

# 《算法原理》（2013秋）试题库

(注：按学号排序，答案仅供参考，可能有误)

## 简答题

### 1 孙炜程

动态规划和贪心算法的异同

贪心算法和动态规划算法都要求问题具有最有子结构性质，这是两类算法的一个共同点

动态规划算法通常以自底向上方法解各个子问题，而贪心算法则通常以自顶向下方法进行，以迭代的方法做出相继的贪心选择，每做一次贪心选择就将所求问题简化为规模更小的子问题。

### 2 吕源皓

问题：请叙述动态规划算法与贪心算法的异同，并说明什么是最优子结构。

答：相同点：两者都需要问题具有最优子结构性质。

不同点：贪心算法的每一步贪心决策都无法改变，每一步的最优解一定包含上一步的最优解。动态规划算法的全局最优解中一定包含某一个局部最优解，但不一定包含前一个局部最优解。

最优子结构：当一个问题最优解包含其子问题的最优解时，称此问题具有最优子结构。动态规划保存递归时的结果，因而不会在解决同样的问题时花费时间。

### 3 李敏

请清楚的解释二分图。

答案：二分图又称作二部图，是图论中的一种特殊模型。设  $G=(V,E)$  是一个无向图，如果顶点  $V$  可分割为两个互不相交的子集  $(A,B)$ ，并且图中的每条边  $(i, j)$  所关联的两个顶点  $i$  和  $j$  分别属于这两个不同的顶点集  $(i \in A, j \in B)$ ，则称图  $G$  为一个二分图。

#### 4 季聪

问题：任何思想方法都有一定的局限性，超出了特定条件，它就失去了作用。那么一个问题可以使用动态规划算法需要满足什么条件。

答案：1、最优化原理（最优子结构性质） 最优化原理可这样阐述：一个最优化策略具有这样的性质，不论过去状态和决策如何，对前面的决策所形成的状态而言，余下的诸决策必须构成最优策略。简而言之，一个最优化策略的子策略总是最优的。一个问题满足最优化原理又称其具有最优子结构性质。

2、无后效性将各阶段按照一定的次序排列好之后，对于某个给定的阶段状态，它以前各阶段的状态无法直接影响它未来的决策，而只能通过当前的这个状态。换句话说，每个状态都是过去历史的一个完整总结。这就是无后向性，又称为无后效性。

3、子问题的重叠性 动态规划将原来具有指数级时间复杂度的搜索算法改进成了具有多项式时间复杂度的算法。其中的关键在于解决冗余，这是动态规划算法的根本目的。动态规划实质上是一种以空间换时间的技术，它在实现的过程中，不得不存储产生过程中的各种状态，所以它的空间复杂度要大于其它的算法。

#### 5 凌云昊

问：在无向图的深度优先搜索中，边被分为哪几类？请根据  $\text{edge}(u, v)$  给出定义

答：

$\text{type edge}(u, v)$

$\text{tree}[u][v][v]u$

$\text{back}[v][u]u]v$

$\text{forward}[u][v][v]u$

$\text{cross}[v][v][u]u$

#### 6 徐赞佳

给定一个无向图  $G(V, E)$ ，给出一个算法，在  $O(V)$  时间内找出是否有一条回路，时间独立于  $|E|$

解法：用 DFS，如果有回边，就有回路。

## 7 陈志超

以比较为基础的排序算法的时间下界是多少？基数排序的时间复杂度是什么？这个下界适用于基数排序吗？为什么？

答案：基于比较的排序算法的时间下界为  $\Omega(n \log n)$ 。基数排序的时间复杂度为  $\theta(kn)$ ，其中  $k$  为元素的位数， $n$  为元素个数。这个下界不适用于基数排序，因为基数排序是各位数之间的比较，并不是基于元素之间比较的排序，所以该下界不适用。

## 8 吴强强

What's the time complexity of  $f(n)$  where

$$f(n) = 2f(n/2) + n \text{ (for } n \geq 2; f(1) = 1)$$

\_\_\_\_\_ (express the solution using the  $\Theta$  notation)

解：  $\Theta(n^2)$

## 9 姚岚

分而治之和动态规划均是基于递归技术的算法。请分别列举出一种采用上述思想的经典算法，并比较两种思想的差别。

解：

分而治之：快速排序

动态规划：Floyd 算法

分而治之的思想是自顶向下把问题实例划分成若干无重叠子实例，递归得到子实例的解然后组合。

动态规划：自底向上，有重叠子问题。

## 10 段景耀

题目：

请简述计算最小耗费生成树的 Kruskal 算法。

答案：

1. 设  $G=(V,E)$ ，对  $G$  的边以非降序权重排列；
2. 对于排序表中的每条边，如果放入  $T$  中不会形成回路的话，则将他加入  $T$  中，否则将它丢弃。

## 11 乔梁

请简述有向深度图中边的种类分别有哪些，并举例说明。

答：4 类，树边（tree edge）回边（back edge）前向边（forward edge）横跨边（cross edge），举例只要正确即可。

## 12 陈慧

请叙述动态规划算法与贪心算法的异同。

答：

相同点：两者都需要问题具有最优子结构性质。

不同点：贪心算法的每一步贪心决策都无法改变，每一步的最优解一定包含上一步的最优解。

动态规划算法的全局最优解中一定包含某一个局部最优解，但不一定包含前一个局部最优解。

## 13 张晟

请举出一例不是基于比较的排序算法，并给出其复杂度。

解答：基数排序， $O(KN)$

## 14 曹明轩

有  $m$  种糖果，有  $n$  个人，每个人随意抓糖果，一人只能抓两种且一种只有一个，

问是否存在一个裁判抓任意种糖果，使得他和每个人都有共同的糖果。证明是 NP 完全问题。

解：换下输入就可以规约到顶点覆盖

## 15 李超

题目：当我们玩扑克牌的时候，我们将牌从小到大排起来，这个过程是在抓牌的过程中完成的。即依次地拿起牌堆上的牌，放在手上，手中的牌永远是排好序的牌，下一张牌拿到手上我们会将它与手上的牌比较，从而放到一个恰当的位置。这种排序的方式是下列那种排序算法的体现：

a 基数排序 b 插入排序 c 冒泡排序 d 选择排序

答案：b 插入排序

## 16 张凯源

题目

什么是主定理？它的作用是什么？

答案：

定义：假设有递推关系式  $T(n) = aT\left(\frac{n}{b}\right) + f(n)$ ，其中  $n$  为问题规模， $a$  为递推的子问题数量， $\frac{n}{b}$  为每个子问题的规模（假设每个子问题的规模基本一样）， $f(n)$  为递推以外进行的计算工作。 $a \geq 1$ ， $b > 1$  为常数， $f(n)$  为函数， $T(n)$  为非负整数。则有以下结果：

(1) 若  $f(n) = O(n^{\log_b a - \epsilon})$ ， $\epsilon > 0$ ，那么  $T(n) = \Theta(n^{\log_b a})$

(2) 若  $f(n) = \Theta(n^{\log_b a})$ ，那么  $T(n) = \Theta(n^{\log_b a} \log n)$

(3) 若  $f(n) = \Omega(n^{\log_b a + \epsilon})$ ， $\epsilon > 0$ ，且对于某个常数  $c$  和所有充分大的  $n$  有  $af\left(\frac{n}{b}\right) \leq cf(n)$ ，那么  $T(n) = \Theta(f(n))$ 。

作用：可以非常方便地分析分治法所得到的递推式的时间复杂度

## 17 胡雪阳

划分问题：给出一个  $n$  个整数的集合  $S$ ，是否可能划分  $S$  成为两个子集合  $S_1$   $S_2$ ，使  $S_1$  中的整数和等于  $S_2$  中的整数和？

背包问题：给出具有大小  $s_1, s_2, \dots, s_n$  和值  $v_1, v_2, \dots, v_n$  的  $n$  项，背包的容量和整数  $k$ ，

是否可能用这些项的一部分来装背包，是他们的总大小不超过  $C$ ，同时总价值不小于  $k$ ？

请将划分问题规约到背包问题。

解答：对于划分问题的实例  $U=\{u_1, u_2, \dots, u_n\}$ ，构造背包，对于  $i$  小于  $n$ ，背包大小  $s_i=v_i=u_i$ ，

计算  $U$  集合的和为  $Sum$ ，构造  $C=k=Sum/2$ 。

PS.同样的问题可以考虑将子集和（subset sum）问题规约到规划问题，将 3SAT 规约到子集和问题

## 18 舒弋

问题：什么是最大堆？怎么用最大堆实现排序？

解答：最大堆就是父节点的值大于等于所有子节点的值几乎完全的二叉树。

先把数组变换成一个最大堆，然后依次取出根节点，就能完成排序。

## 19 闫方戈

题目：

现有两个字符串  $A = \text{"xzyzzyx"}$  和  $B = \text{"zxyyzxz"}$ 。请写出这两个字符串的最长公共子序列的长度。

回答：4

## 20 陈光

分析算法 MERGESORT 的时间复杂度并给出过程（可假设  $n=2^k$ ）

答： 时间复杂度  $\Theta(n \log n)$

根据合并操作需要的比较次数

每次合并最小比较次数为

$$C(n) = \begin{cases} 0 & (n == 1) \\ 2C(n/2) + n/2 & (n \geq 2) \end{cases}$$

解递推式得  $C(n) = n \log n / 2$

最大比较次数为

$$C(n) = \begin{cases} 0 & (n == 1) \\ 2C(n/2) + n - 1 & (n \geq 2) \end{cases}$$

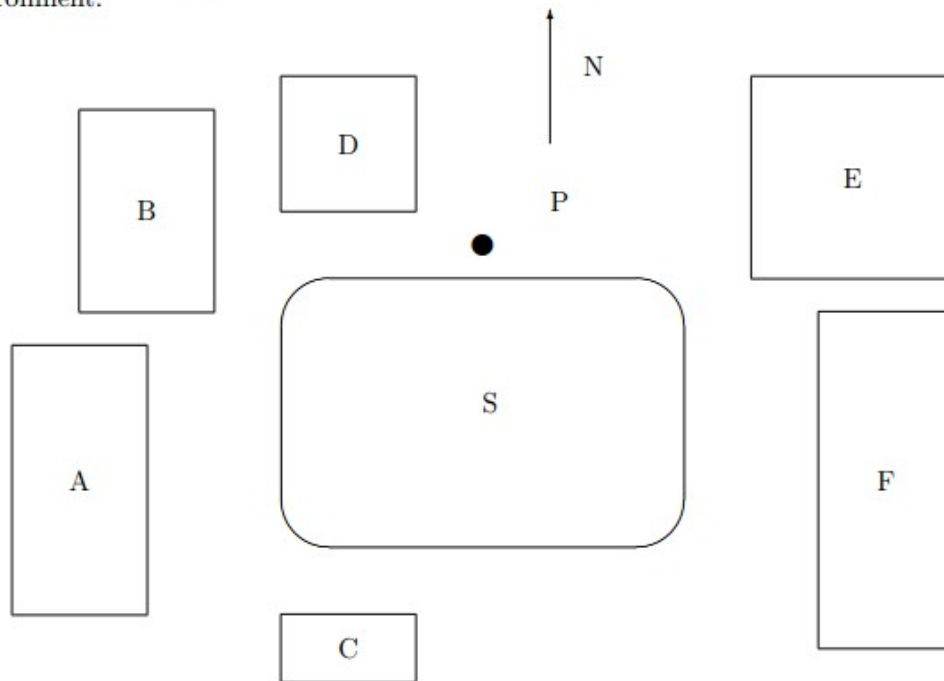
解递推式得  $C(n) = n \log n - n + 1$

综上算法 MERGESORT 的时间复杂度为  $\Theta(n \log n)$

## 21 陈钧

### Question

The 2014 World cup is coming. And the FIFA committee are always worried about the security in Brazil, and they want you to help design a scheme to guarantee the security with as few as police. Suppose the picture below is the top view of the Maracana stadium and its environment.



As shown above, we can see the west side of **E**, both south and east sides of **D**, and east side of **B** if standing at Point **P**.

Prove the question is NPC.

### Solution

Change the top view to an undirected graph  $G=(V,E)$ , where  $V$  contains all the sides of a building and  $E$  denotes the edge( $u,v$ ) from which we can see  $v$  from Point  $u$ . It's NP because we can check whether edge( $u,v$ ) is in set **E** in  $O(|E|)$ . Then we let Vertex Cover reduce to the problem.



## 22 毛驾臣

问：以二叉树为存储结构，写出求一棵二叉树所有节点个数的递归算法，并计算此算法的时间复杂度与空间复杂度。

求结点数的递归定义为：

若为空树，结点数为 0

若只有根结点，则结点数为 1；

否则，结点数为根结点的左子树结点数+右子树结点数+1

```
typedef char DataType;//定义 DataType 类型
typedef struct node{
    DataType data;
    struct node *lchild, *rchild;//左右孩子子树
}
BinTNode; //结点类型
typedef BinTNode *BinTree //二叉树类型
```

```
int Node(BinTree T)
{
    //算结点数
    if(T)
        if (T->lchild==NULL)&&(T->rchild==NULL)
            return 1;
        else return Node(T->lchild)+Node(T->rchild)+1;
        else return 0;
}
```

时间复杂度：  $O(\log n)$

空间复杂度： $O(n)$

## 23 汪伟滨

问题:(1)背包问题

(2)快速排序

(3)二分搜索

(4)矩阵链相乘

(5)合并排序

(6)需找第  $k$  小元素

(7)最长公共子序列问题

请将上叙 7 个算法按照不同的算法思想分为两类,并简要阐述这两个算法思想的不同.

