2010 北邮网研院复试上机题目：

第一题：查找

输入数组长度 n

输入数组 a[1...n]

输入查找个数 m

输入查找数字 b[1...m]

输出 YES or NO 查找有则 YES 否则 NO

如(括号内容为注释)

输入：

5(数组长度)

1 5 2 4 3(数组)

3(查找个数)

2 5 6(查找具体数字)

输出：

YES

YES

NO

```
 #include <stdio.h>
#include <stdlib.h>

int main()
{
   int n,m,i,j,a[2001]={0},b[2001]={0},flag=0;
   scanf("%d",&n);
   for(i=0;i<n;i++)
   scanf("%d",&a[i]);
   scanf("%d",&m);
   for(i=0;i<m;i++)
   scanf("%d",&b[i]);
   for(i=0;i<m;i++)
   {
            for(j=0;j<n;j++)
       {
            if(b[i]==a[j])
           {
            flag=1;
            printf("YES\n");
            break;
            }
```

```
        }
    if(flag!=1)
    printf("NO\n");
    flag=0;
    }
    // // system("PAUSE");
    return 0;
}
```

第二题：查找第 K 小数

查找一个数组的第 K 小的数，注意同样大小算一样大

如 2 1 3 4 5 2 第三小数为 3

如(括号内容为注释)

输入：

6(数组长度 n)

2 1 3 5 2 2(数组)

3(K 即为第三小数)

输出：

3

Code

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
    int n,k,i,j,a[1001],temp,m=1;
    scanf("%d %d",&n,&k);
    for(i=0;i<n;i++)
    scanf("%d",&a[i]);


    for(i=0;i<n-1;i++)
        for(j=0;j<n-i-1;j++)
            if(a[j]>a[j+1])
                {
                    temp=a[j];
                    a[j]=a[j+1];
                    a[j+1]=temp;
                    }

    for(i=0;i<n-1;i++)

    if((a[i]==a[i+1])&&(i<k))
```

```
    k++;

    printf("%d\n",a[k-1]);
```

```
// system("PAUSE");
return 0;
```

- }

第三题：打牌

牌只有 1 到 9，手里拿着已经排好序的牌 a，对方出牌 b，用程序判断手中牌是否能够压过对方出牌

规则：出牌牌型有 5 种

[1]一张 如 4 则 5...9 可压过

[2]两张 如 44 则 55，66，77，...，99 可压过

[3]三张 如 444 规则如[2]

[4]四张 如 4444 规则如[2]

[5]五张 牌型只有 12345 23456 34567 45678 56789 五个，后面的比前面的均大

压过输出 YES 否则 NO

如(括号内容为注释)

输入：

12233445566677(手中牌)

33(出牌)

输出：

YES

Problem Id: 1820

Submit time: 2010-04-25 20:28:36

User_id: jyjyjy1989

Memory:204K Time:19MS

Language:G++ Result:Accepted

- Code

```
#include <iostream>
  #include<string.h>
  using namespace std;

