**CCF CSP认证（**CCF计算机软件能力认证 Certified Software Professional**）**

中国计算机学会（CCF）联合华为、360、滴滴等十余家知名IT企业以及清华、北航、国防科大等15所著名高校于2014年推出CCF CSP（计算机软件能力）认证标准，用于评价业界人士的计算机软件能力

# CSP-J ---- NOIP 普及组（初赛 、复赛）

# CSP-S ---- NOIP 提高组（初赛、复赛）

# C:\Users\ADMINI~1\AppData\Local\Temp\1599548813(1).png

分数组成：选择题15题 共：30分

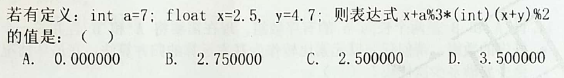
阅读程序题：3题（判断、选择） 共40分

完善程序题：2题 （选择） 共30分

目 录

1. 填空题………………………………………………………………………………………..3
2. 阅读程序 ………………………………………………………………………………….18
3. 完善程序 ………………………………………………………………………………….26

# 填空题

1．

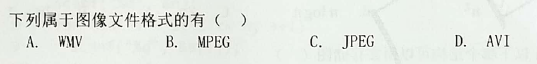
答案D

a%3 =1;

(int)(x+y) = 7;

7%2=1;

2.5+1\*1=3.5

2.

答案C

常见的视频文件格式：AVI、MOV/.QT、ASF、RM、NAVI、DivX、MPEG、WMV等

常见的图像文件格式：JPEG、TIFF、RAW、BMP、GIF、PNG

其它非主流图像格式：PCX、DXF、WMF、EMF、LIC、EPS等

### C:\Users\ADMINI~1\AppData\Local\Temp\1599559098(1).png3.

答案D

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

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

**逻辑”或”运算 || （or）**

两种情况：其中任意一种为真，或两种都为真时，结果为真，这种运算为或运算。

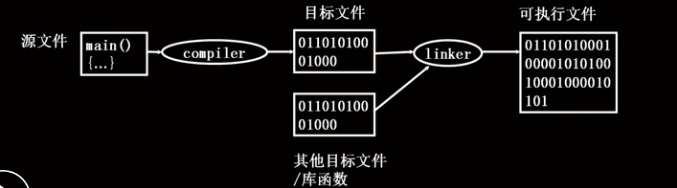
**比如：教室里有两个门前门、后门， A:前门开门，B:后门开门**

|  |  |  |
| --- | --- | --- |
| A | B | 结果 |
| 真 | 真 | 真 |
| 真 | 假 | 真 |
| 假 | 真 | 真 |
| 假 | 假 | 假 |

### C:\Users\ADMINI~1\AppData\Local\Temp\1599559716(1).png4.

答案B

编译：就是把高级语言变成计算机可以识别的2进制语言，计算机只认识1和0，编译程序把人们熟悉的语言换成2进制的。



### C:\Users\ADMINI~1\AppData\Local\Temp\1599560524(1).png

### 5. C:\Users\ADMINI~1\AppData\Local\Temp\1599560524(1).png

答案：B

1): (int) (x) 取浮点数的整数部分

(int)(x\*100)/100.0 取小数点后两位

(int)(x\*100+0.5)/100.0 四舍五入到小数点后两位

2): 可以用代入法排错：

x=2.6;

代入到A、B、C、D 4个答案中。

A：2.605 B: 2.6 C:52.6 D: 260.005

### 6. C:\Users\ADMINI~1\AppData\Local\Temp\1601365292(1).png

答案：B

1 2 4 8 组成的4位数字有4！=24 种

1 1 2 4 组成的4位数组有4!/2! (因为有两个1相同所以 / 2！) = 12种

1 1 2 8 、 1 1 4 8、 2 4 8 8 、1 2 8 8 、1 4 8 8 共有 6\*12 = 72种

1 1 8 8组成的4位数字：4！ / 2! /2 =6种

共有：24 + 72 +6 = 102种

### 7. C:\Users\ADMINI~1\AppData\Local\Temp\1601366069(1).png

答案：C

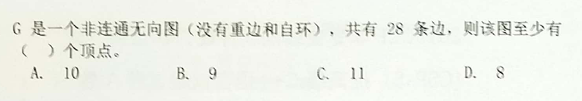
**稳定性的定义**

      假定在待排序的记录序列中，存在多个具有相同的关键字的记录，若经过排序，这些记录的相对次序保持不变，即在原序列中，ri=rj，且ri在rj之前，而在排序后的序列中，ri仍在rj之前，则称这种排序算法是稳定的；否则称为不稳定的。

排序的稳定性特点是排序完成后，之前相同的元素排序不会改变。快速排序在排序时在交换中间元素时可能会打乱顺序。如3、1、1、2、1、6、7、8、9，在一开始3与中间1交换后，稳定性已被打破。

