

Instruction for Computing Program II

CS225 Group D2
















Ren, Junjie; Chen, Yang; Cai, Peiqi; Li, Xinzhao

This is the instruction for computing program II of group D2 which helps professor and TAs to test our program conveniently. We will list some assumptions of our medical system, with some details and how we complete the exercises. At last, we give an example of demonstration.

1. Main description

1.1 Files

Our program contains in total 15 necessary files (exclude the `.exe` file), which are:

 <code>appointment.h</code>	C Header 源文件
 <code>B_Tree.h</code>	C Header 源文件
 <code>B+_Node.h</code>	C Header 源文件
 <code>B+_tree.h</code>	C Header 源文件
 <code>FibHeap.h</code>	C Header 源文件
 <code>Register.h</code>	C Header 源文件
 <code>appointment.cpp</code>	C++ 源文件
 <code>B_Tree.cpp</code>	C++ 源文件
 <code>B+_Node.cpp</code>	C++ 源文件
 <code>B+_tree.cpp</code>	C++ 源文件
 <code>FibHeap.cpp</code>	C++ 源文件
 <code>Medicial_System_Simulator.cpp</code>	C++ 源文件
 <code>Register.cpp</code>	C++ 源文件
 <code>thanks.cpp</code>	C++ 源文件
 <code>registry.csv</code>	Microsoft Excel 逗号分隔值文件

1.2 Assumptions

We assume that the registry center receives **10 patients for twice** (that is, 20 patients a day in total), while every hospital can cure **5 people a day**. There are in total 3 hospitals, that is, we can cure **15 patients** in total.

Additionally, we randomly give the people in our input list risk status in the ratio of no: low: medium: high = 4:3:2:1. Basically, our input list (*register.csv*) provides 1500 random patients with their personal information. We partly used the random generated data from open source on GitHub, we also modified it a lot, such as risk, profession generation, preferred hospital generation.

Also, every patient has a unique id, that is the order of them in the input list, starting from 1.

We assume the deadline for every patient is 20 days when he or she is pushed into the centralized queue. After a patient has reached the deadline, he or she will be immediately pulled out of the centralized queue and be assignment an

appointment. In this case we may treat more than 15 people a day, thus the additional patients will be scheduled to other hospitals (for example hospitals in the downtown).

After a patient gets treated, he or she will be pushed into the B or B+ tree for storage.

1.3 Priority Descriptions

Priority: 0. DDL (different); 1. Risk; 2. Prof; 3.age; 4. Date/time; 5. Treatment Type

In case of medium risk, remove from queue, and add to the queue after 20 days.

2. Registration

We assume every day first 10 patients insert into local queue 1 (morning), last 10 patients insert into local queue 2 (afternoon).

Every patient has a relative class person. It stores the patient's:

Id, name, address, phone, WeChat, email, profession, birth, risk

Timestamp, the date and the time of the registration

1 to 8 profession categories (I II III IV V VI VII VII)

Seven age groups 12 18 35 50 65 75

Four risks 0 1 2 3

Location preferred (which hospital to choose)

Every time the centralized queue is ready to receive patients, registration queue checks: people who change risk, medium people who waits for 20 days, high risk people who wait for the centralized queue to be empty, re-registered people who have waited for 20 days, at last is the normal patients in the input list. (We assumed the priority.)

3. Queue

The centralized queue is built as a Fibonacci heap. And every node in the Fibonacci heap stores the pointer of the “person” class. In this case, we can save abundant space and the information of the patients are easy to renew or update. Some functions are added except those that usually exist in an ordinary Fibonacci heap. The queue receives the pointers of the patients by the function “record_in” and orders them by the function “comparePriority”. Then it can provide a patient with the function “record_out”. Furthermore, there are some other functions that cooperate with the main function, such as removing all of the patients reaching DDL, changing the profession and risk of the patient, withdrawal, and so on. All in all, the names of the functions explain them well, otherwise, there should be some detailed comments.

