CS1002 – Programming Fundamentals

Lecture # 28
Tuesday, December 06, 2022
FALL 2022
FAST – NUCES, Faisalabad Campus

Muhammad Yousaf

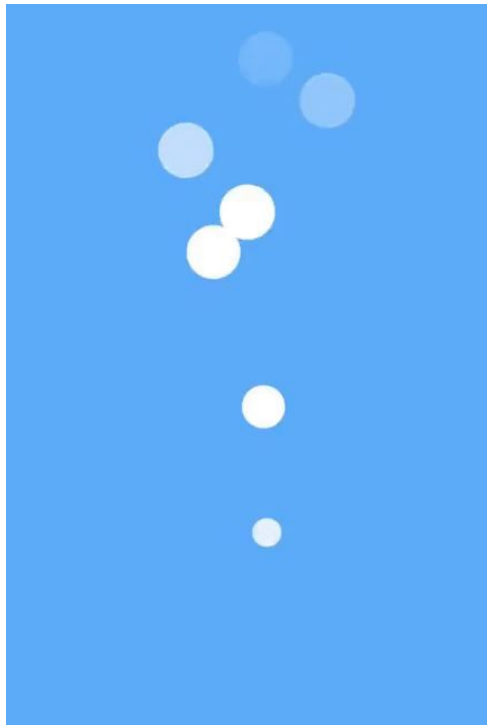
Outline

- Sorting Algorithms
  - Sorting 1D-Arrays
  - Bubble Sort
  - Insertion Sort

Bubble Sort

- Compares two adjacent items and if the item at index *i* is greater than item at index *i+1* then swap those two
- In this way largest element bubbles up to the end

```
bubbleSort(array)
    for i=0 to size - 1
            if leftElement > rightElement
                    swap leftElement and rightElement
end bubbleSort
```



Bubble Sort

- Sort an unsorted array
- Compares two adjacent items and if the item at index *i* is less than item at index *i+1* then swap those two
- In this way largest element bubbles up to the end

```
void bubblesort(int A[],int size){
    for ( int i = 0; i < size; ++i ){
        for(int j = 0 ; j < (size-i)-1 ; ++j){
        if(A[j] > A[j+1]){
            int temp = A[j] ;
            A[j] = A[j+1];
            A[j+1] = temp;
        }
        }
    }
}
```

Bubble Sort (First Pass)

First Pass with i = 0 and j = 1 to size-1									
[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]		j
16	30	24	7	62	45	5	55		
16	30	24	7	62	45	5	55	No Exchange	1
16	***24***	***30***	7	62	45	5	55	Exchange	2
16	24	***7***	***30***	62	45	5	55	Exchange	3
16	24	7	30	62	45	5	55	No Exchange	4
16	24	7	30	***45***	***62***	5	55	Exchange	5
16	24	7	30	45	***5***	***62***	55	Exchange	6
16	24	7	30	45	5	***55***	***62***	Exchange	7

Bubble Sort (Second Pass)

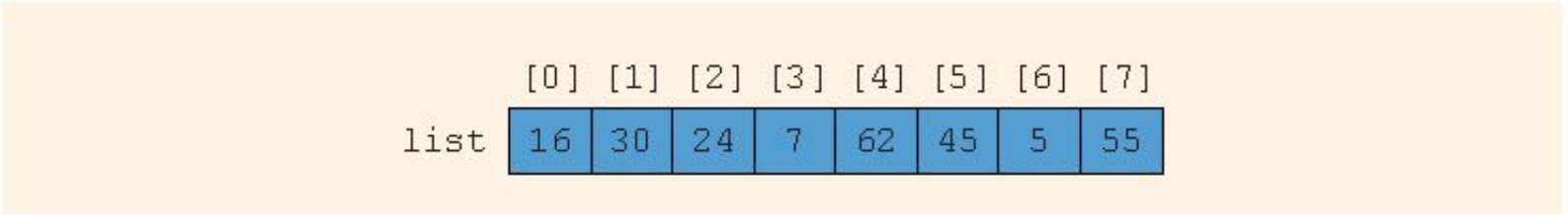
Second Pass with i = 1 and j = 1 to size-1									
[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]		j
16	24	7	30	45	5	55	62		
16	24	7	30	45	5	55	62	No Exchange	1
16	***7***	***24***	30	45	5	55	62	Exchange	2
16	7	24	30	45	5	55	62	No Exchange	3
16	7	24	30	45	5	55	62	No Exchange	4
16	7	24	30	***5***	***45***	55	62	Exchange	5
16	7	24	30	5	45	55	62	No Exchange	6
16	7	24	30	5	45	55	62	No Exchange	7