稳定的排序算法：[基数排序](https://baike.baidu.com/item/%E5%9F%BA%E6%95%B0%E6%8E%92%E5%BA%8F)、[冒泡排序](https://baike.baidu.com/item/%E5%86%92%E6%B3%A1%E6%8E%92%E5%BA%8F)、[直接插入排序](https://baike.baidu.com/item/%E7%9B%B4%E6%8E%A5%E6%8F%92%E5%85%A5%E6%8E%92%E5%BA%8F)、[折半插入排序](https://baike.baidu.com/item/%E6%8A%98%E5%8D%8A%E6%8F%92%E5%85%A5%E6%8E%92%E5%BA%8F)、[归并排序](https://baike.baidu.com/item/%E5%BD%92%E5%B9%B6%E6%8E%92%E5%BA%8F)

不稳定的排序算法：[堆排序](https://baike.baidu.com/item/%E5%A0%86%E6%8E%92%E5%BA%8F)、[快速排序](https://baike.baidu.com/item/%E5%BF%AB%E9%80%9F%E6%8E%92%E5%BA%8F)、[希尔排序](https://baike.baidu.com/item/%E5%B8%8C%E5%B0%94%E6%8E%92%E5%BA%8F)、[直接选择排序](https://baike.baidu.com/item/%E7%9B%B4%E6%8E%A5%E9%80%89%E6%8B%A9%E6%8E%92%E5%BA%8F)



### 8.

答案：B

无向连通图： n个顶点 有 n\*(n-1)/2 个顶点。

n=8 时 无向连通图的顶点是28个，非连通无向图，共有28条边，至少有9个顶点。

### C:\Users\ADMINI~1\AppData\Local\Temp\1601368408(1).png9.

答案：B

能被3整除的数，各数字之后是3个倍数。

不考虑被3整除，共有5\*5\*3 =75种选择。

第3位数的可选项是：0 1 8而这3个数整除3 分别余：0 1 2

所以其他4位数确定后，第3位数只能有一种选择。 5\*5\*1 =25 种。

也可以通过列举法：

第3位是0时： 第1 位 第2位可以选：60 90 06 09 66 99 69 96 18 81 00 共11种选法。

第3位是1时： 第1位 第2位可以选：61 16 91 19 10 01 88 共7种选法

第3位是8时： 第1位 第2位可以选：68 86 89 98 80 08 11 共7种选法

共：11 + 7 + 7 = 25种选法

### C:\Users\ADMINI~1\AppData\Local\Temp\1601369771(1).png10.

答案：A

15+12 = 27 减去语文、数学都得满分的人4， 27-4= 23

### 11. C:\Users\ADMINI~1\AppData\Local\Temp\1601369996(1).png

答案：D

归并算法：

数组a={1,3,5,7} n=4

数组b={2,4,6,8} n=4

比较：

1： 1->2 1

2: 2->3 2

3: 3->4 3

4: 4->5 4

5: 5->6 5

6: 6->7 6

7: 7->8 7

第最后一个数 8不需要再比较直接放在最后一个元素：

比较次数是：2n-1

两个数组从小到大依次比较，哪边小哪边入数组，当某一数组全部计入结果数组后，剩下的也依次进入。最好的情况是数组A所有数都比数组B第一个数小，只要比较n次。最坏情况是全部比较完，最后AB只剩最后一个数比较，总比较次数就是2n-1。

### 12．C:\Users\ADMINI~1\AppData\Local\Temp\1601370935(1).png

答案：D

图的存储可以用邻接矩阵、邻接链表

栈、二叉数、队列属于数据结构。

常用的数据结构：数组、栈、队列、链表、树、图、堆、散列表等。

### 13. C:\Users\ADMINI~1\AppData\Local\Temp\1601372024(1).png

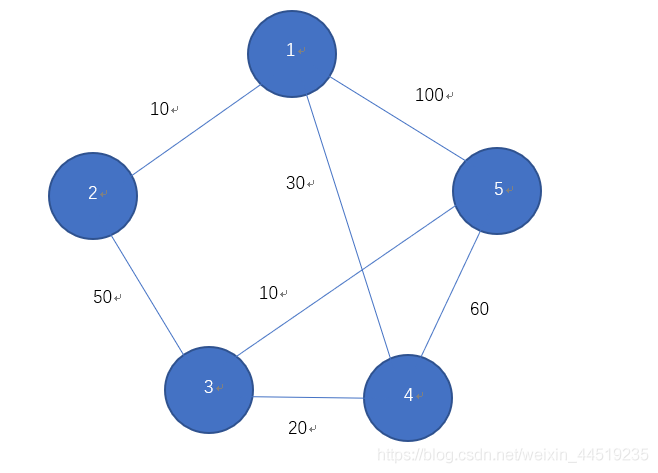
答案：B

Floyd算法利用动态规划属于动态规划算法。

Dijkstra算法是用于求解图中某源点到其余各顶点的最短路径的算法

Dijkstra算法求解单源最短路径的基本算法思想：  
首先需要做的是创建一个空集S，表示已经遍历过的节点的集合。  
选定好源点O之后，便要把源点O放入集合S中。之后进行如下步骤：

