



中南大學
CENTRAL SOUTH UNIVERSITY

算法设计与分析

实验报告(贪心)

学生姓名	杨凯楠
学 号	8208201004
专业班级	信息安全 2002 班
指导教师	石峰
学 院	计算机学院
完成时间	2021 年 11 月 18 日

一、 实验目的

掌握贪心算法

二、 实验内容

1. 搬桌子问题

某教学大楼一层有 n 个教室，从左到右依次编号为 1、2、...、 n 。现在要把一些课桌从某些教室搬到另外一些教室，每张桌子都是从编号较小的教室搬到编号较大的教室，每一趟，都是从左到右走，搬完一张课桌后，可以继续从当前位置或往右走搬另一张桌子。输入数据：先输入 n 、 m ，然后紧接着 m 行输入这 m 张要搬课桌的起始教室和目标教室。输出数据：最少需要跑几趟。

Sample Input

```
10 5
1 3
3 9
4 6
6 10
7 8
```

Sample Output

```
3
```

2. 分发饼干

假设你是一位很棒的家长，想要给你的孩子们一些小饼干。但是，每个孩子最多只能给一块饼干。

对每个孩子 i ，都有一个胃口值 $g[i]$ ，这是能让孩子们满足胃口的饼干的最小尺寸；并且每块饼干 j ，都有一个尺寸 $s[j]$ 。如果 $s[j] \geq g[i]$ ，我们可以将这个饼干 j 分配给孩子 i ，这个孩子会得到满足。你的目标是尽可能满足越多数量的孩子，并输出这个最大数值。

输入：数组 g 和数组 s

输出：最多能满足的孩子个数

样例：

输入： $g = [1,2,3]$, $s = [1,1]$

输出：1

输入： $g = [1,2]$, $s = [1,2,3]$

输出：2

3. 加工生产调度

某工厂收到了 n 个产品的订单，这 n 个产品分别在 A、B 两个车间加工，并且必须先在 A 车间加工后才可以到 B 车间加工。

某个产品 i 在 A、B 两车间加工的时间分别为 A_i, B_i 。怎样安排这 n 个产品的加工顺序，才能使总的加工时间最短。

这里所说的加工时间是指：从开始加工第一个产品到最后所有的产品都已在 A、B 两车间加工完毕的时间。

输入：

第一行仅一个数据 n ，表示产品的数量；
接下来 n 个数据是表示这 n 个产品在 A 车间加工各自所要的时间；
最后的 n 个数据是表示这 n 个产品在 B 车间加工各自所要的时间。

输出：

第一行一个数据，表示最少的加工时间；
第二行是一种最小加工时间的加工顺序。

输入样例：

```
5
3 5 8 7 10
6 2 1 4 9
```

输出样例：

```
34
1 5 4 2 3
```

4. 课程表

这里有 n 门不同的在线课程，他们按从 1 到 n 编号。每一门课程有一定的持续上课时间（课程时间） t 以及关闭时间第 d 天。一门课要持续学习 t 天直到第 d 天时要完成，你将会从第 1 天开始。

给出 n 个在线课程用 (t, d) 对表示。你的任务是找出最多可以修几门课（你不能同时修两门课程）。

输入： n 和 n 对数据 (t, d)

输出：最多可以修多少门课

样例：

输入：4 [[100, 200], [200, 1300], [1000, 1250], [2000, 3200]]

输出：3

样例解释：

这里一共有 4 门课程，但是你最多可以修 3 门：

首先，修第一门课时，它要耗费 100 天，你会在第 100 天完成，在第 101 天准备下门课。
第二，修第三门课时，它会耗费 1000 天，所以你将第 1100 天的时候完成它，以及在第 1101 天开始准备下门课程。

第三，修第二门课时，它会耗时 200 天，所以你会在第 1300 天时完成它。

第四门课现在不能修，因为你将会在第 3300 天完成它，这已经超出了关闭日期。

三、 具体过程

1.搬桌子

分析：本题采用贪心算法，即从第一个教室开始走，到达目标教室后从目标教室向后搜索，以此类推，直到所有教室需要搬的课桌都被处理。在存储类型中，我将课桌原本在的教室设为数组下表，目标教室为数组中存贮的信息。若该教室不搬桌子，就将其设为 0。通过递归函数，走一步就将数组内容变换为下一次处理的数组下表。

