浙江大學

Cars on Campus

Fundamentals of Data Structures

Laboratory Projects 3

December 30, 2016

Contents

Chapter 1

Introduction

For the problem of cars on campus with all the information available, we are asked to tell that at any specific time point, the number of cars parking on campus, and at the end of the day find the cars that have parked for the longest time period.

First we should decide which car are available. Because when car leaves campus, it must first enter the campus, but there are more than one entry for the same car. In order to solve the problem we make the same plate number car to be adjacent in an array by dictionary sorting algorithm. Then we compare the time to sort the same plate number car, finally confirm each entry or leave is valid. For example, if two consecutive inputs are the same car into the campus, on other hand between the same car two enter the campus don't exist leave the campus input. We must ignore the first input. And if one car want to leave campus but before it did not enter, we also should ignore it. At the same time, we can record the time of each car stays campus.

After deciding which input is valid and record time, we start to study how to record the number of cars in the campus at a certain moment. We create an array, and its index represent different moments of the day. Then we create another array to record the change number of car. Through add the change in the moment to the previous moment, we can make the first array record the number of car at each moment.

Finally, we get an algorithm to solve this problem. Accroding to the analysis in this report, we know the time complexity is $O(NlogN + K + T)$, and the space complexity is $O(N + T)$. What's more, if the total time period of this problem is too large, we can also improve our program. We will get a faster algorithm, which has $O(NlogN + K)$ time complexity, and $O(N)$ space complexity.

Chapter 2

Algorithm Specification

There are three steps to deal with this problem.

1. Sort the input data by the plate number and by time point.
2. Calculate the time length which cars have parked for.
3. Calculate the total number of cars parking on campus at any time point.

After finishing these three steps, the plate number of the car that has parked for the longest time period and the total number of cars parking on campus at a time point could be easily calculated by the information we get.

2.1 Hash and sort

2.1.1 Specification

Because the plate number of cars is not an integer, and it is not easy for us to sort input records, we convert the plate number to integer by a hash function.

Set the plate number input as P, $P_i \in \{0, 1, 2, ..., 9, \text{A, B, ..., Z}\}$.

And then we define a hash function $H(P)$ and another function $f(c)$.

Figure 2.1: $c - f(c)$

c	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	G	H	I
$f(c)$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18

c	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
$f(c)$	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35

$$H(P) = 36^{n-1}P_0 + 36^{n-2}P_1 + ... + 36P_{n-2} + P_{n-1} = \sum_{i=0}^{n-1} 36^{n-1-i}P_i$$

By the hash function $H(P)$, we can map plate number (string) to an integer.

2.1.2 Pseudo code

Figure 2.2: convert functions

Algorithm 1 Hash

```
1: function CONVERT-PLATE-NUM-TO-TNT($P$)
2:     $r := 0$
3:     for $i = 1$ to $|P|$ do
4:         $r := r \times 36 + f(P_i)$
5:     end for
6:     return $r$
7: end function
```

Algorithm 2 Hash

```
1: function CONVERT-INT-TO-PLATE-NUM($p$)
2:     for $i = \max\{n|36^n \leq p\}$ to 0 do
3:         output($M^{-1}(\frac{p}{36^i} \text{ Mod } 36)$)
4:     end for
5: end function
```

2.2 Calculate the time length

2.2.1 Specification

After sort the records and wipe the illegal records (not well paired records), we get cars C_i $(0 \leq i < n)$ and records $RIN_{ij}, ROUT_{ij}$.

The time length which car C_i have parked for

$$T_i = \sum_{j=0}^{m-1} ROUT_{ij} - RIN_{ij} \quad (0 \leq i < n)$$

2.3 Calculate the total number of cars

2.3.1 Specification

First, we have a naive algorithm. To know the number of cars in the campus at a time t, we can check all the records and calculate the number. But the time complexity of each query is $O(N)$ (the number of records), total complexity is $O(NK)$.

