The World's Richest

Kuan Lu

Huang-shi Tian

Guan-chun Zhao

Date: 2014-10-23

Chapter 1: Introduction

In this project We are assigned to simulate this job of Forbes magazine, it publishes every year its list of billionaires based on the annual ranking of the world's wealthiest people, but concentrate only on the people in a certain range of ages. That is, given the net worths of N people, we must find the M richest people in a given range of their ages. For example if we get 4 25 50 from the input file, then we must list out the top four richest people range in age from 25 to 50

However, considered that the age and net worth of some people might dulplicate, we sort the people as follows. That is, the order is in decreasing order of the net worths. In case there are equal worths, it must be in increasing order of the ages. If both worths and ages are the same, then the output must be in increasing alphabetical order of the names. It is guaranteed that there is no two persons share all the same of the three pieces of information. In case no one is found, output "None".

To find an optimum algorithm to solve the problem, we have adopted two method of data structure and different ways of implementation, each will be well illustrated in the following content, we also tested our program with different input file that contains very extreme situations that we took thought of.

Chapter 2: Algorithm Specification

I. Algorithm I, using an array and sort by qsort

Data structure

In algorithm I, we use an array to store the information of the rich people, struct info

```
{
    char name[10];
    int age;
    long int net;
}info[100000];
```

Using an array is the easiest method to implement compared to other data structure we used in this project, however, sorting the array consumes great time, and lacks efficiency.

Algorithm Description:

We first read the data in the input file and put it in an array using a for

loop, then we use Quicksort to sort the people in accordance with the given order. In the algorithm we used the qsort tools provided by stdlib.h because our group has discussed in detail the implementation of the tools, and to use it we just have to code a simple function cmp.

Note that in the Project 2 description document, the maximum number of outputs is never more than 100, so we designed an extra step to simplify the algorithm in extreme situations, which is that if more than a hundred people with the same age are in the input file, since the array is already sorted, only the richest 100 people are taken in to consideration because the rest is definitely not gonna be printed out.

At last we run a for loop starting from the head of the array, if the age of a person meet the requirement, we print it out, if the number of people printed reaches maxnumber, we break the loop.

Function (In pseudo-code)

```
void ArrayImplementation()
```

1: Get the number of people nPeople;

2: Get the number of queries nQueries;

3: for(i=0;i<nPeople;i++)

{

4: Get the name of the people info[i].name;

5: Get the age of the people info[i].age;

6: Get the net worths of the peoplr info[i].net;

}

7: qsort(info, nPeople, sizeof(struct info), cmp);

8: for (k = i = 0; i < nPeople; i++)

{

9: If (++ageCount[info[i].age] < 101)

10: pos[k++] = i;

}

11: for (i = 0; i < nQueries; i++)

{

12: count=0;

13: Get the maxnumber of output maxNum;

14: Get the range of age Amin, Amax;

15: for (j = 0; j < the totoal number of the new array k; j++)

{

16: if (info[pos[j]].age is in the required range)

{

17: print the name, age, net-worths;

18: count++;

19: if (count==maxNum);

20: break;

}

21: if(count==0) print "None";

}

II. Algorithm II, using a binary tree

Data structure

In algorithm II, we use a tree to store the information of the rich people, struct tree

{

long int key;

```
    struct tree *l;
    struct tree *r;
} *T = NULL, *s[100000];
```

Using a tree makes the inserting procedure more complicated, but it has advantage over array implementation, for the data is stored in certain order and the sorting procedure is just a problem of traversal..

Algorithm Description:

In this algorithm we will build a tree to store all the data, so the first step is insertion, we read data from the input file, if the net worth of a person is small than the root, we place it as the left child of the root, otherwise we place it as the right child of the root, for every node in the tree, the left child is always smaller than the root and the right child is always bigger. Thus when we finish reading the data, the tree is already in order. All we need to do is an in-order traversal and store the ordered people's information in a new array, after which we selected out the people in the given range exactly the same with algorithm I.

