2023 年 12 月认证 C++ 七级真题解析

CCF 编程能力等级认证，英文名 Grade Examination of Software Programming（以下简称 GESP），由中国计算机学会发起并主办，是为青少年计算机和编程学习者提供学业能力验证的平台。GESP 覆盖中小学全学段，符合条件的青少年均可参加认证。GESP 旨在提升青少年计算机和编程教育水平，推广和普及青少年计算机和编程教育。

GESP 考察语言为图形化（Scratch）编程、Python 编程及 C++编程，主要考察学生掌握相关编程知识和操作能力，熟悉编程各项基础知识和理论框架，通过设定不同等级的考试目标，让学生具备编程从简单的程序到复杂程序设计的编程能力，为后期专业化编程学习打下良好基础。

本次为大家带来的是 2023 年 12 月份 C++ 七级认证真题解析。

一、单选题（每题 2 分，共 30 分）

题号	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
答案	B	D	B	C	D	D	B	C	B	C	D	C	B	D	B

1、定义变量 double x，如果下面代码输入为 100，输出最接近(　)。

```cpp
#include <iostream>
#include <string>
#include <cmath>
#include <vector>
using namespace std;

int main()
{
    double x;

    cin >> x;
    cout << log10(x) - log2(x)  << endl;

    cout << endl;
    return 0;
}
```

A. 0

B. -5

C. -8

D. 8

【答案】B

【解析】log10(x)表示 10 的多少次方是 x，log2(x)表示 2 的多少次方是 x，这里的 x 是输入的 100，所以，log10(100)=2，又因为 26=64，所以 log2(100)是 6.多，两者作差，约为-4.多，选 B。

2、对于下面动态规划方法实现的函数，以下选项中最适合表达其状态转移函数的为(　)。

```
1   int s[MAX_N], f[MAX_N][MAX_N];
2   int stone_merge(int n, int a[]) {
3       for (int i = 1; i <= n; i++)
4           s[i] = s[i - 1] + a[i];
5       for (int i = 1; i <= n; i++)
6           for (int j = 1; j <= n; j++)
7               if (i == j)
8                   f[i][j] = 0;
9               else
10                  f[i][j] = MAX_F;
11      for (int l = 1; l < n; l++)
12          for (int i = 1; i <= n - l; i++) {
13              int j = i + l;
14              for (int k = i; k < j; k++)
15                  f[i][j] = min(f[i][j], f[i][k] + f[k + 1][j] + s[j] - s[i - 1]);
16          }
17      return f[1][n];
18  }
```

A. $f(i,j) = \min_{i \le k < j}(f(i,j), f(i,k) + f(k+1,j) + s(j) - s(i-1))$

B. $f(i,j) = \min_{i \le k < j}(f(i,j), f(i,k) + f(k+1,j) + \sum_{k=i}^{j} a(k))$

C. $f(i,j) = \min_{i \le k \le j}(f(i,k) + f(k+1,j) + \sum_{k=i}^{j+1} a(k))$

D. $f(i,j) = \min_{i \le k < j}(f(i,k) + f(k+1,j)) + \sum_{k=i}^{j} a(k)$

【答案】D

【解析】首先看代码，s 数组是前缀和数组，f 数组是 dp 数组，初始化 f 数组为正无穷，只有 f[i][i]=0（1<=i<=n）的值为 0，接着进行了区间 dp，i 和 j 分别是区间 dp 的两个端点，k 是枚举的分界点，k 的取值范围是[i,j)，所以选项 C 错误，根据第 15 行转移方程，发现后面的 s[j]-s[i-1]是 a[i]+a[i+1]+…+a[j]的和，且与 k 无关，可以单独拎出来，所以转移方程为 f[i][j]=min(f[i][k]+f[k+1][j])+$\sum_{k=i}^{j} a(k)$ ，选项 D 正确。选项 A,B 的错误点在于 f(i,j)的初始值为正无穷，所以 f(i,j)是不参与转移方程的。