Actually, we have a algorithm with time complexity $O(1)$ of each query. The records are static (there will be no more record after query begins), so we could count the number of cars at all time point before query.

We record the number of cars changed at time point t $A_t(0 \leq t < 86400$, 86400 is the seconds in a day). Obviously, we could calculate A_t by going through the all records.

$$A_t = \sum_{RIN_{ij}.time=t} 1 - \sum_{ROUT_{ij}.time=t} 1$$

After we get the number of cars changed at any time point, the number of cars at time point t B_t will be sum of the number of cars changed from time 0 to t.

$$B_t = \sum_{i=0}^{t} A_i$$

Finally, we can answer each query in $O(1)$ time.

2.3.2 Pseudo code

Algorithm 3 Calculate the total number of cars at a time point

```
1: for $i = 0$ to 86399 do
2:     set $A_i, B_i = 0$
3: end for
4: for each record $r$ do
5:     if $r$.status is in then
6:         $A_{r.time} := A_{r.time} + 1$
7:     else if $r$.status is out then
8:         $A_{r.time} := A_{r.time} - 1$
9:     end if
10: end for
11: $B_0 := A_0$
12: for $i = 1$ to 86399 do
13:     $B_i := B_{i-1} + A_i$
14: end for
```

Chapter 3

Testing Results

3.1 Correctly test

3.1.1 Test data 1

Purpose

Example input and output.

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
```

Output

```
1
4
5
2
1
0
1
JH007BD ZD00001 07:20:09
```

Time	04:09:59	05:10:33	06:30:50	06:30:50	10:23:00	11:30:08	11:42:01
Car	ZD00001	JH007BD	DB8888A	ZA3Q625	ZA133CH	ZD00001	ZA3Q625
Status	in	in	in	in	in	out	out

Time	12:23:42	13:00:00	17:11:22	18:00:01	18:07:01	23:55:00	23:59:50
Car	JH007BD	DB8888A	ZA133CH	JH007BD	JH007BD	ZA3Q625	ZA3Q625
Status	out	out	out	in	out	in	out

3.1.2 Test data 2

Purpose

Test if two car has parked for the same largest time period.

Input

```
6 3
JH007BD 12:00:00 in
JH007BD 05:09:59 in
ZA3Q625 12:00:00 in
JH007BD 18:30:00 out
ZA3Q625 18:30:00 out
DB8888A 06:30:50 in
05:10:00
06:30:50
18:00:00
```

Output

```
0
0
2
JH007BD ZA3Q625 06:30:00
```

Time	12:00:00	12:00:00	18:30:00	18:30:00
Car	JH007BD	ZA3Q625	JH007BD	ZA3Q625
Status	in	in	out	out

3.1.3 Test data 3

Purpose

Test if there is a case in the input that the car drive in the campus twice in the same time and out the campus twice in the same time.

Input

```
8 4
JH007BD 12:00:00 in
JH007BD 12:00:00 in
JH007BD 12:24:23 out
JH007BD 12:24:23 out
ZA133CH 17:11:22 out
ZA133CH 17:11:22 out
ZA133CH 10:23:00 in
DB8888A 06:30:50 in
12:22:00
12:25:00
17:10:00
17:13:00
```

Output

```
2
1
1
0
ZA133CH 06:48:22
```

Time	10:23:00	12:00:00	12:24:23	17:11:22
Car	ZA133CH	JH007BD	JH007BD	ZA133CH
Status	in	in	out	out

3.1.4 Test data 4

Purpose

Test the case that if there are cars in or out the campus when we have a certain timing. And compared before and after the moment.

Input

```
6 6
JH007BD 18:00:00 in
ZD00001 11:30:08 out
ZD00001 04:09:59 in
JH007BD 06:00:00 in
JH007BD 13:00:00 out
```

```
JH007BD 18:07:01 out
12:59:59
13:00:00
13:00:01
17:59:59
18:00:00
18:00:01
```

Output