Function (In pseudo-code)

```
void TreeImplementation()

    1: Get the number of people nPeople'
    2: Get the number of queries nQueries
    3: for(i=0;i<nPeople;i++)
        {
    4:   Get the name of the people info[i].name;
    5:   Get the age of the people info[i].age;
    6:   Get the net worths of the peoplr info[i].net;
    7:   Insert(i);      /*insert the index of people in the tree*/
        }
    8: traverse();      /*in-order traversal, the sorted data is stored
                          in the array temp[]*/
```

```
 9: for(i=0;i<nPeople;i++)
10: info[i]=temp[i];    /* transfer information from array 'info' to
                           array 'temp' in order */
11: for (k = i = 0; i < nPeople; i++)
    {
12:          If (++ageCount[info[i].age] < 101);
13:          pos[k++] = i;
    }
14: for (i = 0; i < nQueries; i++)
    {
15: count=0;
16: Get the maxnumber of output maxNum;
17: Get the range of age Amin, Amax;
18: for (j = 0; j < the totoal number of the new array k; j++)
      {
19: if (info[pos[j]].age is in the required range)
        {
20:         print the name, age, net-worths;
21:         count++;
22:         if (count==maxNum);
23:         break;
        }
24: if(count==0) print “None”;
```

```
        }
void insert(int i)/* insert a node in the tree */
```

1: if(the node is empty)

2: {create a new node;

3: store i in the node;

4: point the left and right child to NULL;}

5: while(1)

{

6: if(the key to insert is less than current node's)

7: find the right location in the left tree and insert it;

8: break;

9: else

10: find the right location in the right tree and insert it;

11: break;

}

```
void traverse()/* in-order traverse a tree using
                  iteration in case of segment fault
                  caused by too many recursions */
```

```
1: struct tree *pos = T;

2: do

  {

3:   while (pos != NULL)

4:      {push all nodes on the left into the stack}

5:   if (the stack is empty)

6:        break;

7:   pop all the element from stack

8:   transfer information in sorted order

9:   pos = pos->r; /* go right by one step */

}

while (1);
```

Chapter 3: Testing Results(Based on algorithm I)

The requirements of this work is that we are supposed to rank the wealthiest people, but concentrate only on the people in a certain range of ages. That is, given the net worths of N people, we must find the M richest people in a given range of their ages,and output the M richest people with their ages in the range [Amin, Amax].

Considering these requirements,I design 8 groups of dates to test our programme,and they can test 16 conditions which programme may face to.In 16 conditions,15 of them are special conditions,and 1 of them is a normal condition which has a lot of disorder ed dates.The condition,result of the test,and the name of the files with test dates are showed in following table:

The test condition	Result illustrate	File with dates
1、Name、age and worth are all different	Succeed to read 10^5 people's dates,and find 4 right people,then have the rank of worth.	test1.in
2、Only worth is the same	Succeed to read 200 people's dates, and rank rightly.	test2.in
3、Only age is the same	Succeed to have the rank of worth.	test3.in
4、Only name is the same	Succeed to have the rank of worth.	test4.in
5、Only name is different	Succeed to have the rank of name.	test5.in
6、Only worth is different	Succeed to have the rank of worth.	test6.in
7、Only age is different	Succeed to have the rank of worth.	test7.in
8、If worth has 10^6or-10^6	Succeed to reveal	test6.in
9、If age has 200	Succeed to reveal	test3.in
10、If name has special character	Succeed to reveal, and should compare the characters by their ASCII value	test8.in
11、When have 100 more people who have the same worth	Succeed to reveal.In our programme,there is a loop to get the vaild people(eliminate the people which have the same age but rank above 100), but it seems not working	test2.in
12、When wealth are all minus	Succeed to reveal	test4.in
13、If no one is found	Succeed to output "None"	test7.in
14、If need to find 4 people but only have 2 right man	Only output 2 right people,nothing else (it meets teacher's requirements)	test8.in
15、Two people,one name is abc,and the other is abcd(their age and worth are all the same),what can that be	abc will be in front of abcd	test5.in
16、Will the people who are at the bound of the requirements be showed?	Succeed to show	test1.in

We think our method of testing is reasonable. Because We have taken almost all conditions of the extreme situations in to consideration, and we also have a test for normal condition but with higher complexity. The test programme runs perfectly.

We also tested the running time of the program if the input file are positive sequence, inverted sequence or disordered, using the programme in project 1 to measure the time.

positive sequence	0.4825	0.483	0.482	0.488	0.4846	0.4837	test1.3.in
inverted sequence	0.4908	0.487	0.486	0.488	0.4903	0.4883	test1.2.in
disorder	0.6735	0.666	0.667	0.668	0.6693	0.6687	test1.in

The experiment shows that the running time positive sequence<inverted sequence<disorder, our programme takes less running time in positive sequence than in inverted sequence or disodered sequence, that is because we've used qsort for sorting.

