CSCI-2270
Recitation 4
Instructor: Boese

Sorting

Objectives

- Understand how the different sorting algorithms work
- Use <ctime> and generate random number to discuss the Big-O for each algorithm

---

Preparation:

Download the code from the website.

Define an array and initialize it as {9, 4, 7, 2, 8, 3, 5 }. Then add a printList function that takes the array and the size of the array , loops through the list and prints their values with a space between each value.

Selection Sort:

The algorithm can be described as follows: find smallest element in unsorted portion of array; swap it into its final position; continue until all elements sorted.

```
void selectSort(int arr[], int n)
{
    int index_min_value;
    int temp;

    for (int i=0; i<n-1; i++)   // outer loop tracks the sorted portion of the list
    {
        index_min_value = i; //set index_min_value to the current index of array

        for (int j=i+1; j < n; j++) // inner loop finds the next smallest value
        {
            if (                )
                    index_min_value=j;
        }

        if (index_min_value != i)    // swap to the correct position
        {
           temp = arr[i];
           arr[i] = arr[index_min_value];
           arr[index_min_value] = temp;
        }
    }
}
```

Complete the code and call the selection sort algorithm to sort your defined array. Then call printList function to see the sorting results step by step.

Insertion Sort:

The algorithm can be described as follows: assume first element in list is the sorted list; take first element from unsorted portion of list; shift all elements in sorted list that are

larger than the element; put the element into position in sorted list; continue for all elements in unsorted portion of list.

```
void insertion_sort (int arr[], int length)
{
    int j, temp;

    for (int i = 0; i < length; i++) // outer loop tracks the sorted portion of the list
    {
        j = i;
        while (                    )
        // move elements over until find the position for next element
        {
              temp = arr[j];
              arr[j] = arr[j-1];
              arr[j-1] = temp;
              j--;
        }
    }
}
```

Complete the code and call the selection sort algorithm to sort your defined array. Then call printList function to see the sorting results step by step.

Bubble Sort:

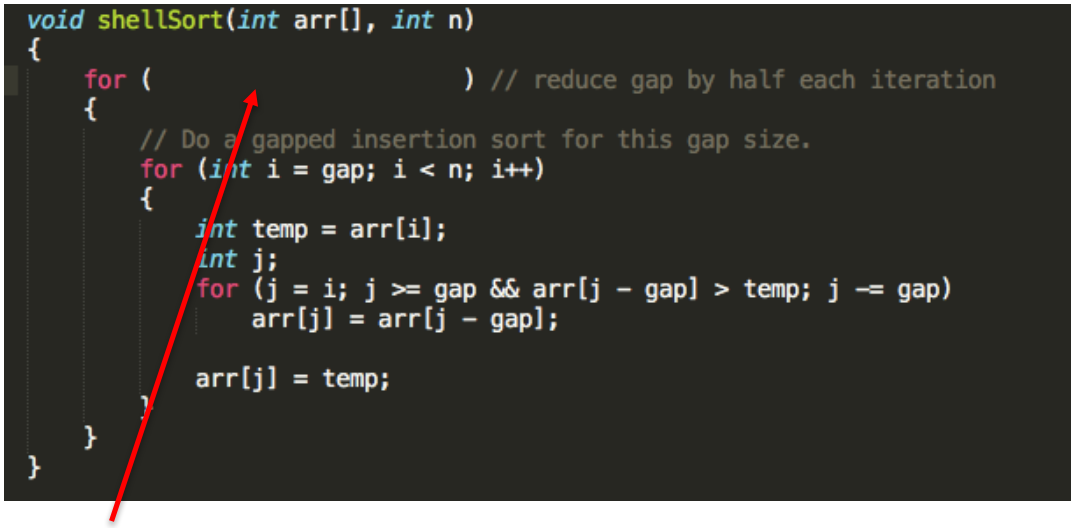
The algorithm can be described as follows: step through the list to be sorted and compare each pair of adjacent items; swap them if they are in the wrong order; pass through the list and repeat until no swaps are needed.

```
void swap(int *xp, int *yp)
{
    int temp = *xp;
    *xp = *yp;
    *yp = temp;
}
void bubbleSort(int arr[], int n)
{
   int i, j;
   for (i = 0; i < n-1; i++)
   {
       // Last i elements are already in place
       for (j = 0; j < n-i-1; j++)
           if (                  )
              swap(&arr[j], &arr[j+1]);
   }
}
```

Complete the code and call the selection sort algorithm to sort your defined array. Then call printList function to see the sorting results step by step

Shell Sort:

The algorithm can be described as follows: set gap size to (size of sequence)/2; then sort all subsequences determined by gap size by insertion sort; then set gap size to (gap size)/2 and repeat until gap size =1.



Complete the code and call the selection sort algorithm to sort your defined array. Then call printList function to see the sorting results step by step

Time complexity:

This part is only for TAs to demonstrate and students do not need to code it.

Use rand() and srand() to generate **n**(n=10000) random numbers. Store these numbers into an array and call these four sorting algorithms to sort. Use clock() to calculate the running time.

Efficiency: Shell sort > Insertion sort ~ Selection sort > Bubble sort

Question: What is the Big-O for each of these sorts and why?

To get credit for this lab exercise, show the TA the results of time complexity and sign the lab's completion log.