```
1
0
0
0
1
1
ZD00001 07:20:09
```

Time	04:09:59	06:00:00	11:30:08	13:00:00	18:00:00	18:07:01
Car	ZD00001	JH007BD	ZD00001	JH007BD	JH007BD	JH007BD
Status	in	in	out	out	in	out

3.1.5 Test data 5

Purpose

Test the hash function of the program is well defined. For example, if the hash function of program is define as $H(P) = \sum_{i=0}^{n-1} f(P_i)$. The `AA12345` and `AA54321` have the same hash value, and it may get the wrong answer.

Input

```
4 6
AA12345 10:00:00 in
AA54321 11:00:00 in
AA54321 12:00:00 out
AA12345 13:00:00 out
10:00:00
10:30:00
11:00:00
11:30:00
12:30:00
13:00:00
```

Output

```
1
1
2
```

```
2
1
0
AA12345 03:00:00
```

Time	10:00:00	11:00:00	12:00:00	13:00:00
Car	AA12345	AA54321	AA54321	AA12345
Status	in	in	out	out

3.1.6 Test data 6

Purpose

Test if the program correctly sort the records by time.

Input

```
4 6
JH007BD 13:00:00 in
JH007BD 15:00:00 out
JH007BD 14:00:00 in
JH007BD 16:00:00 out
13:00:00
13:30:00
14:00:00
15:00:00
15:30:00
16:00:00
```

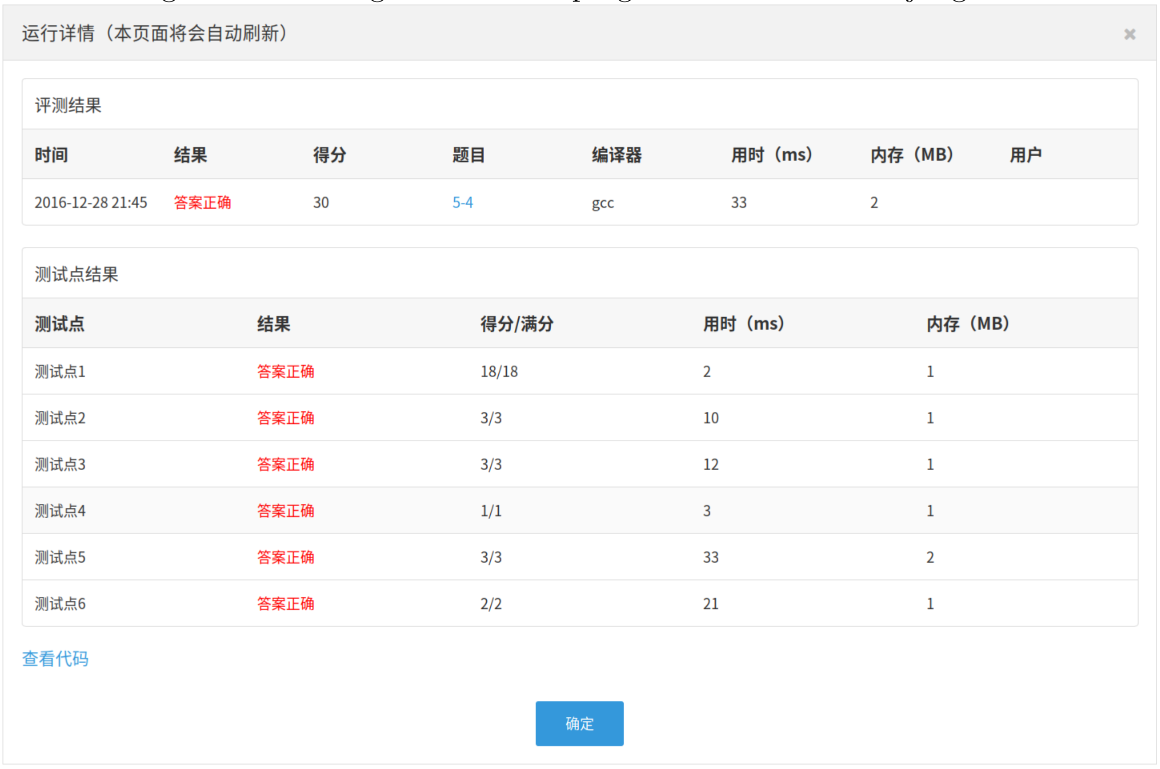
Output

