

Heap Sort



Modifies the input to be in **descending** (reverse sorted) order!

Possible fixes:
1) Reverse the output (in O(n))
2) Use a max heap
3) Reverse heap compare function to emulate a max heap

```
public void inPlaceHeapSort(collection) {
    E[] heap = buildHeap(collection)
    for (n)
        heap[n - i - 1] = removeMin(heap)
}
```

In-place Heap Sort

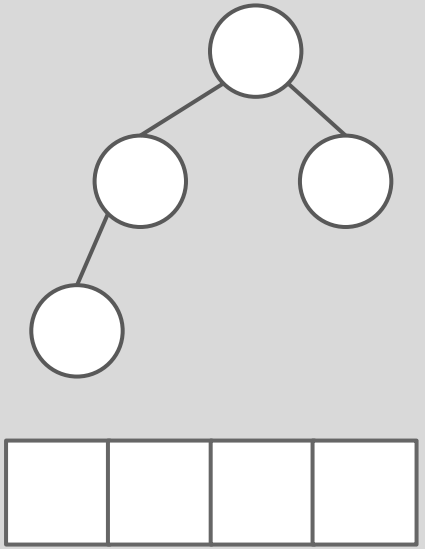
Avoid extra copies of data to save memory by treating the input array as a heap. We'll use a max heap for this sort.

1. **Floyd's buildHeap**. Efficient heap construction by percolating down on nodes in reverse level order (starting from the back of our input array).
2. Once heap-ified, call removeMax() and place max item after remainder of heap in array. Repeat this step N times.

Heap Sort Stability

Question

1. **Floyd's buildHeap**.
2. Repeat N times:
   a. Call removeMax().
   b. Put max item after heap in the array.

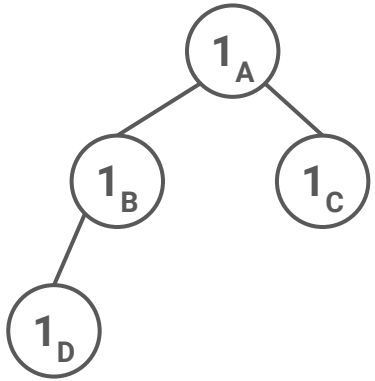


Heap sort is not stable. Give an example.

Heap Sort Stability

Answer

1. **Floyd's buildHeap**.
2. Repeat N times:
   a. Call removeMax().
   b. Put max item after heap in the array.

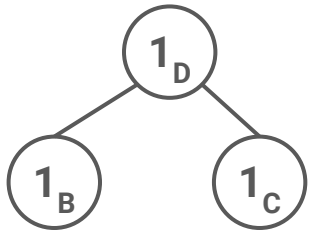


Heap sort is not stable. Give an example.

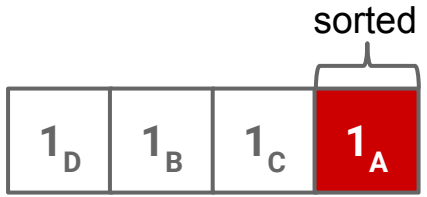
Heap Sort Stability

Answer

1. **Floyd's buildHeap**.
2. Repeat N times:
   a. Call removeMax().
   b. Put max item after heap in the array.



Heap sort is not stable. Give an example.

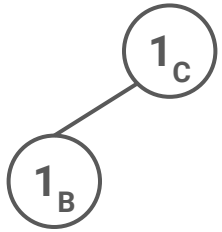


Relative order messed up!

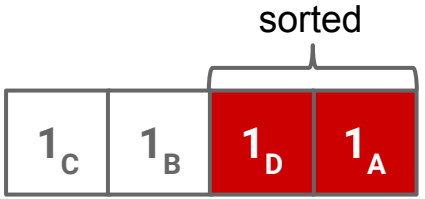
Heap Sort Stability

Answer

1. **Floyd's buildHeap**.
2. Repeat N times:
   a. Call removeMax().
   b. Put max item after heap in the array.



Heap sort is not stable. Give an example.



Relative orders messed up!

And so on...