3、下面代码可以用来求最长上升子序列（LIS）的长度，如果输入是：5 1 7 3 5 9，则输出是(　)。

```cpp
int a[2023], f[2023];
int main()
{
    int n,i,j,ans = -1;

    cin>>n;
    for( i=1; i<=n; i++){
        cin >> a[i];
        f[i] = 1;
    }

    for( i=1; i<=n; i++)
        for( j=1; j<i; j++)
            if(a[j] < a[i])
                f[i] = max(f[i], f[j]+1);
    for( i=1; i<=n; i++){
        ans = max(ans, f[i]);
        cout << f[i] << "  ";
    }

    cout << ans << endl;
    return 0;
}
```

A. 9 7 5 1 1 9

B. 1 2 2 3 4 4

C. 1 3 5 7 9 9

D. 1 1 1 1 1 1

【答案】B

【解析】题目已经提示我们这是在求最长上升子序列，f 数组的含义是以 i 结尾的最长上升子序列长度，ans 是整个序列的最长上升子序列长度，代码中先依次输出了 f[1],f[2],…,f[n],最后再输出 ans，接着我们可以进行手算，1 7 3 5 9 序列的 f 值分别为 1,2,2,3,4,ans=4，所以正确答案为 B。

4、C++语言中，下列关于关键字 static 的描述不正确的是(　)。

A. 可以修饰类的成员函数。

B. 常量静态成员可以在类外进行初始化。

C. 若 a 是类 A 常量静态成员 ，则 a 的地址都可以访问且唯一。

D. 静态全局对象一定在 main 函数调用前完成初始化 ，执行完 main 函数后被析构。

【答案】C

【解析】static 是静态意思，可以修饰成员变量和成员方法，static 修饰成员变量表示该成员变量在内存中只存储一份，可以被共享访问，修改。选项 C 中 a 的地址都可以访问是不对的，所以本题选 C。

5、G 是一个非连通无向图，共有 28 条边，则该图至少有(　)个顶点。

A. 6

B. 7

C. 8

D. 9

【答案】D

【解析】注意到题目里说的是非连通无向图，那么在同样的点数 n 下，为了有尽量多的边，可以分为两张连通图，一张 n-1 个点的完全图，另一张只有单独一个点，手算后可以发现，8 个点的完全图有 8*7/2=28 个点，正好满足题目要求，所以总点数为 9 个，选 D。

6、哈希表长 31，按照下面的程序依次输入4 17 28 30 4，则最后的 4 存入哪个位置？　(　　)

```cpp
#include <iostream>
#include <string>
#include <cmath>
#include <vector>
using namespace std;

const int N=31;
int htab[N],flag[N];
int main()
{
    int n,x,i,j,k;

    cin >> n;
    for(i=0; i<n; i++){
        cin >> x;
        k=x%13;
        while(flag[k]) k = (k+1)%13;
        htab[k]=x;
        flag[k]=1;
    }

    for(i=0; i<N; i++)
        cout << htab[i] << "  ";

    cout << endl;
    return 0;
}
```

A. 3

B. 4

C. 5

D. 6

【答案】D

【解析】题目提示我们这是哈希表，根据代码，发现是按照%13 进行哈希并且在发生冲突的情况下， 对应放到下一个位置，我们依次计算 17 28 30 4 会放置在什么位置，17 放置在 4,28 放置在 2,30 本来放置在 4，但是发生冲突，最终放置在 5,4 本来放置在 4，但是 4 和 5 都被占用了，所以最终放置在 6，选 D。