1） 如果!S不为空，选择距离源最小的点，称为点V，放入集合S中。  
2） 对V的所有后继(针对无向图则为所有相连的节点)进行遍历，当V到某一后继U的距离加上V到源点O的距离要小于从源点O到U的直接距离时，更新源点O到U的距离为OV距离+VU距离。  
3）重复进行前两步，直到!S为空为止。

如下面的一张图，求解点1到其他所有点的最短路径：

我们首先先把与点1相连的所有点的距离加进去。以数组dist[i]记录。  
首先dist[0] = 0 , dist[1] = 10, dist[2] = INF, dist[3] = 30, dist[4] = 100;  
用数组mark[i]表示第i-1个节点是否进入了集合S。  
mark[0] = true;  
接下来选择距离源最近并且还没有进入S的点。点2。

那么 mark[0] = true; mark[1] = true;  
接着，dist [1] + edge[1][2] = 10 + 50 = 60 < dist[2] = INF.  
也就是说点1到点2的距离加上点2到点3的距离小于点1到点3的距离(因为他们不相连)。  
dist[0] = 0 , dist[1] = 10, dist[2] = 60, dist[3] = 30, dist[4] = 100;

接下来选择距离源最近并且还没有进入S的点。点4。

那么 mark[0] = true; mark[1] = true; mark[3] = true;  
接着，dist[3] + edge[3][2] = 30 + 20 = 50 < dist[2] = 60  
也就是说点1到点4的距离加上点4到点3的距离小于点1到点3的距离。  
dist[0] = 0 , dist[1] = 10, dist[2] = 50, dist[3] = 30, dist[4] = 100;

接着，dist[3] + edge[3][4] = 30 + 60 = 90 < dist[4] = 100  
也就是说点1到点4的距离加上点4到点5的距离小于点1到点5的距离。  
dist[0] = 0 , dist[1] = 10, dist[2] = 50, dist[3] = 30, dist[4] = 90;  
至此，点4的所有后继搜寻完毕。

接下来选择距离源最近并且还没有进入S的点。点3。  
那么 mark[0] = true; mark[1] = true; mark[2] = true; mark[3] = true;  
接着，dist[2] + edge[2][5] = 50 + 10 = 60 < dist[4] = 90  
也就是说点1到点3的距离加上点3到点5的距离小于点1到点5的距离。  
dist[0] = 0 , dist[1] = 10, dist[2] = 50, dist[3] = 30, dist[4] = 60;

点3的其余后继不用更新，因为相加的大小都比之前的距离长，不能更新。

最后点5也同上，没有可以更新的距离，至此，所有的点都进入了集合S。

最终点1到其他所有点的最短距离为：  
dist[0] = 0  
dist[1] = 10  
dist[2] = 50  
dist[3] = 30  
dist[4] = 60

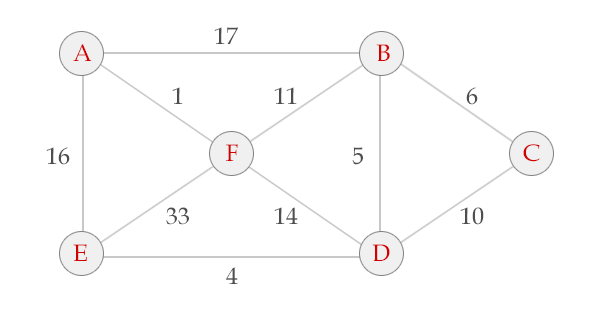
[Prim算法](https://www.cnblogs.com/nbalive2001/p/4970636.html)

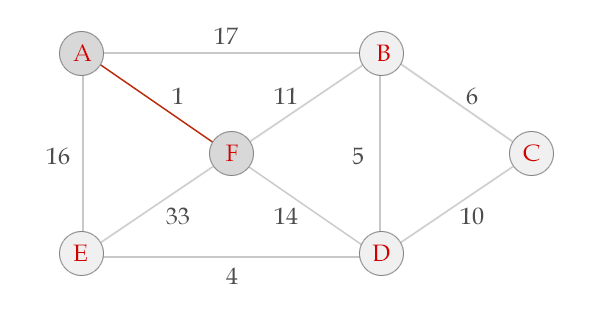
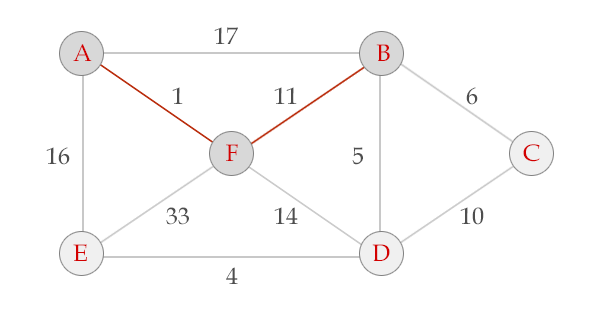
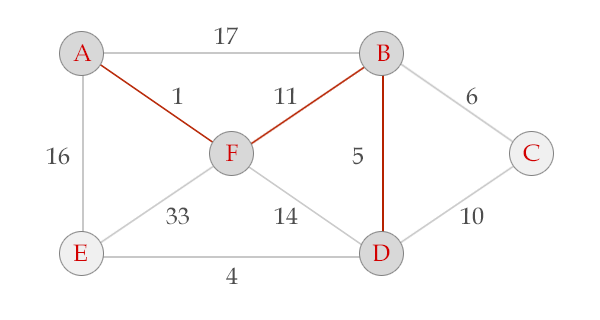
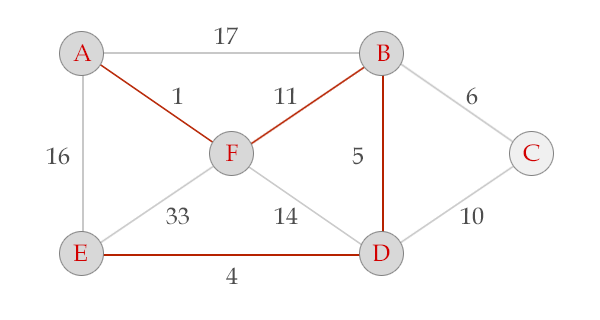
最小生成树的Prim算法也是贪心算法的一大经典应用。Prim算法的特点是时刻维护一棵树，算法不断加边，加的过程始终是一棵树。