4. Appointment

For every day, we first check whether there are some people approaching the deadline. If so, we treat it right away, no matter what their priority is. Under the circumstance that over 15 people have deadline today, we move the exceeding ones to the city town since the daily capacity in the local hospitals is fixed. On the other hand, if less than 15 people approach deadline, we pop the person with highest priority in the Fibonacci heap until local hospitals have been filled or the Fibonacci heap gets empty.

Then according to the preference, we choose the appropriate hospital and time slots for every person treated locally.

5. B-tree and B+-tree

The basic design of our B-tree and B+ tree is completely based on the theories introduced in lecture slides. Some useful functions including insert、remove、split、merge have been realized in our program. Some small modifications concerning the additional overflow block will be explained in the part 8---explanations about

exercises.

6. Main program and Reporting

Weekly Report

At the end of each week, we need a weekly report that is printed on the screen (Basically, output to a file is of the same function here, so we just use “cout” to print them out). What to print out is shown below:

The treated people

Registered people with appointment

Queueing people without appointments

(Including their name + ID + profession category + age category + risk + time)

Monthly Report

At the end of each month, we need a monthly report that is also printed on the screen. In monthly report, we will not print the people one by one. On the other side, we will count the total number of each group and print them out. What to print out is shown below:

Number of registered people this month

Number of waiting people (in the central queue now)

Number of appointments in all three hospitals of people who have withdrawn

7. Demonstration Example

First, make sure the necessary files are under the same folder. Then you can choose to run the complied executable named *Medicial_System_Simulator.exe* or run *Medicial_System_Simulator.cpp* in VScode. Once succeed, the screen below will be displayed.

```
Computing Program II--CS 225--Group D2
All rights reserved

A new day has begun, Day 1
Please choose the operation you want:
1.update profession category
2.update risk status
3.withdraw a patient
4.re-register a patient that has withdrawn
Press anything but 1,2,3,4 to continue without any operation
```

This menu will appear every new day. You can choose what you want to operate. For example, if you want to update the profession of patient having ID 1. It will say:

```
1
Please enter the ID of the patient
1
This person is not in the centralized queue.
Please choose the operation you want:
1.update profession category
2.update risk status
3.withdraw a patient
4.re-register a patient that has withdrawn
```

Which is expected since no one has been ever in the local queue nor the centralized queue.

Then you can type any numbers other than 1,2,3,4 to continue to Day 2.

```
5
A new day has begun, Day 2
Please choose the operation you want:
1.update profession category
2.update risk status
3.withdraw a patient
4.re-register a patient that has withdrawn
Press anything but 1,2,3,4 to continue without any operation
```

For example, we keep doing so to pass Week 1. When passing Day 7, there will be a weekly report.

```

A new day has begun, Day 7
Please choose the operation you want:
1.update profession category
2.update risk status
3.withdraw a patient
4.re-register a patient that has withdrawn
Press anything but 1,2,3,4 to continue without any operation
5

*****WEEKLY REPORT*****
This is weekly report for week 1!
Which category do you want to sort?
Enter 1 for treated people, 2 for appointed people, 3 for queueing people
If you DO NOT want to sort, please choose other than 1,2,3.

```

Weekly report will display treated, appointed, queueing people of this week and you are free to choose which one you want to see that is sorted. For example, let's choose sort treated people.


```

1
Next, use what sort to output?
1. Sort by name
2. Sort by profession category
3. Sort by age group
Type anything but 1,2,3: Without any sort


```

Here you also can choose how to sort. Let's sort it by name.

Then there will produce a long list with two additional files you may notice.

 week 1 B-Tree report.csv

Microsoft Excel 逗号分隔值文件

 week 1 report.csv

Microsoft Excel 逗号分隔值文件

```

1
the person with id:87 has name: Allcott
the person with id:84 has name: Andrews
the person with id:88 has name: Atkinson
the person with id:5 has name: Benfield
the person with id:154 has name: Brennan
the person with id:133 has name: Brock
the person with id:169 has name: Bullock
the person with id:121 has name: Callan
the person with id:73 has name: Casey
the person with id:8 has name: Cassidy
the person with id:24 has name: Cavanagh

```

With another 3 unsorted treated, appointed, queueing people lists. They have also been outputted as a file (in .csv).