答案:

分治:(2)快速排序 (3)二分搜索 (5)合并排序 (6)需找第  $k$  小元素

动态规划:(1)背包问题 (4)矩阵链相乘 (7)最长公共子序列问题

算法思想的不同点:动态规划算法是将所给问题的过程按时间或空间特征分解成若干相互联

系的阶段,按次序求每阶段的解;而贪心算法原问题分解成若干规模较小而与原问题相似的子

问题,迭代进行求解.

## 24 张一帆

Question:

某个算法 A 的运行时间由递归式  $T(n) = 7T\left(\frac{n}{2}\right) + n^2$  表示；另一个算法 A' 的运行时间为  $T'(n) = aT'\left(\frac{n}{4}\right) + n^2$ 。若要 A' 比 A 更快，那么 a 的最大整数数值是多少？

Answer:

利用主定理：

对 A,  $\Theta(n^{\log_2 7})$

对 A'，要使 a 最大，也需在  $a > b^d$  时取得，所以  $\Theta(n^{\log_4 a})$

所以 a 最大为 48

## 25 干祯超

给两个烧杯，容积分别是 m 和 n 升 ( $m \neq n$ )，还有用不完的水，用这两个烧杯能量出什么容积的水？

Answer: m,n,m-n,m+n 以及它们的线性组合

## 26 代政

简答题：请解释什么是 NP 完全问题？

答案：首先，NP 完全问题是难解问题的一个子类，这一类含有许多问题，这些问题有一个共同的特性，即如果它们中一个是多项式可解的，那么所有其他的问题也是多项式可解的。

## 27 李智

用动态规划解 TSP 问题

TSP 问题：旅行家要旅行  $n$  个城市，要求各个城市经历且仅经历一次，最后回到出发城市，并要求所走路程最短。

动态规划方程：

$$d(i, V') = \begin{cases} t_{ik}, & V' = \emptyset \text{ and } i \neq k \\ \min_{k \in V'} \{t_{ik} + d(k, V' - \{k\})\}, & V' \neq \emptyset \end{cases}$$

其中  $d$  函数表示从顶点  $i$  出发经过  $V'$  中各个顶点各一次且仅一次，最后回到出发点  $s$  的最短路径。

其中  $t_{ik}$  则表示当  $V'$  为空时，从  $i$  不需要经过任何点就回到  $k$ ，这就是城市  $i$  到城市  $k$  的距离。

## 28 赵子濠

简答题：

简单描述一个算法， $O(n)$  时间复杂度的条件下，在数组中找到出现频率大于  $1/4$  的所有数

答：每次删去不同的 4 个数，判断最后剩下的那些元素是否出现次数超过  $1/4$ 。

## 29 吴可宁

以比较为基础的排序算法的时间下界是  $\Omega(n \log n)$ 。基数排序的时间复杂度是什么？这个下界适用于基数排序吗？为什么？

答：基数排序的时间复杂度为  $\Theta(kn)$ ，其中， $k$  是元素的位数， $n$  是元素的个数。

此下界不适用于基数排序，因为基数排序不属于基于元素的比较来排序的算法类。

### 30 刘仑

简答题：

春游的时候，一群小伙伴（ $n$  人）决定围成一圈玩击鼓传花，每个人只能传给自己左边或右边的小伙伴，停止的时候，花在谁手上，谁就要表演骑马舞。小明是一个宅男，十分羞涩，十分害怕自己会要表演骑马舞。请帮他设计一个算法，假设从他开始传，当传递  $m$  次的时候，共有多少种传法最后还是会回到他手上。

解答：

动态规划。对小伙伴从  $1$ - $n$  编号。 $F[k,p]$  表示传了  $p$  次，花在  $k$  手中的状态下，有多少种传法花会在第  $m$  次时回到小伙伴  $1$  手中（假设从  $1$  开始传）。

递推式：

$$\begin{aligned} F(k,p) &= 1 \quad (p = m, k = 1) \\ &0 \quad (p = m, k \neq 1) \\ &F(R[k],p+1) + F(L[k],p+1) \quad (1 \leq k \leq n, 0 \leq p < m) \end{aligned}$$

求  $F(1,0)$  即可

( $R[k]$  是  $k$  右边的同学的编号， $L[k]$  是  $k$  左边的同学的编号)

### 31 龚立航

证明一个问题是 NPC 问题一般采用哪几个步骤？

答：

第一，证明该问题是 NP 问题

第二，选取一个已知的 NPC 问题

第三，构造一个从二中的问题到题目中问题的变换

第四，证明这个变换是多项式变换

### 32 董俊翼

简答题:

问题：如何判断一个排序的算法是不是稳定的？以下算法中有哪些是稳定的排序算法：快速排序、基数排序、希尔排序、堆排序、冒泡排序、归并排序。

答：假定在待排序的序列中有若干个相同的元素，若经过排序后这些元素的相对次序不变，则这种排序算法是稳定的。

上述算法中，基数排序、冒泡排序、归并排序是稳定的。

### 33 魏诚

什么是  $P$ ? 什么是  $NP$ ? 什么是  $NP^c$ ?

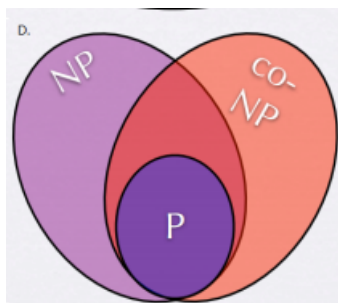
### 34 徐炜宸

简答题：分别描述  $NP$  和  $co-NP$  类，并试画出  $NP$ ， $co-NP$  及  $P$  类的关系。

答案：

对于一个问题的断言解，能在线性时间内验证其正确性的问题是  $NP$  类问题；

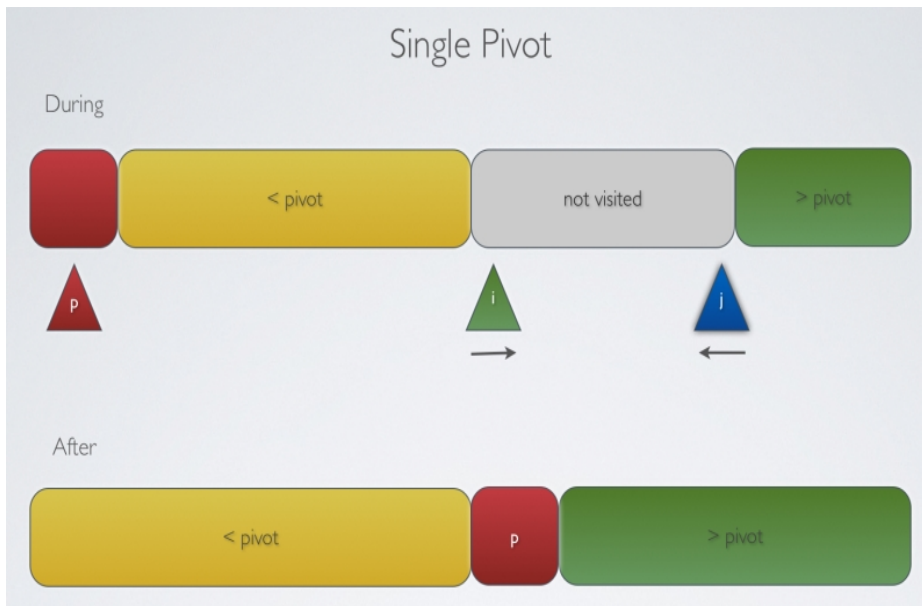
若一个问题的补问题为  $NP$  问题，其为  $co-NP$  问题；



### 35 肖迪

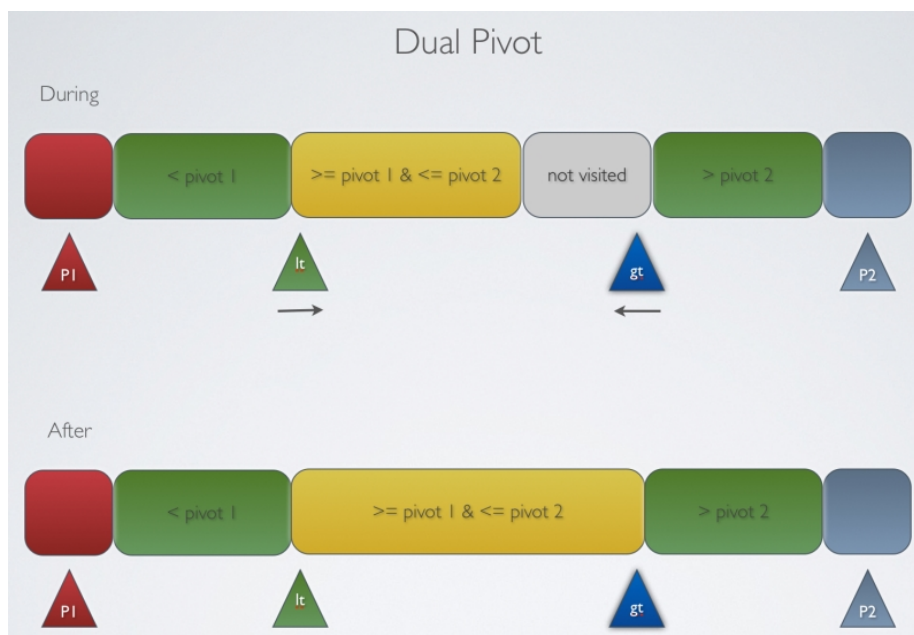
问题：

快排是很好的算法，在现实中（如 Java 7 Arrays.sort）有一种的实现方法效率更高：Dual-pivot。



如果 Single Pivot 的方法可以表示为：

请表示 Dual-pivot。



答案:

## Reference

[Algorithm of the Week: Quicksort - Three-way vs. Dual-pivot | Javalobby](<http://java.dzone.com/articles/algorithm-week-quicksort-three>)

### 36 姓名: 周丞

简答题:

Q:  $O$ 、 $\Omega$  和  $\Theta$  都是表示时间复杂度的符号, 请简单的描述一下他们各自表示的意义。

A: 简单地说, " $O$ " 符号, 就是"至多是"或"不超过"的意思, 一般用于表示运行时间上界; " $\Omega$ " 符号, 就是"至少是"或"不低于"的意思, 一般用于表示运行时间下界; " $\Theta$ " 符号, 就是"等于"的意思, 一般用于精确表示某一算法的时间耗费。

### 37 姓名: 古裕

设用于通讯的电文由 `abcdefgh` 共 8 个字母组成, 字母在电文中出现的频率分别为 7, 9, 2, 6, 32, 3, 21, 10, 试为这 8 个字母设计不等长 Huffman 编码。

答案:

a:000; b:001; c:01100; d:0111; e:11; f:01101; g:10; h:010

## 设计题

### 1 陶丝嘉

一幢大楼的底层有 1001 根电线, 这些电线一直延伸到大楼楼顶, 你需要确定底层的 1001 个线头和楼顶的 1001 次线头的对应关系。你有一个电池, 一个灯泡, 和许多很短的电线, 你需要上下楼几次才能确定电线接头的对应关系。

解答:



首先将底层一对接口（这里假设为（bottom1, bottom2））连接起来，然后上楼，根据提供的电池和灯泡的亮灭，确定顶层的一对（这里假设为（top1, top2）），接着将顶层的另一对连接起来（假设为（top3, top4）），然后下底层，确定和（top3, top4）对应的一对（假设为（bottom3, bottom4）），然后将底层的 bottom1 和 bottom3 连接，底层的 bottom2 和 bottom4 连接，上楼，分别将确定过的两对交换对接，即依次测试（top1, top4），（top2, top3）或者（top1, top3),(top2,top4)，直接灯泡亮为止即可确定这四个接口的对接关系。这样第一次确定 4 个接口需要上下楼 3 次。

然后根据第一次确定的 4 个接口，在顶层分别和剩余的接口中的其中四个接口连接，下到底层，和第一次确定四个接口一样，即可确定 8 个接口。这时确定 8 根只需要在上面的基础上加 1 次就可以。接下来就可以确定 16 个接口，并以此指数增加，从而到 2 的 10 次方，即 1024，即可全部确定 1001 个接口，而从 2 的 3 次方到 2 的 10 次方，共 8 次。

最后得出第一次确定的 3 次加上接下来的 8 次，共需 11 次即可确定他们的对应关系。

## 2 郝仁

Suppose there is an array  $A$  with  $n$  ( $\geq 2$ ) numbers.

(a) Describe a  $O(n)$  time algorithm to find two elements  $x, y \in A$  such that

$$|x - y| \geq |u - v| \text{ for all } u, v \in A.$$

(b) Describe a  $O(n \log n)$  time algorithm to find two elements  $x, y \in A$  such that  $|x - y| \leq |u - v|$  for all  $u, v \in A$ .