详细设计：

```
1. #include<iostream>
```

```

2. #include<algorithm>
3. using namespace std;
4. int mov[100] = { 0 };
5. int n,m;
6. void set() {
7.     cin >> n>>m;
8.     int j = 0, k = 0;
9.     for (int i = 0; i < m; i++) {
10.         cin >> j >> k;
11.         mov[j] = k;          //mov 上线到 n
12.     }
13. }
14. void solution(int i) {
15.     int j=0;
16.     for (i; i <= n; i++) {
17.
18.         if (mov[i] != 0) {
19.             j = mov[i];
20.             mov[i] = 0;
21.             if (j <= n)
22.                 solution(j);
23.             break;
24.         }
25.     }
26. }
27.
28. int main(void) {
29.     set();
30.     int c = 0;
31.     for (int i = 1; i <= n; i++) {
32.         if (mov[i] != 0) {
33.             c++;
34.             solution(i);
35.         }
36.     }
37.     cout << c;
38.     return 0;
39. }

```

实现:

```
Microsoft Visual Studio 调试控制:
10 5
1 3
3 9
4 6
6 10
7 8
3
D:\杨凯楠\c++\算法\实验报告
```

测试分析：本类问题需要将数对按出发教室排序，而由于我将出发教室设为了数组下标，故而不需要进行排序，因为数组下标本身就是排好序的。如图有十个教室，五组数对最后需要三次。

2. 分发饼干

分析：本题先对输入的胃口值、饼干尺寸进行非递减排序，然后建立一个二重循环，以人的胃口值为基准，直到找到一个能满足该胃口的饼干为止，当饼干遍历一次后算法结束。在此算法中可以进行相应的优化，比如在对胃口值和饼干大小排好序后，对于每一个胃口值找饼干不必要从头开始，因为下一个人的胃口一定大于等于此人胃口，故而从该处的饼干往后寻找饼干即可。如此其实算法的时间复杂度可能只有 $O(n)$, n 为饼干个数。

详细设计：

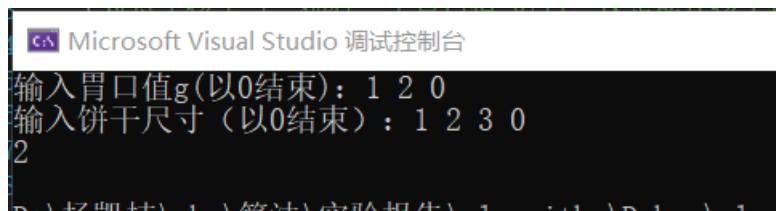
```
1. #include<iostream>
2. #include<stdio.h>
3. #include<algorithm>
4. using namespace std;
5. int g[100] = { 0 };
6. int s[100] = { 0 };
7. int leng, lens, countnum = 0;
8.
9. void set() {
10.     int i;
11.     printf("输入胃口值 g(以 0 结束): ");
12.     for ( i = 0; i<100; i++) {
13.         cin >> g[i];
14.         if (g[i] == 0) {
15.             break;
16.         }
17.     }
18.     leng = i;
19.     printf("输入饼干尺寸 (以 0 结束): ");
20.     for ( i = 0; i<100; i++) {
21.         cin >> s[i];
22.         if (s[i] == 0) {
23.             break;
24.         }
25.     }
```

```

25.     }
26.     lens = i;
27. }
28. void solution() {
29.     int j = 0;
30.     for (int i = 0; i < leng; i++) {
31.         for ( j ; j < lens; j++) {
32.             if (g[i] <= s[j] ) {
33.                 countnum++;
34.                 break;
35.             }
36.         }
37.     }
38. }
39. int main(void) {
40.     set();
41.     sort(s, s + lens);
42.     sort(g, g + leng);
43.     solution();
44.     cout << countnum << endl;
45.     return 0;
46. }

```

实现：



```

Microsoft Visual Studio 调试控制台
输入胃口值g(以0结束): 1 2 0
输入饼干尺寸(以0结束): 1 2 3 0
2

