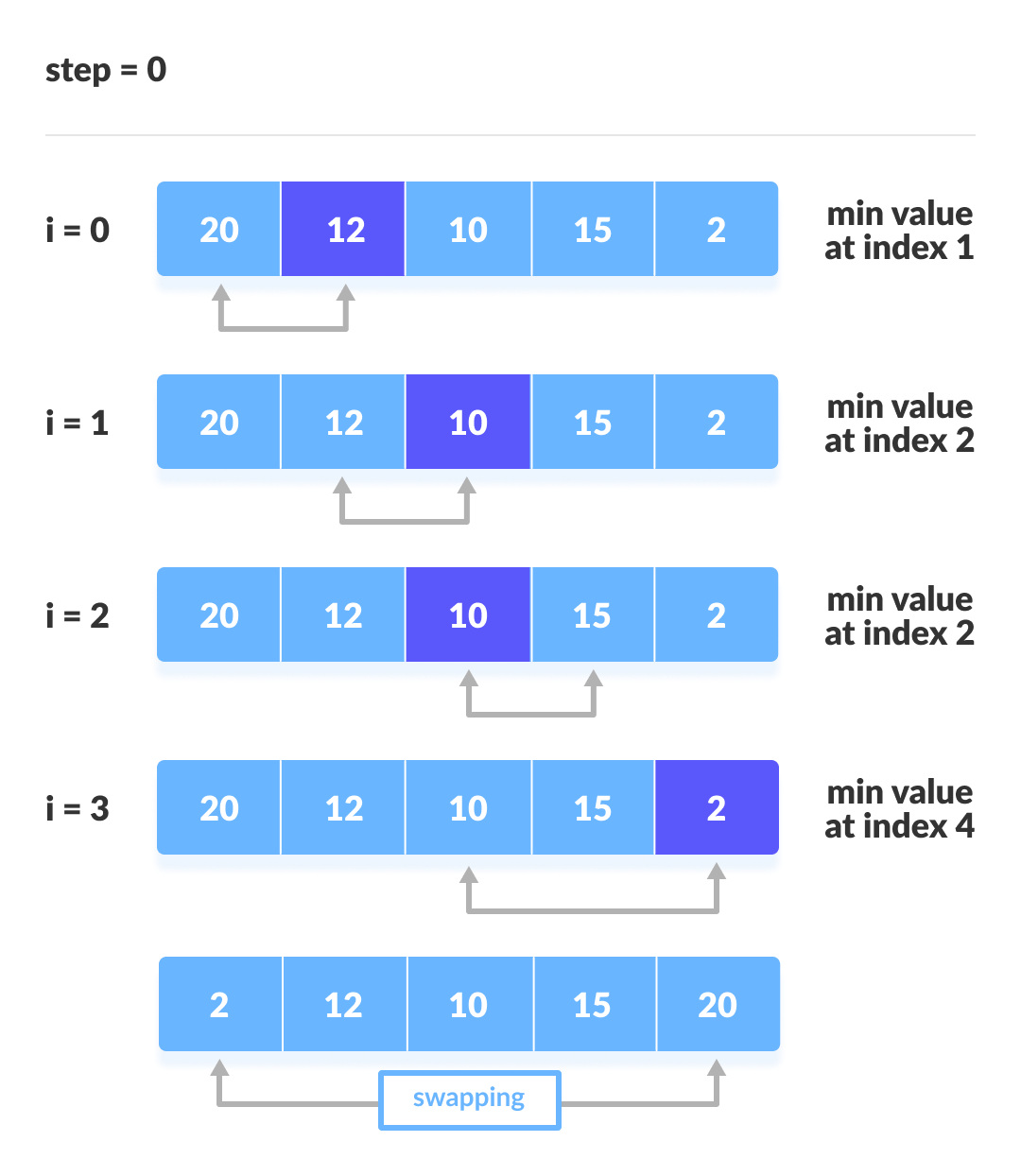
Sorting

**Selection Sort:**

এই সর্ট এর মাধমে একটি unsorted arrey কে sorted করা যায়



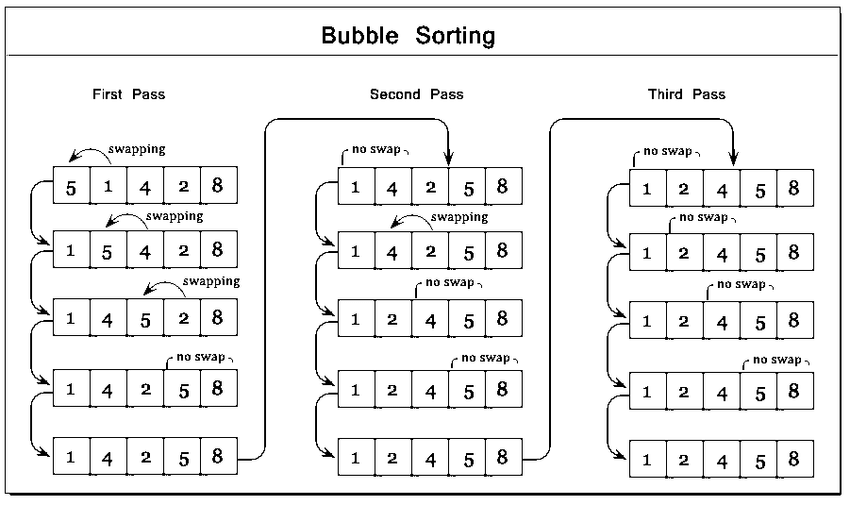
Code:

|  |
| --- |
| #include<bits/stdc++.h> using namespace std;  int main() {  int ary[]={5,3,4,7,2,1,6,8,9,13,12,11,10};  int sz=sizeof(ary)/sizeof(ary[0]);   for(int i=0;i<sz;i++)  {      for(int j=i+1;j<sz;j++)      {          if(ary[i]>ary[j])          {              int temp=ary[i];              ary[i]=ary[j];              ary[j]=temp;          }      }  }   for(int i=0;i<sz;i++)  {      cout<<ary[i]<<" ";  }  } |

Time Complexity : O( n^2 )

**Bubble sort:**

Bubble sort এ একটি array এর দুইটা সংখ্যা এর মধ্যে compare হয় যদি প্রথম সংখ্যাটি দ্বিতীয় সংখ্যার থেকে বড় হয় তবে প্রথম সংখ্যা টি দ্বিতীয় সংখ্যার জায়গাই বসবে এবং দ্বিতীয় সংখাটি প্রথম সংখ্যার জায়গায় বসবে । এভাবে চলতে থাকে যতক্ষন না সবচেয়ে বড় সংখ্যাটি সবার পিছনে না যায় এবং এভাবে চলতে থাকে যতক্ষননা সবগুলো সংখ্যা sort হয়ে না যায় ।



Code:

|  |  |
| --- | --- |
| |  | | --- | | #include<bits/stdc++.h> using namespace std;  int main() {     int ary[]= {5,3,4,7,2,1,6,8,9,13,12,11,10};     int sz=sizeof(ary)/sizeof(ary[0]);      for(int i=0; i<sz; i++)     {         for(int j=0; j<sz-i-1; j++)         {             if(ary[j]>ary[j+1])             {                 int temp=ary[j+1];                 ary[j+1]=ary[j];                 ary[j]=temp;             }          }      }      for(int i=0; i<sz; i++)     {         cout<<ary[i]<<" ";     }  } | |

Complexity : O(n^2)

**Insertion Sort:**

How Insertion Sort Works?

We take an unsorted array for our example.

Unsorted Array

Insertion sort compares the first two elements.

Insertion Sort

It finds that both 14 and 33 are already in ascending order. For now, 14 is in sorted sub-list.

Insertion Sort

Insertion sort moves ahead and compares 33 with 27.

Insertion Sort

And finds that 33 is not in the correct position.

Insertion Sort

It swaps 33 with 27. It also checks with all the elements of sorted sub-list. Here we see that the sorted sub-list has only one element 14, and 27 is greater than 14. Hence, the sorted sub-list remains sorted after swapping.