Solution:

(a) For this we can find the minimum and the maximum simultaneously which we can do in  $3 \lceil n/2 \rceil = O(n)$  time.

(b) For this we sort the numbers first, then  $x$  and  $y$  must be consecutive elements in the sorted order. We go through the sorted list and we find the smallest difference between two neighbor element.

### 3 任德威

动态规划问题：设计一个  $O(n^2)$  的算法，使之能够找出一个  $n$  个数的序列中最长的单调递增子序列。

解答：构造新的序列  $y$ ，是的  $y$  是经过排序的原序列。则只要求  $y$  和原序列的最长公共子序列就可以了。

### 4 杜佳薇

动态规划（0\1 背包问题）

假设小王在作业截止时间前还有四项作业（算法、数据结构、操作系统和 C++）尚未完成，现在距离这些作业截止时间还有 9 小时

其中完成算法作业预计需要花费 2 小时，其作业在期末总成绩（所有科目的成绩之和）中占 30 分；

完成数据结构作业预计需要花费 3 小时，其作业在期末总成绩中占 40 分；

完成操作系统作业预计需要花费 4 小时，其作业在期末总成绩中占 50 分；

完成 C++ 编程作业预计需要花费 5 小时，其作业在期末总成绩中占 70 分。

请问小王应该如何安排时间能够在期末总成绩获得更高的分数？

解：（《算法设计技巧与分析》P139 7.6 章）

动态规划范式：

第  $i$  项作业:  $s_i$ : 花费时间,  $v_i$ : 获得分数

$j$ : 总时间

$$V[i,j] = \begin{cases} 0 & \text{若 } i=0 \text{ 或 } j=0; \\ V[i-1,j] & \text{若 } j < s_i; \\ \max\{V[i-1,j], V[i-1, j-s_i] + v_i\} & \text{若 } i > 0 \text{ 和 } j \geq s_i; \end{cases}$$

## 5 张杨

设计一个算法

输入: 序列  $A$  (size of  $A$  equals  $N$ )

输出:  $A$  的所有长度为  $M$  的连续子序列中的最小值 ( $M \leq N$ )

$x$  from 1 to  $N$

if  $A(y)$  in quene and  $y \leq x - M$  then quene  $\rightarrow A(y)$

$A(x) \rightarrow$  quene

if  $A(y)$  in quene and  $y < x$  and  $A(y) \geq A(x)$  then quene  $\rightarrow A(y)$

$\min(A(x-M+1), \dots, A(x)) = \text{first element of quene} = A(U)$ ;

if  $x \geq M$  then output  $A(U)$

## 6 李昶宇



题目:

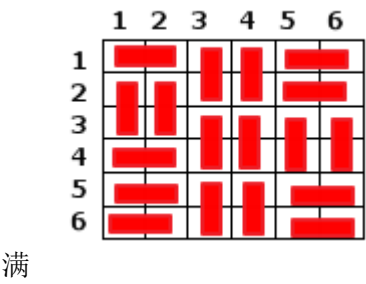
大学毕业后的我找不到工作, 于是乎找了一份搬砖的工作。准确来说是砌砖, 已知交大有一块人行道上需要更新砖瓦, 原本有一些砖还可以用, 所以就不能拿掉。先给出一个

$m \times n$  的路面， ( $m \leq 14, n \leq 1000$ ), 现在学校提供给我的新砖大小为  $1 \times 2$ ，由于美观问题，不允许将一块砖分成两块。问能否按照学校要求将  $m \times n$  的路面刚好铺满。如果可以有多少种方法将人行道铺满。

解题思路：

由于  $m \times n$  的数据量比较大，显然不能用普通深搜的办法来解决这个问题。于是能想到的是运用动态规划来解决。

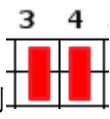
砖块有两种放的方法一种是  一种是 。比如讲一片  $m \times n$  的原本没有砖的道路填



1	1	0	0	1	1
0	0	1	1	1	1
1	1	0	0	0	0
1	1	1	1	1	1
1	1	0	0	1	1
1	1	1	1	1	1

，我们将横着的砖视作 1，竖着的砖视作 0，便有如下图：。

引入  $f[i][j]$  数组， $i$  表示第  $i$  行， $j$  表示第  $j$  行这么铺所能出现的可能数。那么我们对于第一

行进行举例， $f[1][110011]$  表示第 1 行所有位置全被占用，并且第 3,4 列  是会占用下一行的空间的。首先可以说的是 110011 这个数字过大，我们可以将这个数字理解为 2 进制，从而可以将 110011 转换为 51（2 进制向 10 进制转换）这样数字便好储存。当我们枚举第二行的所有状态时，便只用枚举（ $0 \rightarrow (2^{(m-1)} - 1)$ ）即可。那么状态转移方程便是

For (int  $i = 1; i \leq n$ ) {

    将第  $i$  行的已经有砖块的位置转换成二进制，得到 tmp。例如砖块为 001001 即转换成 9

```

for (int j = 0; j < 2^(m-1); ++j)
    For (int k = 0; k < 2^(m-1); ++j)
        If ((j & k == 0) && j & (tmp == 0) && (k & tmp == 0) (j + k + tmp ==
2^m - 1))
            F[i][j] += f[i - 1][k];
    }

```

Ps,  $j \& k == 0$  表示  $j$  状态和  $k$  状态在一行的每个砖块位置没有冲突的砖块，即砖块位置没有重叠。 $j + k + tmp == 2^m - 1$  表示刚好将这一排砖块铺满。即二进制显示为 11111111..1 ( $m$  个 1)

综上这一题算是解决的差不多，但是由于  $2^m * n$  依旧非常大，所以我们可以将状态滚动，因为每一行的状态只与上一行状态有关。所以可以将方程写成  $f[i \% 2][j] = f[(i - 1) \% 2][...]$ 。每次计算完一行时，将下一行状态清空，所以需要的空间是  $2 * 2^{(m-1)}$ ，便不是很大。

## 7 吴自怡

在  $n$  个人中，一个被所有人知道但却不知道别人的人，被定义为社会名流。现在的问题是如果存在，试找出该社会名流。你可以使用的唯一方式是询问：“请问你知道那个人吗？”请给出提问次数为  $O(n)$  的算法，写出伪代码，分析算法的正确性，并给出算法运行时间的精确分析（即  $O(n)$  中隐藏的系数）。

（提示：当你问  $A$  是否认识  $B$  时，如果  $A$  认识  $B$ ，则  $A$  不是社会名流；如果  $A$  不认识  $B$ ，则  $B$  不是社会名流）

A:

定义数组  $R[n][2]$ ， $R[i][0]$  记录  $i$  所认识的人数， $R[i][1]$  记录认识  $i$  的人数。

对每行输入（每行一对数字  $i$  和  $j$ ，表示  $i$  认识  $j$ ）

```
R[i][0]++; R[j][1]++;
```

最后找出  $R[i][0] == 0$   $R[i][1] == n$  的人就是名流

## 8 刘颖珊

（NOIP2013 Day1 第三题改编）某国有  $n$  座城市，编号从 1 到  $n$ 。城市之间有  $m$  条双向道路，每一条道路对车辆都有重量限制，即要经过的车辆载重不能超过道路的重量限制。某物流公司需要从城市 1 运货到城市  $n$ ，希望得到每辆车的最大载货量。请给出一个多项式时间复杂度的算法，并分析。

解法 1:

最大生成树。

【实现方法】将所有道路按限重从大到小排序，按顺序取道路，如果这条道路连接了已经相通的两个城市，那么舍弃这条道路，否则，选取它。

【正确性验证】加入现在需要决定取舍的道路为  $(u, v)$ ，且  $u$  和  $v$  已经有路径能使其连通，说明连通  $u$  和  $v$  的道路的限重都比  $(u, v)$  的限重大，所以  $(u, v)$  的加入不会使答案更优。

【时间复杂度分析】最大生成树的时间复杂度瓶颈在对道路的排序，下限为  $O(N\log N)$ 。DFS 的最多每条道路搜索两遍，时间复杂度为  $O(m)$ 。所以总的时间复杂度为  $O(N\log N)$ 。

【伪代码】

MST-KRUSKAL( $G, w$ )

$A = \{\}$

for each vertex  $v \in V[G]$  do

    MAKE-SET( $v$ )

sort the edges of  $E$  into nonincreasing order by weight limit  $w$

for each edge  $(u, v) \in E$ , taken in nonincreasing order by weight limit do

    if FIND-SET( $u$ )  $\neq$  FIND-SET( $v$ ) then

$A = A \cup \{(u, v)\}$

        UNION( $u, v$ )

return  $A$

DFS( $A, \text{start}$ )

    visited[start] = true

```

for each edge(start, v) ∈ A and visited[v] is false
    weight_limit[v] = min(weight_limit[start], w[u,v])
    DFS(A, v)

```

MAX-WEIGHT(G, w)

```

A = MST-KRUSKAL(G, w)
weight_limit[1] = ∞
DFS(A, 1)
return weight_limit[n]

```

解法 2:

类 SPFA (Shortest Path Faster Algorithm) 。

【实现方法】从城市 1 开始往相邻的城市拓展节点，每次被更新了 weight\_limit 的城市加入队列中，再按顺序出列继续拓展，直到没有城市 weight\_limit 值被更新。

【正确性验证】当拓展的路径为简单路径时，算法正确性显而易见。当拓展的路径出现环时，例如 1->2->3->4->2，此时城市 2 不会被更新，因为 1->2->3->4->2 的限重一定小于等于 1->2，这就满足了最短路性质。

【时间复杂度分析】平均时间复杂度  $O(km)$ ，m 为边数，k 一般为 2 或 3。最坏情况下会退化到  $O(n^2)$ 。

【伪代码】

SPFA(G, s, t)

```

Q = {}
enqueue(Q, s);
while not empty(Q) do
    u = dequeue(Q)
    for each (u, v) ∈ E do
        temp_min = min(w(u, v), weight_limit[u])
        if temp_min > weight_limit[v] then
            weight_limit[v] = temp_min
            enqueue(Q, v)

```

```
return weight_limit[t]
```

## 9 董明凯

僵尸大战太阳

=====

Mingkai Dong

（上接僵尸大战植物，见 42 邱喆）

Ted 在绞尽脑汁（如果还有脑汁的话）之后终于知道自己能够收集到多少价值的植物了。

然而就在 Ted 刚要跳出第一步的时候，Sophie 决定要给 Ted 增加难度，于是在花园（还是那个  $N \times N$  个小正方形）的某个区域内“种了一些太阳”。

众所周知，僵尸最讨厌太阳了。但是阴险的 Sophie 不愿意告诉 Ted 哪些小正方形里面有太阳。

无奈之下，Ted 只能根据平日对 Sophie 的了解进行猜测：

0. Sophie 的头是圆的，因此花园还是由  $N \times N$  的小正方形格子组成，第  $i$  行第  $j$  列的格子里面有一个价值为  $v[i,j]$  的植物；
1. Sophie 的头是圆的，因此他一定会在一个\*\*矩形\*\*区域内种太阳；
2. Sophie 的头是圆的，因此在他种太阳的区域中，一定\*\*没有价值为负\*\*的植物；
3. Sophie 的头是圆的，因此他种太阳的区域一定\*\*面积最大\*\*（会包含尽可能多的格子）。

Ted 的头不是圆的，因此他只能想到这么多了。为了躲开讨厌的太阳，Ted 希望能够猜测出哪些格子里面有太阳。



请设计一个算法，帮助 Ted。（Bonus: 保守的 Ted 不会跳到任何一个有可能有太阳的区域，那么在躲开太阳的情况下，Ted 最多能收集多大价值的植物？）

## Solution: ##

$up[i,j]$ 表示从 $[i,j]$ 这个格子向上（包括 $[i,j]$ 这个格子）有几个连续的没有负值的格子。

$left[i,j]$ 表示从 $[i,j]$ 这个格子向左（包括 $[i,j]$ 这个格子）有几个连续的“ $up$  值不小于  $up[i,j]$ ”的格子。

$right[i,j]$ 表示从 $[i,j]$ 这个格子向右（包括 $[i,j]$ 这个格子）有几个连续的“ $up$  值不小于  $up[i,j]$ ”的格子。

$f[i,j]$ 表示底边在第  $i$  行，高度为  $up[i,j]$ 的矩形中，包含没有负值格子的最大矩形区域的面积（即格子数）。

则可计算如下：

$$\begin{aligned} up[i,j] &= \{ \\ &0 \quad (v[i,j] < 0) \\ &1 \quad (i=0 \text{ and } v[i,j] \geq 0) \\ &up[i-1,j] + 1 \quad (i>0 \text{ and } v[i,j] \geq 0) \\ &\} \\ left[i,j] &= \{ \\ &1 \quad (j=0) \\ &1 \quad (j>0 \text{ and } up[i,j] > up[i,j-1]) \\ &left[i,j-1] + 1 \quad (j>0 \text{ and } up[i,j] \leq up[i,j-1]) \\ &\} \\ right[i,j] &= \{ \\ &1 \quad (j=n-1) \\ &1 \quad (j<n-1 \text{ and } up[i,j] > up[i,j+1]) \end{aligned}$$

```

        right[i,j+1] + 1  (j<n-1 and up[i,j] <= up[i,j+1])
    }
    f[i,j] = (up[i,j]) * (left[i,j] + right[i,j] - 1)

## Final Answer ##

令 maxf = max {f[i,j]}