7、某二叉树 T 的先序遍历序列为：{A B D F C E G H} ，中序遍历序列为：{B F D A G E H C} ，则下列说法中正确的是(　)。

A. T 的度为 1

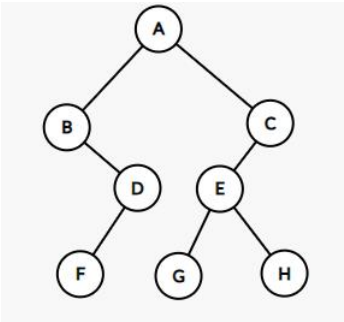
B. T 的高为 4

C. T 有 4 个叶节点

D. 以上说法都不对

【答案】B

【解析】先序遍历是根左右，中序遍历是左根右，首先可以根据先序遍历和中序遍历画出完整的树，如下图：



所以正确答案为 B。

8、下面代码段可以求两个字符串 s1 和 s2 的最长公共子串（LCS），下列相关描述不正确的是（ ）。

```
while (cin >> s1 >> s2)
{
    memset(dp, 0, sizeof(dp));
    int n1 = strlen(s1), n2 = strlen(s2);
    for (int i = 1; i <= n1; ++i)
        for (int j = 1; j <= n2; ++j)
            if (s1[i - 1] == s2[j - 1])
                dp[i][j] = dp[i - 1][j - 1] + 1;
            else
                dp[i][j] = max(dp[i - 1][j], dp[i][j - 1]);
    cout << dp[n1][n2] << endl;
}
```

A. 代码的时间复杂度为 $O(n^2)$

B. 代码的空间复杂度为 $O(n^2)$

C. 空间复杂度已经最优

D. 采用了动态规划求解

【答案】C

【解析】题目告诉我们代码是在求解最长公共子串，代码中使用了双重 for 循环，且循环范围为[1~n1]以及[1~n2]，所以选项 A 正确，使用了二维数组 dp，两维的长度也均为字符串长度，所以选项 B 正确，空间复杂度还可以使用滚动数组进一

步优化为 O(n)，所以选项 C 错误，本题选 C，选项 D 正确，代码中使用的正是动态规划算法。

9、图的广度优先搜索中既要维护一个标志数组标志已访问的图的结点，还需哪种结构存放结点以实现遍历？(　　)

A. 双向栈

B. 队列

C. 哈希表

D. 堆

【答案】B

【解析】图的广度优先搜索是从若干点出发，依次向外进行逐层扩展的算法，使用队列存放待遍历节点，本题选 B。

10、对关键字序列 {44， 36，23， 35， 52，73，90， 58} 建立哈希表 ，哈希函数为 h(k)=k%7 ，执行下面的 Insert 函数 ，则等概率情况下的平均成功查找长度（即查找成功时的关键字比较次数的均值）为(　)。

```
#include <iostream>
#include <string>
#include <cmath>
#include <vector>
using namespace std;

typedef struct Node{
    int data;
    struct Node *next;
}Node;
Node* hTab[7];
int key[]={44, 36, 23, 35, 52, 73, 90, 58, 0};
void Insert()
{
    int i,j;
    Node *x;

    for(i=0; key[i];i++){
        j = key[i] % 7;
        x=new Node;
        x->data = key[i];
        x->next = hTab[j];
        hTab[j] = x;
    }

    return;
}
```

A. 7/8

B. 1

C. 1.5

D. 2

【答案】C

【解析】代码采用链地址法来存储哈希，即将所有哈希地址相同的记录都链接在同一链表中，哈希方式为%7，我们依次对每个元素进行判断：44， 36，23， 35，52，73，90， 58，每个数字的哈希地址分别是 2,1,2,0,3,3,6,2，即哈希值为 0~6 的元素个数分别有 1,1,3,2,0,0,1，对于之前的 8 个数字，它们查找成功的次数分别是 1,1,2,1,1,2,1,3，总次数为 12 次，平均次数=12/8=1.5 次，答案为 C。

11、学生在读期间所上的某些课程中需要先上其他的课程 ，所有课程和课程间的先修关系构成一个有向图 G，有向边<U, V>表示课程 U 是课程 V 的先修课，则要找到某门课程 C 的全部先修课下面哪种方法不可行？ （ ）