Prim算法过程：  
  
一条边一条边地加， 维护一棵树。

初始 E ＝ ｛｝空集合， V = ｛任选的一个起始节点｝  
  
循环（n – 1）次，每次选择一条边（v1,v2）， 满足：v1属于V , v2不属于V。且（v1,v2）权值最小。  
  
E = E + （v1,v2）  
V = V + v2  
  
最终E中的边是一棵最小生成树， V包含了全部节点。

以下图为例介绍Prim算法的执行过程。



Prim算法的过程从A开始 V = {A}, E = {}  
  
选中边AF , V = {A, F}, E = {(A,F)}   
  
选中边FB, V = {A, F, B}, E = {(A,F), (F,B)}  
  
选中边BD, V = {A, B, F, D},   E = {(A,F), (F,B), (B,D)}  
  
选中边DE, V = {A, B, F, D, E},   E = {(A,F), (F,B), (B,D), (D,E)}  
   
选中边BC, V = {A, B, F, D, E, c},   E = {(A,F), (F,B), (B,D), (D,E), (B,C)}, 算法结束。

# 最小生成树（kruskal算法）

最小生成树问题顾名思义，概括来说就是路修的最短。

接下来引入几个一看就明白的定义：

最小生成树相关概念：

带权图：边赋以权值的图称为网或带权图，带权图的生成树也是带权的，生成树T各边的权值总和称为该树的权。

最小生成树（MST）：权值最小的生成树。

最小生成树的性质：假设G＝(V,E)是一个连通网，U是顶点V的一个非空子集。若(u,v)是一条具有最小权值的边，其中u∈U，v∈V－U，则必存在一棵包含边(u,v)的最小生成树。

完成构造网的最小生成树必须解决下面两个问题：

      （1）尽可能选取权值小的边，但不能构成回路；

      （2）选取n－1条恰当的边以连通n个顶点；

**prim算法适合稠密图，kruskal算法适合简单图。**

Floyd算法  
**求最短路径一般有两种算法：**  
**1.弗洛伊德  
2.迪杰斯特拉**

关于迪杰斯特拉算法→[最短路径](https://blog.csdn.net/Doro_/article/details/105747564)