  int main(void)
```

```
{
    char str1[100],str2[5];
    scanf("%s",str1);
    char ch;ch=getchar();
    scanf("%s",str2);
    int i=0;int count[10]={0};
    while(str1[i]!='\n')
    {
    if((str1[i]-'1')==0) count[0]++;
    else if((str1[i]-'1')==1) count[1]++;
    else if((str1[i]-'1')==2) count[2]++;
    else if((str1[i]-'1')==3)  count[3]++;
    else if((str1[i]-'1')==4)  count[4]++;
    else if ((str1[i]-'1')==5) count[5]++;
    else if((str1[i]-'1')==6) count[6]++;
    else if((str1[i]-'1')==7) count[7]++;
    else if((str1[i]-'1')==8)  count[8]++;
    else break;
    i++;
    }
int s1,s2,s3,s4,s5;int flag=1;
if(strlen(str2)==1)
{  s1=*str2-'1';
for(;s1<9;s1++)
   if(count[s1+1]>0)
   {printf("YES\n",s1);flag=0;break;}
 }
else if(strlen(str2)==2) {s2=*str2-'1'; for(;s2<9;s2++)if(count[s2+1]>=2) {printf("YES\n",s2);flag=0;break;} }
else if(strlen(str2)==3) {s3=*str2-'1'; for(;s3<9;s3++)if(count[s3+1]>=3) {printf("YES\n");flag=0;break;} }
else if(strlen(str2)==4) {s4=*str2-'1'; for(;s4<9;s4++)if(count[s4+1]>=4) {printf("YES\n");flag=0;break;} }
else if(strlen(str2)==5){s5=*str2-'1'; for(;s5<9;s5++) if(count[s5+5]>0&&count[s5+1]>0&&count[s5+2]>0&&count[s5+3]>0&&count[s5+4]>0&&((s5+5)<9)) {printf("YES\n");flag=0;break;}}
if(flag==1) printf("NO\n");
    // system("PAUSE");
    return EXIT_SUCCESS;
}
```

第四题：树 查找

简单说就是一棵树，输出某一深度的所有节点，有则输出这些节点，无则输出 EMPTY，具

体描述得借助图形比较好，懒得写了，基本就是这个样子的。

2010 北邮计算机学院上机题目回忆版

题目大意（回忆版）：第一行输入一个数，为 n，第二行输入 n 个数，这 n 个数中，如果偶数比奇数多，输出 NO，否则输出 YES。

Sample:

Input:
5
1 2 3 4 5

Output:
YES

Problem Id: 1814
Submit time: 2010-04-10 14:06:48
User_id: bupt111310352
Memory:72K Time:17MS
Language:G++ Result:Accepted

Code
*/

```
#include <stdio.h>

int main()
{
    // 所有数的数目
    int m_Num_Count;
    scanf("%d", &m_Num_Count);

    // 输入数
    int m_Input_Num;

    // 偶数 奇数的个数
    int m_Oushu_Count = 0, m_Jishu_Count = 0;

    // 循环输入
```

```
    while(m_Num_Count > 0)
    {
        scanf("%d", &m_Input_Num);
        getchar();

        // 判断奇偶并计数
        if( (m_Input_Num % 2) == 0 )
            ++ m_Oushu_Count;
        else
            ++ m_Jishu_Count;

        -- m_Num_Count;
    }

    // 输出
    if(m_Jishu_Count >= m_Oushu_Count)
        printf("YES\n");
    else
        printf("NO\n");

    return 0;
}

/*
```

题目大意（回忆版）：第一行输入一个数 n，1 <= n <= 1000，下面输入 n 行数据，每一行有两个数，分别是 x y。输出一组 x y，该组数据是所有数据中 x 最小，且在 x 相等的情况下 y 最小的。

Sample:

Input:
5
3 3
2 2
5 5
2 1
3 6

Output:
2 1

Problem Id: 1815
Submit time: 2010-04-10 14:24:39
User_id: bupt111310352

Memory:104K Time:19MS
Language:G++ Result:Accepted

Code
*/

```
#include <stdio.h>

typedef struct LNode
{
    int m_X;          // x 坐标值
    int m_Y;          // y 坐标值
    bool m_Used;     // 该位是否被使用
};

int main()
{
    // 样例个数
    int m_Case_Count;
    scanf( "%d", &m_Case_Count );

    // 因为 1 <= N <= 1000，建结点
    LNode m_Case[ 1000 ];

    // 循环数
    int m_Cycle;

    // 最小的 x, y 值，并初始化，1 <= x,y <= 1000
    int m_X_Min = 1000, m_Y_Min = 1000;

    // 初始化结点被使用状态
    for( m_Cycle = 0; m_Cycle < 1000; ++ m_Cycle )
        m_Case[ m_Cycle ].m_Used = false;

    // 初始化循环数，下面使用
    m_Cycle = 0;

