浙江大学

Fundamental Data Structure
Project 3 Report

Cars on Campus

Contents

Part1: Introduction to the project

The topic of project 2 is Cars on campus. In this project, we are going to find the number of the cars parking on campus at every given time point as well as find the car(s) which stays on campus in this day for the longest time.

Therefore, in this project, we will get the following information:

1. The number of the records been recorded.
2. The number of the time points need to query.
3. All the records been recorded in this day from 00:00:00 to 23:59:59. Every record includes the plate number of the car, the time this record been recorded as well as the status of the car at this time point.
4. All the time points need to query.

Our task is to give out the number of cars parking on campus at the time point given in every query. What's more, we need to output the car which stays on campus for the longest time. If there are more than one qualified cars, output their plate numbers in alphabetic order. After that, output the time these cars stay for on campus.

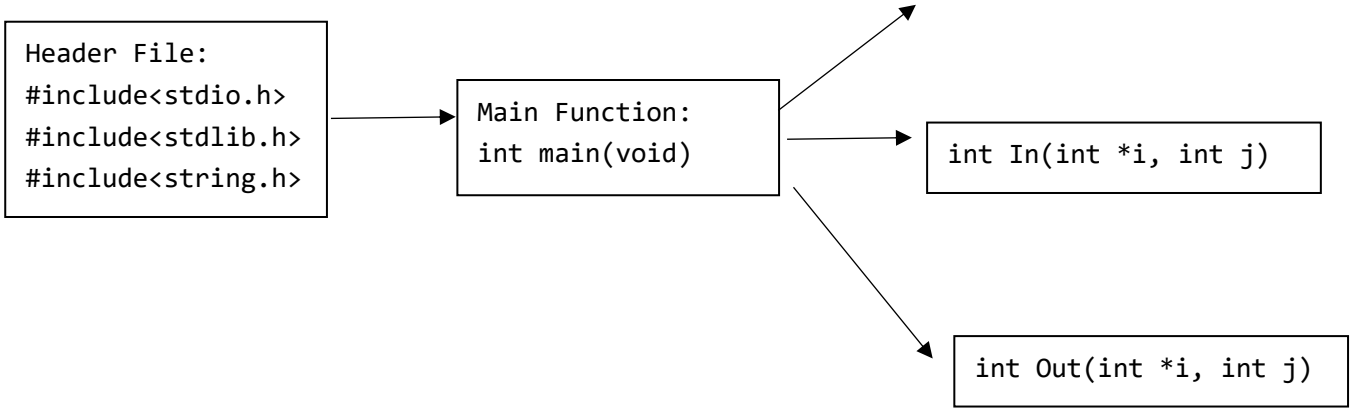
Part2: Algorithm and Program Instructions

At the beginning of our program, we read N records from the input data and store them in the struct array. And then In we use quick sort to sort the records in alphabetic order firstly, and if the alphabetic order of two records are the same, it means they are the same car. We will then sort the records in time-ascending order. The next step is using bucket sort to calculate how many cars are in campus at every moment. We also calculate the max stay time of cars at the same time. What's more, we design 2 functions In() and Out() to eliminate the influence of those extra "in" and "out", which can save a lot of time.

Here is the procedure code of our Algorithm:

```
ALGORITHM: Cars on campus
Procedure Cars on campus
INPUT: The data stated in the questions
OUTPUT: The result of each check: the number of cars in the school at
that time with the car ID which stayed for the longest time in the
campus and the max time.

Read N and K;(the number of records and checks)
for(i=0;i<N;i++)
    Read the N records

Sort the records in alphabetic order and time order ascendingly

While i<N
    Calculate the number of cars at the time of record[i]
    Calculate the max stay time of car

While i<K
    Read the check time
    OUTPUT the result according to the array EverySecond[]
OUTPUT(CAR IDs who stayed longest, the max time)
```

The framework of the code is



Part3: Testing Results

We wrote a small program (P3 test_code.c) to get some test data, you can get a set of data by input the N and K. We made up some special conditions to test our program. These are the results.

```
int   compare(const
void *a, const void
*b)
```

TEST1

Data input	16 7 JH007BD 18:00:01 in ZD00001 11:30:08 out DB8888A 13:00:00 out ZA3Q625 23:59:50 out ZA133CH 10:23:00 in ZD00001 04:09:59 in JH007BD 05:09:59 in ZA3Q625 11:42:01 out JH007BD 05:10:33 in ZA3Q625 06:30:50 in JH007BD 12:23:42 out ZA3Q625 23:55:00 in JH007BD 12:24:23 out