```

所有  $f[i,j]=\max f$  的以  $[i-\text{left}[i,j]+1,j]$  为左下角的高度为  $\text{up}[i,j]$  的矩形均有可能包含太阳。

时间复杂度  $O(n^2)$

空间复杂度  $O(n^2)$

### And... ###

其实可以发现， $\text{up}[i,j], \text{left}[i,j], \text{right}[i,j]$  均仅仅对计算某一行时候有用，因此可以缩小到一维。

$f[i,j]$  可以不用实际记录，但是如果要求出方案的话，需要把算法做两次（第一次求出  $\max f$ ，第二次求出方案）。

Format copied from 5110379077.md

## 10 孟繁宇

Osmo

题目描述：

Osmo 是由 Hemisphere Games 开发的一款游戏，我们在这里考虑一个关于他的改变版本。

游戏开始玩家需要控制一个体积为  $V$  的气泡，通过吸收体积比它小的气泡从而扩大自己的体积，以吸收更多的气泡。当吸收了一个体积为  $V_1$  ( $V_1 < V$ ) 的气泡时，玩家控制的气泡体积增大为  $V+V_1$ ，因此可以吸收任何体积小于  $V+V_1$  的气泡。

例如：玩家控制的一个气泡体积为 5，游戏中还有其它体积为 3，7，10 的气泡。那么玩家可以依次吸收这三个气泡，从而获得整个游戏的胜利。

（吸收 3 之后体积变为  $5+3=8$ ，因此之后可以吸收体积为 7 的气泡 ( $8>7$ )，吸收之后它的体积变为  $8+7=15$ ，因此可以吸收体积为 10 的气泡 ( $15>10$ )，这时场上没有剩余其它的气泡，玩家获得胜利）

当然玩家也有可能无法获得游戏的胜利。例如在上面的例子中如果剩下的三个气泡体积为 3，8，100 的话，那么玩家就无法吸收所有的气泡（因为玩家能达到的最大体积为 8，并不大于第二个气泡体积 8）。

在本例中，我们考虑玩家可以进行以下两种操作以达到自己的目的：

1. 增加一个任意体积的气泡 ( $V_i \geq 1$  且为正整数)
2. 删除一个任意体积的气泡

两种操作的代价是相等的。也就是说，增加或删除一个气泡都需要进行一次操作。

对于上面气泡为 3，8，100 的例子来说，我们可以选择：①删除体积为 8，100 的气泡或者②增加一个体积为 2 的气泡，同时删除体积为 100 的气泡。这样两种操作都可以达到吸收所有气泡的目的，并且它们需要操作数是相等且最小的（均为 2）。

我们关心的内容是玩家需要多少的最少操作数，能够达到游戏胜利的目标（场上没有剩余其它的气泡）

### 【输入格式】

第一行是两个整数  $V$  和  $N$ ，分别代表玩家控制气泡的初始体积和场上其它气泡的数量 ( $V \geq 1$  且为正整数)。

第二行是  $N$  个整数，分别代表场上其它气泡的体积 ( $V_i \geq 1$  且为正整数)。

### 【输出格式】

一个整数，代表玩家获得胜利所需要的最小操作数。

【样例输入 1】

5 3

3 7 10

【样例输出 1】

0

【样例输入 2】

10 4

25 20 9 100

【样例输出 2】

2

【样例输入 3】

1 4

1 1 1 1

【样例输出 3】

4

题解：

首先对第二行数据进行排序，然后可以发现要吸收所有的气泡也就是要按照气泡体积的升序吸收所有的气泡。

之后可以通过观察得到一个结论：

如果玩家删掉了体积为  $V$  的气泡，则体积大于等于  $V$  的气泡也都需要删除。（结论 1）

因此有一个线形解决的方法：从前往后进行扫描，记录  $f[i]$  为吸收前  $i$  个气泡需要的最小操作数。

$$f[i] = f[i - 1] + C(i - 1, i)$$

当发现当前气泡无法吸收时，则增加一个体积为当前气泡体积-1 的气泡并吸收之，同时边际函数 C 值加 1。重复以上操作，直到可以吸收此气泡为止。

另一方面，根据观察得到的结论 1，令  $g[i]$  表示比元素  $i$  大的元素个数。也就是  $g[i] = \text{total} - i + 1$ 。

则最后所得的答案为  $\min(f[i]+g[i])$ ，其中  $1 \leq i \leq n$

需要注意对于样例 3 的特殊处理：最简单的方式是删除所有的元素，所以直接返回所有元素个数即可

代码：

---

```
if (m == 1) ost1<<"Case #"<<i<<": "<<t<<endl;else {
    sortit(totd,0,totd.size()-1);
    taild = totd.size()-1;
    for (int i1 = 0;i1<=taild;i1++)
    {
        if (m > totd[i1]) m += totd[i1];
        else {
            m1 = m;
            while (m1 <= totd[i1])
            {
                m1 = m1 * 2 -1;
                cnt1++;
            }
            m1 += totd[i1];
            m = m1;
        }
        a[i1] = cnt1;
    }

    for (int i1 = 0;i1<=taild;i1++)
    {
        if (ans > a[i1] + b[i1]) ans = a[i1] + b[i1];
    }
    ost1<<"Case #"<<i<<": "<<ans<<endl;
}
```

xx 学院将要举行一年一度的社会实践活动总结报告会，xx 学院抽取了过去一年中各个年级学生中的优秀社会实践活动进行公开展示。已知共有  $n$  个小组进行展示，每个小组的展示时间为  $T[i], 1 \leq i \leq n$ ，问如何安排各个小组的展示顺序，使得每个小组的平均等待时间最小？

解答

贪心算法：

1. Sort  $T[i]$

2.  $x[i] = x[i-1]$  （每组的展示时间）

3.  $T_{\text{总}} += x[i]$

$T_{\text{总}}$ 即为总等待时间

平均时间 =  $T_{\text{总}}/n$

## 12 童吉

### 【问题描述】

有一款经典 APP 游戏，游戏角色向上跳需要消耗能量，每跳 1 米就会消耗 1 点能量。

改编这个游戏，假设角色的正上方每 100 米处就有一个能量球（也就是这些能量球位于海拔 100, 200, 300.....米处），每个能量球所能提供的能量是不同的，一共有  $N$  个能量球（也就是最后一个能量球在  $N \times 100$  米处）。你要控制你的角色吃完所有能量球，并且让自己最后剩余能量最大。

你可以每次自由控制自己每次跳的高度，接着跳起把这个高度下的能量球全吃了，便能获得能量球内的能量，接着吃到的能量球消失。

注意，你不能二段跳，并且每次跳完都会掉回原地(0 米处)。

求在吃完所有的能量球的前提下，最多还能保留多少能量。

### 【输入格式】

第 1 行包含两个正整数  $N, M$ ，表示了能量球的个数和角色的初始能量。

第 2 行包含  $N$  个非负整数，从左到右第  $i$  个数字依次从下向上描述了位于  $i \times 100$  米位置能量球包含的能量。

### 【举例】

5 300

100 100 200 500 400

第一次跳起 300 米，用掉原有 300 个能量，吃掉前三个能量球，共剩 400 能量。

第二次跳起 400 米，用掉原有 400 个能量，吃掉第四个能量球，共剩 500 能量。

最后一次跳起 500 米，吃完所有能量球。剩下 400 能量。

### 【问题】

1. 考虑此贪心算法：每次跳跃把现有能量全部用完。  
请举出一个反例说明此算法可能导致无法吃完所有的能量球。  
请举出一个反例说明此算法可能导致无法得到最大值。
2. 设  $dp[i]$  为，吃完前  $i$  个能量球，角色剩下的最大能量。 $Sum[p,q]$  为，第  $p+1$  个球至第  $q$  个球的能量和。  
请设计一个算法，使得在  $O(n)$  时间的预处理后，每次计算  $Sum[p,q]$  的时间复杂度为  $O(1)$ 。  
请写出状态转移方程（或伪代码），并写出此算法的时间复杂度。
3. 已知此算法可以进一步优化至  $O(n)$ ，请问使用哪一种数据结构可以优化此算法？  
（栈/队列/堆/树……）

### 【答案】

1.

3 200

200 0 200

按贪心算法，第一次跳起 200 后，只吃到 200 能量球，无法继续跳起吃第 3 个了。

而第一次跳起 100，就能剩 300 能量刚好够再跳起吃第三个。

3 200

200 200 200

最优解是第一次跳起 100，吃掉第一个能量球。再跳起 300 吃完所有能量球。

2.

$S[i]$ 表示前  $i$  个能量球的总和。可以在  $O(n)$ 时间内预处理。 $SUM[p,q]=s[q]-s[p]$

For  $i:=1$  to  $n$  do

For  $j:=0$  to  $i$  do

$dp[i] = \max(dp[i], dp[j] + sum[j,i] - i * 100);$

3.

队列。

解释： 化简为  $dp[i] = \max(dp[i], dp[j] - s[j]) + s[i] - i * 100;$

因为每次都是找  $\max(dp[j] - s[j]);$  而且要花费  $N$  次。但是，  $i, i + 1$  找的区间只是错开一位。

所以可以用队列优化。维护一个单调递减队列。

【附队列优化版程序 pascal 版。。高中时候写的。。 其实类似于伪代码】

var  $n,m,i,j,x,l,r$ :longint;

que,f,sum:array[0..2000000] of longint;

function max(a,b:longint):longint;

begin

if  $a > b$  then  $max:=a$  else  $max:=b$ ;

end;

begin

readln( $n,m$ );

$f[0]:=m$ ;

$l:=1;r:=2;que[1]:=0;que[2]:=1$ ;



```

for i:=1 to n do begin
    read(x);
    sum[i]:=sum[i-1]+x;    //部分和预处理
    if i=1 then f[1]:=f[0]+x-100;    //边界
    while f[que[l]]<i*100 do inc(l);    //删除队列中不可能的元素
    if i<>n then f[i]:=max(f[i-1],f[que[l]]-sum[que[l]]+sum[i]-i*100) //状态转移
    else f[i]:=f[que[l]]-sum[que[l]]+sum[i]-i*100;    //状态转移
    while f[que[r]]-sum[que[r]]<=f[i]-sum[i] do dec(r);    //删除队列中不可能的元素
    inc(r);que[r]:=i;    //更新队列
end;
writeln(f[n]);
end.

```

### 13 魏星达

一伙人在海岛上寻宝，希望能找到到每个宝藏点的路径图，使得路径图加起来权重最小。

解法：最小生成树。

### 14 张君儒

某英超的世界知名俱乐部想要得到一名小球会的球员，但是由于英国的劳工证制度，导致该球员无法直接从该球会引进。于是该俱乐部想到通过多次其他球队的交易使该球员得到劳工证资格后再引入，同时该球会负担关联球会的转会费用，已知从小球会到该俱乐部之间的各个球队的转会费是一条含权有向无回图  $G(V,E)$ ，请设计一个算法，使该俱乐部的花费尽可能小，并上报球队经理预算值。

输入：含权有向图  $G(V,E)$ ， $V=\{1,2,3,\dots,n\}$

输出：顶点 1 到 n 的最小值

解：

$X = \{1\}; Y \leftarrow V - \{1\}; v[1] \leftarrow 0;$

for  $y \leftarrow 2$  to  $n$

    if  $y$  相邻于 1 then  $v[y] \leftarrow l[1, y]$

    else  $v[y] \leftarrow \infty$

end for

for  $y \leftarrow 1$  to  $n-1$

    令  $y \in Y$ ，使得  $v[y]$  最小

$X \leftarrow X \cup \{y\}$

$Y \leftarrow Y \cup \{y\}$

    for 每条边  $(y, w) \in E$

        if  $w \in Y$  and  $v[y] + l[y, w] < v[w]$  then

$v[w] \leftarrow v[y] + l[y, w]$

        end for

end for

return  $v[n]$

## 15 徐邦

学校新修了一条商业街，总共有  $L$  个门面，现在有  $A$  个商家报名想要开店，第  $A_i$  个商家需要  $B_i$  个门面，能够创造  $C_i$  的利润。请求出该如何商家使得利润最高。

解答：

同背包问题

## 16 王若愚

在一个大厅里有很多桌子，排成了一个三角形的形状，其中第一排有一张桌子，第二排有两张桌子，以此类推，桌子上有数量不等的香蕉，阿邦站在第一排的那张桌子前，想要吃香蕉，但是他有点懒，每次吃完一张桌子上的香蕉他只会继续吃左前方或者右前方的桌子上的香蕉，请你设计一个算法帮助阿邦找到一条路径使他能吃到数量最多的香蕉。

