

School of Computers and Information Engineering**STUDENT LEARNING ACTIVITY**

Fall Semester 2019

Data Structures (SOC 2010)**SM**

Team 12

Submitted by:

U1810032, Allanazarov Madiyorbek (001)

U1810048, Makhamadaliev Ibrokhimjon (001)

U1810064, Akhtamov Amal (001)

U1810108, Anvarkhujaev Saidkamol (001)

U1810123, Asrolkhujayeva Barno (001)

**TOSHKENT SHAHRIDAGI INHA UNIVERSITETI**
INHA UNIVERSITY IN TASHKENT

- Problem Introduction

One of the nowadays' main problems that students face up is the *timetable*. Which causes further problem:

First, fresh meal due to different schedule of groups. Canteens cook their meal in fixed time, which is the same for all five studying days in a week. The thing is that, when (in particular time frame) the largest number of students have free time (break) go to the canteen to have a meal, they usually come up against not so serious but very irritating problem.

-There is no meal yet in canteen, if this time frame earlier than fixed one.

-OR There is no fresh (hot) meal in canteen, it's already get cold, if this time later than fixed one.

As a consequence both student and canteen will be in lost, since if there is no food cooked yet vast majority of student's will stay hungry and canteen will lose in terms of finances, else if there is no fresh meal vast majority will have to eat microwave preheated meal, which is harmful for the health.

As there is nothing we can do with timetable, we can simply arrange things according to students' spare time. Doing it manually will be really difficult, therefore we need some algorithm and data structure to perform the task. Resulting application was done by combining linear and nonlinear data structures; it has lots of applications, including:

- In last couple of years the number of conferences and other different events is being noticeable increased in IUT life. Most of them include useful and interesting speech, information. This type of events should be held with large audience, there for It is supposed to organize them during five studying days, without making students miss the lectures. Unfortunately, due to tied and not so comfortable timetable majority of interested student are torn between their desire to participate and studies, together with attendance, assignments, catching up the topic, quiz and so on. In addition, it will be impossible to keep a fine line between social life and study, which leads to stress and other serious health problems, which again effect on academic performance.

Some might argue that a student has weekends to have a fun, but that who tells it is not a student, definitely. Being outside the whole week takes the energy so they need a rest, in addition there will be housework and of course deadlines.

The problem with timetable does not only touch students.

- Operations included in the application

➤ Data structure:

In *our* timetable related *application*, due to the absence of basic operations like deletion and addition, we used Array because of it's time complexity($O(1)$). To express the day, group and exact time we used 3D array-(array[days][group][time]) we

added all the days which is constant 5. `temp[n] = array[days][0][0]+.... array[days][n][n]`. After that, to find the minimum of day which is leftmost we used BST and we used it because it is one of the most efficient way to find minimum at unsorted data and in the worst-case it may take only $O(n)$ time complexity. Additionally it has some functions:

1. **Node* create(int data,int time)**. In this function it creates leaves which would be added to our BST.
2. **void insert(Node* &root, int data,int time)**. In this function it add leaves to BST related to their data.
3. **Node* displayLeftMost(Node* root)**. In this function it find the node which have the least data.
4. **Switch(displayLeftMost(root)->time)**. In the final function it makes our program little bit user-friendly, which shows time like "There have free time until 11-12, 13-14 ... 17-18".

➤ How useful application is:

The purpose is to inform the canteen staff about the time when most of the students have breaks. So the staff will be ready for the big stream of students and serve effectively.

Without this system canteen workers will not be able to orient (especially in the beginning of each semester) on food quantity to keep hot, as if they do it with all amount, it might spoil and of course maintain chemical changings. On the other hand, not heating any amount effects on service and canteen budget, in

addition heating all of the food with microwave ovens causes the huge consumption of electricity and food's getting waves. In fact it might cause further health problems.

If any health problem occurs, it will definitely effect on both: student's and teacher's academic performance together with university itself and canteen.

As you see, just a simple problem, as it seams, might lead to serious problems. Because of its simplicity, high speed, and making life easier, our application is beneficial for all sides.

➤ Limitation:

The application is not *fully completed*, not all the groups are included. Also, manual insertion per semester is needed.

➤ Scope:

Regarding it's simplicity, the application(finding the exact day and time when most of the students have break) can be used not only within IUT, but In other universities, cafes for students and private schools. In addition it can be used for conferences and other events that require students as participants and organizers. Additionally, it may be useful anywhere when you should find the most free time of the group of people.