Casey	73	6	5	0	4
Robinson	123	6	5	0	3
Hill	131	6	5	0	3
Waiting people with appointments at the end of this week					
name	ID	profession	age	group	risk
Egerton	210	1	2	0	0
Eastwood	215	1	3	1	0
Penn	202	3	4	0	0
Chester	220	3	4	0	0
Shepherd	207	4	3	0	0
Emmott	211	4	3	0	0
Hopkins	213	4	4	1	0
Neal	203	5	6	1	0
Murray	198	6	1	0	0
Rosenbloo	209	6	2	1	0
Wilton	218	6	2	0	0
Edwards	222	6	3	0	0
Newton	153	6	5	0	2
Olson	201	6	5	0	0
Steer	205	6	5	1	0
Queueing poeple without appointments this week					
name	ID	profession	age	risk	waiting-time-until-today
Long	124	6	68	0	3
Blackburn	106	7	51	0	3

Also, it will also display the B-tree and output it as a file (in .csv)

Display the B-Tree as follows

B-Tree root node at left most:
profession ID

1 2
1 3
1 5
1 24
1 31

	A	B
1	profession	ID
2	1	2
3	1	3
4	1	5
5	1	24

6 110
6 117
6 123
6 131
6 171
6 197

B-Tree has 90 keys.

*****END DISPLAY*****

86	6	110
87	6	117
88	6	123
89	6	131
90	6	171
91	6	197

Then that's the end of the weekly report. Let's continue with Day 8.

*****END DISPLAY*****

A new day has begun, Day 8

Please choose the operation you want:

1.update profession category

2.update risk status

3.withdraw a patient

4.re-register a patient that has withdrawn

Press anything but 1,2,3,4 to continue without any operation

Now we want to change the profession of one specific patient. It can only be changed if this patient is in the centralized queue (Fibonacci Heap). Say we want to change the profession of the patient Long (ID 124) from 6 to 5.


```
Press anything but 1,2,3,4 to continue without any operation
1
Please enter the ID of the patient
124
Please enter the profession you want to change
5
Update successful
```

Then the priority of patient Long will also be changed. The location of him or her in the centralized heap also changes.

And if say we want to change the risk status of patient Blackburn (ID 106) from no risk (0) to medium risk (2).

```
Press anything but 1,2,3,4 to continue without any operation
2
Please enter the ID of the patient
106
Please enter the risk you want to change
2
```

Then this patient will be pulled out of the centralized queue and be put into a temporary local queue to wait for being processed. At Day 9, Blackburn will be pushed into local queue again with new risk 2. (Then Blackburn will be required one month extension, of course)

And, for example, we want to withdraw a patient, let's choose Mackenzie (ID 113).

```
Press anything but 1,2,3,4 to continue without any operation
3
Please enter the ID of the patient
113
Withdraw successful.
```

Then Mackenzie has been removed from the centralized queue and will not back unless we re-register him or her.

Then let's continue to the end of the first month (Day 30). (You will first get reports from Week 2, 3, 4)

```


A new day has begun, Day 30
Please choose the operation you want:
1.update profession category
2.update risk status
3.withdraw a patient
4.re-register a patient that has withdrawn
Press anything but 1,2,3,4 to continue without any operation
5

*****MONTHLY REPORT*****
This is the monthly report for month 1!
The number of registered people is: 600
The number of people with appointment this month is: 450
The average waiting time is: 3.13111
The whole number of person in the Centralized Queue 148
The whole number of person who had withdrawn 1

*****Monthly Report is done****

