Improved Selection Sort

In a selection sort - the algorithm takes the minimum on every pass on the array and places it at its correct position. The idea is that the maxium is also taken and placed in its correct position. So in every pass, we keep track of both max and min, sorting the array from both ends.

Code for Improved Selection Sort

```
for(int i = 0; i < size -1; i++)
int pos = i
for(int k = i+1; k<size; k++)
if(arr[pos] > arr[k])
pos = k
end k loop
arr [i] = int temp
arr [i] = arr[pos]
arr[pos] = temp
```

Bubble Sort with Flag

Bubble sorts swap adjacent elements and swaps them if the first is smaller or greater than the other. At the end the smallest and largest will have 'bubbled' to the end of the array, this process continues with the next smallest and largest element to second last position

Code for Bubble Sort with a Flag

```
boolean sorted;
int i = size - 1;
do {        do while must execute at least once to ensure the array is sorted
sorted = true;
for(int j = 0; j < i; j++){
if(strArr[j].compareTo(strArr[j+1])>0){
String temp = strArr[j];
strArr[j] = strArr[j+1];
strArr[j+1] = temp;
sorted false;   if any swapping occurs, sorted is set to false. Since the array is not complete, thus needs to be do while must execute again to check the array is sorted.
}}
i--;}
while (sorted = false);
```

Sequential searching with a flag

Sequential search / Linear search

If we want to see if an array has a specific element we compare the first element and the value until we find the matching value or come to the end of the array.

Code for Sequential Search

```
int pos = 0;
String temp = "";
for(int i = 0; i<size; i++){
if(strArr[i].equalsIgnoreCase(st)){
pos = i;
temp = strArr[i];}}
return temp;
```

Binary Search

This algorithm assumes the array is already sorted and then divides the array in half. The search is done in the half of the array where the element resides. The process is repeated until the element is found.

Code for Binary Search

```
bubbleSortWithFlag();
int mid = 0, start = 0, pos = -1, end = size -1;
boolean found = false;
while(start <= end && found == false){   the while loop checks if the element has been found and checks for the end of the array
mid = (start + end) /2
if(st.compareTo(strArr[mid])<0)
end = mid -1;
}else{
found = true;
pos = mid;}}
return pos;
```


Sponsored by **Readable.com**
Measure your website readability!
https://readable.com