A. BFS 搜索

B. DFS 搜索

C. DFS+BFS

D. 动态规划

【答案】D

【解析】查询有向图上有多少个点能够到达该点，可以在反图上进行搜索，所以选项 A,B,C 都可以，正确答案是 D。

12、一棵完全二叉树有 2023 个结点 ，则叶结点有多少个？ （ ）

A. 1024

B. 1013

C. 1012

D. 1011

【答案】C

【解析】设一棵完全二叉树有 k 层，则前 k-1 都是满二叉树，第 k 层的节点需要从左往右排列，那么第 1 层有 1 个节点，第 2 层有 2 个节点，第 3 层有 4 个节点。。。第 9 层有 512 个节点，此时总节点个数为 1023，第 10 层放置剩余的 1000 个节点，那么叶节点个数为第 10 层的 1000 个节点，再加上第 9 层除去被第 10 层消耗的 500 个节点外剩余的 12 个节点，总共为 1012 个，选 C 。也可以根据完全二叉树的节点编号性质来计算，即：第 2023 号结点的双亲是最后一个非叶结点，序号是 2023/2=1011，所以叶节点个数为：2023-1011=1012。

13、用下面的邻接表结构保存一个有向图 G，InfoType 和 VertexType 是定义好的类。设 G 有 n 个顶点、e 条弧，则求图 G 中某个顶点 u （其顶点序号为 k）的度的算法复杂度是(　)。

```
typedef struct ArcNode{
    int             adjvex;    // 该弧所指向的顶点的位置
    struct ArcNode  *nextarc; // 指向下一条弧的指针
    InfoType        *info;     // 该弧相关信息的指针
} ArcNode;
typedef struct VNode{
    VertexType  data;       // 顶点信息
    ArcNode     *firstarc; // 指向第一条依附该顶点的弧
} VNode, AdjList[MAX_VERTEX_NUM];
typedef struct{
    AdjList vertices;
    int     vexnum, arcnum;
    int     kind;           // 图的种类标志
} ALGraph;
```

A. O(n)

B. O(e)

C. O(n ＋e)

D. O(n + 2＊8)

【答案】B

【解析】代码中使用了邻接表来存储边的信息，查找某个点的度时需要计算出度和入度。出度直接从该点出发，遍历该点出发的边即可。同时查询入度，可以在反图上进行类似操作，总复杂度为 O(e),选 B。也可以遍历整个邻接表，包含点顶点 u 的弧的数目就是该顶点的度。

14、给定一个简单有向图 G，判断其中是否存在环路的下列说法哪个最准确？（ ）

A. BFS 更快

B. DFS 更快

C. BFS 和 DFS一样快

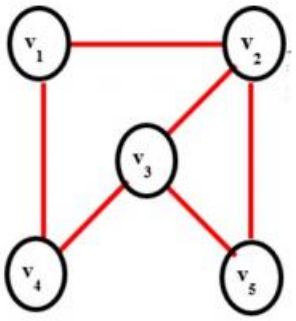
D. 不确定

【答案】D

【解析】对于不同的图，BFS 和 DFS 的效率也不一样，有可能 DFS 更快，也有可能 BFS 更快，所以本题正确答案为 D。

15、从顶点 v1 开始遍历下图 G 得到顶点访问序列 ，在下面所给的 4 个序列中符合广度优先的序列有几个？（ ）

{v1 v2 v3 v4 v5} ， {v1 v2 v4 v3 v5} ， {v1 v4 v2 v3 v5} ， {v1 v2 v4 v5 v3}



A. 4

B. 3

C. 2

D. 1

【答案】B

【解析】广度优先遍历会首先搜索和 s 距离为 k 的所有顶点，然后再去搜索和 s 距离为 k+1 的其他顶点，所以第 1 个序列不是广度优先搜索的序列，因为 v3 和 v1 的距离超过了 v4 和 v1 的距离，但是序列中 v3 确排在 v4 前面，其余 3 个序列都是广度优先搜索的序列，本题选 B。