```

Monthly report is like as above. Also, you will receive output and file for the display of B-tree of the first month (*profession* as the secondary key) (Details omitted since it is similar to weekly B-tree report).

 month 1 B-Tree report.csv

Microsoft Excel 逗号分隔值文件

Then we are at Day 31. Say we want Mackenzie back, let's re-register him or her.

```

A new day has begun, Day 31
Please choose the operation you want:
1.update profession category
2.update risk status
3.withdraw a patient
4.re-register a patient that has withdrawn
Press anything but 1,2,3,4 to continue without any operation
4
Please enter the ID of the patient
113
Re-register successful. This patient is no or low or medium risk.

```

Luckily, Mackenzie is not high risk, so he or she only needs to wait for two weeks. We can find him or her in the centralized queue 14 days later (Day 45)

Let's skip 14 days then. We can find Mackenzie in the report of Week 7 (Day 49).

```

the person with id:113 has name: Mackenzie

```

277	Mackenzie	113	7	25	1	4
-----	-----------	-----	---	----	---	---

We can find that Mackenzie has waited for 4 days in the queue, i.e., he or she was pushed into the queue at Day 45 as expected.

That's the end of the main operations of our simulator. You are free to do the operations you want (such as let Mackenzie be withdrawal again).

The program terminates when there are no one in the local queue and centralized queue. The end of the day will differ if you do operations. In our case above, we skip to the end.

At one specific day, we have processed all the patients that register (some of them are still waiting to be pushed into centralized queue since medium or high risk), and you will see display like:

```
A new day has begun, Day 62
Please choose the operation you want:
1.update profession category
2.update risk status
3.withdraw a patient
4.re-register a patient that has withdrawn
Press anything but 1,2,3,4 to continue without any operation
5
*****
All people registered processed, now only consider people with medium and high risk and specials(change risk,profession or withdrawn)
*****
```

But you are still able to withdraw patients, feel free to test our program.

There are also special cases like:

```
the 3 local hospitals today have been fully occupied
Move the person (id 1223) with deadline today to another hospital in the city town
```

This is due to assumptions of our deadline for every patient is 30 days. When over 15 patients reach deadline in one day, we need to cure them in other hospital.

When all the queues are cleared and all the patients are cured, you will see display like:

```
A new day has begun, Day 106
Please choose the operation you want:
1.update profession category
2.update risk status
3.withdraw a patient
4.re-register a patient that has withdrawn
Press anything but 1,2,3,4 to continue without any operation
5
*****
All patients are treated! Congratulations !
*****
*****End Day 106*****
*****
Do you want some info about the treating details about some people?
Enter 1 for yes, 0 for no
```

Our system allow administers to check the detailed of the cured patients. Say we want to know Mackenzie (again ID 113).

```
1
Which people do you want to know?
Enter his id: 113
The person with id:113 was treated on day:66, in hospital:2, in the time slots:17:00-19:00
```

It seems that poor Mackenzie waited a bit long in the centralized queue.

You are free to check the information as you like. Thanks, and have fun!



8. Explanations about exercises

Exercise 1.

(i)

We have divided the class “person” we established in CA1 into four different classes to form the relational database schema.

```
//The next four class are designed for the database, i.e., B+ and B tree.
class person_specific
{
public:
    person_specific();
    int id;
    string name;
    int profession;
    string birth;
    int age;
    int age_group;
    string address;
    int phone;
    string wechat;
    string email;
    int preferred_hos1;
    int preferred_hos2;
    int preferred_hos3;
};
```

```
class medical_status
{
    public:
        medical_status();
        int risk;
        int treatment_type;
        int ddl_day;
};
```

```
class registration
{
    public:
        registration();
        int register_order;
        int register_day;
        bool if_queueing;
        bool if_withdrawed;
        bool if_re_registered;
        int wait_before_in_queue;
        int wait_re_register;
};
```

```
class treatment
{
    public:
        treatment();
        int apponitment_loc;
        int treated_order;
        int treated_date;
        string treated_time;
        bool if_treated;
        bool if_appointed;
        bool if_treated_locally;
};
```