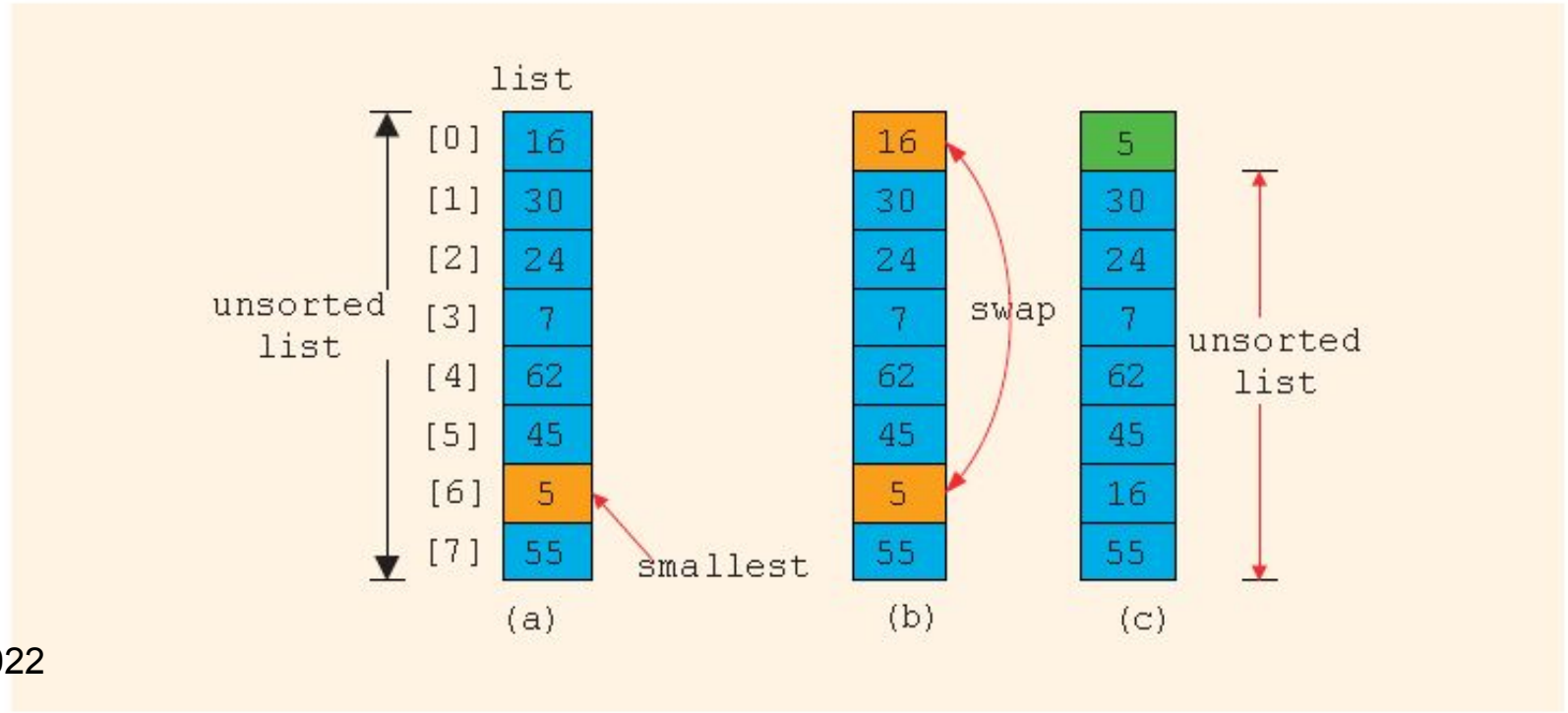
Selection Sort

- This algorithm finds the location of the smallest element in the unsorted portion of the list
  - Moves it to the top of the unsorted portion of the list
- The first time, we locate the smallest item in the entire list
- The second time, we locate the smallest item in the list starting from the second element in the list

