1. 单项选择题（共15题，每题2分，共计30分；每题有且仅有一个正确选项）
2. 下列关于NOIP的说法，错误的是（ ）
3. NOIP中文名称为全国青少年信息学奥林匹克联赛，将于今年恢复举行。
4. 参加NOIP是参加NOI的必要条件，不参加NOIP将不具有参加NOI的资格。
5. NOIP竞赛全国前五十名将获得进入国家集训队的资格。
6. NOIP竞赛费0元，但本着谁受益谁承担成本的原则，参加竞赛所需的餐饮、住宿、交通、保险等费用由参加者自行承担。

解析：NOI前50

1. 二进制数001001与100101进行按位异或的结果为（ ）

A. 101000 B. 100100 C. 101101 D. 101100

解析：异或：对应的2个二进制位相同为0，不同1，不进位的加法

1. 在8为二进制补码中，10101011表示的数是十进制下的（ ）

A. 43 B. -43 C. -85 D. -84

解析：符号位为1是负数，剩余数字部分，反码0101010，原码1010101，-(5\*16+5)=-85

1. 以下算法使用倍增思想的是（ ）
2. 树状数组
3. 线段树
4. 快速排序
5. 三分法

解析：常识，其它都与分治思想相关。

1. 设树有7片叶子，其余结点度均为3，则T中3度结点有多少个（ ）

A. 3 B. 7 C. 9 D. 4

解析：树的度数是孩子节点数，题中相当于是一个“完全三叉树”

1. 组合数C(n,k)为从n件有标号物品中选出k件物品的方案数，例如C(3,2)=3，已知n,k皆为自然数，下列说法错误的是（ ）
2. C(n,k)=C(n-1,k)+C(n-1,k-1)。
3. C(2n,k)（0≤k≤2n）在 k=n时取得最大值。
4. 卡特兰数C\_n=C(2n,n)/n
5. 包含n个0和k个1，且没有两个1相邻的字符串的数量为C(n+1,k)

解析：卡特兰数H(n)=C(2n,n)/(n+1)。D选项可用插板法解释。

1. 下列有关CPU的说法，正确的是（ ）
2. CPU的用途是将计算机系统所需要的显示信息进行转换驱动显示器。
3. CPU 的性能和速度取决于时钟频率（一般以赫兹或千兆赫兹计算，即 hz 与 Ghz）和每周期可处理的指令（IPC），两者合并起来就是每秒可处理的指令（IPS）。
4. AMD 是世界上最大的半导体公司，也是首家推出 x86 架构处理器的公司。
5. 目前的 CPU 一般都带有3D 画面运算和图形加速功能，所以也叫做“图形加速器”或“3D加速器”。

解析：A、D都是显卡，C是~~牙膏厂~~INTEL

1. 下列算法中，没有用到贪心思想的算法为（ ）
2. 计算无向图最小生成树的Kruskal算法。
3. 计算无向图点中每对节点之间最段路的Floyd算法。
4. 计算无向图单源最短路路径的Dijkstra算法。
5. 以上算法均使用了贪心的思想。