    // 循环输入样例
    while( m_Case_Count > 0 )
    {
        // 输入坐标值
        scanf( "%d %d", &m_Case[ m_Cycle ].m_X, &m_Case[ m_Cycle ].m_Y );

        // 置状态值，并且 m_Cycle ++
```

```
            m_Case[ m_Cycle ++ ].m_Used = true;

            --  m_Case_Count;
        }

        // 循环找最小的 x,y 值
        for( m_Cycle = 0; ( m_Cycle < 1000 ) && ( m_Case[ m_Cycle ].m_Used ) ; ++ m_Cycle )
        {
            if( m_Case[ m_Cycle ].m_X  < m_X_Min )
            {
                // 如果找到更小的 x 值，记录
                m_X_Min = m_Case[ m_Cycle ].m_X;
                m_Y_Min = m_Case[ m_Cycle ].m_Y;
            }
            else if ( m_Case[ m_Cycle ].m_X  == m_X_Min )
            {
                // 如果 x 值相等，判断 y 值，如果在相等的 x 值的条件下，找到更小的的 y 值，
记录 y
                if( m_Case[ m_Cycle ].m_Y  < m_Y_Min )
                    m_Y_Min = m_Case[ m_Cycle ].m_Y;
            }
        }

        // 输出找到的最小值
        printf("%d %d\n", m_X_Min, m_Y_Min );

        return 0;
}
```

/*
题目大意（回忆版）：该题是要翻转数据。首先输入一个 5 * 5 的数组，然后输入一行，这一行有四个数，前两个代表操作类型，后两个数 x y 代表需操作数据为以 x y 为左上角的那几个数据。

操作类型有四种：
1 2 表示：90 度，顺时针，翻转 4 个数
1 3 表示：90 度，顺时针，翻转 9 个数
2 2 表示：90 度，逆时针，翻转 4 个数
2 3 表示：90 度，逆时针，翻转 9 个数

Sample:

Input:
1 2 3 4 5
6 7 8 9 10
11 12 13 14 15
16 17 18 19 20
21 22 23 24 25
1 3 1 1

Output:
11 6 1 4 5
12 7 2 9 10
13 8 3 14 15
16 17 18 19 20
21 22 23 24 25

Problem Id: 1816
Submit time: 2010-04-10 15:06:04
User_id: bupt111310352
Memory:88K Time:14MS
Language:G++ Result:Accepted

解题思路：--。自己细心点就行。。。

Code
*/

```
#include <stdio.h>

int main()
{
    // 输入数据
    int m_Case[ 5 ][ 5 ];
    scanf( "%d %d %d %d %d", &m_Case[ 0 ][ 0 ], &m_Case[ 0 ][ 1 ], &m_Case[ 0 ][ 2 ], &m_Case[ 0 ][ 3 ], &m_Case[ 0 ][ 4 ]);
    scanf( "%d %d %d %d %d", &m_Case[ 1 ][ 0 ], &m_Case[ 1 ][ 1 ], &m_Case[ 1 ][ 2 ], &m_Case[ 1 ][ 3 ], &m_Case[ 1 ][ 4 ]);
    scanf( "%d %d %d %d %d", &m_Case[ 2 ][ 0 ], &m_Case[ 2 ][ 1 ], &m_Case[ 2 ][ 2 ], &m_Case[ 2 ][ 3 ], &m_Case[ 2 ][ 4 ]);
    scanf( "%d %d %d %d %d", &m_Case[ 3 ][ 0 ], &m_Case[ 3 ][ 1 ], &m_Case[ 3 ][ 2 ], &m_Case[ 3 ][ 3 ], &m_Case[ 3 ][ 4 ]);
    scanf( "%d %d %d %d %d", &m_Case[ 4 ][ 0 ], &m_Case[ 4 ][ 1 ], &m_Case[ 4 ][ 2 ], &m_Case[ 4 ][ 3 ], &m_Case[ 4 ][ 4 ]);