	ZA133CH 17:11:22 out JH007BD 18:07:01 out DB8888A 06:30:50 in 05:10:00 06:30:50 11:00:00 12:23:42 14:00:00 18:00:00 23:59:00
Explain	The data is same to the sample
Actual output	05:10:00 06:30:50 11:00:00 12:23:42 14:00:00 18:00:00 23:59:00 1 4 5 2 1 0 1 JH007BD ZD00001 07:20:09
Testing result	Pass

TEST2

Data input	3 1 ZZZZZZZ 01:01:01 in ZZZZZZZ 01:01:02 in ZZZZZZZ 10:10:10 out 10:10:09
Explain	The “in” records is not in pairs
Actual output	1 ZZZZZZZ 09:09:08
Testing result	Pass

TEST3

Data input	3 1 ZZZZZZZ 03:03:03 in ZZZZZZZ 03:04:08 out ZZZZZZZ 03:05:08 out 10:00:00
Explain	The “out” records is not in pairs
Actual output	0 ZZZZZZZ 00:01:05
Testing result	Pass

TEST4

Data input	4 1 ZZZZZZZ 08:00:00 in ZZZZZZZ 09:00:00 out AAAAAAA 10:00:00 in AAAAAAA 11:00:00 out 10:00:01
Explain	More than 2 cars have the same max staying time
Actual output	1 AAAAAAA ZZZZZZZ 01:00:00
Testing result	Pass

TEST5

Data input	20 10 L0MFFIO 10:55:48 in L0MFFIO 05:19:17 out P5C7LLW 11:37:14 in P5C7LLW 00:44:49 out 5OI2OIJ 18:21:27 in KBGDHKL 09:31:04 in AWS21ZR 09:37:26 in P5C7LLW 15:16:54 in KBGDHKL 18:13:35 in AWS21ZR 10:31:22 out 9K7INK6 18:47:38 out TCGMK8P 14:12:24 out 5OI2OIJ 12:08:57 in P5C7LLW 06:48:54 out

	5OI2OIJ 05:26:48 out AWS21ZR 12:10:33 out TCGMK8P 09:01:15 out KBGDHKL 23:04:11 out KBGDHKL 03:29:15 out TCGMK8P 02:37:53 out 04:20:38 06:42:25 12:11:40 12:36:01 14:17:31 18:31:42 19:54:56 21:42:13 22:12:23 22:48:03
Explain	A large amount of input data (created in test program)
Actual output	0 0 0 0 0 1 1 1 1 1 KBGDHKL 04:50:36
Testing result	Pass

Part4: Analysis of Algorithm and Codes

I. Time complexity:

The time complexity depends on the main steps of our code, they are: read, sort, calculate the max time and check the time.

(i). read: In this step we will read N records from the input data and store them in the struct array, so it is O(N)

(ii). sort: In this step we use quick sort to sort the records in alphabetic order and time-ascending order, we call the function qsort() in the header file <stdlib.h> and define a new function compare to be used as the compare rule,so the Time Complexity is O(NlogN)

(iii). Calculate the max time: In this step we will calculate the number of cars in campus at each minute of a day and compare the staying time of all the cars. We use 2 functions In() and Out() to eliminate the influence of those extra “in” and “out”, the worst condition of the 2 functions is O(N)

(iv). check: in evert check we will read a time (cost O(1)) and print the result of the number of cars at that time. However, we use a array Everysecond to store the number of cars, to get the result from this array costs O(1), so each check costs O(1) and K times of check costs O(K)

Totally, the time complexity is O(N logN + K)

II. Space complexity

The extra space we use are struct array Record[], nomal array EverySecond[ ] and check[ ],plate_number[][8] and some other single variables. Besides, EverySecond[ ] has 86400 elements in it and others have O(N) elments. Though 86400 is a big number but it is still a constant. So the space complexity is O(N). In our algorithm, you even do not need input the check time ascendingly.

Part5: Source Code

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>

#define MaxRecord 10000 /*The max number of the records be
en recorded.*/
#define MaxQuery 80000  /*The max number of the queries be
ing recorded.*/
#define MaxTime 86400   /*The number of seconds per day.*/

/*An array to save every record.*/
struct Node {
    char plate_number[8];/*The plate number of the car
in this record.*/
    int t;               /*The time when the record is
recorded.*/
    int status;          /*The status of the car when
the record is recorded.*/
}Record[MaxRecord];

/*An array to save current number of cars on campus of
every second in a day.*/
int EverySecond[MaxTime];
/*An array to save the points of time needed to calculate
the number of cars on campus.*/
int check[MaxQuery];