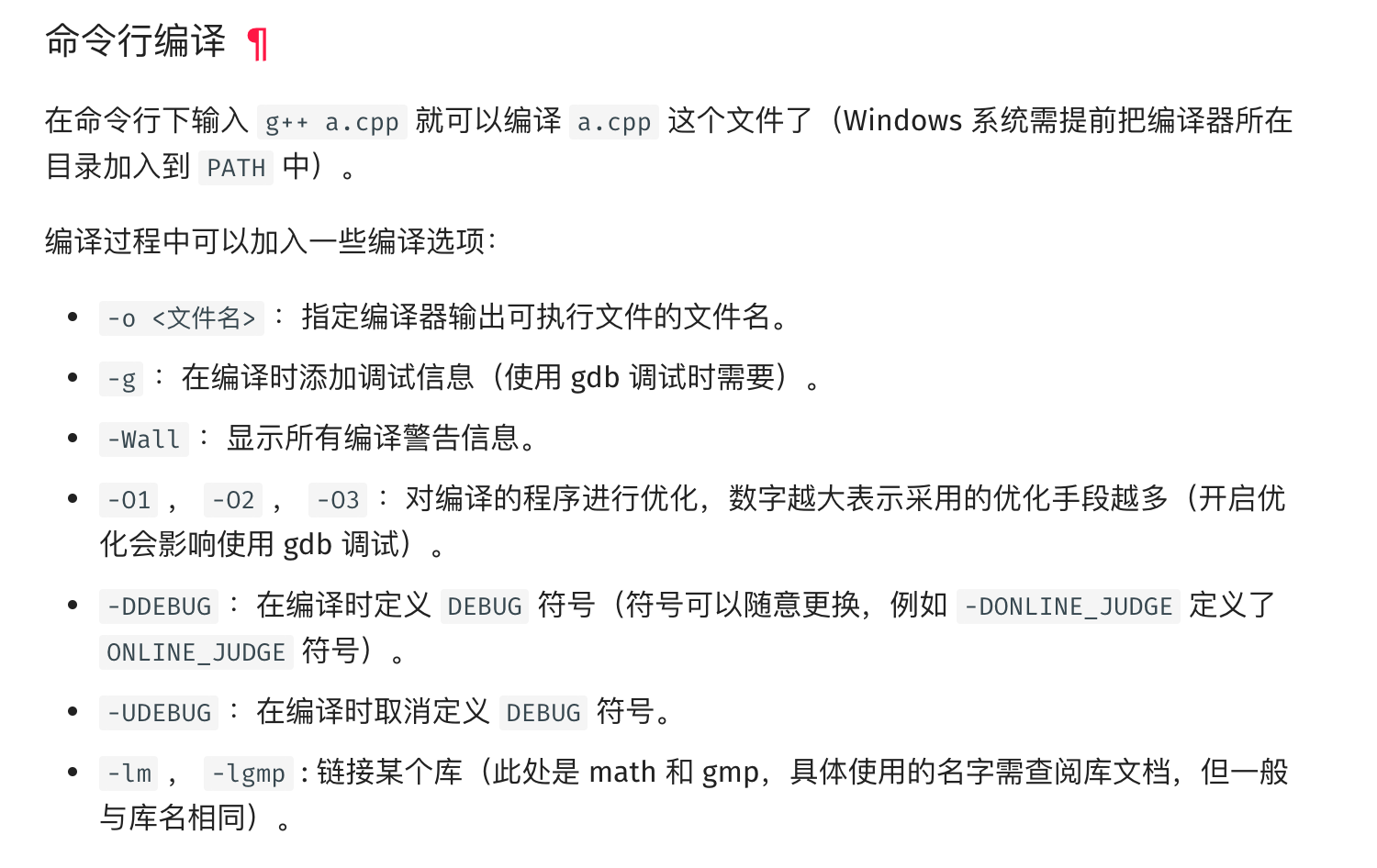
解析：Floyd是DP

1. 使用快速排序算法对序列进行排序，最坏时间复杂度为（ ）
2. O(n) B. O(n log n) C. O(n\*n) D. O(n log n log n)

解析：常识，数据全有序时为O(n^2)

1. 若要使用 g++ 编译器，开启 -Ofast 优化，且使用 C++ 11 标准，将源文件 prog.cpp 编译为可执行程序 exec，且保留调试信息，则需要使用的编译命令为（ ）
2. g++ prog.cpp -Ofast exec -std=c++11 -debug
3. g++ prog.cpp -Ofast exec -std=c++11 -g
4. g++ prog.cpp -o exec -Ofast -std=c++11 -debug
5. g++ prog.cpp -o exec -Ofast -std=c++11 -g

解析：



1. 已知袋子 α 中装有 4 张 5 元纸币和 3 张 1 元纸币，袋子 β 内装有 2 张 10 元纸币与 3 张 1 元纸币，袋 子 γ 中装有 3 张 20 元纸币与 3 张 50 元纸币。现在从每个袋子中随机选出 2 张纸币丢弃，记第 i 个袋子此时剩下的纸币面值之和为 vi ，则 vα < vβ < vγ 的概率为（ ）

A. 8/35 B. 9/35 C. 11/35 D. 12/35

解析：Vγ永远最大，vβ只能取两张1。vα 可取两张5，或一张5一张1。C(3,2)/C(5,2)\*(C(4,2)+4\*3)/C(7,2)