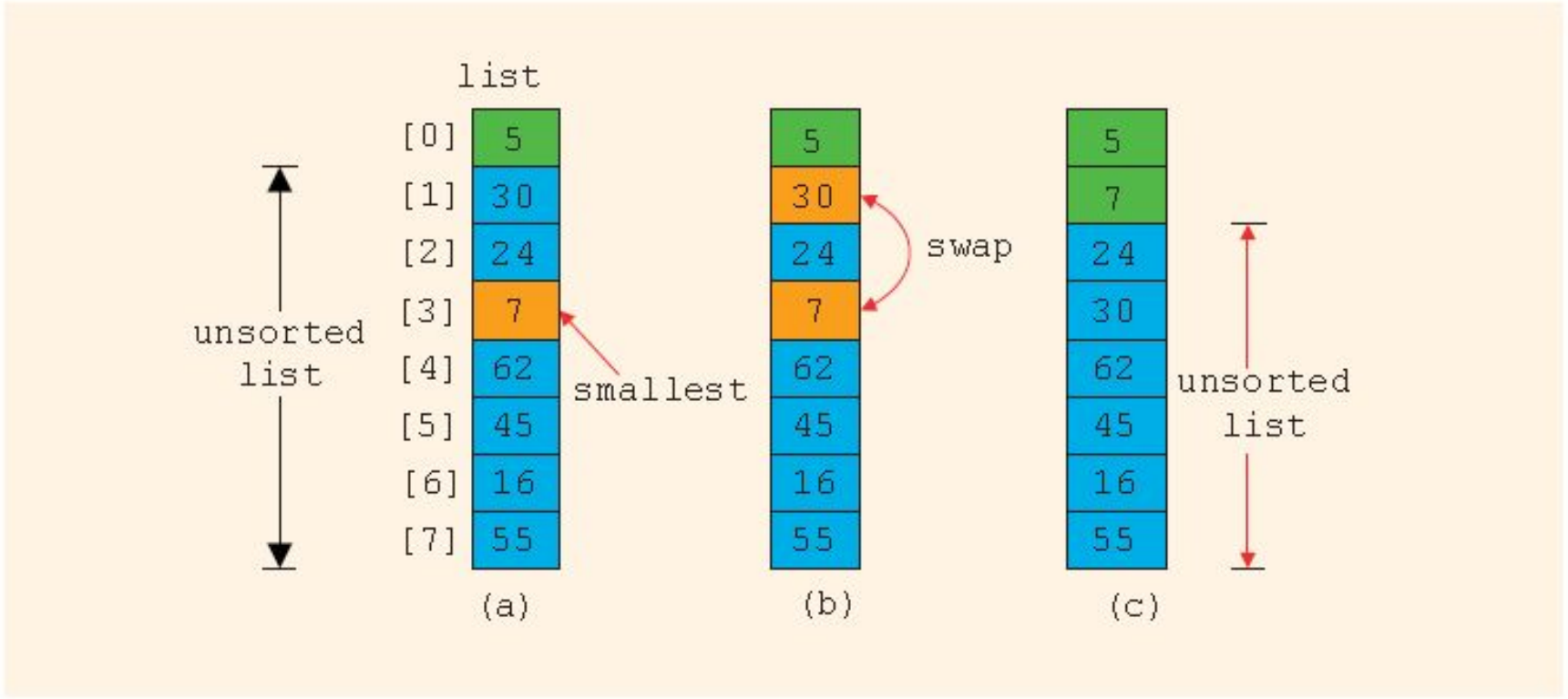
Contd..

Suppose you have the list shown in Figure 10-6.





Contd..


list
[0]
[1]
[2]
[3]
[4]
[5]
[6]
[7]
5
30
24
7
62
45
16
55
unsorted
list
smallest
(a)
swap
(b)
unsorted
list
(c)

Selection Sort

- In the unsorted portion of the list:

a. Find the location of the smallest element.

b. Move the smallest element to the beginning of the unsorted list.

```
for (index = 0; index < LENGTH; ++index)
{
    a. Find the location, smallestIndex, of the smallest element in the
       list[index] ... list[LENGTH -1]
    b. Swap the smallest element with array[index]. That is swap
       list[smallestIndex] with list[index]
}
```

Contd..

- Step a is similar to the algorithm for finding the index of the largest item in the list

```
smallestIndex = index; //Assume first element is the smallest
for (location = index + 1; location < LENGTH; ++location)
{
    if (list[location] < list[smallestIndex])
        smallestIndex = location;
//Current element in the list                                    //is
smaller than the smallest                                  //so far, so
update Smallest Index
}
```

Contd..

- Step b swaps the contents of **list[smallestIndex]** with **list[index].** The following statements accomplish this task:

```
temp = list[smallestIndex];
list[smallestIndex] = list[index];
list[index] = temp;
```

Algorithms – Selection Sort

- The following function, **selectionSort**, implements the selection sort algorithm:

```
void selectionSort(int list[], int length)
{
    int i, j, smallestIndex , temp;
    for (i = 0; i < length - 1; i++)
    {
        smallestIndex = i;
        for (j = i + 1; j < length; j++)
        {
            if (list[j] < list[smallestIndex])
                smallestIndex = j;
        }
        temp = list[smallestIndex];
        list[smallestIndex] = list[i];
        list[i] = temp;
    }
}
```

Contd..

```
#include <iostream>
using namespace std;

const int LENGTH = 20;

void selectionSort(int[], int);
int main()
{
    int list[LENGTH];

    selectionSort(list, LENGTH);

    cout << "After Sorting the elements are\n";
    for (int i = 0; i < LENGTH; ++i) {
        cout << list[i]<<" ";
    }
    cout << endl;

    return 0;
}
```

Questions