二、判断题（每题 2 分，共 20 分）

题号	1	2	3	4	5	6	7	8	9	10
答案	√	√	×	√	√	×	×	×	√	×

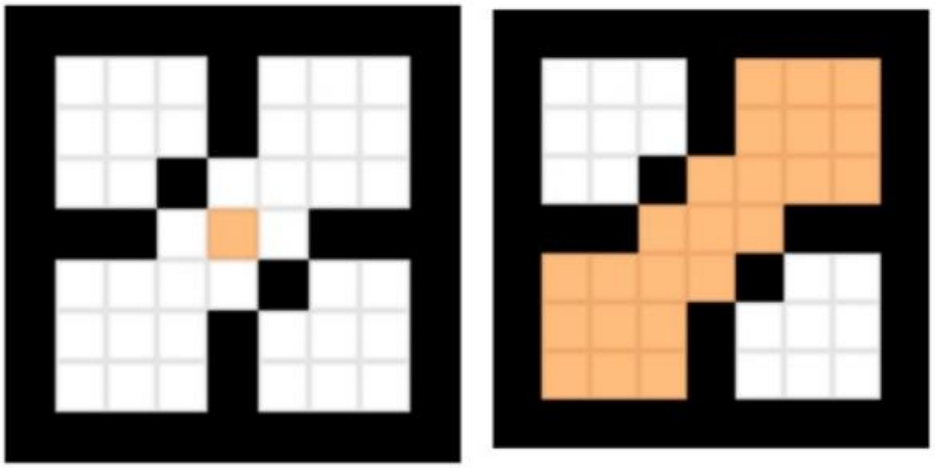
1、 小杨这学期准备参加 GESP 的 7 级考试 ，其中有关于三角函数的内容 ，他能够通过下面的代码找到结束循环的角度值。( )

```
int main()
{
    double x;

    do{
        cin >> x;
        x=x/180*3.14;
    }while(int(sin(x)*sin(x)+cos(x)*cos(x)) == 1);
    cout << "//" << sin(x) << "   " << cos(x);

    cout << endl;
    return 0;
}
```

【答案】正确

【解析】正确，代码将输入的角度转换成弧度，虽然对于任意的弧度，数学上均有，但 int（）转换对某些 x 可能出现截断的情况，能够导致循环结束。

2、小杨在开发画笔刷小程序(applet)，操作之一是选中黄颜色，然后在下面的左图的中间区域双击后，就变成了右图。这个操作可以用图的泛洪算法来实现。( )



【答案】正确

【解析】泛洪算法是从某个点出发，向周边相邻的区域进行扩展，和操作的要求是一致的，正确。

3、假设一棵完全二叉树共有N个节点 ，则树的深度为 log(N)+1。( )

【答案】错误

【解析】树的深度应该是$[\log_2 N+1]$,直接写 log 会以 e 为底，错误。

4、给定一个数字序列 A1，A2，A3， ...，An，要求 i 和 j（1<=i<=j<=n)，使 A i+...+Aj 最大，可以使用动态规划方法来求解 。( )

【答案】正确

【解析】问题为最大子段和，动态规划的经典例题，设 f[i]为以 i 结尾的子段最大值，则 f[i]=max(a[i],f[i-1]+a[i]);正确。

5、若变量 x 为 double 类型正数 ，则 log(exp(x)) > log10(x) 。( )

【答案】正确

【解析】式子的左侧 exp(x),是,所以 log(exp(x))就等于 x，等价于询问对于任意大于 0 的正实数 x，是否有 x>log10(x),当 0<x<1 时，log10(x)为负数，必然成立，当 x==1 时，log10(x)=0,成立，当 x>1 时，log10(x)的增长远远慢于 x 的增长，也成立，所以 x>log10(x)成立，正确。

6、简单有向图有 n 个顶点和 e 条弧 ，可以用邻接矩阵或邻接表来存储 ，二者求节点 u 的度的时间复杂度一 样 。( )