    // 输入操作
```

```
int m_Op_First, m_Op_Second, m_X, m_Y;
scanf( "%d %d %d %d", &m_Op_First, &m_Op_Second, &m_X, &m_Y );

// 临时数据
int m_Temp_First, m_Temp_Second;

if( ( m_Op_First == 1 ) && ( m_Op_Second == 2 ) )
{
	// 如果是 1 2，翻转 4 个数，90 度，顺时针
	m_Temp_First = m_Case[ m_X - 1 ][ m_Y - 1 ];
	m_Case[ m_X - 1 ][ m_Y - 1 ] = m_Case[ m_X ][ m_Y - 1 ];
	m_Case[ m_X ][ m_Y - 1 ] = m_Case[ m_X ][ m_Y ];
	m_Case[ m_X ][ m_Y ] = m_Case[ m_X - 1 ][ m_Y ];
	m_Case[ m_X - 1 ][ m_Y ] = m_Temp_First;
}
else if( ( m_Op_First == 1 ) && ( m_Op_Second == 3 ) )
{
	// 如果是 1 3，翻转 9 个数，90 度，顺时针
	m_Temp_First = m_Case[ m_X - 1 ][ m_Y - 1 ]; // 00
	m_Temp_Second = m_Case[ m_X ][ m_Y - 1 ];	// 10
	m_Case[ m_X - 1 ][ m_Y - 1 ] = m_Case[ m_X + 1 ][ m_Y - 1 ];; // 00 = 20
	m_Case[ m_X ][ m_Y - 1 ] = m_Case[ m_X + 1 ][ m_Y ];		// 10 = 21
	m_Case[ m_X + 1 ][ m_Y - 1 ] = m_Case[ m_X + 1 ][ m_Y + 1 ];	// 20 = 22
	m_Case[ m_X + 1 ][ m_Y ] = m_Case[ m_X ][ m_Y + 1 ];		// 21 = 12
	m_Case[ m_X + 1 ][ m_Y + 1 ] = m_Case[ m_X - 1 ][ m_Y + 1 ];	// 22 = 02
	m_Case[ m_X ][ m_Y + 1 ] = m_Case[ m_X - 1 ][ m_Y ];		// 12 = 01
	m_Case[ m_X - 1 ][ m_Y + 1 ] = m_Temp_First;			// 02 = 00
	m_Case[ m_X - 1 ][ m_Y ] = m_Temp_Second;			// 01 = 10
}
else if( ( m_Op_First == 2 ) && ( m_Op_Second == 2 ) )
{
	// 如果是 2 2，翻转 4 个数，90 度，逆时针
	m_Temp_First = m_Case[ m_X - 1 ][ m_Y - 1 ];
	m_Case[ m_X - 1 ][ m_Y - 1 ] = m_Case[ m_X - 1 ][ m_Y ];
	m_Case[ m_X - 1 ][ m_Y ] = m_Case[ m_X ][ m_Y ];
	m_Case[ m_X ][ m_Y ] = m_Case[ m_X ][ m_Y - 1 ];
	m_Case[ m_X ][ m_Y - 1 ] = m_Temp_First;
}
else if( ( m_Op_First == 2 ) && ( m_Op_Second == 3 ) )
{
	// // 如果是 2 3，翻转 9 个数，90 度，逆时针
	m_Temp_First = m_Case[ m_X - 1 ][ m_Y - 1 ];			// 00
	m_Temp_Second = m_Case[ m_X - 1 ][ m_Y ];			// 01
	m_Case[ m_X - 1 ][ m_Y - 1 ] = m_Case[ m_X - 1 ][ m_Y + 1 ]; // 00 = 02
```