➤ Code

```
#include <iostream>
using namespace std;
struct Node {
    int data;
    int time;
    Node* left;
    Node* right;
};

Node* create(int data,int time)

{
    Node* temp = new Node();
    temp->left=temp->right = NULL;
    temp->data = data;
    temp->time = time;
    return temp;
}

void insert(Node* &root, int data,int time)
{
    if(root== NULL)
        root = create(data,time);

    else if(root->data > data)
        insert(root->left, data,time);

    else
        insert(root->right, data,time);
}

int displayLeftMost(Node* &root)
{
    while(root->left!=NULL)
```

```

    {
        root = root->left;
    }

    return root->time;
}

int main() {
    //1st Group 19-01, 2nd Group 18-04, 3rd Group 16-02

    //array[x][y][z] = 'x' for days, 'y' for 3 groupd which
    is above, 'z' for timeline which is below

    //0) 9-10 1)10-11 2) 11-12 3) 12-13 4) 13-14 5)14-15 6)
    15-16 7)16-17 8)17-18
    int array[5][3][9] = {
        //Monday
        {

            //1st Group
            {0,1,1,1,1,0,0,0,0},

            //2nd Group
            {1,1,1,1,1,1,1,1,0},
            //3rd Group
            {0,1,1,0,1,1,0,0,0} },
        //Tuesday
        {
            //1st Group
            {1,1,0,0,0,1,1,1,0},
            //2nd Group
            {0,0,1,1,0,0,0,1,1},
            //3rd Group
            {0,1,1,0,0,0,0,0,0}
        },
        //Wednesday
        {
            //1st Group

```

```

        {0,0,0,0,1,1,1,1,1},
        //2nd Group
        {0,1,1,1,1,0,1,1,0},
        //3rd Group
        {1,1,1,1,1,1,0,1,1}
    },
    //Thursday
    {
        //1st Group
        {0,1,1,0,1,1,0,0,0},
        //2nd Group
        {1,1,1,0,0,0,0,0,0},
        //3rd Group
        {0,0,0,1,1,1,0,0,0}
    },
    //Friday
    {
        //1st Group
        {1,1,0,0,1,1,0,0,0},
        //2nd Group
        {0,0,0,0,1,1,1,1,0},
        //3rd Group
        {1,1,0,1,1,0,0,1,1}
    }
};

cout<<"Choose the day: "<<endl;
cout<<"1.Monday\t 2.Tuesday\t 3.Wednesday\t 4.Thursday\t
5.Friday"<<endl;
int x;
cin>>x;
x= x-1;
int temp[9] = {};
for(int i =0; i < 3; i++)
{
    temp[0] += array[x][i][0];
    temp[1] += array[x][i][1];
    temp[2] += array[x][i][2];
    temp[3] += array[x][i][3];
    temp[4] += array[x][i][4];
}

```



```
temp[5] += array[x][i][5];
temp[6] += array[x][i][6];
temp[7] += array[x][i][7];
temp[8] += array[x][i][8];
}
```

```
// for(int i =0; i<9; i++)
// {
//     cout<<temp[i]<<endl;
// }
Node* root = NULL;
//int n;
//int number;
//cout<<"Enter the value: "<<endl;
//cin>>n;
for(int i = 0; i<9; i++)
{
    insert(root,temp[i], i);
    //cin>>number,insert(root, number);
}
string appTime;
switch(displayLeftMost(root))
{
    case 0: {
        appTime = "9-10";
    }
    break;
    case 1: {
        appTime = "10-11";
    }
    break;
    case 2: {
        appTime = "11-12";
    }
    break;
    case 3: {
```

```
        appTime = "12-13";
    }
    break;
case 4: {
    appTime = "13-14";
}
    break;
case 5: {
    appTime = "14-15";
}
    break;
case 6: {
    appTime = "15-16";
}
    break;
case 7: {
    appTime = "16-17";
}
    break;
case 8: {
    appTime = "17-18";
}
    break;
}
    cout<<"The most efficient time to make a meeting is:
"<<appTime<<endl;
    return 0;
}
```

The importance of data structures in CS.

Any problem in Computer Science requires use of Algorithms to solve it. One problem can be solved using different algorithms with different efficiency. And the goal of any Computer Scientist is to use the most efficient one. And there arises the importance of Data Structures.

Data Structures are the key part of many computer algorithms as they allow the programmers to manage data in an efficient way. A right selection of data structure can enhance the efficiency of computer program or algorithm in a better way. Data Structure can be defined as the collection of data objects which provides a way of storing and managing data in the computer so that it can be used. Various Data Structures types are arrays, Linked List, Stack, Queue, etc. Data Structures are widely used in almost every aspect of Computer Science for simple as well as complex computations. Data structures are used in all areas of computer science such as Artificial Intelligence, graphics, Operating system etc.

Increase in data usage and in algorithm's complexities, which can affect the performance of the application and can create some aspects to concern about:

Speed: To handle very large data, high-speed processing is required, but with growing data processor may fail to achieve required processing speed.

Data Search: Getting a particular record from database should be quick and with optimum use of resources.

Multiple requests: To handle simultaneous requests from multiple users

In order to work on concern areas, data structures are used. Data is organized to form a data structure in such a way that all items are not required to be searched and required data can be searched instant

Memory usage efficiency: With efficient use of data structure memory usage can be optimized, for instant we can use linked list versus arrays when we are not sure about the size of data. When there is no more use of memory, it can be released.

Reusable: Data structures are reusable, i.e. once we have implemented a particular data structure, we can use it at any other place. Implementation of data structures can be compiled into libraries which can be used by different clients.