On the other hand,there are such codes in our programme:

```
for (k = i = 0; i < nPeople; i++)
    if (++ageCount[info[i].age] < 101)
        pos[k++] = i;
```

The codes are originated from M<=100,and it may have a great optimization for extreme condition.But the code has a pos array,so that its essence is to sacrifice space for time.

Condition of codes	1	2	3	4	5	average	File with dates
Don't use this code	0.0143	0.0143	0.0141	0.0142	0.0143	0.0142	test2.1.in
Use this code	0.0145	0.0145	0.0144	0.0145	0.0143	0.0144	test2.1.in

The experiment indicates that when the dates are not large enough(or not extremely large enough), this code still has a bad effect.

Chapter 4: Analysis and Comments

Time complexity

Although it use two 'for' loop, but its time complexity should be o(N),because there is only one loop is K(approximately N),another is the number of queries, so the whole time complexity should be o(N).At the same time, the time complexity of the sort is o(N*logN),so the whole

time complexity is o(N*logN).

Space complexity

Because only the 'for' loop of the input has effect on space complexity,so the space complexity is o(N).

Optimization

We have two other programme,use other ways to make the sort.

Programme	1	2	3	4	5	Average
Qsort	0.6735	0.6663	0.6667	0.6675	0.6693	0.6687
Tree	0.7134	0.7277	0.722	0.7056	0.7141	0.7166
Radix	0.7398	0.7397	0.7296	0.7318	0.7334	0.7348

The experiment shows that the qsort programme is the best one in three programmes.When the dates is as big as possible,the tree programme may have the same effect as the qsort one.

Appendix: Source Code (in C)

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

struct info /* structure storing every person's information */
{
    char name[10];
    int age;
    long int net;
}info[100000];

int cmp(const void *a, const void *b)/* function for qsort */
{
    if (((struct info *)a)->net - ((struct info *)b)->net)      /* whether their
net worths are different */
        return ((struct info *)b)->net - ((struct info *)a)->net;
    else if(((struct info *)a)->age - ((struct info *)b)->age)  /* whether their
ages are different */
        return ((struct info *)a)->age - ((struct info *)b)->age;
    else
        return strcmp(((struct info *)a)->name, ((struct info *)b)->name);
}

int main(void)
{
    int i, j, k, maxNum, Amax, Amin, count;
```

```
    long int nPeople, nQueries, ageCount[205], pos[100000];
    memset(ageCount, 0, sizeof(ageCount));
    memset(pos, 0, sizeof(pos));
    /* array 'ageCount' records number of appearance of every distinct age */
    /* array 'pos' records the position of person whose age only appears no more
than 100 times in previous positions */

    /* read in information */
    scanf("%ld %ld", &nPeople, &nQueries);
    for (i = 0; i < nPeople; i++)
        scanf("%s %d %ld", info[i].name, &(info[i].age), &(info[i].net));

    /* sort all information in order required in output */
    qsort(info, nPeople, sizeof(struct info), cmp);

    /* for maximun output number is 100, so it indicates redundency that a specific
age appears more than 100 times */
    for (k = i = 0; i < nPeople; i++)
        if (++ageCount[info[i].age] < 101)
            pos[k++] = i;

    for (i = 0; i < nQueries; i++)
    {
        count = 0; /* reset the counter for output information */
        printf("Case #%ld:¥n", i+1);
        scanf("%d %d %d", &maxNum, &Amin, &Amax);
        for (j = 0; j < k; j++) /* 'k' records the number of useful information */
        {
            if (info[pos[j]].age >= Amin && info[pos[j]].age <= Amax)/* check age
location */
            {
                printf("%s  %d  %ld¥n",  info[pos[j]].name,  info[pos[j]].age,
info[pos[j]].net);
                if (++count == maxNum) /* increment 'count' and check whether it's
adequate */
                    break;
            }
        }
        if (!count) /* when 'count' is zero, that is, no information has been printed
*/
            printf("None¥n");
    }
    return 0;
}
```

Declaration

We hereby declare that all the work done in this project titled "World's Richest" is of our independent effort as a group.

Duty Assignments:

Programmer: Huang-shi Tian
Tester: Guan-chun Zhao
Report Writer: Kuan Lu