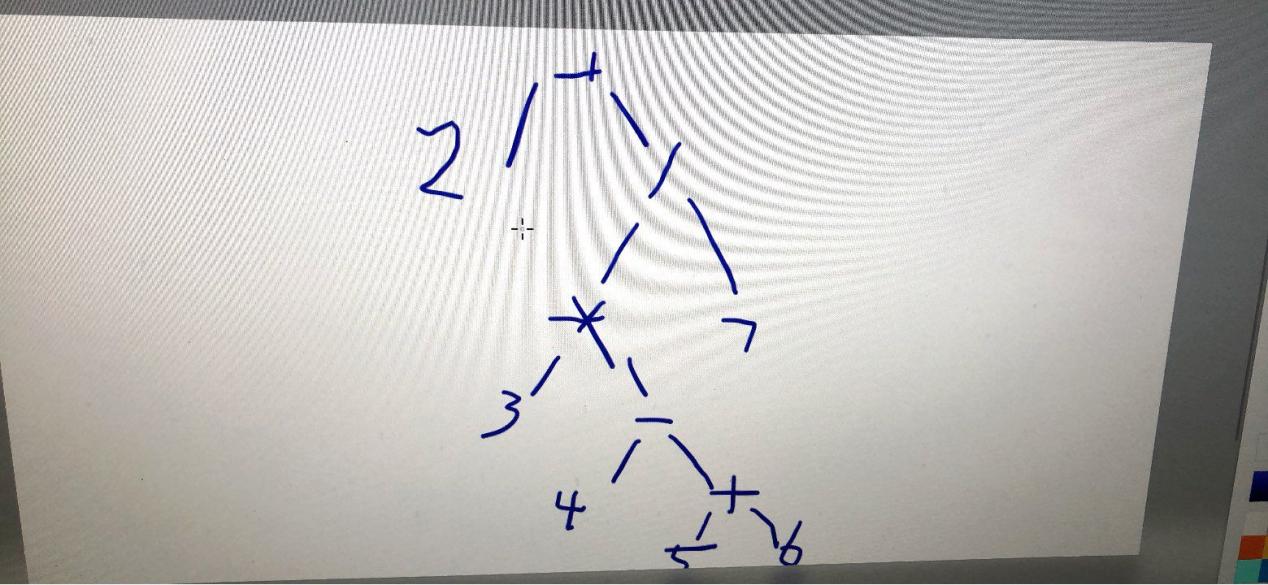
1. 对于一个数字串S，规定每一位单独为一个数字。我们从第一个数字开始扫描，标记出现的数字，若当前处理位的数字被标记过，则答案ans加上这最近两次出现该数字的位置之间（包含端点）的所有数字的和。现在请你求出对于S=”1926081720050831”的答案为（ ）

A. 98 B. 107 C. 134 D. 141

解析：模拟即可

1. 2+3\*(4-(5+6))/7的逆波兰表达式为（ ）
2. 2 3 4 5 6 - + \* 7 / +
3. 2 3 4 5 6 - + \* / 7 +
4. 2 3 4 5 6 + - \* 7 / +
5. 2 3 4 5 6 + + \* / 7 -

解析：



1. 现在定义一种新的数据结构“折半表”，本质为一个有n个元素的数列{ai}(1≤i≤n)，支持两个基本操作：
2. 入表操作，让元素个数n加一，将一个元素an插入到数列的末尾。
3. 出表操作：提取数列的第n/2（上取整）项的值，然后删除这一项，后面的每一项都前移一项，最后让元素个数n减一。

现在有以下的数列{bi}={26 62 54 58 13 6 49 37 40 12 31}，让这些元素按此顺序入表。现在得到的对于折半表的操作列表为（A代表入表，B代表出表）：AABABBAAAABAABBBABA，则最后一个出表的元素是：

A. 54 B. 49 C. 40 D. 6

解析：模拟即可

1. 在xxy的面前摆了4包不同品牌的薯条（用a代替）和5包不同品牌的蕃茄酱（用b代替），其中有4个b的品牌与4个a一一对应，另一个b的品牌则无法对应。每次操作，xxy从剩下的a中随机选择一个，从剩下的b中随机选择一个，一起吃掉。这样4此以后，a已经没有了，b还有一包，xxy就会把这包b送给小y。问xxy恰好只吃到一组同品牌的a和b的概率约为（ ）