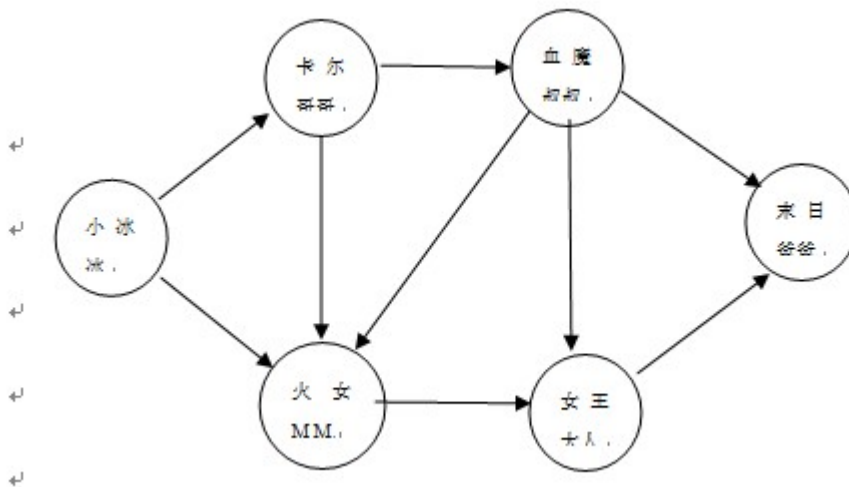
解答：动态规划：

建立一张三角形的表，表里的每一项对应一张桌子，里面存储的是走到这张桌子前阿邦可以吃到的香蕉的最大数量，递推式可以这么表示：第一排的桌子对应的数值是它本身有的香蕉的数量，其余的桌子对应的数值等于它自身有的香蕉数量加上它左后或者右后方桌子对应的数值里较大的那个然后记录一下是从哪个桌子上过来的，以此类推。在整张表格的都填写完之后我们在最后一排的所有桌子寻找数值最大的那张桌子，就可以找到阿邦应该走的路线了。

## 17 贺字明

倒不了的塔：

近卫军团的小冰冰的梦想是拯救世界，有朝一日能击杀天灾军团的黑色叉腰大魔头影魔导演。可小冰冰的实力不足，只好另寻他法杀死影魔。据小冰冰的多方打听，天灾军团的末日爸爸与影魔导演关系不和，因此想通过贿赂末日爸爸，让其击杀影魔。小冰冰的人际关系有限，希望通过打通一层层关系收买到末日爸爸，以下是人际关系图：



每人向自己能联系到的人雇佣所需要的钱如图所示（括号内为雇佣所需金钱：

小冰冰 火女 MM (12) 卡尔哥哥 (2)

火女 MM 女王大人 (5)

卡尔哥哥 血魔叔叔 (4) 火女 MM (8)

血魔叔叔 火女 MM (3) 女王大人 (13) 末日爸爸 (15)

女王大人 末日爸爸 (2)

小冰冰希望通过最少的钱收买到末日爸爸，请设计一个算法，在  $O(n^2)$  策划出收买方案( $n$  为所牵涉到的人数，这里为 6)。

解：采用 DIJKSTRA 算法，最惠收买方案为

小冰冰——卡尔哥哥——血魔叔叔——火女 MM——女王大人——末日爸爸。

## 18 徐熠琳

问题：将  $n$  个正整数存放于一个一维数组  $a$  中，试设计一个算法，将所有的奇数移动并存放于数组的前半部分，将所有的偶数移动并存放于数组的后半部分，要求尽可能少地使用临时存储单元，并使计算时间达到  $O(n)$ 。

解答：头尾两个指针不断往中间遍历，若从前往后遍历的指针碰到偶数，便与从后往前的指针碰到的奇数交换。

## 19 姜德志

无向图  $G$  中有  $N$  个顶点，并通过一些边相连接，边的权值均为正数。初始时你身上有  $M$  元，当走过  $i$  点时，需要支付  $S(i)$  元，如果支付不起表示不能通过。请找出顶点 1 到顶点  $N$  的最短路径。如果不存在则返回一个特殊值，如果存在多条则返回最廉价的一条。限制条件：  $1 < N \leq 100$ ;  $0 \leq M \leq 100$  ; 对任意  $i$ ,  $0 \leq S[i] \leq 100$ 。

解法：

如果不考虑经过顶点时的花费，这就简化成了一个一般的两点间最短路径问题，可以用 Dijkstra 算法求解。加入了花费限制之后，就不能直接求解了。

考察从顶点 0 到达顶点 i 的不同状态，会发现它们之间的区别是：总花费相同但路径长度不同、总花费不同但路径长度不同。为了寻找最短路径，必然要保存到达 i 点的最短路径；同时为了找到最小开销，应该把到达 i 点的开销也进行保存。根据题目的数值限制，可以将总开销作为到达顶点 i 的一个状态区分。这样，就可以把  $\text{Min}[i][j]$  表示为到达顶点 i（并经过付钱）时还剩余 j 元钱的最短路径的长度。在此基础上修改 Dijkstra 算法，使其能够保存到达同一点不同花费时的最短长度，最终的  $\text{Min}[N-1][0...M]$  中最小的即为所求。以下是求解过程的伪代码。

```
//初始化
对所有的(i,j),  $\text{Min}[i][j] = \infty$ ,  $\text{state}[i][j] = \text{unvisited}$ ;
 $\text{Min}[0][M] = 0$ ;

while(1) {
    for 所有 unvisited 的(i,j)找出  $\text{Min}[i][j]$ 最小的，记为(k,l)
    if  $\text{Min}[k][l] = \infty$ 
        break;

     $\text{state}[k][l] = \text{visited}$ ;
    for 所有顶点 k 的邻接点 p
        if ( $1-S[p] \geq 0$  &&  $\text{Min}[p][1-S[p]] > \text{Min}[k][l] + \text{Dist}[k][p]$ )
             $\text{Min}[p][1-S[p]] = \text{Min}[k][l] + \text{Dist}[k][p]$ ;

    //通过 Dijkstra 算法寻找不同花费下的最小路径
}

for 所有 j，找出  $\text{Min}[N-1][j]$ 最小的
    如果存在多个 j，那么选出其中 j 最大的
```

## 20 邹赫

给出一个  $O(N)$  时间的算法来计算一个有向图  $G(V,E)$  的传递闭包。

```
for i ← 1 to n do
  for j ← 1 to n do
     $t_{ij} \leftarrow 0$ 
  for each vertex i of V do
    run breadth-first search
     $t_{ii} \leftarrow 1$ 
  for each vertex j with color black do
     $t_{ij} \leftarrow 1$ 
```

## 21 胡轶文

考虑下面特殊的整数线性规划问题

$$\begin{aligned} & n \\ \max \sum_{i=1}^n c_i x_i \end{aligned}$$

$$\begin{aligned} & n \\ \sum_{i=1}^n a_i x_i & \leq b, \quad x \in \{0,1,2\}, \quad 1 \leq i \leq n \end{aligned}$$

试设计一个解此问题的动态规划算法.

解：该问题是一般情况下的背包问题，具有最优子结构性质。

$$\begin{aligned} & i \\ \text{Max } & \sum_{k=1}^i c_k x_k \\ & K=1 \end{aligned}$$

$$\begin{aligned} & i \\ & \sum_{k=1}^i a_k x_k \leq j \\ & K=1 \end{aligned}$$

设所给背包问题的子问题的最优值为  $m(i,j)$ , 即  $m(i,j)$  是背包容量为  $j$ , 可选物品为  $1, 2, \dots, i$  时背包问题的最优解。由背包问题的最优子结构性质, 可建立计算  $m(i,j)$  的递归式如下:

$$m(i,j) = \begin{cases} \max\{m(i-1,j), m(i,j-a_i) + c_i\} & a_i \leq j \\ m(i-1,j) & 0 \leq j < a_i \end{cases}$$

$$m(0,j) = m(i,0) = 0; m(i,j) = -\infty, j < 0$$

按此递归式计算出来的  $m(n,b)$  为最优值, 算法所需的计算时间为  $O(nb)$ 。

## 22 刘串

有一个大小为  $m \times n$  的矩阵, 矩阵中每个格子内有一个整数 (其中  $(1,1)$  和  $(m,n)$  处为 0), 现在请找出从  $(1,1)$  到  $(m,n)$  的两条路径 (只能向下或向右), 使得两条路径上的整数之和最大。  
(格子内的整数取过一次后变为 0)

Input:

第一行:  $m \ n$

以下  $m$  行每行  $n$  个整数

Output:

两条路径的最大和

Solution:

$f[p][x1][x2]$ ,  $p$  表示经过的格子数. 整个表示经过  $p$  步后, 两条路径分别到达横坐标为  $x1$ ,  $x2$  时取得的最大值。

$$f[p][x1][x2] = \max \{ f[p-1][x1][x2], f[p-1][x1][x2-1], f[p-1][x1-1][x2], f[p-1][x1-1][x2-1] \} + v[x1, p+1-x1] + v[x2, p+1-x2]$$

用三重循环解出  $f[m+n-1][m][n]$  即可

### 23 吴佳俊

小明家离学校  $k$  公里, 设小明家到学校公交车费用  $C_i$  ( $i$  为公里数), 求一个方案使得小明到学校的车费最少。

解答: 同背包问题

### 24 杨朔

设计一个算法, 找出在一个有向图中的最短回路, 并找出其长度, 如果是有向无回路图则输出 0。(图中每一条边的权重设为 1, 最短路即通过边最少的路径) 分析算法的时间复杂度。

参考答案:

用广度优先搜索, 首先每一个点都标记为未到达, 每次遍历到一个点, 如果此点未到达过, 则标记未到达, 到达过记为回路(第一次找到的有标记过的点即为最短的回路), 如果搜到了出度为 0 的点则结束。如果所有的都遍历结束后仍没有回路, 则此图为无回路图, 输出 0。



## 25 徐宇

问题：一群男孩和女孩共  $N$  人，男孩  $a$  人，女孩  $b$  人。某些男孩和女孩之间会发生恋爱关系（满足一定的关系），现在希望找到最多的孩子，他们之间不会发生恋爱关系。

分析：找到最多的孩子，没有恋爱关系，这实质上是找最大独立集。男孩在左有  $a$  个，女孩在右有  $b$  个，那么如果某男孩和某女孩之间有关系，就连线。最大独立集就是找到最多顶点，顶点之间没有联系，正好就是所求，而最大独立集就是  $N$ -最大匹配，所以问题得到解决。试想，如果二分图中没有连线，那么所有的孩子都可选，最大独立集也是  $N$ ，他们是等价的。如果存在一条连线，那么去掉一个孩子就是所找的孩子，最大独立集此时是  $N-1$ 。依次类推。在试想，最大匹配其实就找到了几对恋爱对象，假设是这样的：

$a \quad b$

$B_0 \quad G_0$

$B_1 \quad G_1$

... ..

$B_i \quad G_i$

... ..

我们只需要把他们的另一半去掉，就是我们找的孩子。不过会有这样的疑问，如果我取出了  $B_1$  的另一半  $G_1$ ， $B_1$  会不会和其余的孩子恋爱呢，比如说  $B_1$  和  $M$  会恋爱（ $M$  没有出现在选中的  $b$  集合中），那么好吧，去掉  $G_1$  的另一半  $B_1$ ，保留  $G_1$ ，这样就不会有问题了吧。还有担心？ $G_1$  会不会和其余的孩子恋爱呢，比如说  $G_1$  和  $N$  会恋爱（ $N$  没有出现在选中的  $a$  集合中），不过这样的情况是不会出现的。假如  $G_1$  和  $N$  好， $B_1$  和  $M$  好，那么最大匹配中出现的是两条边  $G_1 \text{---} N$ ， $B_1 \text{---} M$ ，而不是现在的  $B_1$  和  $G_1$ 。所以，既然最大匹配中选择了  $B_1$  和  $G_1$ ，去掉他们中间的一个肯定是可行的。所以答案就是  $N$ —最大匹配了。

## 26 付周望

题目：求一个字符序列最少要增加几个字母才能变成回文序列？

解：设原字符串为  $S$ ， $S'$  为其反串，可以通过求  $S$  和  $S'$  的最长公共子序列，然后用长度减去最长公共子序列的长度即可以求得需要添加的字母个数。

算法：

Input: String  $S$

Output: num of letter that make S a palindromic string

```
REVERSE(String s)
```

```
String result = "";
```

```
for (int i = s.length() - 1; i >= 0; i --) {
```

```
    result += s.charAt(i);
```

```
}
```

```
return result;
```

```
PALINDROMIC(String s)
```

```
String s2 = REVERSE(s);
```

```
for i = 0 to s.length() {
```

```
    L[i, 0] = 0;
```

```
}
```

```
for i = 0 to s.length() {
```

```
    L[0, i] = 0;
```

```
}
```

```
for i = 1 to s.length() {
```

```
    for j = 1 to s.length() {
```

```
        If (s[i] == s2[j]) {
```

```
            L[i,j] = L[i-1, j-1] + 1;
```

```
        }
```

```
        else
```

```
            L[i,j] = max {L[i-1, j], L[i,j-1]};
```

```
    }
```

```
}
```

```
return s.length()-L[s.length(), s.length()];
```

