Institute of Technology of Cambodia

Department Information and Communication Engineering

Algorithm and Programming

Report: Research about sorting algorithm

Lecturer : BOU Channa

Student's Name	ID
Roeun Pacharoth	e20170731
Tho Channarith	e20170958
Ros Seireywathna	e20170740
Kheng Sokuntheary	e20170337
Group :	I3B-GICB

2019-2020

Semester II

Table of Contents

Objective

- Want us to do some research what the sorting algorithm be like
- Understand how the sorting algorithm work by many different way to simplify the complexity of algorithm.
- And we also understand the big' O is the main component of those sorting algorithm.

Insert sort:

- First we talk about insert sorting algorithm , Insert sort algorithm is algorithm that we simplify the data to run as the order from left to the right and swap the data.
- For example we have like 5 3 2 4 1 as the value in index of the array so we can say that the value we start we insert sorting, we compare from left to the right.
- First we start from the number 5, if 5 bigger than the next index value so we swap the data or we exchange the data like 3 5 2 4 1 so we do it again 2 3 5 4 1. And it makes sure they are in the correct position. And then it will do again it will output 2 3 4 5 1 → 1 2 3 4 5 . That is the sorting algorithm
- So how to simplify as code ? The thing we just choose the big' O (O(N^2)) method to compare the data in each elements.
- Example

```
for (int initialize = 1; initialize<size-1; initialize++) do //we start index one to compare with
                                                                  index zero
    index = initialize
    while (index>0  and array[index-1]> array[index]) do // we check left value in index has
                                                                  data bigger we swap the data
        swap(array[index], array[index-1]) //so data has been swapped
        index = index -1
```

Bubble Sort:

- Bubble sort is algorithm to sort the data as compare data by compare two by two data in each array. When it meet big data in the left so it will compare to other data at the right and then it will swap and that big data will compare to other next element of the array if it is smaller so it wont swap and it will check next data to find big data and swap them to the right until end and it will check again and do that process again.
- For example we have data in array of each elements, we wont explain much in this point so it might be easier by see the output : 2 10 8 1 7 → 2 10 8 1 7 → 2 8 10 1 7 → 2 8 1 10 7 →

2 8 1 7 10 so we got 10 last element so we sort them again: 2 8 1 7 10 → 2 1 8 7 10 → 2 1 7 8 10 → 1 2 7 8 10, so we get them as order .

- To do as code we just use big’s O square

```
for initialize from 1 to n_element
    for index from 0 to n_element -1
        if array[index]>array[index+1]
            swap(array[index],array[index+1]) //if they bigger so we swap data to the right
                                                 element until cant find big it is stop and
                                                 check again after this loop
```

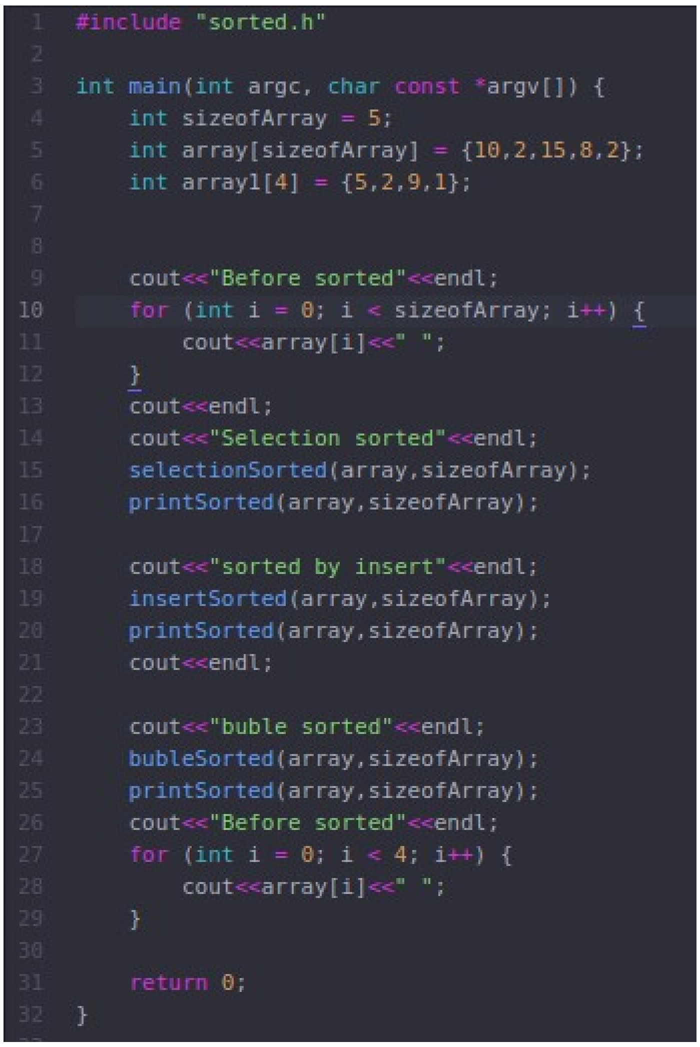
Selection sort:

- This selection sort is algorithm that we select the smallest and unsorted item and move it to sorted position. So mostly it check one element to another elements if it find another elements smaller so it will swap small to the left element and those big to right element.
- To understand that so we just give example and output how it is selected and compare to other item: 2 9 1 5 8 check 2 & 9 check 2& 1 check 1& 5 check 1& 8 and then swap 2 to 1 like: 1 9 2 5 8 and then check 9&2 ,2&5, 2&8 and then swap 9 to 2, 2to9: 1 2 9 5 8 and then check 9&5, 5&8 swap 9 to 5 ,5 to 9: 1 2 5 9 8 and then check 9 & 8 and swap :1 2 5 8 9, so this is the result.
- So to do this algorithm we use big O square:

```
for index from 0  to n_element-1
    index_Min = index //minimum to select smallest data
    for initialize from index+1 to n_element
        if array[initialize]<array[index_Min]
            index_Min = initialize            //change the index we set first time equal
                                                index to initial
    if index_Min != initialize
        swap(array[index],array[index_Min]) //check if it is changed to initialize so we
                                              we swap data
```

Source code:

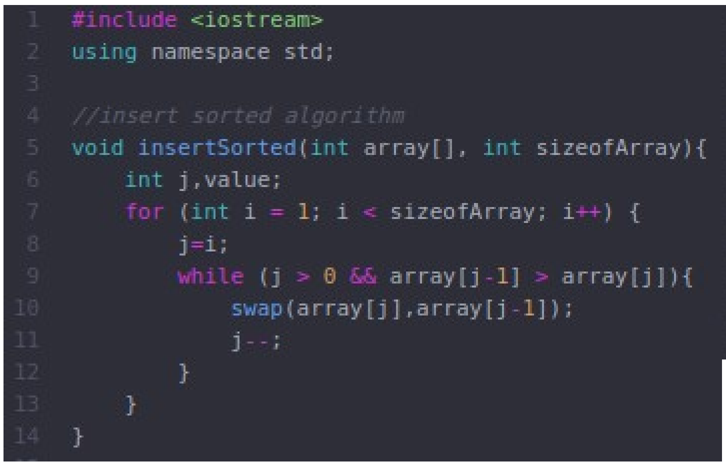
Main Program



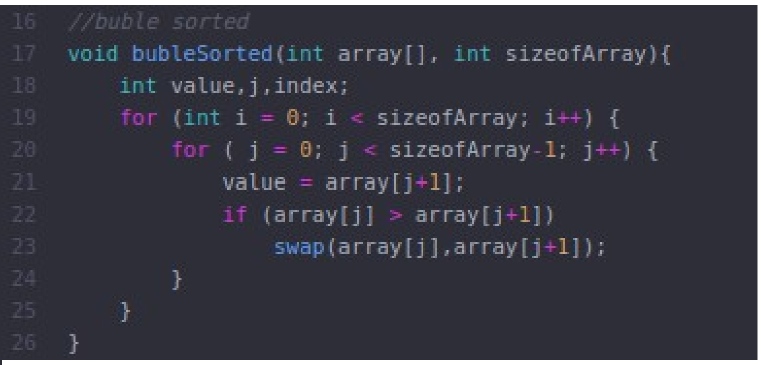
```cpp
#include "sorted.h"

int main(int argc, char const *argv[]) {
    int sizeofArray = 5;
    int array[sizeofArray] = {10,2,15,8,2};
    int array1[4] = {5,2,9,1};


    cout<<"Before sorted"<<endl;
    for (int i = 0; i < sizeofArray; i++) {
        cout<<array[i]<<" ";
    }
    cout<<endl;
    cout<<"Selection sorted"<<endl;
    selectionSorted(array,sizeofArray);
    printSorted(array,sizeofArray);

    cout<<"sorted by insert"<<endl;
    insertSorted(array,sizeofArray);
    printSorted(array,sizeofArray);
    cout<<endl;

    cout<<"buble sorted"<<endl;
    bubleSorted(array,sizeofArray);
    printSorted(array,sizeofArray);
    cout<<"Before sorted"<<endl;
    for (int i = 0; i < 4; i++) {
        cout<<array[i]<<" ";
    }

    return 0;
}
```