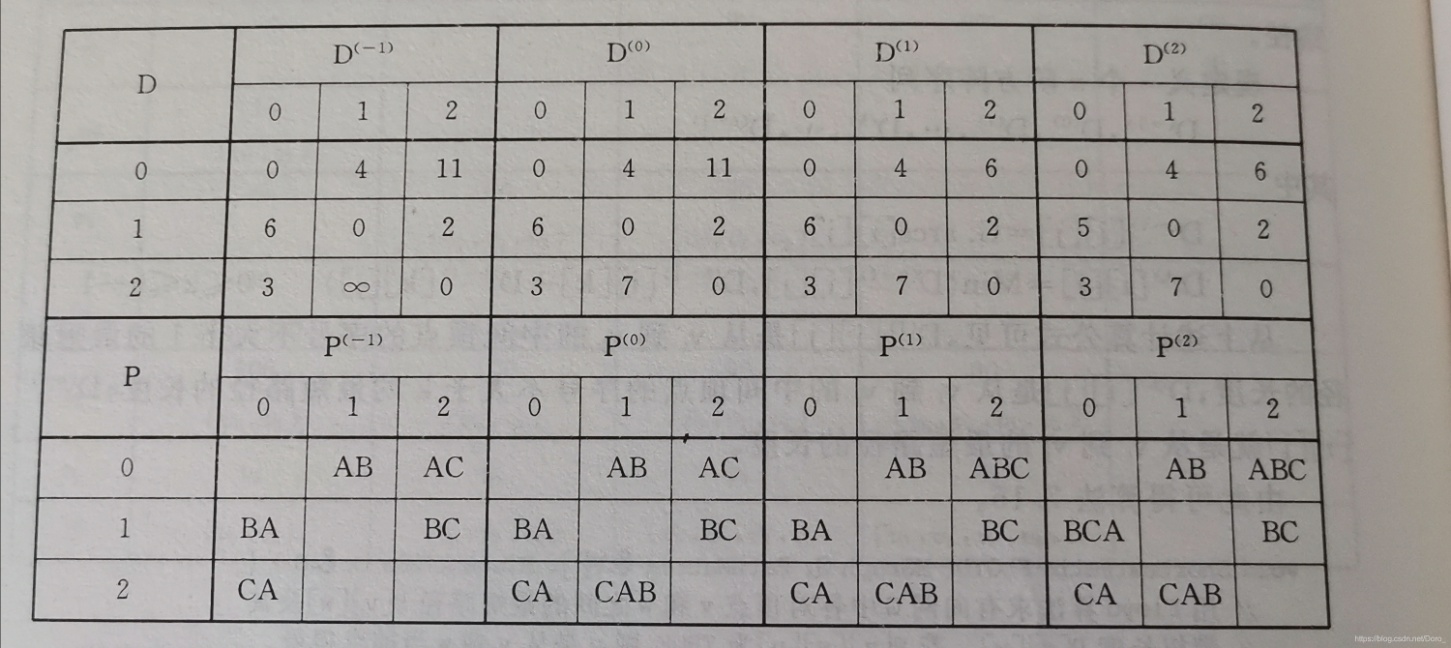
**两算法的区别:**  
弗洛伊德算法是可以解决任意两点间的最短路径的一种算法  
Dijkstra算法是用于求解图中某源点到其余各顶点的最短路径的算法

那我们来看看弗洛伊德算法

## 思想

通过Floyd计算图中各个顶点的最短路径时，需要引入两个矩阵，矩阵D中的元素表示顶点i到顶点j的距离。矩阵P中的元素表示顶点i到顶点j经过了P记录的值所表示的顶点。但在此题中不需要记录P

**过程**  
顶点数为n的话，就对初始矩阵进行n次更新，每次分别以第i个顶点为中转点，符合条件就更新。  
具体看图吧重在理解==这里以有向图为例，无向图更简单



### 14. C:\Users\ADMINI~1\AppData\Local\Temp\1601375279(1).png

答案：B

118098/486 = 243

486/2=243

243不是2，4，5的倍数而是3的倍数，所以可能的公比是3

### C:\Users\ADMINI~1\AppData\Local\Temp\1601375453(1).png15. C:\Users\ADMINI~1\AppData\Local\Temp\1601375429(1).png

答案：A

每个点只能从上方两个点过来，自然取最大的加a(i,j) 加上 上一行中经过的较大的路径max(c[i-1][j-1],c[i-1][j] )

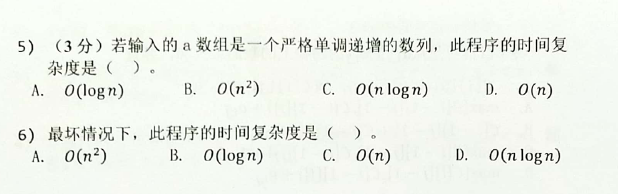
# 阅读程序题

### C:\Users\ADMINI~1\AppData\Local\Temp\1601375964(1).png]

思路：在数组a中找到a[i]后面第一个大于a[i]的位置。



1. 错： 11行执行，i=1, ans=1,都不满足条件时，ans = I;
2. 对，ans 是数组中 位置是 小于等于n的
3. 正确，程序的关键在于14，15行，大于a[i]的位置
4. 正确，a[i]后面a[i+2] 才是大于a[i]的值，那说明a[i+1]<=a[i]

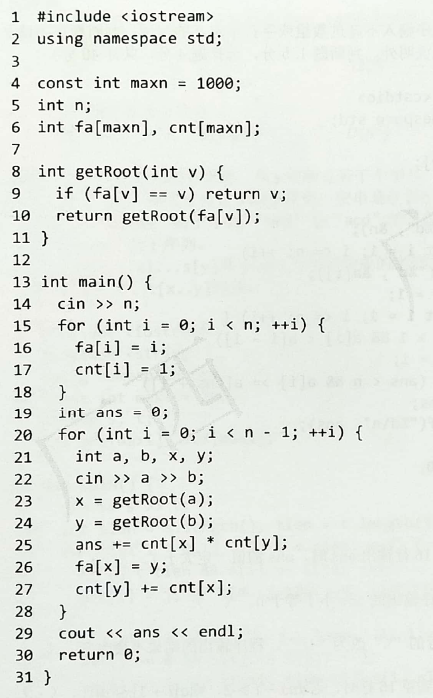


5). 答案D

严格的递增，14行不执行，while循环每次只执行一次。

时间复杂度，D

6). 答案A

最坏的情况为严格单调递减，14行if判断每次都执行，while循环每次都查找到n，时间复杂度为n+（n-1）+……+2+1=n\*（n+1）/2,即O(n^2)。