```
0
0
1
0
0
0
JH007BD 01:00:00
```

Time	14:00:00	15:00:00
Car	JH007BD	JH007BD
Status	in	out

3.2 Testing result on online judge

Figure 3.1: testing result of the program on PTA online judge



运行详情（本页面将会自动刷新）

评测结果

时间	结果	得分	题目	编译器	用时（ms）	内存（MB）	用户
2016-12-28 21:45	答案正确	30	5-4	gcc	33	2	

测试点结果

测试点	结果	得分/满分	用时（ms）	内存（MB）
测试点1	答案正确	18/18	2	1
测试点2	答案正确	3/3	10	1
测试点3	答案正确	3/3	12	1
测试点4	答案正确	1/1	3	1
测试点5	答案正确	3/3	33	2
测试点6	答案正确	2/2	21	1

Chapter 4

Analysis and Comments

4.1 Time complexity

4.1.1 Hash

According to figure 2.2, the time which for-loop run for is determined by the length of plate number P. We consider all operations of integer is $O(1)$, so the time complexity of getting input data and hashing is $O(|P|)$.

4.1.2 Sort

In our program, we use `qsort` in `stdlib.h` in ANSI C. As we all know, the average time complexity of quick sort is $O(NlogN)$, and the worst case is $O(N^2)$. In this section, we regard the time complexity as $O(NlogN)$.

4.1.3 Calculate the time length

To calculate the time length carshave parked for, we have to go through every record. The number of records is N, so the time complexity is $O(N)$.

4.1.4 Calculate the total number of cars

First, we go through the records to calculate the number of cars changed at any time point, the time complexity of it is $O(N)$. And then we use the number of cars changed to calculate the number of cars at a time point i, for $i = 0to$ the seconds of a day. Moreover, there are K queries we should answer. The total complexity is $O(N + K + T)$. (T means the seconds of a day)

4.1.5 Conclusion

Finally, the total time complexity is the sum of all above time complexity, which is $O(|P| + NlogN + K + T)$. In this section, since $|P|$ is much smaller than T, N and K, we take the time complexity as $O(NlogN + K + T)$.

4.2 Space complexity

4.2.1 Hash

We only use a for-loop to calculate the result, and the number of variables is fixed, so the space complexity is $O(1)$. We also use a array to store input strings, and the size of it is $|P|$. The total space complexity is $O(|P|)$.

4.2.2 Sort

Because there is no buffer we have to use in quick sort, the space complexity is $O(1)$.

4.2.3 Calculate the time length

We will not use extra variables or space, so the space complexity is $O(1)$.

4.2.4 Calculate the total number of cars

We have to use two arrays in this section, the size of array is determined by the seconds of the all records, and we define it as T. The space complexity of this is $O(T)$.

4.2.5 Conclusion

The space we use in this program is the sum of all above space complexity and the space we store all records, which is $O(N + |P| + T)$. In this section, since $|P|$ is much smaller than N, T, the time complexity is $O(N + T)$.

4.3 Comments

There is a possible improvement on the algorithm.

Although the T in this problem, which is the seconds in a day (86400), is not too large, T may be larger than this in another problem. If T is very large, the space is not enough for us to simply use an array to store data. To deal with the problem, we can discretize the data, because the space we really use is the time points of all records. The space complexity and time complexity can be reduce from $O(T)$ to $O(N)$.

Appendices

Appendix A

Source Code (in ANSI C)

Listing A.1: main.c

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

#define N_MAX 10005
#define TIME_SEC_MAX 86400

#include "define_structures.c"
#include "hash.c"
#include "count_park_time.c"
#include "io.c"

int main(void)
{
    struct record records[N_MAX];
    int change_car_num[TIME_SEC_MAX], cur_car_num[TIME_SEC_MAX];
    int park_time[N_MAX];
    long long int car_id[N_MAX];
    int car_num;

    int n, k;
    int i;
    scanf("%d%d", &n, &k);

    getData(records, n);
    qsort(records, n, sizeof(struct record), compareRecord);
    car_num = countParkTime(n, records, car_id,
                            park_time, change_car_num, cur_car_num);
    outputNumberOfCars(k, cur_car_num);
    outputLongestTime(car_num, park_time, car_id);
    return 0;
}
```

