Unit 13 Algorithms

Sorting

Bubble Sort Description

The bubble sort is really easy to write but it is terribly slow. If you write a bubble sort, I would not be impressed. Infact, I would die of not impression.

The engine for the bubble sort is swapping two values. If you go down the array doing this enough times, you will be guaranteed to sort it. But you wouldn’t be impressing anyone.

Bubble Sort -1

1. Write a **`for`** loop using `j` that goes from `0` to `n-1`

   ```
   for (int j=0; j<a.length-1; j++)
   ```

2. In the loop, if `a[j]>a[j-1]` swap the values
3. Print the result.

   ```
   System.out.println(Arrays.toString(ar));
   ```

4. Copy this loop two more times. See what happened?
5. Each time you carry out a loop, one number gets sorted.

Bubble Sort

1. Now that you can see what is going on, keep the first loop and remove the rest.
2. Write a for loop outside the first one using i. Have it go from 1 to n-1
3. Now you should have a working bubble sort.
4. If you have i go down from N-1 to 1 (inclusive) and have j go from 0 to i, you can cut the time in half!

Selection Sort : Description

The engine for the selection sort is searching for a minimum value. You look for the smallest value in an array and put it into the first place. Then you look for the smallest value in the remaining array and place it in the second place. And so on and so on.

Selection sort 1

1. Write a **`for`** loop that uses the variable j and goes from `0` to `n-1`
2. Use the **`for`** loop to find the minimum value of the array and store its index.
3. Swap the values at 0 and the index (it will work if idx=0)

Print the array and you should see that the first number is the smallest number.

Selection sort 2

1. Write a **`for`** loop that uses the variable j and goes from `1` to `n-1`
2. Use the **`for`** loop to find the minimum value of the array and store its index.
3. Swap the values at 1 and the index.

Print the array and you should see that the first two numbers are sorted.

Selection sort 3

1. Write a **`for`** loop that uses the variable j and goes from `2` to `n-1 (humor me)`
2. Use the **`for`** loop to find the minimum value of the array and store its index.
3. Swap the values at 2 and the index.

Print the array and you should see that the first three numbers are sorted.

Selection sort - 4

If you want to, we can keep on copying and pasting the same code over and over again. So let's just turn it into a loop.

1. You know all those for loops we had? Keep the first one (the one going from 0 to n-1) and delete the rest.
2. Next write a for loop using i (`int i=0 ; i<array.length-1; i++`) and put it before the first for loop.
3. Change the index of the j for loop to i. (`int j=i ; j<array.length; j++`)
4. Everywhere you see a 0, replace with an i. You should have a working sort.

Insertion Sort Desc

Insertion is in many ways a very natural sorting algorithm.

You start with the first element and assume it is sorted. Then you get the next element and keep swapping it with the sorted elements until it is in its proper place. It is kind of like bubble sort except it stops comparisons earlier. It is still slow, though.

Insertion Sort 1.

Start with the following code. Print it out and see what it does.

```
Int j=1;
While (j>0 && a[j-1]>a[j])
{
    Int tmp=a[j];
    a[j]=a[j-1];
    a[j-1]=tmp;
  j--;
}
```

Insertion Sort 2

Copy this the code so you get to codes. Except change

```
int j=1
```

to

```
j=2
```

Insertion sort 3

See what it does? Each loop goes in partially orders the loop.

Delete the second j loop. Put the first under an i loop

```
for (int i=1 ; i<a.length ; i++ ) {

    j=i;

  while (j>0 ….
```