### 2.

思路：并查集操作

并查集：并查集被很多人认为是最简洁而优雅的数据结构之一，主要用于解决一些**元素分组**的问题。它管理一系列**不相交的集合**，并支持两种操作：

**合并**（Union）：把两个不相交的集合合并为一个集合。

**查询**（Find）：查询两个元素是否在同一个集合中。

### 初始化

**int** fa[MAXN];

**inline** **void** **init**(**int** n)

{

**for** (**int** i **=** 1; i **<=** n; **++**i)

fa[i] **=** i;

}

假如有编号为1, 2, 3, ..., n的n个元素，我们用一个数组fa[]来存储每个元素的父节点（因为每个元素有且只有一个父节点，所以这是可行的）。一开始，我们先将它们的父节点设为自己。

### 查询

**int** **find**(**int** x)

{

**if**(fa[x] **==** x)

**return** x;

**else**

**return** find(fa[x]);

}

我们用递归的写法实现对代表元素的查询：一层一层访问父节点，直至根节点（根节点的标志就是父节点是本身）。要判断两个元素是否属于同一个集合，只需要看它们的根节点是否相同即可。

### 合并

**inline** **void** **merge**(**int** i, **int** j)

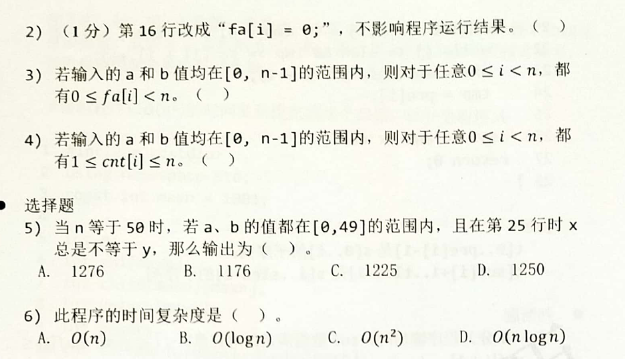
{

fa[find(i)] **=** find(j);

}

合并操作也是很简单的，先找到两个集合的代表元素，然后将前者的父节点设为后者即可。当然也可以将后者的父节点设为前者，这里暂时不重要。本文末尾会给出一个更合理的比较方法。

C:\Users\ADMINI~1\AppData\Local\Temp\1601377015(1).png



程序：8-10 行查询

15-18 初始化

20-28 构建集合

29 输出深度

1). 正确 a b 是集合的下标，范围【0，n-1】

2). 错识， fa[i]=i, 初始化所有元素的根结点是本身，如果改为fa[i]=0 意味着所有的结点的根结点是0，都在集合0内，与题目的初衷相悖，并且影响后面的ans的计算结果。

3). 正确 所有的结点的根结点都在【0，n)

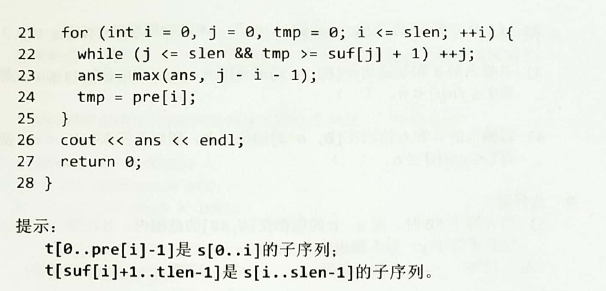
4). 错误 cnt[i]是合并之后集合的元素个数，严谨的并查集操作cnt[i]是不会超过n的，但是本题没有判断是否重复合并。所以如果先输入（1,2），（3,4），之后一直不断输入（2,4），最后cnt[4]会超过n。

5). cnt[i]表示当前集合的元素个数，每次执行都是两个集合合并，我们可令每次都是单个集合合并进入大集合，ans=1\*1+1\*2+1\*3+……1\*48+1\*49=49\*（49+1）/2=1225。

6). getRoot是依次查找上一个父元素，没有“压缩路径”，时间复杂度最差为n，所以总的时间复杂度为n^2。

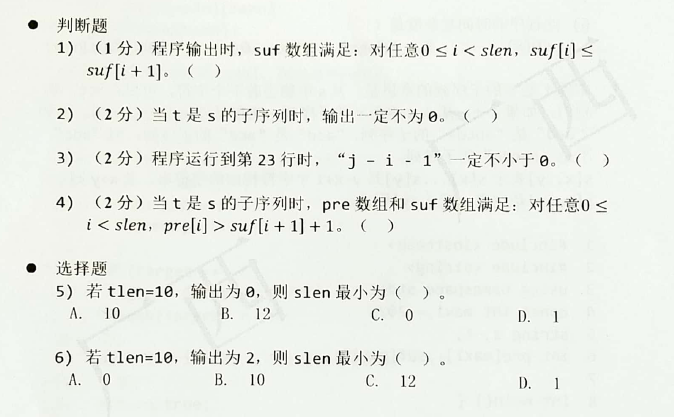
1\*1 +1\*2+1\*3+1\*n=n\*(n-1)/2 ～n^2

### C:\Users\ADMINI~1\AppData\Local\Temp\1601384639(1).png3.



思路：求解字符串s删除连续最多几个字符后，字符串t仍是s的子序列。

pre[i]表示s[0..i]至多可以从前往后匹配到t串的哪一个字符，此时t[0..pre[i]-1]是s[0..i]的子序列。sub[i]用于记录s[i..slen-1]至多从后到前匹配到t的哪一个字符，此时t[suf[i]+1..tlen-1]是s[i..slen-1]的子序列。