## 27 陈荣

### Subset Sum Problem (Pro)

#### Problem Description

有  $n$  个正整数， $a[1]...a[n]$ 。平均分到两个集合，使之和相等。问有多少种分法？给出一个算法来解决这个问题。

例如：1, 2, 3, 4, 5, 6, 7 的划分方案有4种：

$\{1,6,7\}$  与  $\{2,3,4,5\}$

$\{2,5,7\}$  与  $\{1,3,4,6\}$

$\{3,4,7\}$  与  $\{1,2,5,6\}$

$\{1,2,4,7\}$  与  $\{3,5,6\}$

#### Solution:

和容易联想到 Subset Sum Problem，回顾一下这个问题的 DP 状态转移方程：

$L[I, j] = 0$	..... $i == 0$
$L[I, j] = 1$	..... $j == 0$
$L[I, j] = L[I - 1, j]$	..... $i > 0, j > 0, j < a[i]$
$L[I, j] = L[I - 1, j - a[i]] \text{ or } L[I - 1, j]$	..... $i > 0, j > 0, j \geq$

$a[i]$

其中  $L[I, j]$  表示是否前  $i$  个数存在和为  $j$  的情况。

不过这道题不是问是否存在解，而是问存在多少个这样的解。状态转移方程：

for  $i = 1$  to  $n$

for  $j = \text{sum}$  downto  $a[i]$

$f[j] = f[j] + f[j - a[i]]$

这个迭代可以用  $f[i, j]$  来写，表示前  $i$  个数使和为  $j$  有多少种方法，对于空间来说可以优化成一维的（回顾一下 XJ 老师在说背包问题空间优化的方法）

$f[j]$  表示对于一个集合  $S$  来说，使之和为  $j$  有多少种方法

边界:  $f[0] = 1$ ,  $sum = a[1] + \dots + a[n]$

最后, 问题的答案:  $(f[sum / 2] / 2)$ ,  $sum/2$  表示平分了这一堆的数,  $f / 2$  是因为集合划分的时候重复计算了, 需要把方法数除以 2。

## 28 李正坤

一种新型的防卫导弹可截击多个攻击导弹。它可以向前飞行, 也可以用很快的速度向下飞行, 可以毫无损伤地截击进攻导弹, 但不可以向后或向上飞行。但有一个缺点, 尽管它发射时可以达到任意高度, 但它只能截击比它上次截击导弹时所处高度低或者高度相同的导弹。现对这种新型 防卫导弹进行测试, 在每一次测试中, 发射一系列的测试导弹 (这些导弹发射的间隔时间固定, 飞行速度相同), 该防卫导弹所能获得的信息包括各进攻导弹的高度, 以及它们发射次序。现要求一个动态规划算法, 求在每次测试中, 该防卫导弹最多能截击的进攻导弹数量。

输入: 按进攻次序排列的进攻导弹的高度  $H[n]$ 。

输出: 最多能够截击的导弹数。

Solution:

另  $F[i]$  为若截击第  $i$  个导弹则前  $i$  个导弹中最多能截击多少个。

则:

$F[i] =$

- 1、1            if for each  $j = 1$  to  $i-1$ ,  $H[j] > H[i]$
- 2、 $\max\{f[j]\} + 1$    where  $1 \leq j < i$ ,  $H[j] < H[i]$

## 29 何冲

犹豫的小官

小官是上海交大软院第一男神, 人生至高的追求就是各种女神。然而, 他掐指一算, 最近将有  $n$  个女神出现在他的身边。小官辛辛苦苦将各个女生的信息调查到手后 (在此感

谢他的网管室友），得知他能在  $T_i$  时刻遇到第  $i$  个女神（错过的话将不再遇到），而追第  $i$  个女神需要的时间为  $D_i$ ，追到第  $i$  个女神的幸福感为  $H_i$ 。小官不是一般的男神，他既专一又花心。专一是指他在追一个女神的时候就不会追求其他的人，而花心则是指他把女神追到手之后便会失去兴趣，转向下一个目标。但面对眼前这么多目标，小官觉得十分犹豫，无从下手。所以他请求你帮他设计一个算法，算出他接下来的日子能得到的最大幸福感是多少。

解答：

用  $F[t]$  表示  $t$  时刻小官能获得的最大幸福感，可以得出递推式

如果存在  $i$  是的  $T_i + D_i = t$ ，则  $a = \text{Max}\{F[T_i] + H_i\}$  对于所有符合  $T_i + D_i = t$  的  $i$ ，否则  $a = 0$

$F[t] = \text{Max}\{F[t-1], a\}$

其意义为，当前  $t$  的幸福感到  $a$ 。选择追求能在今天追到的某个女神加上追之前的幸福感的和  $b$  不追 两个选择中较优的按个。

然后根据递推式动态规划求解，最后  $F[\text{max}(T_i + D_i)]$  中即是最大的幸福感

### 30 王鹏

#### 算法设计题

##### 【题目描述】

佳儿爷爷经常给她讲故事，某天就讲了一个采摘西瓜的故事（因为她闹着要买...）。某年某村的瓜农把  $n$  个西瓜放在像一条直线的水库大坝上，叫本村的小朋友去大坝搬西瓜，谁的西瓜搬走得多，谁就是胜者。搬西瓜必须遵守的原则是：西瓜一个一个搬，可以从任何位置开始搬运，按西瓜所摆放的位置,只能往后取西瓜,取走的西瓜重量不得大于前面已经搬走西瓜的重量（除第一个西瓜）。请设计一个算法，计算出做多一次能搬走的西瓜个数。

##### 【pascal 代码】

```
var  
    n:longint;
```

```

data,f:array[0..10001] of longint;

procedure init;
var i:longint;
begin
  readln(n);
  for i:=1 to n do
    read(data[i]);
end;

procedure main;
var i,j,max:longint;
begin
  for i:=1 to n do
    f[i]:=1;
  for i:=n-1 downto 1 do      #计算倒数 n-i 个西瓜能取走的最大数量
    for j:=i+1 to n do
      if (data[i]>=data[j]) and (f[j]+1>f[i])
        then f[i]:=f[j]+1;
  max:=0;
  for i:=1 to n do
    if f[i]>max then max:=f[i];
  writeln(max);
end;

begin
  init;
  main;
end.

```

### 31 陈彦纶

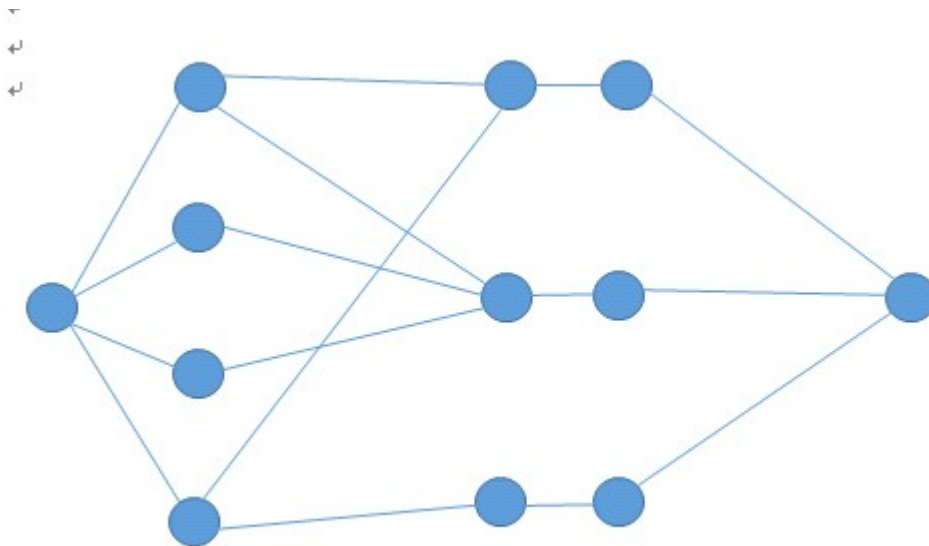
洗澡问题：

现在有  $n$  个同学去浴室洗澡，浴室有  $m$  个隔间。每个洗澡的同学都是有原则的同学，他们只会选择在自己喜欢的一个或多个隔间里洗澡。请问现在最多有多少同学可以同时洗到澡？

解答：

这是个典型的网络流问题。可将每个同学抽象成一个与源相连的节点，每个浴室抽象成中间有一条容量为 1 的边的两个节点，右边的节点与汇相连。对于 A 同学喜欢在 x 隔间洗澡，则从 A 连一条边到 x 的左节点。

接下找出最大流即是问题的解。



### 32 孟祥来

代码合并

某学渣写代码的时候总是习惯先按照功能把代码一段一段写好，再将代码按次序合并到一起。但是由于学渣的编程习惯不好，代码合并十分的困难：每次他将相邻两段长度分别为  $m$  行和  $n$  行的代码合并到一起的时候，都要花  $m+n$  分钟来 debug（果然 debug 最痛苦），并得到一段新的长度为  $m+n$  行的代码。假设学渣一共写了编号为  $C_1 \dots C_n$  的  $n$  段代码（其中第  $i$  段代码和第  $i-1$  行和第  $i+1$  行代码相邻， $1 < i < n$ ），它们的长度分别为  $L_1 \dots L_n$ 。请设计一个算法，计算该学渣要把这些代码合并到一起最少需要多长时间来 debug。

输入：

$L_1 L_2 L_3 \dots L_n$

输出：

最短必要 debug 时间

例：

输入：1 3 1 2 3 5

输出：37

思路：

$f[i, j]$  代表将  $C_i \dots C_j$  合并到一起所需的最小时间

$s[i, j]$  代表  $L_i + L_{i+1} + L_{i+2} + \dots + L_j$

则：

$f[i, j] = \min(f[i, k] + f[k, j] + s[i, j]) \quad (1 < i < k < j < n)$

$f[i, i] = 0 \quad (1 \leq i \leq n)$

循环计算即可

算法复杂度为  $O(n^3)$



给出一个有向图  $G(V,E)$ ，地图上有  $n$  个点， $m$  条路，同时每条路有权重  $d[1]...d[m]$ ，地图上有一个宝物，位置为  $C$ 。同时，地图上有  $p$  个警察，位置分别为  $a[1]...a[p]$ ， $q$  个小偷，位置分别为  $b[1]...b[q]$ 。假设宝物出现后，警察与小偷同时向宝物出发，警察只走大路（马路），即每条路的长度都不小于  $S$ ，而小偷只走小路(下水道、通风口)，即路的长度都小于  $S$ 。警察事先知道宝物会出现，所以在小路上设置路障，但由于预算不足，只能在长度小于  $X$  的路上设置路障。

给出  $G(V,E)$ ,  $d[]$ ,  $C$ ,  $p$ ,  $q$ ,  $a[]$ ,  $b[]$ ,  $S$

求：最小的  $X$  使得存在警察比小偷先找到宝藏。

Hint.  $O(N*N*\log S)$

Input:

第 1 行:  $n, m, p, q, C, S$

第 2 至  $m+1$  行: 每行 3 个数  $h, t, d$  表示  $h$  到  $t$  有一条长度为  $d$  的路

第  $m+2$  至  $m+p+1$  行: 每行 1 个数  $a$  表示  $a$  点有一个警察

第  $m+p+1$  至  $m+p+q+1$  行: 每行 1 个数  $b$  表示  $b$  点有一个小偷

Output:

1 个数:  $X$

解: 1 首先对  $G$  求反向图  $G'$

2 接下来都在  $G'$  上操作:

3 首先以  $C$  为起点求全图最短路，找到  $C$  到  $a[]$  的最短路中最小的值记为  $mind$

4 之后在  $[0,S]$  区间内二分枚举  $X$

5 每次确定  $X$  后，删除小于  $X$  的边记为  $G''$

6 在  $G''$  求最短路，若从  $C$  到  $b[]$  的最短路中存在小于  $S$  的值，则增大  $X$ ，返回 5，否则

缩小  $X$ ，返回 5。若  $X$  确定，且无法增大或减小，且  $X$  小于  $S$ ，则  $X$  即为解，否则无解。

时间复杂度  $O(n*n*\log S)$

### 34 刘已豪

在很久很久很久以前，有一只青蛙要过河，河面上排满了石子。青蛙一次可以跳跃过  $\leq k$  个石子的地方，但不能回头。已知河面总宽有  $n$  个石子的宽度，但有  $m$  个位置的石子是活动的不能到达，分别在离出发点  $A[1\dots m]$  的地方，那么这只坑爹的青蛙有多少种过河的方式？

参考答案：

一道基本的 DP 题目

```
s[-k..n] = 0;
b[1..n] = true;
For i = 1 to m
    b[A[i]] = false;
For i = 1 to n
    if b[i] then
        for j = 1 to k
            s[i] += s[i-j]
        endfor
    endif
endfor
print s[n]
```

### 35 华志超

金明搬了新家，妈妈对他说：“你的房间需要购买哪些物品，怎么布置，你说了算，只要不超过  $N$  元钱就行”。今天一早，金明就开始做预算了，他把想买的物品分为两类：主件与附件，附件是从属于某个主件的，例：鼠标从属于电脑

如果要买归类为附件的物品，必须先买该附件所属的主件。每个主件可以有 0 个、1 个或 2 个附件。附件不再有从属于自己的附件。金明想买的东西很多，肯定会超过妈妈限定的 N 元。于是，他把每件物品规定了一个重要度，分为 5 等：用整数 1~5 表示，第 5 等最重要。他还从因特网上查到了每件物品的价格（都是 10 元的整数倍）。他希望在不超过 N 元（可以等于 N 元）的前提下，使每件物品的价格与重要度的乘积的总和最大。