```
        m_Case[ m_X - 1 ][ m_Y ] = m_Case[ m_X ][ m_Y + 1 ];              // 01 = 12
        m_Case[ m_X - 1 ][ m_Y + 1 ] = m_Case[ m_X + 1 ][ m_Y + 1 ]; // 02 = 22
        m_Case[ m_X ][ m_Y + 1 ] = m_Case[ m_X + 1 ][ m_Y ];              // 12 = 21
        m_Case[ m_X + 1 ][ m_Y + 1 ] = m_Case[ m_X + 1 ][ m_Y - 1 ]; // 22 = 20
        m_Case[ m_X + 1 ][ m_Y ] = m_Case[ m_X ][ m_Y - 1 ];              // 21 = 10
        m_Case[ m_X + 1 ][ m_Y - 1 ] = m_Temp_First;                              // 20 = 00
        m_Case[ m_X ][ m_Y - 1 ] = m_Temp_Second;                          // 10 = 01
    }

    // 输出
    printf( "%d %d %d %d %d\n", m_Case[ 0 ][ 0 ], m_Case[ 0 ][ 1 ], m_Case[ 0 ][ 2 ],
m_Case[ 0 ][ 3 ], m_Case[ 0 ][ 4 ]);
    printf( "%d %d %d %d %d\n", m_Case[ 1 ][ 0 ], m_Case[ 1 ][ 1 ], m_Case[ 1 ][ 2 ],
m_Case[ 1 ][ 3 ], m_Case[ 1 ][ 4 ]);
    printf( "%d %d %d %d %d\n", m_Case[ 2 ][ 0 ], m_Case[ 2 ][ 1 ], m_Case[ 2 ][ 2 ],
m_Case[ 2 ][ 3 ], m_Case[ 2 ][ 4 ]);
    printf( "%d %d %d %d %d\n", m_Case[ 3 ][ 0 ], m_Case[ 3 ][ 1 ], m_Case[ 3 ][ 2 ],
m_Case[ 3 ][ 3 ], m_Case[ 3 ][ 4 ]);
    printf( "%d %d %d %d %d\n", m_Case[ 4 ][ 0 ], m_Case[ 4 ][ 1 ], m_Case[ 4 ][ 2 ],
m_Case[ 4 ][ 3 ], m_Case[ 4 ][ 4 ]);

    return 0;
}

/*
// 我最纠结的题。前 3 个题一个小时。这个题一个小时都没有弄完。我实在是很郁闷啊。
当时真感觉自己想法太多了。还不如好好分析分析。这个题主要是要设 parent 指针。如果这
题还需要输出哈夫曼编码的话，则需要同时设 rchild 与 lchild 指针。太纠结了这道题。这都
没做出来，强烈 bs 自己。。。回家弄了会就弄出来了。。。
```

题目大意（回忆版）：哈夫曼树，第一行输入一个数 n，表示叶结点的个数。需要用这些叶结点生成哈夫曼树，根据哈夫曼树的概念，这些结点有权值，即 weight，题目需要输出所有结点的值与权值的乘积之和。

Sample:

Input:
5
1 2 2 5 9

Output:
37

解释：即生成如下图哈夫曼树，结点 1 的权值为 4，结点 2 的权值为 4，结点 2 的权值为 3，

结点 5 的权值为 2，结点 9 的权值为 1，和为 37

Problem Id: 1817
Submit time: 2010-04-14 10:49:17
User_id: bupt111310352
Memory:96K Time:16MS
Language:G++ Result:Accepted

Code
*/

```
#include <stdio.h>

// 结点的数据结构
typedef struct HNode
{
    unsigned int weight;        // 权值
    unsigned int parent;        // 父结点
};

// 在 m_Node 的 0 到 m_Cycle 之间寻找 parent 指针为 0 且最小的两个结点
// 并在数组最后存放这两个结点形成的父结点
void Select( HNode* m_Node, int m_Cycle )
{
    // 循环数
    int temp;

    // 记录下最小的两个数的位置
    int m_FirstMin_Pos = -1, m_SecondMin_Pos = -1;