1). 正确 从15至19行可以看出，sub数组的值是从尾部往前减小或不变，所以suf[i] ≤suf[i+1]。

2). 错误 有题目目的可知，当t==s时输出为0。

3). 错误 若第一循环时while不执行，则j-i-1为-1，如s=“ bacb”,t=”ac”

4). 错误。由题义可知若t是s子序列，t[0..pre[i]-1],t[sub[i+1]+1..lent-1]是s[0..i],s[i+1..lens-1]的子序列，不会重叠，即pre[i]-1< sub[i+1]+1,即pre[i] <= sub[i+1]+1。

或者用s=t=‘a’ 代入即可

5). 答案D

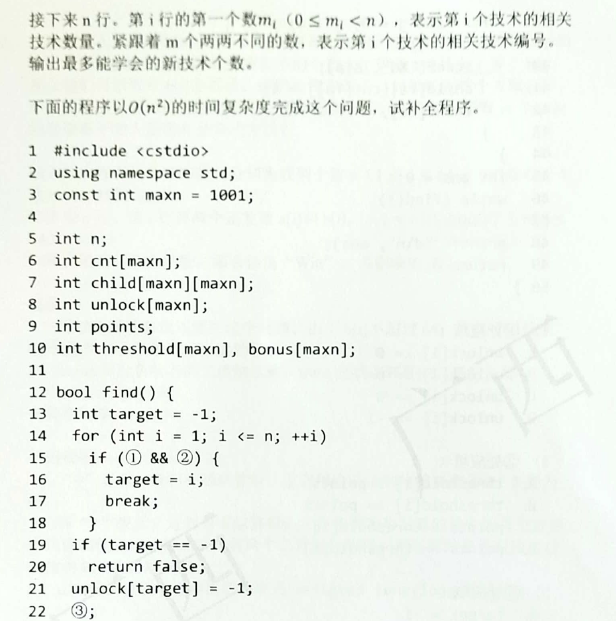
当t不是s子串（或t==s）输出都为0，但为保证程序执行，最少应输入一个字符。

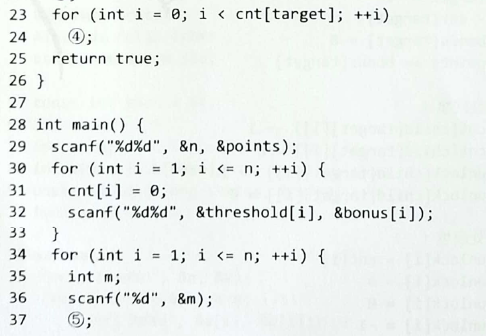
6). 答案C

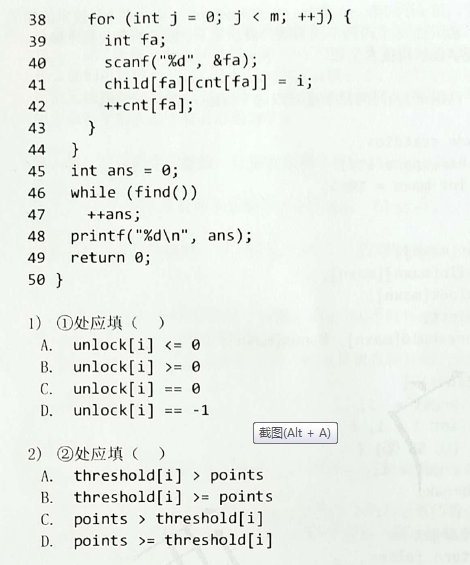
输出为2说明slen最多连续删除2个后为10，所以最小为12。

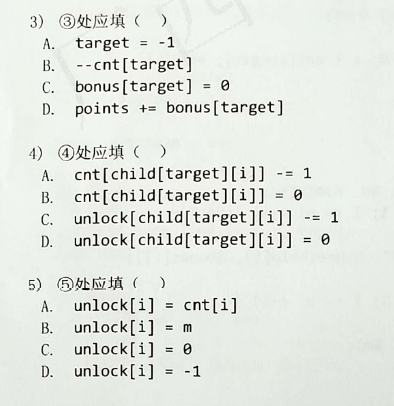
# C:\Users\ADMINI~1\AppData\Local\Temp\1601387991(1).png完善程序