Listing A.2: io.c

```c
/* the function to get input data */
void getData(struct record *records, int num)
{
    int i;
    char buf1[100], buf2[100], buf3[100];

    for (i = 0; i < num; i ++) {
        scanf("%s%s%s", buf1, buf2, buf3);

        /* convert plate number (string) to integer */
        records[i].plate_number = convertPlateNumToInt(buf1);

        /* convert time format from "HH:MM:SS" to second */
        records[i].moment.data = convertToSecond(buf2);

        /* convert "in", "out" to (int) 1, 0 */
        if (strcmp(buf3, "in") == 0) {
            records[i].status = status_in;
            /* status_in =  1 */
        } else if (strcmp(buf3, "out") == 0) {
            records[i].status = status_out;
            /* status_out = 0 */
        }
    }
}

/* the function to output the total number of cars parking on campus at a time */
void outputNumberOfCars(int k, int *cur_car_num)
{
    int i;
    char buf[100];
    for (i = 0; i < k; i ++) {
        scanf("%s", buf);

        /* convert time format from "HH:MM:SS" to second */
        int park_time = convertToSecond(buf);
        printf("%d\n", cur_car_num[park_time]);
    }
}

/*
 * the function to output the plate number of the car that has parked for
 * the longest time period, and the corresponding time length.
 */
void outputLongestTime(int car_num, int *park_time, long long int *car_id)
{
    int i, max_park_time = 0;

    /* calculate the maximum number of park time period */
    for (i = 0; i < car_num; i ++) {
        if (max_park_time < park_time[i]) {
            max_park_time = park_time[i];
        }
    }

    /*
     * output all the plate number of cars which has
     * parked for the longest time period
     */
    for (i = 0; i < car_num; i ++) {
        if (park_time[i] == max_park_time) {
            convertIntToPlateNum(car_id[i]);
            putchar(' ');
        }
    }
    /* output the time length */
    printf("%02d:%02d:%02d\n",
           max_park_time / 3600, (max_park_time / 60) % 60, max_park_time % 60);
}
```

Listing A.3: count-park-time.c

```c
/*
 * the function to count times which the cars park for
 *
 * @param (int) n, (struct record *) records, (long long int *) car_id,
 *        (int *) park_time, (int *) change_car_num, (int *) cur_car_num
 * @return (int) car_num
 */
int countParkTime(int n, struct record *records, long long int *car_id,
                  int *park_time, int *change_car_num, int *cur_car_num)
{
    /* initialize park_time[i] = 0 */
    int i;
    for (i = 0; i < N_MAX; i ++) {
        park_time[i] = 0;
    }

    int last_status;
    int car_num = -1;

    for (i = 0; i < n; i ++) {
        /* check if record[i] is a new (different) car */
        if (i == 0 || records[i].plate_number != records[i - 1].plate_number) {
            /* car_id[k] = the plate number of k-th car */
            car_id[++ car_num] = records[i].plate_number;
            last_status = status_out;
            /*set last status as out */
        }

        /* check if it is a legal pair by last_status */
        char illegal = 0;
        if (records[i].status == last_status || records[i].status == status_in) {
            illegal = 1;
        }
        last_status = records[i].status;
        if (illegal) {
            continue;
        }

        /* calculate time the car park for */
        park_time[car_num] += records[i].moment.data - records[i - 1].moment.data;

        /* the number of cars is enhanced at the moment when the car comes in */
        change_car_num[records[i - 1].moment.data] ++;
        /* the number of cars is reduced at the moment when the car comes out */
        change_car_num[records[i].moment.data] --;
    }
    /*
     * because the array is 0-index, the number of
     * different cars should be car_num + 1
     */
    car_num ++;

    /* the number of cars at the time t is sum [i = 0 to t] {change_car_num[i]} */
    cur_car_num[0] = change_car_num[0];
    for (i = 1; i < TIME_SEC_MAX; i ++) {
        cur_car_num[i] = cur_car_num[i - 1] + change_car_num[i];
    }

    return car_num;
}
```