In addition, we use the class “person_union” to integrate four separate classes above, which stores the information in the node of B-tree and B+ tree.

```
//person_union collects the pointers to four separate database
class person_union
{
public:
    person_union();
    person_specific *info;
    medical_status *status;
    registration *reg;
    treatment *treat;
};
```

(ii)

For a general Fibonacci heap, the sorting is done by comparing the size of each node key value. But since our program needs to compare multiple priority attributes, our group came up with two methods to reach the condition. The first method is to place the higher priority attributes on higher bits, and this method we applied in the B-tree. But in the Fibonacci heap, we wrote a separate function specifically to compare the priorities of individual patients. For the two arguments node0 and node1, the function returns 0 if node0 has higher priority, otherwise it returns 1. For all attributes, we arrange the rule that the priority is to compare the occupation, followed by the age, followed by the data at the time of registration, followed by the date of registration, and finally by the type of treatment. And for each attribute, we set the rule that the smaller the number corresponding to the type, the higher the priority.

```
// if node0 has higher priority return 0, else return 1
// assume that for every criterion, the smaller number
// means the higher priority
template <class T> int FibHeap<T>::comparePriority(FibNode<T> *node0, FibNode<T> *node1) {
    if (node0->loc->profession < node1->loc->profession) return 0;
    else if (node0->loc->profession > node1->loc->profession) return 1;
    else {
        if (node0->loc->age_group < node1->loc->age_group) return 0;
        else if (node0->loc->age_group > node1->loc->age_group) return 1;
        else {
            // id is the data of registration
            if (node0->loc->id < node1->loc->id) return 0;
            else if (node0->loc->id > node1->loc->id) return 1;
            else {
                if (node0->loc->register_day < node1->loc->register_day) return 0;
                else if (node0->loc->register_day > node1->loc->register_day) return 1;
                else {
                    if (node0->loc->treatment_type < node1->loc->treatment_type) return 0;
                    else if (node0->loc->treatment_type > node1->loc->treatment_type) return 1;
                    else return 0;
                }
            }
        }
    }
}
```

(iii)

We have used a linear array `m_Key_Values` to store the information, which satisfies the requirement of “an ordered sequence of blocks”.

```
NODE_TYPE m_Type;
int m_KeyNum;
KeyType m_KeyValues[MAXNUM_KEY];
```

(iv)

To enhance the performance of different operations used in B+ tree, we introduced the overflow block to temporarily store the information. Our design is that the size of the overflow block is half of the tree’s order. In our situation, we set the order of the tree to be 5, so the size of overflow block is 2, which is appropriate in our case.

```
// Additional Overflow Block
KeyType overflow_key[ORDER/2];
DataType overflow_data[ORDER/2];
int overflow_temp_size;
```

(v)

Six operations required can be found in our B+ tree data structure.

```
void INSERT_IN_MAIN(KeyType key, const DataType data);
void insert(KeyType key, const DataType data);
```

```
virtual void removeKey(int keyIndex, int childIndex);
```

```
virtual void mergeChild(CNode<KeyType, DataType> *parentNode, CNode<KeyType, DataType> *childNo
```

```
virtual void split(CNode<KeyType, DataType> *parentNode, int childIndex);
```

```
DataType getData(int i);
```

```
DataType *getDataAddr(int i);
```

No specific function written for “sort”, as it is an internal part for “merge” “split” and so on. In other words, “sort” has been realized in other functions.