请你帮助金明设计一个满足要求的购物单。

答：考虑到每个主件最多只有两个附件，因此我们可以通过转化，把原问题转化为 01 背包问题来解决，在用 01 背包之前我们需要对输入数据进行处理，把每一种物品归类，即：把每一个主件和它的附件看作一类物品。处理好之后，我们就可以使用 01 背包算法了。在取某件物

品时，我们只需要从以下四种方案中取最大的那种方案：只取主件、取主件+附件1、取主件+附件2、既主件+附件1+附件2。很容易得到如下状态转移方程：

$$\begin{aligned} f[i,j] &= \max \{ f[i-1,j], \\ f[i-1,j-a[i,0]]+a[i,0]*b[i,0], \\ f[i-1,j-a[i,0]-a[i,1]]+a[i,0]*b[i,0]+a[i,1]*b[i,1], \\ f[i-1,j-a[i,0]-a[i,2]]+a[i,0]*b[i,0]+a[i,2]*b[i,2], \\ f[i-1,j-a[i,0]-a[i,1]-a[i,2]]+a[i,0]*b[i,0]+a[i,1]*b[i,1]+a[i,2]*b[i,2] \} \end{aligned}$$

其中， $f[i,j]$ 表示用 j 元钱，买前 i 类物品，所得的最大值， $a[i,0]$ 表示第 i 类物品主件的价格， $a[i,1]$ 表示第 i 类物品第 1 个附件的价格， $a[i,2]$ 表示第 i 类物品第 2 个附件的价格， $b[i,0], b[i,1], b[i,2]$ 分别表示主件、第 1 个附件和第 2 个附件的重要度。 $f[i-1,j]$ 表示把 j 元钱全部投入前 i-1 类物品所得的最大值，即不取第 i 类物品这一方案， $f[i-1,j-a[i,0]]+a[i,0]*b[i,0]$ 表示只取第 i 类物品的主件这一方案， $f[i-1,j-a[i,0]-a[i,1]]+a[i,0]*b[i,0]+a[i,1]*b[i,1]$ ，表示取第 i 类物品的主件和第 1 个附件这一方案， $f[i-1,j-a[i,0]-a[i,2]]+a[i,0]*b[i,0]+a[i,2]*b[i,2]$ ，表示取第 i 类物品的主件和第 2 个附件一方案， $f[i-1,j-a[i,0]-a[i,1]-a[i,2]]+a[i,0]*b[i,0]+a[i,1]*b[i,1]+a[i,2]*b[i,2]$ ，则表示取第 i 类物品的主件和两个附件这一方案。

### 36 祁磊

#### 题目

因政策制订需要，现国家卫计委需要根据各国历史各年份的人口数据绘制一种全新的“人口趋势图”。这种人口-年份图的要求如下：

1. 在人口到达峰值前，剔除部分数据点使得绘制出的人口-年份图保持单调上升(包括持平)趋势；在人口到达峰值后，剔除部分数据点使得绘制出的人口-年份保持单调下降(包括降低)趋势。

2. 峰值点可以任选，但是剔除点的数量应为理论的最小情况。

现在卫计委委托交大软件学院来设计一个解决方案，你需要设计一个算法，在  $n^2$  的时间内找到保留下来的数据。

答案：

这是最长上升和最长下降子序列问题。正向求一遍最长不降子序列，再逆序求一遍最长不增子序列。再把各个位数的正序和逆序的最长不增子序列相加-1，找出最大值即可找到最大数。在找最大数的过程中，我们可以通过一个指向前一个数的指针记录下所有数据。由于年份是递增的，我们还可以把年份抽象成顺序增长的数。

### 37 俞飞樾

1. 给出随机获得整数 1 到  $m$  的函数， $m > 1$ ，求随机获得整数 1 到  $n$  的函数， $n > m$ 。（归约）

2. Jason 喜欢玩打砖块游戏，一开始有  $n$  行  $m$  列砖块，Jason 有  $k$  发子弹，打掉一个砖块有对应的得分，有一些砖块在打碎后，可以奖励一发子弹。Jason 知道每一块砖块的分数，以及是否有奖励子弹，求 Jason 的最大得分

解答：

动态规划

(1) 如果没有奖励子弹的话，比较简单

$b[i][j]$  表示对着前  $i$  列打  $j$  发子弹的最大得分

$s[i][k]$  表示对第  $i$  列打  $k$  发子弹的得分

$b[i][j] = \max \{b[i-1][j-k] + s[i][k]\}, (0 \leq k \leq j, k \leq n)$

## (2)有奖励子弹

有奖励子弹的砖块可以理解为不用耗费子弹，就可以打,这个当然要打，但是如果现在手里已经没有子弹了，那么不能打

$b1[i][j]$ 表示对着前  $i$  列打  $j$  发子弹且最后一发子弹不是对着前  $i$  列发的最大得分

$s1[i][k]$ 表示对第  $i$  列打  $k$  发子弹，最后一发子弹还没打的得分

$b2[i][j]$ 表示对着前  $i$  列打  $j$  发子弹且最后一发子弹对着前  $i$  列发的最大得分

$s2[i][k]$ 表示对第  $i$  列打  $k$  发子弹，最后一发子弹已用掉的得分

$b1[i][j]=\max\{b1[i-1][j-k]+s1[i][k]\}, (0\leq k\leq j, k\leq n)$

$b2[i][j]=\max\{b1[i-1][j-k]+s2[i][k], b2[i-1][j-k]+s1[i][k]\}, (0\leq k\leq j, k\leq n)$

最大得分  $b2[m][k]$

## 38 张一鸣

有一组整数数组  $a[n]$ ，请设计一个算法求出数组元素之间差值的绝对值的最小值，并给出平均时间复杂度。（遍历所有数据的差不得分）

解答：可以先利用快排对数组进行排序，再遍历相邻数据的差值的绝对值，求出最小值。  
平均时间复杂度为  $O(n\log n + 3(n-1)/2)$

## 39 魏剑辉

问题描述□

设有  $n$  个程序  $\{1, 2, \dots, n\}$  要存放在长度为  $L$  的磁带上。程序  $i$  存放在磁带上的长度是  $l_i (1 \leq i \leq n)$ 。程序存储问题要求确定这  $n$  个程序在磁带上的一个存储方案，使得能够在磁带上存储尽可能多的程序。

编程任务□

对于给定的  $n$  个程序存放在磁带上的长度，编程计算磁带上最多可以存储的程序数。

数据输入：

第一行是 2 个正整数，分别表示文件个数  $n$  和磁带的长度  $L$ 。接下来的 1 行中，有  $n$  个正整数，表示程序存放在磁带上的长度。

结果输出：

最多可以存储的程序数。

输入示例：

6 50

2 3 13 8 80 20

输出示例：

5

参考答案：

```
int greedy( vector<int> x, int m){
    int i=0, sum=0, n=x.size();
    sort(x.begin(),x.end());
    while(i < n){
        sum += x[i];
        if(sum <= m) i++;
        else return i;
    }
    return n;          //所有的程序都没有磁带长
}
```

小明和小红比赛心算，小红报一个数，小明就要在一堆数中找到几个数令它们加起来的和为小红说报的数，为此小明需要先判断有没有这么几个数满足条件，请你给出一个算法帮助小明判断能否找到这几个数。

---

这就是个典型的 Subset sum problem 用动态规划可以很好的解决这个问题。

假设小红报出的数为 sum，一堆数都在集合  $S=\{a_1, a_2, \dots, a_i\}$  中。那么就是判断能否在集合 S 中找到一个子集使得该子集的和为 sum。

**分析：**

假设  $L(i, j)$  为从集合  $S=\{a_1, a_2, \dots, a_i\}$  中能否找到一个子集使得该子集的和等于 j。

那么

$$L[i, j] = \begin{cases} 0 & \text{if } i = 0 \\ 1 & \text{if } j = 0 \\ L[i-1, j] & \text{if } i > 0, j > 0, j < a_i \\ L[i-1, j-a_i] \vee L[i-1, j] & \text{if } i > 0, j > 0, j \geq a_i \end{cases}$$

**代码：**

//假设所有的集合元素都在数组 a[N]中

```
bool SubSetSum_Equal(int a[N])
```

```
{
```

```
int L[N][sum];
```

```
memset(L, 0, sizeof(L));
```

```
for(int i=0 ; i<N ; i++)
```

```
    L[i][0] = 1;
```

```
for(int i=1 ; i<N ; i++)
```

```
{
```

```
    for(int j=sum ; j>a[i] ; j--)
```

```
    {
```

```
        L[i][j] = L[i-1][j-a[i]] || L[i-1][j] ;
```

```
    }
```

```
for(j; j>0 ; j--)
```

```

        {
            L[i][j] = L[i-1][j];
        }
    }
return L[N][sum];
}

```

最后的函数返回 1 则为找得到；返回 0 则为找不到。

#### 41 刘聪

算法设计题：

今有  $k$  个质量均为正整数的砝码和一台天平（天平左右均可以放砝码），对于一个输入的整数  $n$ ，

- (1) 请设计一个算法判断是否能够用所给砝码和天平测出质量为  $n$  的物品重量？
- (2) 如果要能测出质量为 1、2...、 $n$  的所有物品质量，那么  $k$  的最小值应为多少（用  $n$  表示）？
- (3) 当  $n$  满足什么条件时，使得 (2) 成立的  $k$  个砝码有唯一解？

SOL: (1) 可以对  $k$  个砝码进行加权计算

$$W = \sum l_i w[i] \text{ 其中 } w[i] \text{ 为第 } i \text{ 个砝码的质量, } l_i \in \{-1, 0, 1\}$$

$W$  的取值最多有  $3^k$  个，用遍历方法最差复杂度  $O(3^k)$

(2) 不难看出如果  $W$  能表示  $i$ ，则也能表示  $-i$ ，

$$\text{所以有 } (3^{k-1} - 1)/2 < n \leq (3^k - 1)/2$$

$$\text{得到 } k \geq \log_3(2n + 1)$$

实际上 我们也可以给出使得  $\log_3(2n+1)$  取上整的构造 (EX: 1, 3, 9, 27, ...)

(3) 不必证明，根据 (2) 中给出的构造我们知道，

使得不等式  $k \geq \log_3(2n + 1)$  取等号时，即

$$n = 3^{k-1} - 1/2 \text{ 时, 有唯一解 (如构造)}$$



## 42 邱喆

僵尸大战植物

=====

Qiu Zhe

Sophie 住在 California，她有一个正方形的花园。花园分为  $n * n$  个小正方形，共有  $n$  行，每行  $n$  个。Sophie 说，要有植物，便有了植物。

Sophie 觉得只有植物太单调了，就招了个僵尸 Ted 陪植物玩。于是有一天，Sophie 站在第  $n-1$  行第  $n-1$  列的小正方形里，僵尸站在第 0 行第 0 列。Sophie 让

僵尸从第 0 行第 0 列走到她所在的那个正方形，然后再走回去第 0 行第 0 列。

僵尸说，我不能白走。Sophie 觉得有道理，就让僵尸在路上可以采植物。僵尸走到任何一个正方形内，都可以采走里面的植物。但是，植物只能采一次，而且

每次必须取空整个方格。所以下次经过那个格子的时候就没有植物可以采。第  $i$  行第  $j$  列的植物有一个价值  $v[i,j]$ ， $v[i,j]$  是一个整数（不保证非负）。

因为僵尸 Ted 成为僵尸的原因，就是还不是僵尸的时候头脑就被另一只僵尸吃掉了，所以 Ted 每一步走都不会兜远路。具体来说，当 Ted 从第 0 行第 0 列走向 Sophie 的时候，

每一步只能向右（列数+1）或者向下（行数+1）移动，此外没有别的选择。当 Ted 返回出发点时，每一步只能向左或向上，依此类推。

Ted 想知道最多能收集到多少价值的植物呢？请设计一个尽可能高效的 DP 算法，试分析其时间复杂度。

Solution:

$f[i,j,k] = \max(f[i-1,j-1,k-1], f[i-1,j-1,k], f[i-1,j,k], f[i-1,j,k-1]) + v[j,i-j] + v[k,i-k]$  ( $0 < i \leq 2n-2$  &&  $j \neq k$ ,  $0 \leq j, k < n$ )

$= \max(f[i-1,j-1,k-1], f[i-1,j-1,k], f[i-1,j,k], f[i-1,j,k-1]) + v[j,i-j]$   
 ( $0 < i \leq 2n-2$  &&  $j = k$ ,  $0 \leq j, k < n$ )

$= 0$  ( $i = 0$  &&  $j = 0$  &&  $k = 0$ )

$= -\infty$  otherwise

Final Answer  $f[2n, n-1, n-1]$

$O(n^3)$

提示:

来回走等价于两个僵尸同时从左上走向右下。 $f[i,j,k]$ 表示走  $i$  步，两只僵尸分别在第  $j$  行和第  $k$  行时，最多能获得的价值。