Listing A.4: define-structures.c

```c
/* define status_out = 0, status_in = 1 */
enum { status_out, status_in };

/* define the structure of time format */
struct time_format
{
    int data;
};

/* define the structure of input record */
struct record
{
    long long int plate_number;
    struct time_format moment;
    char status;
};

/* the function to determine the sign of a long long int
 *
 * @param (long long int) x
 * @return (int) y = 0,  if x = 0
 *                 = 1,  if x > 0
 *                 = -1, if x < 0
 */
int sgn(long long int x)
{
    return x == 0 ? 0 : (x < 0 ? -1 : 1);
}

/* the function for qsort to compare two records
 *
 * the rule of comparing two records:
 * 1. compare the alphabet order of plate numbers of two records
 * 2. compare the time of two records
 */
int compareRecord(const void *a, const void *b)
{
    struct record *A = (struct record*)a;
    struct record *B = (struct record*)b;
    return (A->plate_number == B->plate_number) ?
           sgn(A->moment.data - B->moment.data) :
           sgn(A->plate_number - B->plate_number);
}
```

Listing A.5: hash.c

```c
/*
 * convert plate number (string type) to integer
 *
 * @param (char) *str
 * @return (long long int) hash of the plate number
 *
 * because plate number has 7 character
 * 36 ^ 7 = 78364164096 > 2147483647 (maximum of 32 bit integer)
 * we have to use a long long int (64 bit integer) to save the plate number
 */
long long int convertPlateNumToInt(char *str)
{
    int i;
    long long int result = 0;
    for (i = 0; str[i]; i ++) {
        result *= 36;
        if (str[i] >= '0' && str[i] <= '9') {
            /* map '0' ~ '9' to 0 ~ 9 */
            result += str[i] - '0';
        } else if (str[i] >= 'A' && str[i] <= 'Z') {
            /* map 'A' ~ 'Z' to 10 ~ 35 */
            result += str[i] - 'A' + 10;
        }
    }
    return result;
}

/* convert integer to plate number (directly output) */
void convertIntToPlateNum(long long int p)
{
    long long int i;
    for (i = 1; i <= p; i *= 36);
    for (i /= 36; i; i /= 36) {
        int c = (p / i) % 36;
        if (c < 10) {
            /* map 0 ~ 9 to '0' ~ '9' */
            c = c + '0';
        } else {
            /* map 10 ~ 35 to 'A' ~ 'Z' */
            c = c - 10 + 'A';
        }
        putchar(c);
    }
}

/*
 * convert time format from "HH:MM:SS" (string) to second
 *
 * @param (char) *str
 * @return (int) second
 *
 * string: HH:MM:SS
 * index:  01 34 67
 *
 * hour   = str[0] * 10 + str[1]
 * minute = str[3] * 10 + str[4]
 * second = str[6] * 10 + str[7]
 */
int convertToSecond(char *str)
{
    int h = (str[0] - '0') * 10 + (str[1] - '0');
    int m = (str[3] - '0') * 10 + (str[4] - '0');
    int s = (str[6] - '0') * 10 + (str[7] - '0');
    return (h * 60 + m) * 60 + s;
}
```

Appendix B

Declaration

Declaration

We hereby declare that all the work done in this project titled "Cars on Campus" is of our independent effort as a group.