A. 37% B. 36% C. 33% D. 31%

解析：分两种清况。第一种，抛弃的b刚好是多余品牌。那么选一对a,b对应正确，其余错排即可。第二种，选一对a,b对应正确，再选一个a对应多余b。剩余两个可简单列出。

(4\*3+4\*3\*3)/P(5,4)

1. 阅读程序（程序输入不超过数组或字符串定义的范围；判断题正确填 T，错误填F；除特殊说明外，判断题1.5分，选择题4分，共计40分）

1）

1. #include<bits/stdc++.h>
2. using namespace std;
3. int n,m;
4. int dfs(int x)
5. {
6. if (x<=0) return 1;
7. int ii,ss=0;
8. for(ii=1;ii<=m;++ii)
9. ss+=ii\*dfs(x-ii);
10. return ss;
11. }
12. int main(){
13. scanf("%d%d",&n,&m);
14. printf("%d\n",dfs(n));
15. }
16. （1分）若m=1，则输出为∑(1≤i≤n)i2。 F

解析：全为1

1. （1分）若n一定，则程序输出结果随m的增大而增大(m≤n)。 T

解析：ss返回全为正数，显然。

1. （1分）若将第6行的<=改为==，程序运行结果依然始终不变。 F

解析：m>1时会出现形参x<0

1. （2分）输入5 3，则输出150 。 F

解析：151。求的东西是一个和等差数列相关的级数。

1. 输入7 2，程序dfs函数调用的次数为（ ）

A. 157 B. 861 C. 171 D. 67

解析：对最深的分支进行模拟，其它的“人工记忆化”即可。

1. 本程序不能用哪一种思想优化：
2. 贪心 B. 递归转递推 C. 记忆化 D. 找规律

解析：略

2）

1. #include<cstdio>
2. using namespace std;
3. int n;
4. int a[100];
5. int main(){
6. scanf("%d",&n);
7. for(int i=1;i<=n;++i)
8. scanf("%d",&a[i]);
9. int ans=1;
10. for(int i=1;i<=n;++i){
11. if(i>1&&a[i]<a[i-1])
12. ans=i;
13. while(ans<n&&a[i]>=a[ans+1])
14. ++ans;
15. printf("%d",ans);
16. }
17. return 0;
18. }
19. 第16行输出ans时，ans的值一定大于i。 F
20. 程序输出的ans小于等于n。 T
21. 若将第12行的“<” 改为“!=”，程序输出的结果不会改变。 T
22. 当程序执行到第16行时，若ans-i>2，则a[i+1]≦a[i] T
23. 若输入的a数组是一个严格单调递增的数列，此程序的时间复杂度是（ ）

A. O(log n) B. O(n^2) C. O(n log n) D. O(n)

1. 最坏情况下，此程序的时间复杂度是（ ）
2. O(n^2) B. O(log n） C. O(n) D. O(n log n)

解析：2019CSP-S原题，略。

3）

1. #include <bits/stdc++.h>
2. const int N = 1e6 + 50;
3. int n, m, min[N], out[N], f[N], siz[N], ans;
4. struct edge
5. {
6. int u, v, w;
7. } e[N];
8. inline int find(int x)
9. {
10. while (x != f[x])
11. x = f[x] = f[f[x]];
12. return x;
13. }
14. inline void merge(int x, int y)
15. {
16. if (siz[x] > siz[y])
17. std::swap(x, y);
18. f[x] = y;
19. }
20. int main()
21. {
22. scanf("%d%d", &n, &m);
23. for (int i = 1; i <= n; ++i)
24. {
25. f[i] = i, siz[i] = 1;
26. }
27. for (int i = 1; i <= m; ++i)
28. {
29. scanf("%d%d%d", &e[i].u, &e[i].v, &e[i].w);
30. }
31. int components = n;
32. while (components > 1)
33. {
34. memset(min, 0x3f, sizeof min);
35. for (int i = 1; i <= m; ++i)
36. {
37. int u = find(e[i].u), v = find(e[i].v), w = e[i].w;
38. if (u != v)
39. {
40. if (w < min[u])
41. min[u] = w, out[u] = v;
42. if (w < min[v])
43. min[v] = w, out[v] = u;
44. }
45. }
46. for (int i = 1; i <= n; ++i)
47. {
48. int x = find(i);
49. if (out[x])
50. {
51. int y = find(out[x]);
52. if (x != y)
53. {
54. merge(x, y);
55. --components;
56. ans += min[x];
57. }
58. }
59. }
60. }
61. printf("%d", ans);
62. return 0;
63. }