【答案】错误

【解析】错误，邻接矩阵求节点 u 的度时间复杂度为 O(n),而邻接表为 O(e)。

7、某个哈希表键值 x 为整数，为其定义哈希函数 H(x)=x%p，则 p 选择素数时不会产生冲突 。( )

【答案】错误

【解析】错误，设 p 为 7，则键值 14 和 21 的 hash 值相同，产生了冲突。

8、动态规划只要推导出状态转移方程，就可以写出递归程序来求出最优解。( )

【答案】错误

【解析】错误，用递归法求解动态规划方程会造成重复计算，可能导致超时。另外，动态规划算法的核心是状态转移方程，但同时也需要定义状态、初始条件和边界条件等。

9、广度优先搜索（BFS）能够判断图是否连通 。( )

【答案】正确

【解析】正确，BFS 是图论中遍历图的算法，可以从任意一个点出发进行 BFS，记录遍历过程中经过的不同点的个数，若不等于总点数，则说明图不连通。

10、在 C++中，如果定义了构造函数，则创建对象时先执行完缺省的构造函数，再执行这个定义的构造函数。( )

【答案】错误

【解析】错误，创建对象时最多只会执行一个构造函数。

三、编程题（每题 25 分，共 50 分）

题号	1	2
答案		

1、商品交易

问题描述

市场上共有 N 种商品，编号从 0至N-1，其中，第 i 种商品价值 vi 元。

现在共有 M 个商人，编号从 0至M - 1。在第 j 个商人这，你可以使用第 x_j 种商品交换第 y_j 种商品。每个商人都会按照商品价值进行交易，具体来说，如果 $v_{xj} > v_{yj}$ ，他将会付给你 $v_{yj} - v_{xj}$ 元钱；否则，那么你需要付给商人 $v_{xj} - v_{yj}$ 元钱。

除此之外，每次交易商人还会收取 1 元作为手续费，不论交易商品的价值孰高孰低。

你现在拥有商品 a ，并希望通过一些交换来获得商品 b 。请问你至少要花费多少钱？（当然，这个最小花费也可能是负数，这表示你可以在完成目标的同时赚取一些钱。）

输入描述

第一行四个整数 N,M,a,b，分别表示商品的数量、商人的数量、你持有的商品以及你希望获得的商品。保证 $0 \le a, b < N$、保证 $a \ne b$。

第二行 N 个用单个空格隔开的正整数 $v_0, v_1, \dots, v_{N-1}$，依次表示每种商品的价值。保证 $1 \le v_i < 10^9$ 。

接下来 M 行，每行两个整数 x_j, y_j，表示第 j 个商人愿意使用第 x_j 种商品交换第 y_j 种商品。保证 $0 \le x_j$ ， $y_j < N$ 保证 $x_j \ne y_j$ 。

输出描述

输出一行一个整数，表示最少的花费。特别地，如果无法通过交换换取商品 ，请输出 No solution

特别提醒

在常规程序中，输入、输出时提供提示是好习惯。但在本场考试中，由于系统限定，请不要在输入、输出中附带任何提示信息。

样例输入 1

```
3 5 0 2
1 2 4
1 0
2 0
0 1
2 1
1 2
```

样例输出 1

```
5
```

样例解释 1

可以先找 2 号商人，花 2-1=1 元的差价以及 1 元手续费换得商品 1，再找 4 号商人，花 4-2=2 元的差价以及 1 元手续费换得商品 2。总计花费 1+1+2+1=5 元。