答出  $f[i,j,k,l]$ 表示两个僵尸分别走到 $(i,j)$ 和 $(k,l)$ 并得到  $O(n^4)$ 的获 80%的分数。

### 43 吴明瑜

设计题:

只给一个数组  $a$ (大小为  $n$ )，用户不断输入  $i,j$  以获取从  $a[i]$ 到  $a[j]$ 的区间和，要求在多次询问时的平均时间复杂度达到  $O(1)$ 。

解答：首先进行预处理：

for  $i = 0$  to  $n-1$  do  $a[i+1] = a[i] + a[i+1]$ ;

此后根据每次询问，返回  $a[j] - a[i-1]$ 即可。

(由于预处理耗费  $O(n)$ ，因此要求多次询问时的平均时间复杂度达到  $O(1)$ )

#### 44 程治谦

给定一个存放整数的数组  $a$ ，大小为  $n$ ，重新排列数组使得数组左边为偶数，右边为奇数。

要求：空间复杂度  $O(1)$ ，时间复杂度为  $O(n)$ 。

答案：

```
void group_oddeven()
{
    int i = 0, j = n-1;
    int buf = 0;
    while (i < j) {
        if (!(a[i]%2))
            { i++; continue; }
        if (a[j]%2)
            { j--; continue; }

        swap(a[i++], a[j--]);
    }
}
```

#### 45 张翱

某班学生男女各有  $n$  人，每天早晨都要进行晨练排队，男生一行女生一行，前后左右对齐。假定从左到右第  $i$  位男生身高为  $a_i$ ，女生为  $b_i$  ( $i=1,2,\dots,n$ )，定义

$$S = \sum_{i=1}^n (a_i - b_i)^2$$

为该队列的“男女高度差”。现需要使用同列男女两两交换的方式更改队列编排，使得  $S$  最小，请给出一个时间复杂度  $O(n \log n)$  的算法求出最小的交换次数。

简解：使用贪心策略，不难证明仅当  $a_i$ ， $b_i$  都按同序排列时  $S$  最大，接下来只要求得  $b_i$  中所有的逆序对数即可，参见 homework 解答。

#### 46 刘玺炜

右手同学上大学了，每周周一至周五会有不同的课程，但是右手又是一个贪玩的孩子，喜欢玩电脑，所以如果右手每天上课时间超过 6 小时就会不高兴，而且上的越久越不高兴。假设右手不会其他原因不高兴，而且他的不高兴不会持续到第二天。现在请你帮忙检查一下右手下周的日程安排 看看他下周会不会不高兴，如果会的话，哪天最不高兴。

**【输入文件】** 输入文件 `unhappy.in` 包括 5 行数据，分别表示周 1 到周 5 的课程安排。每行包括 3 个小于 10 的非负整数，用空格隔开，分别表示右手上课的上午时间和下午时间还有晚上时间。

**【输出文件】** 输出文件 `unhappy.out` 包括一行，这一行只包含一个数字。如果不会不高兴则输出 0，如果会则输出最不高兴的是周几（用 1, 2, 3, 4, 5, 分别表示周 1，周 2，周 3，周 4，周 6）。如果有两天或两天以上不高兴的程度相当，则输出时间最靠前的一天。

**【样例输入】**

4 3 0

3 2 0

4 2 2

3 3 1

2 4 1

0 4 2

0 4 1

**【样例输出】** 3

解答：很简单的贪心问题。输入的时候直接将 3 个数相加保留最大的即可，然后用一个变量 `ans` 记录哪天最不高兴即可。`ans` 初始化为 0，如果 3 个数相加大于 6 并且大于 `ans` 则更新 `ans` 为第几行就可 最后输出 `ans`。

#### 47 曹迪

Problem:

在数组中，数字减去它右边的数字得到一个数对之差。求所有数对之差的最大值。例如在数组 {2, 4, 1, 16, 7, 5, 11, 9} 中，数对之差的最大值是 11，是 16 减去 5 的结果。要求时间复杂度  $O(n)$ 。

Solution:

分治，递归分别得到左右子数组的最大数对之差，以及左子数组的 MAX 和右子数组的 MIN，那么最大数对之差是左右子数组的最大数对之差与 MAX-MIN 这三者的最大值。

$$T(N)=T(N/2)+1 \quad T(1)=1$$

$O(n)$ 。

别的  $O(n)$  算法也可以。

#### 48 李翔

题目：

一群人在玩猜数字的游戏。规则如下：一个人心里想一个数字让其他人猜，猜的人能试着猜测一个数字，回答大了、小了或者正确。问在以下两种情况下如何使猜测次数尽可能小：（1）已知数字范围（2）数字范围未知

参考答案：

- （1） 二分搜索
- （2） 第一次猜 2，第二次猜 4，第三次猜 8，第四次猜 16.....以此类推

#### 49 邱晓雷

盗圣小红有一个背包，容量为  $s$ ， $s$  为正整数。一天他摸进一个店，店里有  $n$  个货物，每个货物都有其占用背包的空间  $c$  和其价值  $v$ ， $c$  为正整数， $v$  为任意实数。小红为了向他老大水哥交差，必须使得他在店内偷并带走的所有货物的价值总乘积最大。请设计算法帮助小红完成他的任务。

解提示：基本动态规划，对于每个背包容量都记录绝对值最大的正数和负数结果，最后结果取结果为正数的方案。

## 50 郁翔

大家应该都玩过推箱子吧，在一张一格一格且只能上下左右移动的地图上，控制主角把箱子推到指定地点。

想要把箱子往某一个方向推的话，需要站在箱子后面一格，然后向前移动。

我们把问题简单化，只需要把一个箱子推到某一个指定地点即可，并且已知排除箱子的话，整张地图都是连通的。

还已知箱子能被推到终点的方法一定是有的。给定箱子初始的位置、目标地点的位置和主角初始的位置，用多项式时间求解主角把箱子推到目标地点的一条移动路线即可。（过程略微复杂，说明自己的算法即可，再慢也无所谓，只要是多项式时间）

解法之一（我自己的解法）：

第一步：

首先将所有非墙壁的格子位置记录下来，给与它们从 1~N 的编号。

第二步：

遍历所有这些位置，进行如下操作：把那个箱子放在这个位置上，记录下该位置的连通状态。

把这个位置作为不可通行，把剩下的地图作为一个无向图，每个格子作为一个节点，与其上下左右的可通行格子对应的节点之间建立权

值为 1 的边，找出连通分量。由于箱子只有一个，除箱子外整张地图都是联通的，方向只有 4 个，所以最多只能被分成 4 个连通分量。

将每个连通分量包含的节点记录在这个状态中，不到 4 个连通分量的话把后几个置为空。

第三步：

建立一个有向图，每个非墙壁的格子对应 4 个节点。先不建立边。

再次遍历所有这些位置，进行如下操作：试着将箱子往上、下、左、右移动一格，找出其对应的主角位置在当前状态的哪个连通分量中，

再找出移动后的主角位置处于移动后状态的哪个连通分量中，往有向图中添加一条边，从当前状态所对应格子的 4 个节点中主角处于的连通

分量所对应的节点，指向目标状态所对应格子的 4 个节点中主角在目标状态中处于的连通分量所对应的节点，权值都为 1。

具体来说，以试着向下推为例，如果该位置（即箱子在该状态下所处位置）下方一格都不是墙的话，找出其下方一格所对应的状态（

也就是箱子在下方一格时所对应的地图的连通分量的状态），然后找出上方一格（也就是主角要这么推需要站在的位置）当前状态和

目标状态中所对应的连通分量分别是第几个连通分量（1~4），然后再向图中添加上述那条有向边即可。

第四步：

找出玩家初始位置和箱子初始位置在有向图中对应的节点（1 个）。再找出目标位置对应的节点（最多 4 个）。

迪杰斯特拉寻路找出最短路径，即箱子可行的最短移动路径。

然后往结果路径里按这个规律添加：

对主角初始位置到第一次箱子移动前主角的位置在该状态对应的无向图中寻路寻得的路径。

第一次箱子移动所对应的主角移动路径。

第一次箱子移动后主角的位置到第二次箱子移动前主角的位置在该状态对应的无向图中寻路寻得的路径。

第二次箱子移动所对应的主角移动路径。

。。。。

最后一次箱子移动所对应的主角移动路径。

这样就搞定了。

稍微改一下应该也可以用多项式时间解决多个箱子的问题，判断是否无解也是可以的。寻找推多个箱子主角移动最短路径可能要多改一些吧。

## 51 张桢宇

临近期末，男神瑞哥准备复习应考。身为男神，瑞哥秉持着要么不复习，复习就要复习到满分的行事准则。但由于每门课的基础不同，每门课在不复习时的得分不同，故每门

课复习取得的成功也不同。已知瑞哥有  $N$  门考试，每门考试需要花费  $t_i$  时间复习，能够取得  $i_i$  分的进步，而瑞哥只剩下  $T$  的时间复习。请为设计算法，为瑞哥提供一个复习方案，在有限的复习时间内取得最多的进步（即取得进步分数之和最多）。

解答：背包问题

```
For i←0 to N
    V[i,0]←0
End for
For j←0 to T
    V[0,j]←0
End for
For i←1 to N
    For j←1 to T
        V[i,j]←V[i-1,j]
        If  $t_i \leq j$  then  $V[i,j] \leftarrow \max\{V[i,j], V[i-1, j-t_i] + i_i\}$ 
    End for
End for
Return V[N,T]
```

## 52 陈凯

小明有  $N$  双精致的筷子，每一双筷子上写着两个相同的数字，有一天小明不小心丢了其中的一根筷子，由于筷子太多而已混在一起，小明不知道丢了哪一双筷子中的一根，于是他找到你帮忙让你用**最快**的方法帮他确认丢失的那根筷子。

输入：  $N$ ，然后  $2N-1$  个数 代表 剩下的筷子上的数字

输出： 1 个数，丢失的筷子上的数字

样例：

输入： 5



9 9 11 4 3 2 11 3 2

输出： 4

解答：

用排序的不给分，用 Hash 的给一半，用 Xor 的满分

### 53 毕舰水

问题：

在  $A[1000]$  中存放了 1-999 之间的元素，只有一对元素是相同的，其它均只出现一次。每个数组元素只能访问一次，设计一个算法，将它找出来（所需的额外空间应为  $O(1)$ ）

解法：

因为有每个数组元素只能访问一次以及空间的限制，因此排序、简单的遍历等方法不可行

一个可行的解法为：对 1000 个元素求和，重复的元素  $= \text{sum} - (1+999)*999/2$ （即 1000 个元素的和减去 1~999 的和）

### 54 刘焰强

大裕神喜欢滑雪。他来到了一个滑雪场，这个滑雪场是一个矩形，为了简便，我们用  $r$  行  $c$  列的矩阵来表示每块地形。为了得到更快的速度，滑行的路线必须向下倾斜。

例如样例中的那个矩形，可以从某个点滑向上下左右四个相邻的点之一。例如 24-17-16-1，其实 25-24-23...3-2-1 更长，事实上这是最长的一条。现给定矩形，求大裕神能获得的最长下滑路线的长度。

输入文件

第 1 行: 两个数字  $r, c (1 \leq r, c \leq 100)$ , 表示矩阵的行列。

第 2.. $r+1$  行: 每行  $c$  个数, 表示这个矩阵。

输出文件

仅一行: 输出 1 个整数, 表示可以滑行的最大长度。

## 55 赖鸿炜

将  $n$  个正整数存放于一个一维数组  $a$  中, 试设计一个算法, 将所有的奇数移动并存放于数组的前半部分, 将所有的偶数移动并存放于数组的后半部分, 要求时间复杂度  $O(n)$ 。

## 56 杨梦筠

课程作业需要完成一个大项目, 假设有  $n$  项独立的模块  $\{1, 2, \dots, n\}$ , 由  $m$  个水平相当的同学编码完成。模块  $i$  所需要的处理时间为  $t_i$ 。约定: 任何一项模块可在由任何一位同学完成, 但未完工前不准中断编码; 任何模块不能拆分更小的子模块。

请给出一种调度方案, 使所给的  $n$  个模块在尽可能短的时间内由  $m$  个同学编码完成。设计算法, 并对其时间复杂度进行分析。

```
1、 if  $n < m$  直接分配
2、 else :
    JobNode time[N+1];           //各作业所需要的处理时间

    MinHeap<MachineNode> H(m); //由  $m$  台机器当前工作时间构成的最小推

    3、 Sort(time,n);              //对  $n$  个作业根据其  $t_i$  排序, 从大到小
    4、 for(int i=1; i<=n; i++)    //将工作分配给各机器
    {
        x = H.RemoveMin();
        cout<<"将机器"<<x.ID<<"从"<<x.avail<<"到"
```

```

        <<(x.avail+a[i].time)<<"的时间段分配给作业"
        <<a[i].ID<<endl;
    x.avail += a[i].time;
    H.Insert(x);          //根据新的 avail 值将 x 插入 Heap 中适当位置
}

```

时间复杂度：

$n > m$  时, 排序耗时  $O(n \log n)$ . 初始化堆耗时  $O(m)$ . 堆的 DeleteMin 和 insert 共需  $O(m \log m)$ . 因此算法所需时间:  $O(n \log n)$ 。