/*The function used to compare the two data structures
when we call the function "qsort" included in stdlib.h.*/
int compare(const void *a, const void *b);
/*The function to avoid the successive "in" status.*/
int In(int *i, int j);
/*The function to avoid the successive "out" status.*/
int Out(int *i, int j);

int main()
```

```
{
    int N, K, i, j, maxtime = -1, top = 0;
    char status[4], plate_number[MaxRecord][8];
    scanf("%d%d", &N, &K);
    for(i = 0; i < N; ++i) {
        int hour, minute, second;
        /*Input the information of each record.*/
        scanf("%s %d:%d:%d %s", Record[i].plate_number, &h
our, &minute, &second, status);
/*To save the times of comparation, transfer the clock
into seconds.*/
        Record[i].t = hour * 3600 + minute * 60 + second;
/*If the status is "in", the status of the record equals
to 1, if not, the status equals to 0.*/
        if(status[0] == 'i') Record[i].status = 1;
        else Record[i].status = 0;
    }
/*Sort the records in alphabetic order and time order,
respectively.*/
    qsort(Record, N, sizeof(struct Node), compare);
    i = 0;/*Initial the template variable.*/
    while(i < N) {
        int m = i, currenttime = 0;
        char temp_number[8];
/*To find the index of the last record of this car.*/
        while((m < N) && (strcmp(Record[i].plate_number, R
ecord[m].plate_number) == 0)) m++;
/*To save the plate number of the current car temporarily.
*/
        strcpy(temp_number, Record[i].plate_number);
/*To calculate the total time the car has been parked on
campus.*/
        while(1) {
            int *tmp = &i;
            if(In(tmp, m) == 0) break;
            int j = i;
            if(Out(tmp, m) == 0) break;
            EverySecond[Record[j].t]++;
            EverySecond[Record[i].t]--;
            currenttime += Record[i].t - Record[j].t;
```

```
        }
/*To compare current time with the max time, if the curren
t time is smaller than the max time, jump to the next
record, if the result is equation, push the current plate
number to the stack named "plate_number", if the result
is bigger, clear all the records in the stack and push the
 current plate number to the bottom of the stack.*/
        if(currenttime > maxtime) {
            maxtime = currenttime;
            top = 0;
            strcpy(plate_number[top++], temp_number);
        }
        else if(currenttime == maxtime) {
            strcpy(plate_number[top++], temp_number);
        }
    }
    /*Now, the elements in the array named "EverySecond" s
ave the change of the number of the cars on campus,
      so to flatter our aim, we need to add the number of
cars of the last second to the next second.*/
    for(i = 1; i < MaxTime; ++i) EverySecond[i] += EverySe
cond[i - 1];
    for(i = 0; i < K; i++) {
        int hour, minute, second;
        scanf("%d:%d:%d", &hour, &minute, &second);
        check[i] = hour * 3600 + minute * 60 + second;
    }
    for(i = 0; i < K; i++) printf("%d\n", EverySecond[chec
k[i]]);
    for(i = 0; i < top; i++) printf("%s ",plate_number[i])
;
    /*Tranfer the max time from the form of seconds into t
he form of clock.*/
    printf("%02d:%02d:%02d\n", maxtime / 3600, maxtime % 3
600 / 60, maxtime % 60);
    system("pause");
    return 0;
}
```

```c
/*If the plate number of record "a" is larger than the num
ber of record "b"
  in alphabetic order, return 1, if smaller, return -
1 and when it comes to
  an equation, compare the time order of the two records.*
/
int compare(const void *a, const void *b)
{
    struct Node *c = (struct Node *)a;
    struct Node *d = (struct Node *)b;
    int p = strcmp(c->plate_number, d->plate_number);
    if(p < 0) return -1;
    if(p > 0) return 1;
    return c->t - d->t;
}

/*To ignore the successive "in" statuses and stop at the l
ast one. If
  we cannot find such a record, return back 0, otherwise,
return 1.*/
int In(int *i, int j)
{
    while((*i < j) && (!Record[*i].status)) (*i)++;
    if(*i >= j) return 0;
    while((*i < j) && (Record[*i].status)) (*i)++;
    (*i)--;
    return 1;
}

/*To find the next fisrt "out" status. If we don't get to
the end of
  the records of the current car, return 1, otherwise, ret
urn 0.*/
int Out(int *i, int j)
{
    while((*i < j) && (Record[*i].status)) (*i)++;
    return *i < j;
}
```

Part6: Summary

Declaration

We hereby declare that all the work done in this project titled "***Public Bike Management*** " is of our independent effort as a group.