样例输入 2

```
3 3 0 2
100 2 4
0 1
1 2
0 2
```

样例输出 2

```
-95
```

样例解释 2

可以找 2 号商人，直接换得商品 2 的同时，赚取 100 - 4 = 96 元差价，再支付 1 元手续费，净赚 95 元。

也可以先找 0 号商人换取商品 1，再找 1 号商人换取商品 2，不过这样只能赚 94 元。

样例输入 3

```
4 4 3 0
1 2 3 4
1 0
0 1
3 2
2 3
```

样例输出 3

```
No solution
```

数据规模

对于 30%的测试点，保证 $N \le 10$，$M \le 20$。

对于 70%的测试点，保证 $N \le 10^3$，$M \le 10^4$。

对于 100%的测试点，保证 $N \le 10^5$，$M \le 2 \times 10^5$。

【解题思路】发现不管通过什么方式换，最终因为价值不同而花费的金币数都是固定的，即 v[b]-v[a],a 为持有的商品，b 为希望获得商品，唯一的区别是每次交易都要额外支付 1 块钱，所以需要最小化交易次数，我们把每件商品看作 1 个点，如果某件商品 x 能够换为某件商品 y，则让 x 和 y 连一条边，我们的目标是从商品 a 出发尽快到达商品 b，即经过的边的数量尽量少，可以通过 bfs 求经过

的最少的边数，设这个值为 min_dist[dst]，那么最终答案为 min_dist[dst]+v[b]-v[a]。

【参考程序】

```cpp
#include <iostream>
#include <cstring>
#include <vector>

using namespace std;

const int max_n = 1e5 + 10;

int n;
vector<int> edge[max_n];
int val[max_n];

int min_dist[max_n];
int queue[max_n], qh, qt;

void bfs(int src) {
    qh = qt = 0;
    memset(min_dist, 127, sizeof(min_dist));
    queue[++qt] = src;
    min_dist[src] = 0;
    while (qh < qt) {
        int u = queue[++qh];
        for (auto v: edge[u]) {
            if (min_dist[u] + 1 < min_dist[v]) {
                min_dist[v] = min_dist[u] + 1;
                queue[++qt] = v;
            }
        }
    }
}

int main() {
    int m, src, dst;
    ios::sync_with_stdio(false);
    cin >> n >> m >> src >> dst;
    for (int i = 0; i < n; ++i) {
        cin >> val[i];
    }
    for (int i = 0; i < m; ++i) {
        int x, y;
        cin >> x >> y;
        edge[x].push_back(y);
    }

    bfs(src);
    if (min_dist[dst] > n) {
        cout << "No solution\n";
    }
    else {
        cout << min_dist[dst] - val[src] + val[dst] << '\n';
    }

    return 0;
}
```

2、纸牌游戏

问题描述

你和小杨在玩一个纸牌游戏。

你和小杨各有 3 张牌，分别是 0、1、2。你们要进行 N 轮游戏，每轮游戏双方都要出一张牌，并按 1 战胜 0，2 战胜 1，0 战胜 2 的规则决出胜负。第 i 轮的

胜者可以获得 $2a_i$ 分，败者不得分，如果双方出牌相同，则算平局，二人都可获得 a_i 分 (i=1,2,...,N)。

玩了一会后，你们觉得这样太过于单调，于是双方给自己制定了不同的新规则。小杨会在整局游戏开始前确定自己全部 n 轮的出牌，并将他的全部计划告诉你:而你从第 2 轮开始，要么继续出上一轮出的牌，要么记一次“换牌”。游戏结束时，你换了 t 次牌，就要额外扣 b1 + ... + bt 分。

请计算出你最多能获得多少分。

输入描述

第一行一个整数 N，表示游戏轮数。

第二行 N 个用单个空格隔开的非负整数 $a_1,...,a_N$，意义见题目描述。

第三行 N-1 个用单个空格隔开的非负整数 $b_1,...,b_{N-1}$，表示换牌的罚分，具体含义见题目描述。由于游戏进行 N 轮，所以你至多可以换 N-1 次牌。