提示：该程序的输入为一张 n 个点 m 条边的连通无向图。输入格式为：

• 输入的第一行包含两个整数 n, m(1 ≤ n ≤ m ≤ 100)。

• 接下来一行，包含三个整数 u, v, w(1 ≤ u, v ≤ n, 1 ≤ w ≤ 100)。

判断题

1. （1 分）该程序实现了用于求无向图 G = (V, E) 的最小生成树的算法。 T
2. （1 分）程序中使用了邻接矩阵来存储图 G = (V, E)。 F
3. 为了确保该算法的结果符合预期，输入需要保证所有的 w 互不相同。 F
4. 对于任意合法的输入，该程序一定会在有限步内返回结果。 T

选择题：

1. 变量 components 在运行过程中可达到的最小值为 ( )

A. 1 B. 0 C. ⎣n/2⎦ D. m-(n-1)

1. 该程序的时间复杂度为（ ）
2. O(nm) B. O(n^2) C. O(n lo n) D. O(m log n)

解析：Boruvka，当原图连通时，每次迭代连通块数量至少减半，算法只会迭代不超过 O(log n) 次，而原图不连通时相当于多个子问题，因此算法复杂度是 O(m log n)  的。

1. 完善程序（每小题3分，共计30分）
2. 学军中学的同学们又要去参加CSP的服务生工作，总共有n个同学参加服务（n<=200），而学军的考场可以看作是2层、无限长的建筑，每一层楼都分布了很多考场，每个考场外面都需要一个服务生。每个同学都有不同的能力值。xxy为了对考生负责，规定了每一层的服务生的能力值总和要尽量接近。同时为了便于安排考场，他还规定两层的服务生个数差必须在1个或以下。现在输入n和n个同学各自的能力值a[i]，请你分别求出两层楼各自的服务生能力值总和，按升序输出。（提示：所有学生能力值总和不超过5000）

例如，现在有3个服务生，其中Oliver的能力值为35，siyuan的能力值为20，hehezhou的能力值为32，为了满足xxy的要求，两层中的服务生个数分别为1个和2个，且两层能力值总和之差要尽可能接近。所以我们把Oliver分在1个考场的一层，能力值总和为35。把hehezhou和siyuan分在两个考场的一层，能力值总和为52。可以证明这是差值最小的分法。

1. #include<cstdio>
2. #include<algorithm>
3. using namespace std;
4. int n,a[251],i,j,k,s,minn=2147483647;
5. int f[251][6001];
6. int main()
7. {
8. scanf("%d",&n);
9. for (i=1;i<=n;++i) scanf("%d",&a[i]);
10. ① ;
11. for (i=1;i<=n;++i)
12. {
13. for (j=5000;j>= ② ;--j)
14. for (k=100;k>=1;--k)
15. f[k][j]= ③ ;
16. s+=a[i];
17. }
18. for (i=0;i<=5000;++i)
19. {
20. if( ④ ) {
21. minn=min(minn, ⑤ );
22. }
23. }
24. printf("%d %d\n",(s-minn)/2,(s+minn)/2);
25. }
26. ①处应填（ ）

A. s=0 B. f[0][0]=0 C. f[0][0]=1 D. f[1][1]=0

1. ②处应填（ ）

A. i B. a[i] C. 1 D. 0

1. ③处应填（ ）
2. max(f[k][j],f[k-1][j-a[i]]+a[i])
3. max(f[k][j],f[k-1][j-a[i]]+1)
4. f[k][j]+f[k-1][j-a[i]]
5. max(f[k][j],f[k-1][j-a[i]])
6. ④处应填（ ）
7. f[n/2][i]&&f[n/2+1][i]
8. f[n/2][i]||f[n/2+1][i]
9. f[n/2][i]||(f[n/2+1][i]&&n&1)
10. f[n/2][i]!=f[n/2+1][i]
11. ⑤处应填（ ）
12. abs(f[i][n/2]-s B. f[i][n]-s C. abs(i\*2-s) D. s+i