```

测试分析：

如图，胃口值为 1,2，饼干尺寸为 1,2,3，则可以满足两个人的胃口。

3.加工生产调度

分析：令 a 为 A 中时间， b 为 B 中时间，设 $A[]$ 为 $a < b$ 的作业集合， $B[]$ 为 $a \geq b$ 的作业集合，将 $A[]$ 的作业按 a 升序排序， $B[]$ 中的作业按照 b 降序排序，则 $A[]$ 作业接 $B[]$ 构成最优顺序。在此问题中，最重要的是考虑两车间的空闲情况，空闲情况分别有1：A车间加工第一个产品时，B车间一定空闲2：在此之后，A车间一直持续工作直到加工完属于自己的任务。3：A车间加工 $a < b$ 的作业时，B车间除开始的空闲外，一定不会空闲，直到A车间加工完 $B[]$ 中第一个，此时分两种情况：B车间已经加工完 $A[]$ 的产品了，此时他可能空闲了一段时间，也可能没有，若B车间还没有加工完 $A[]$ 的产品，则他一定不会空闲4：以此类推，在A加工最后一件产品后，有两种情况B车间能直接加工最后一件，或B还没有加工完除最后一件以外的产品，如此即可以计算出总得时间。

详细设计：

Demo: : [click here](#)

实现：

```
5
3 5 8 7 10
6 2 1 4 9
34
1 5 4 2 3
```

测试分析：如图输入5个产品，上面一组为A车间加工时间，下面一组为B车间加工时间，最后需要时间为34，加工顺序为1 5 4 2 3，与题目所给的测试样例一致。

4.课程表问题

分析：

先按结束时间排序，再依次选择，如果已上课时间加上该次课程时间<该课结束时间则选择上。如果只用贪心的话，如果出现3 6 7, 3 8, 5 8这样的数据，输出结果为1，但是明显结果应该是2，所以应该有所改进。即在已经选择的课程中将时间最长的与该课程时间比较，如果该课程时间较短则替换。在这一过程中可以选择优先队列或者插入排序，可以达到更快的算法效率，否则需要每做一次就排序一次，很麻烦。

详细设计：

```
1. #include<iostream>
2. #include<algorithm>
3. using namespace std;
4. int n;
5. struct Data {
6.     int t;
7.     int d;
8. };
9. Data dat[100];
10. int flag[100] = { 0 };
11. bool cmp(Data x, Data y) {
12.     return x.d < y.d;
13. }
14. int find(int n) {
15.     int temp=0, tempi=0;
16.     for (int i = 0; i < n+1; i++) {
17.         if (temp < dat[i].t && flag[i]==1) {
18.             temp = dat[i].t;
19.             tempi = i;
20.         }
21.     }
22.     return tempi;
23. }
24. int main(void) {
25.     cin >> n;
26.     for (int i = 0; i < n; i++) {
27.         cin >> dat[i].t >> dat[i].d;
```

```

28.     }
29.     //-----输入完毕
30.     sort(dat, dat + n, cmp);
31.
32.     int t1=0,temp=0,tempi=0,cnt=0;
33.     for (int i = 0; i < n; i++) {
34.
35.         t1 += dat[i].t;
36.         flag[0] = 1;
37.         if (t1 <= dat[i].d) {
38.             flag[i]=1;
39.             tempi = find(i);
40.             cnt++;
41.         }
42.         else {
43.             if (dat[tempi].t > dat[i].t) {
44.                 t1 -= dat[tempi].t;
45.                 flag[tempi] = 0;
46.                 flag[i] = 1;
47.                 tempi = find(i);
48.             }
49.             else {
50.                 t1 -= dat[i].t;
51.             }
52.         }
53.     }
54.     cout << cnt;
55. }

```

实现:



```

Microsoft Visual Studio 调试
4
100 200
200 1300
1000 1250
2000 3200
3
D:\杨凯楠\c++\算法\实验1

```

测试分析: 如图所示, 最多可以选择三门, 与所给的样例结果相一致。

四、 心得体会

本次贪心算法实验较为简单, 唯一有难度的就是加工生产调度一题时间计算方法比较烧脑, 而经过本次实验我深刻的体会到了在编写程序时又有一个清晰的逻辑的重要性。这也是算法实验的最后一个心得体会了, 经过这学期的算法学习与实验探究, 我受益颇多, 脑海中产生了一个坚定的想法并且逐渐深信不疑: 算法才是程序的灵魂所在, 这也是我深深爱上了算法。