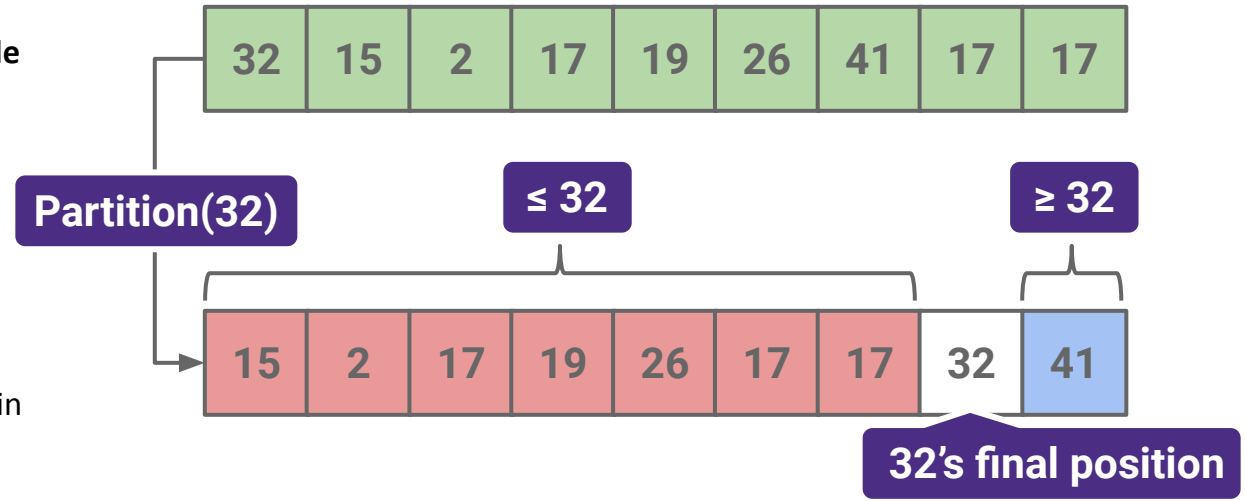
Quick Sort

1. Partition around a **pivot item**, e.g. leftmost item.
2. Quicksort left side, all keys ≤ pivot.
3. Quicksort right side, all keys ≥ pivot (can put equal items on left as well).

Stable?

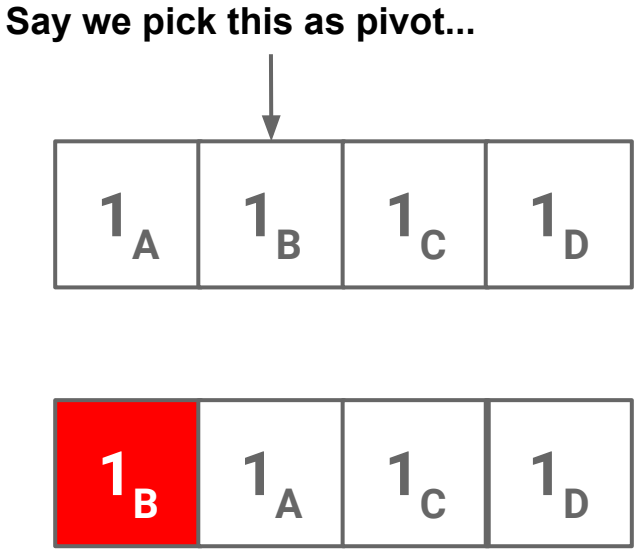
Quick Sort is **not stable** because equal items are arbitrarily placed on the left or right of the pivot so their relative orders may change.

Other partitioning algorithms can result in a stable Quick Sort.



Quick Sort Stability

Not stable!



Relative order messed up!

Level 3: In-Depth Sorting Walkthroughs

These are also Problems **2A-2E** on the Section 8 PDF.

To see the walkthroughs, click the **buttons!**

In-Depth Sorting Walkthroughs

Input:

- **A:** Insertion Sort — Answer
- **B:** Selection Sort — Answer
- **C:** In-place Heap Sort — Answer
- **D:** Merge Sort — Answer
- **E:** Quicksort — Answer