### 









思路：

pionts 为已有经验值

threshold[i]为第i项技能所需要最低经验

bonus[i]第i项技能可获得的经验

unlock数组为对应技能需学习的前置技能数，大于0说明有前置技能要学，为-1表示已学习。

每次都先学习一个已经达到条件但为学习的技能，学习后更新经验值和其他技能与该技能有关的学习条件，不断重复至没有技能可以学。

1). 答案 C

学习技能条件一，需要的前置技能数为0且为学习。

2). 答案D

学习技能条件二，总经验达到该技能的经验要求。

3). 答案D

技能学习之后，更新经验值

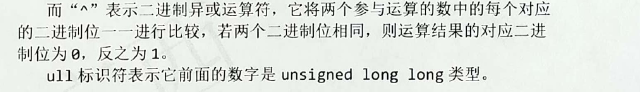
4). 答案 C

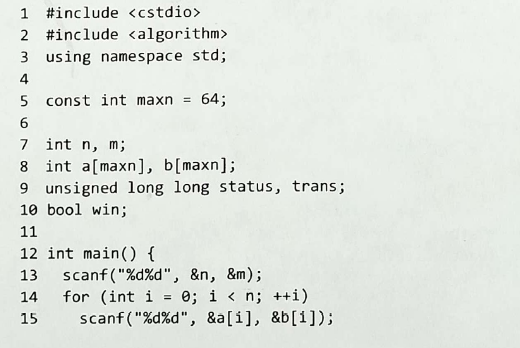
学完一项新技能之后，更新相应后置技能的尚未解锁的前置技能数，相应后置技能unlock值应减1，这里的child数组用于记录每个技能后置技能的编号。

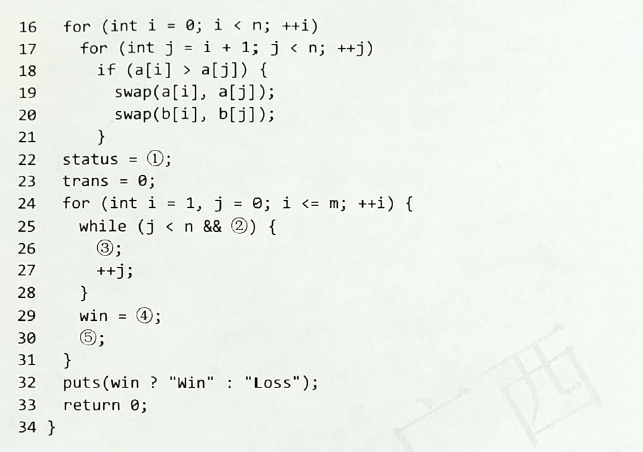
5). 答案B

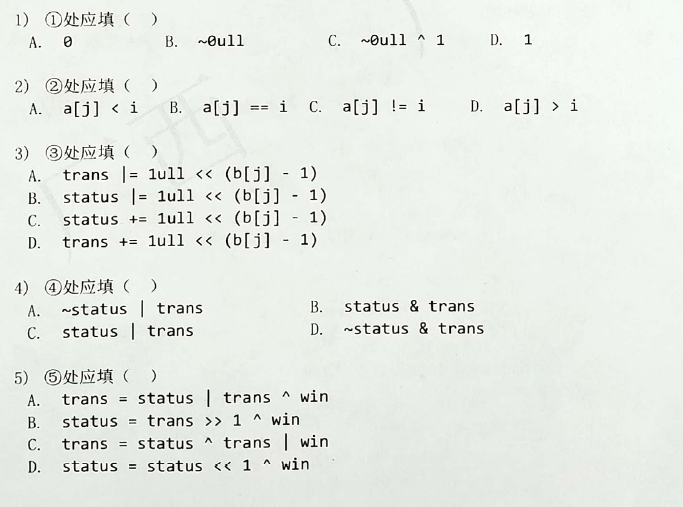
初始化每个技能尚未解锁的前置技能数，unlock[i]应为m。

### 2. C:\Users\ADMINI~1\AppData\Local\Temp\1601389780(1).png









思路：我们可以设置bool数字f[i]表示有i个石子时有无先手必赢策略。若对于i个石子有先手必赢策略，则存在j（a[j]<=i且b[j]<=i）使得i-b[j]个石子先手无必赢策略，则得到转移方程f[i]=OR{!f[i-b[j]} (a[j]<=i且b[j]<=i)。因为本题策略数和数组b数字都不超过64，所以仅考虑f[i-1]..f[i-64],可将其状态压缩至一个64位数中。其中status用于记录对于i个石子，i-1..i-64是否有先手必胜策略。

1). 答案C

由题意可知，状态压缩到64位，A和D默认是32位整数，所以B或者C。最开始石子是0个，应该是输的状态，所以最低位不能是1，选C。

2). 答案 B

a[i]进行从小到大排序，所以可使用规则只增加不减少。此循环用于增加当前可选的规则，当i达到a[j]时规则可以使用。

3). 答案A

在原有规则上，新增”取b[j]个棋子”的规则。二进制新增用|。

4). 答案D

判断当前石子数i能否先手必胜，即要求存在某个前置状态无先手必胜策略。

5). 答案D  
将“i个石子是否先手必胜“添加入压缩后的状态status中，应将当下状态的结果添加到status

的右起第1位。