解析：f[i][j]表示i个人能否取到j值。注意分组人数与n的奇偶性相关。

1. 最小环问题：给定一张无向图，求图中一个至少包含3个点的环，环上的节点不重复，并且环上的边的长度之和最小。该问题称为无向图的最小环问题。在本题中，你需要输出最小环的方案，若最小环不唯一，输出任意一个均可。若无解。输出No solution.

图的节点数不超过100。

输入：第一行两个正整数n,m表示点数和边数。

接下来m行，每行三个正整数x，y，z，表示节点x，y之间有一条长度为z的边。

输出：一个最小环的方案，按环上顺序输出最小环上的点。若最小环不唯一，输出任意一个均可。若无解，输出 No solution。

1. #include<bits/stdc++.h>
2. #define MAXN 105
3. #define INF 0x3f3f3f3f
4. using namespace std;
5. inline int read(){
6. int x=0,f=1;
7. char ch=getchar();
8. while (ch<'0'||ch>'9'){
9. if (ch=='-') f=-1;
10. ch=getchar();
11. }
12. while (ch>='0'&&ch<='9'){
13. x=(x<<3)+(x<<1)+(ch^'0');
14. ch=getchar();
15. }
16. return x\*f;
17. }
18. static int stk[MAXN],top;
19. static int pos[MAXN][MAXN];*//表示i~j的中点节点*
20. #define Push(x) stk[++top]=(x);
21. void GetAns(int i,int j){
22. if (pos[i][j]==0) return;
23. GetAns(i,   ①    );
24. Push(pos[i][j]);
25. GetAns(pos[i][j],   ②    );
26. }
27. static int G[MAXN][MAXN],D[MAXN][MAXN];
28. int main(){
29. int n=read(),m=read();
30. memset(G,0x3f,sizeof(G));
31. memset(D,0x3f,sizeof(D));
32. for(register int i=1;i<=m;++i){
33. int u=read(),v=read();
34. D[v][u]=D[u][v]=G[u][v]=G[v][u]=min(G[u][v],read());
35. }
36. int ans=INF;
37. for (register int k=1;k<=n;++k){
38. for (register int i=1;i<k;++i){
39. for (register int j=i+1;j<k;++j){
40. if (D[i][j]==INF||G[j][k]==INF||G[k][i]==INF) continue;
41. if (D[i][j]+G[j][k]+G[k][i]<ans){
42. ans=    ③    ;
43. top=0;Push(i);GetAns(i,j);Push(j);Push(k);
44. }
45. }
46. }
47. for (register int i=1;i<=n;++i){
48. for (register int j=1;j<=n;++j){
49. if (    ④    ){
50. D[i][j]=D[i][k]+D[k][j];
51. pos[i][j]=k;
52. }
53. }
54. }
55. }
56. if (ans==INF) return puts("No solution."),0;
57. for (register int i=1;i<=top;++i)
58. printf("%d ",   ⑤    );
59. return 0;
60. }
61. ①处应填（ ）

A. j B. pos[i][j] C. i D. pos[j][i]

1. ②处应填（ ）

A. j B. pos[i][j] C. i D. pos[j][i]

1. ③处应填（ ）
2. D[i][j]+G[k][j]+G[i][k]
3. D[i][j]+G[j][k]+G[k][i]
4. D[i][k]+G[k][j]+G[i][j]
5. D[i][j]+G[j][i]+G[i][k]
6. ④处应填（ ）
7. D[k][j]>D[i][k]+D[k][j]
8. D[i][j]>D[i][k]+D[k][j]
9. D[i][j]<D[i][k]+D[k][j]
10. D[i][k]>D[i][k]+D[k][j]
11. ⑤处应填（ ）

A. pos[i][j] B. stk[i] C. pos[1][i] D. pos[i][1]

解析：Floyd求最小环