第四行 N 个用单个空格隔开的整数 $c1,...,c_N$，依次表示小杨从第 1 轮至第 N 轮出的牌。保证 $c_i \in 0,1,2$。

输出描述

一行一个整数，表示你最多获得的分数。

特别提醒

在常规程序中，输入、输出时提供提示是好习惯。但在本场考试中，由于系统限定，请不要在输入、输出中附带任何提示信息。

样例输入 1

```
4
1 2 10 100
1 100 1
1 1 2 0
```

样例输出 1

```
219
```

样例解释

你可以第 1 轮出 0，并在第 2,3 轮保持不变，如此输掉第 1,2 轮，但在第 3 轮中取胜，获得 2x 10 = 20 分;随后,你可以在第 4 轮中以扣 1 分为代价改出 1，

并在第 4 轮中取得胜利，获得 2 100 = 200 分。如此，你可以获得最高的总分 20 +200-1=219。

样例输入 2

```
6
3 7 2 8 9 4
1 3 9 27 81
0 1 2 1 2 0
```

样例输出 2

```
56
```

数据规模

对于 30%的测试点，保证 $N \le 15$。

对于 60%的测试点，保证 $N \le 100$。

对于所有测试点，保证 $N \le 1{,}000$; 保证 $0 \le a_i, b_i \le 10^6$。

【解题思路】考虑使用 dp 算法，设 dp[i][j][k]表示前 i 轮中，第 i 轮出的牌为 j(0<=j<=2),已经换过 k 次牌的最大得分，则

$$dp[i][j][k]=\max\begin{cases} dp[i-1][j][k]+result(j,c[i])*a[i] \\ dp[i-1][j'][k-1]+result(j,c[i])*a[i]-b[k]\ (j!=j') \end{cases}$$

这里的 a 数组是奖励的得分，b 数组是换牌的惩罚，c 数组是小杨的出牌，result(x,y)表示出牌为 x 时和 y 的胜负情况(胜利返回 2，平局返回 1，失败返回 0)

上面一行表示当前轮出牌和上一轮相同

下面一行表示不同，需要额外付出 b[k]的代价

最终答案为 $\max\limits_{j=0\sim2,k=0\sim n-1} dp[n][j][k]$

【参考程序】

```cpp
#include <iostream>
#include <cstring>
#include <vector>

using namespace std;

const int max_n = 1005;

int n;
int a[max_n], b[max_n], c[max_n];

int dp[3][max_n];

int result(int x, int y) {
    if (x == y + 1 || x == y - 2) return 2;
    if (x == y) return 1;
    return 0;
}

int main() {
    ios::sync_with_stdio(false);
    cin >> n;
    for (int i = 1; i <= n; ++i) cin >> a[i];
    for (int i = 1; i < n; ++i) cin >> b[i];
    for (int i = 1; i <= n; ++i) cin >> c[i];
    memset(dp, 128, sizeof(dp));
    for (int k = 0; k < 3; ++k){
        dp[k][0] = result(k, c[1]) * a[1];
    }
    for (int i = 2; i <= n; ++i)
        for (int j = i - 1; j >= 0; --j)
            for (int k = 0; k < 3; ++k) {
                int curr_score = result(k, c[i]) * a[i];
                dp[k][j] = dp[k][j] + curr_score;
                if (j > 0) {
                    for (int l = 0; l < 3; ++l) {
                        dp[k][j] = max(dp[k][j], dp[l][j - 1] + curr_score - b[j]);
                    }
                }
            }
    int ans = -2e9, x = 0;
    for (int j = 0; j < n; ++j)
        for (int k = 0; k < 3; ++k) {
            ans = max(ans, dp[k][j]);
        }
    cout << ans << '\n';
    return 0;
}
```

【联系我们】

1. GESP 微信：关注“CCF GESP”公众号，将问题以文字方式留言即可得到回复。

2. GESP 邮箱：gesp@ccf.org.cn

注：请在邮件中详细描述咨询的问题并留下考生的联系方式及姓名、身份证号，以便及时有效处理。

3. GESP 电话：0512-67656856

咨询时间：周一至周五(法定节假日除外）: 上午 8:30-12:00；下午 13:00-17:30

GESP 第五期认证报名已启动，扫描下方二维码，关注 GESP 公众号即可报名