Data structures are the basic building block of any programming language, complex computations.

2) How we will benefit after finishing this course?

After finishing this course(if we **studied hard**, and **acquired** at least **100%** of what professor taught us) , we will be aware of basics of Data Structures, which will enable us to choose the “most”(sounds quite exaggerated) relevant approach in building algorithm with “best” efficiency in terms of time and

space complexity, for **different** kind of problems. In general then we will be able to see and come up with efficient solution to **particular** problems requiring complex algorithms. However, if put the “efficiency and complexity” away, finishing this course will provide us with the general idea of what problem solving procedure involves.

However, there is still a scope (advanced data structures) to cover. And, studying advanced data structures will be easier. And, the last point, for any software company interview you will have to know the fundamental algorithms involving data structures.

DATA STRUCTURE

BINARY TREE SEARCH

MEMBERS

- U1810032, Allanazarov Madiyorbek (001)
- U1810064, Akhtamov Amal (001)
- U1810123, Asrokhujayeva Barno (001)
- U1810108, Anvarkhujaev Saidkamol (001)
- U1810048, Makhamadaliyev Ibrohim (001)

THE MAIN PROBLEM

SCHEDULES





Most of the students are bounded from participating in valuable lectures by well-known speakers.



In addition:
Extra time for heating the food

CIE-16-02

| | | | | |
|----|-----------|---------|-----------|-----|
| Mo | | Top2 | Art In2 | |
| Tu | | Art In2 | | |
| We | Data Com2 | Top2 | C In2 | MA2 |
| Th | | MA2 | MC2 | |
| Fr | E2 | E2 | Data Com2 | |

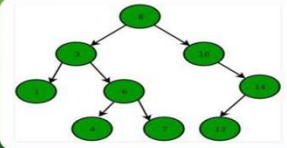
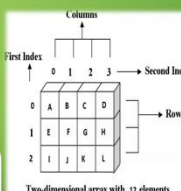
CIE-19-01

| | | | | |
|----|---------|--------|---------|-----|
| Mo | OOP1-1 | OOP1-1 | | |
| Tu | AER1 | | C In1 | BI1 |
| We | | AE1-1 | Calc1-1 | P1 |
| Th | RI1 | | P1 | |
| Fr | Calc1-1 | | AE1-1 | |

CIE-18-04

| | | | | |
|----|-------|-------|-------|------|
| Mo | LA2 | BK1-3 | AE3-3 | DLC1 |
| Tu | | DM2 | | DLC1 |
| We | LA2 | DM2 | C In2 | DS1 |
| Th | Java2 | Java2 | | |
| Fr | | AE3-3 | DS1 | |

USED DATA STRUCTURES:
BINARY SEARCH TREE
3D ARRAY

Two-dimensional array with 12 elements

```

struct Node {
    int data;
    int time;
    Node* left;
    Node* right;
};

// For creating the leaf node
Node* create(int data,int time)
{
    Node* temp = new Node();
    temp->left=temp->right = NULL;
    temp->data = data;
    temp->time = time;
    return temp;
}

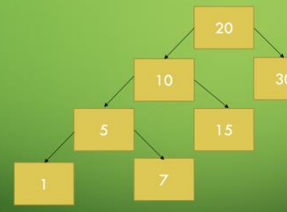
```

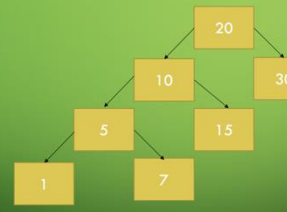
```

void insert(Node* &root, int data,int time)
{
    if(root== NULL)
        root = create(data,time);
    else if(root->data > data)
        insert(root->left, data,time);
    else
        insert(root->right, data,time);
}

```

20 10 5 1 7 15 30 32 40





```

int array[5][3][9] = {
    //Monday
    {
        //1st Group
        {0,1,1,1,0,0,0,0},
        //2nd Group
        {1,1,1,1,1,1,1,0},
        //3rd Group
        {0,1,1,0,1,1,0,0,0}
    },
    //Tuesday
    {
        //1st Group
        {1,1,0,0,0,1,1,1,0},
        //2nd Group
        {0,0,1,1,0,0,0,1,1},
        //3rd Group
        {0,1,1,0,0,0,0,0,0}
    },
    //Wednesday and so on...
};

cout<<"Choose the day: "<<endl;
int x;
cin>>x;
int temp[9] = {};
for(int i=0; i<3; i++)
{
    temp[i] += array[x][i][0];
    temp[i] += array[x][i][1];
    temp[i] += array[x][i][2];
    temp[i] += array[x][i][3];
    temp[i] += array[x][i][4];
    temp[i] += array[x][i][5];
    temp[i] += array[x][i][6];
    temp[i] += array[x][i][7];
    temp[i] += array[x][i][8];
}

```

```

Node* displayLeftMost(Node* root)
{
    Node *temp;
    while(root->left!=NULL)
    {
        root = root->left;
    }
    temp = root;
    return temp;
}

```