In Sorted.h

Insert sorting function and bubble sorting function

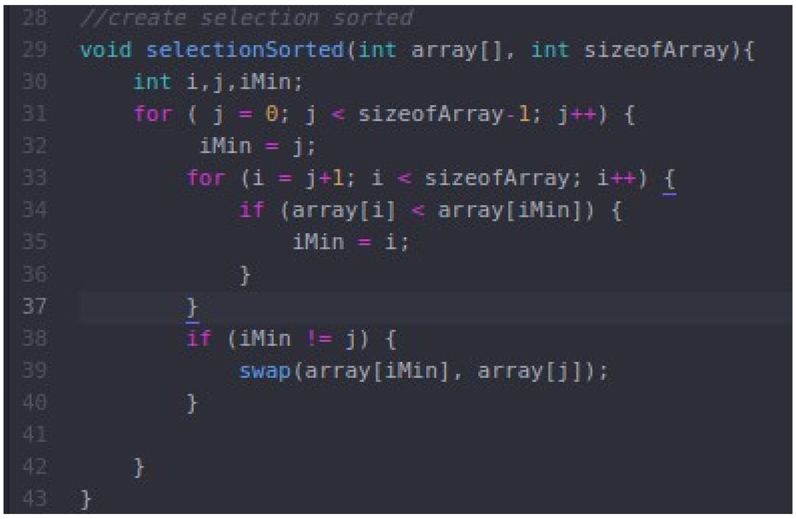


```cpp
#include <iostream>
using namespace std;

//insert sorted algorithm
void insertSorted(int array[], int sizeofArray){
    int j,value;
    for (int i = 1; i < sizeofArray; i++) {
        j=i;
        while (j > 0 && array[j-1] > array[j]){
            swap(array[j],array[j-1]);
            j--;
        }
    }
}
```



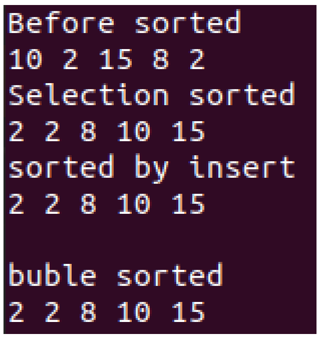
```cpp
//buble sorted
void bubleSorted(int array[], int sizeofArray){
    int value,j,index;
    for (int i = 0; i < sizeofArray; i++) {
        for ( j = 0; j < sizeofArray-1; j++) {
            value = array[j+1];
            if (array[j] > array[j+1])
                swap(array[j],array[j+1]);
        }
    }
}
```

Selected sorting function



```
//create selection sorted
void selectionSorted(int array[], int sizeofArray){
    int i,j,iMin;
    for ( j = 0; j < sizeofArray-1; j++) {
         iMin = j;
        for (i = j+1; i < sizeofArray; i++) {
            if (array[i] < array[iMin]) {
                iMin = i;
            }
        }
        if (iMin != j) {
            swap(array[iMin], array[j]);
        }

    }
}
```

Output:



```
Before sorted
10 2 15 8 2
Selection sorted
2 2 8 10 15
sorted by insert
2 2 8 10 15

buble sorted
2 2 8 10 15
```

In Conclusion

- We gain some knowledge about how these 3 algorithm work but the rest like quick sort merge sort and heap sort , they are really average case which simplify the complexity algorithm. And mostly these three algorithm depend on the program, so we don’t research about these three algorithm much.
- For three above that we just describe and explain, we just explain what we know after doing those research and according to the video that teacher gave.