Exercise 2

(i)

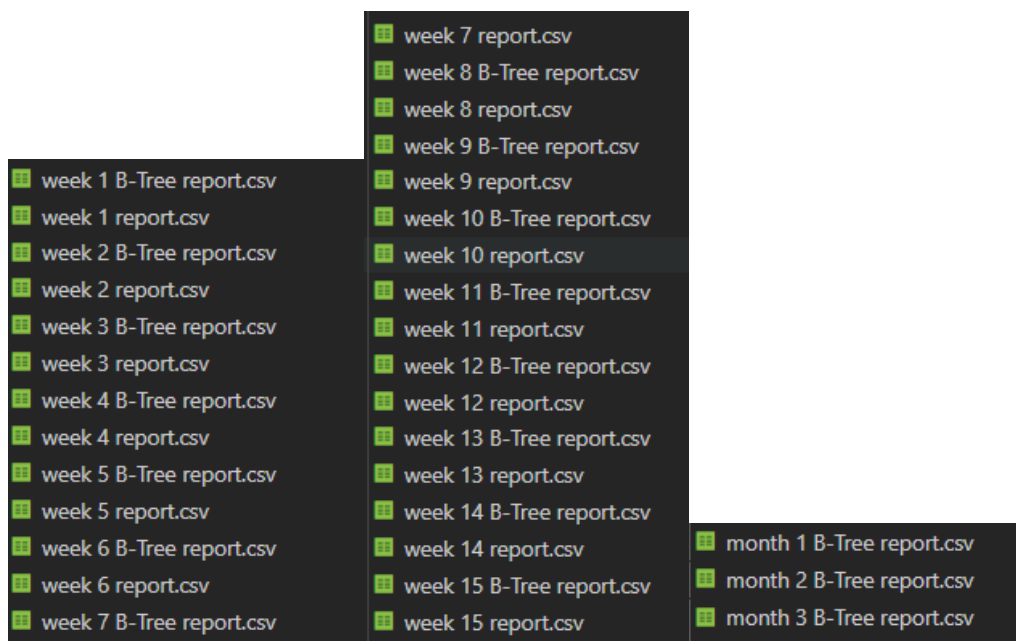
According to the requirements from TA, in particular, we have modified the reporting system using the database schema. That is to say, after people have been treated, their data structures have been converted from “person” to “person_union”, which is the form they are in B-tree and B+ tree.

```
person_union *united = convert_form(person_now);
```

```
person_union* convert_form(person *person_now)
```

(ii)

Our file reports comprise three parts. Weekly reports from CA1, but in the form of file. Weekly reports in the form of B-tree, showing information about people treated this week. Monthly report in the form of B-tree, showing information about people treated this month.



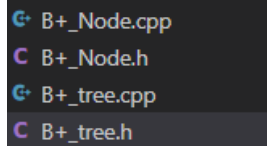
Exercise 3

(i)

B+ tree use the primary key: id. Relevant data structures have been organized in our files.

(ii)

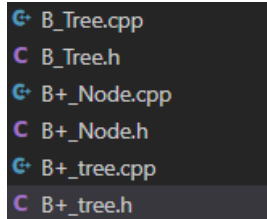
Index structures using B tree and B+ tree have been realized in following files.



```
B+_Node.cpp
B+_Node.h
B+_tree.cpp
B+_tree.h
```

(iii)

Insert and other functions have been achieved in our program. For detailed implementation, refer to following files.



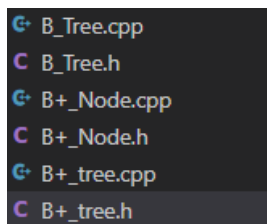
```
B_Tree.cpp
B_Tree.h
B+_Node.cpp
B+_Node.h
B+_tree.cpp
B+_tree.h
```

(iv)

Our secondary key for B tree is a combination of profession and id. To be more specific, secondary key == profession * 10000 + id. Since the patients in our test cases are from id==1 to id==1500, we occupy the last four digits for id information. And use profession * 10000 to identify the profession. In this case, everyone gets a unique secondary key and it can be easily identified.

(v)

They have been achieved in our program. For detailed implementation, refer to following files.



```
B_Tree.cpp
B_Tree.h
B+_Node.cpp
B+_Node.h
B+_tree.cpp
B+_tree.h
```