Insertion Sort

By now we have 14 and 27 in the sorted sub-list. Next, it compares 33 with 10.

Insertion Sort

These values are not in a sorted order.

Insertion Sort

So we swap them.

Insertion Sort

However, swapping makes 27 and 10 unsorted.

Insertion Sort

Hence, we swap them too.

Insertion Sort

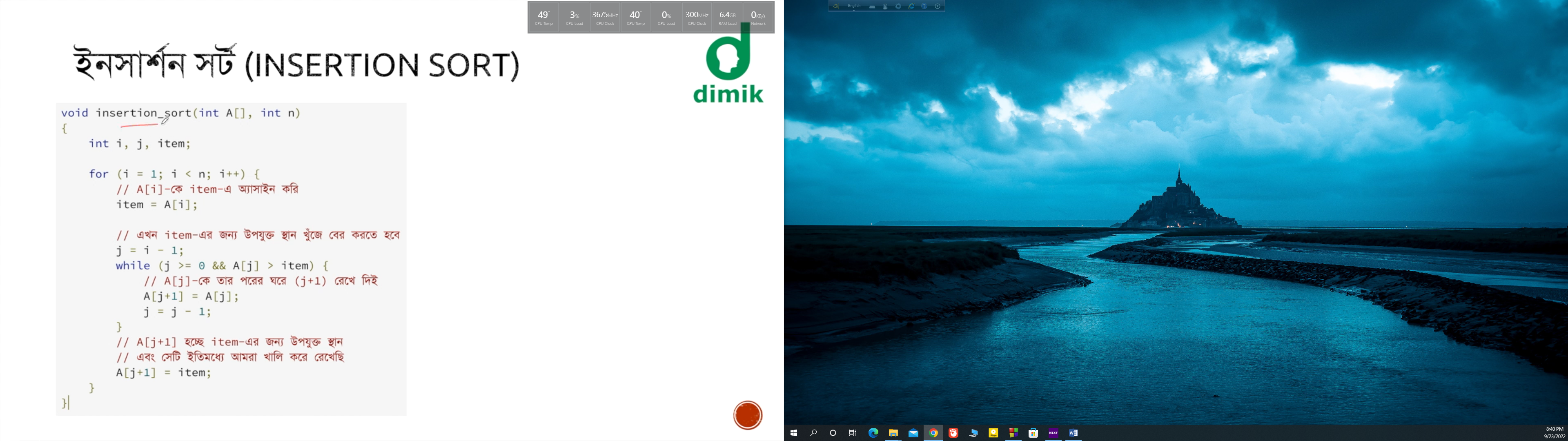
Again we find 14 and 10 in an unsorted order.

Insertion Sort

We swap them again. By the end of third iteration, we have a sorted sub-list of 4 items.

Insertion Sort

This process goes on until all the unsorted values are covered in a sorted sub-list. Now we shall see some programming aspects of insertion sort.



Code:

|  |  |
| --- | --- |
| |  | | --- | | include<bits/stdc++.h> using namespace std; int main() {     int ary[]={13,7,11,8,4,5,2,0,1};      int sz=sizeof(ary)/sizeof(ary[0]);  //get the size of the array      for( int i=1;i<sz;i++)  //make a for loop from index 1 to sz     {         int c=ary[i];    //insert the value of ary[i] into c         int j=i-1;  //initialize j=i-1         while(j>=0 && ary[j]>c)  //while লুপ টা কতক্ষন চলবে , যতক্ষন না  ary[i] c এর থেকে বড় না হয়         {             ary[j+1]=ary[j];  // ary[j] কে তার পরের ঘর এর স্থানান্তর করি । মানে ary[j+1]= ary[j]             j--;                               }         ary[j+1]=c;  // a[j+1] কে খালি করে ফেলছি এবং এর মধ্যে c কে রেখে দেই        }      for(auto u:ary) cout<<u<<" "; } | |

Vedio link: <https://www.youtube.com/watch?v=nKzEJWbkPbQ>