    // 最小的两个数，初始赋值为题目所给最大值加 1
    int m_FirstMin = 1001, m_SecondMin = 1001;

    for( temp = 0; temp <= m_Cycle ; ++ temp )
    {
        if ( m_Node[ temp ].parent == 0 )
        {
            // 如果结点权值比 m_FirstMin 还小，
            // 则 FirstMin 将值传递给 m_Node_SecondMin, FirstMin 结点指向查出的最小值结点
            // 记录该结点位置
            if( m_Node[ temp ].weight < m_FirstMin )
            {
                m_SecondMin = m_FirstMin;
```

```
                m_SecondMin_Pos = m_FirstMin_Pos;
                m_FirstMin = m_Node[ temp ].weight;
                m_FirstMin_Pos = temp;
            }
            else
            {
                // 如果 点权值比 FirstMin 大，但比  m_Node_SecondMin 小，
                // 则 m_Node_SecondMin 指向 查出的最小值结点，并记录位置
                if( m_Node[ temp ].weight < m_SecondMin )
                {
                    m_SecondMin = m_Node[ temp ].weight;
                    m_SecondMin_Pos = temp;
                }
            }
        }
    }

    // 在 m_Cycle + 1 的位置为最小的两个结点的和生成的结点
    m_Node[ ++ m_Cycle ].weight = m_FirstMin + m_SecondMin;

    // 两个最小值结点的 parent 赋值
    m_Node[ m_FirstMin_Pos ].parent = m_Cycle;
    m_Node[ m_SecondMin_Pos ].parent = m_Cycle;
}

int main()
{
    // 叶结点的数目
    int m_Num_Count, m_Num_Count_Temp = 0;
    scanf( "%d", &m_Num_Count );

    // 最后形成的哈夫曼树总结点个数，为叶子结点个数 * 2 - 1
    int m_TotalNode_Count = 2 * m_Num_Count - 1;

    // 存哈夫曼树的数组
    HNode m_Node[ 2 * m_Num_Count - 1 ];

    // 临时输入数
    int m_TempInput;

    // 循环输入叶结点
    while ( m_Num_Count_Temp < m_Num_Count )
    {
        // 输入数据， m_Node 数组的前 m_Num_Count 个存放叶结点
```

```
        scanf( "%d", &m_TempInput );
        getchar();

        // 放入到数组
        m_Node[ m_Num_Count_Temp ].weight = m_TempInput;
        m_Node[ m_Num_Count_Temp ].parent = 0;

        ++ m_Num_Count_Temp;
    }

    // 循环数
    int m_Cycle;

    // 对后   m_Num_Count - 1  个结点初始化
    for( m_Cycle = m_Num_Count; m_Cycle < m_TotalNode_Count; ++ m_Cycle )
    {
         m_Node[ m_Cycle ].weight = 0;
         m_Node[ m_Cycle ].parent = 0;
    }

    for( m_Cycle = ( m_Num_Count - 1 ); m_Cycle < ( m_TotalNode_Count - 1 ); ++ m_Cycle )
    {
        // 在 m_Node 中从 0 到 m_Cycle 之间寻找 parent 指针为 0 且最小的两个结
点
        // 并在数组最后存放这两个结点形成的父结点
        Select( m_Node, m_Cycle );
    }

    // 结果，输出值
    int m_ResultValue = 0;

    // 临时结点
    HNode m_Node_Temp;

    // 结点在哈夫曼树中的长度
    int m_Length;

    for( m_Cycle = 0; m_Cycle < m_Num_Count; ++ m_Cycle )
    {
        // 长度置 0
        m_Length = 0;

        // 临时变量
        m_Node_Temp = m_Node[ m_Cycle ];
```

```
            // 找 parent 直至 parent 不为 0 ，同时计算长度
            while( m_Node_Temp.parent != 0 )
            {
                m_Node_Temp = m_Node[ m_Node_Temp.parent ];
                ++ m_Length;
            }

            // 计算该结点的权值 * 长度，加到总输出值中
            m_ResultValue += ( m_Node[ m_Cycle ].weight * m_Length );
        }

        // 输出
        printf( "%d\n", m_ResultValue );

        return 0;
}
```