**思考题**

#include<vector>

#include <iostream>

#include<stdio.h>

using namespace std;

int main()

{

int n;

vector<int> flower(1000);

cin >> n;

if (n == 0)

{

return 0;

}

else if (n == 1 || n == 2)

{

printf("\nN应该大于等于3");

return 0;

}

for (int i = 0; i < n; i++)

{

cin>> flower[i];

}

//存入花盆信息

int left = 0;

int right = n - 1;

int temp;

//此while用于先将白色移到队伍末尾

while (left < right)

{

if ((flower[left] == 2) && (flower[right] == 1 || flower[right] == 3))

{

temp = flower[left];

flower[left] = flower[right];

flower[right] = temp;

}

else if (flower[left] == 1 || flower[left] == 3)

left++;

else if (flower[right] == 2)

right--;

}

left = 0;

right = right - 1;

//再将白色之前的红蓝两色排好顺序

while (left < right)

{

if (flower[left] == 3 && flower[right] == 1)

{

temp = flower[left];

flower[left] = flower[right];

flower[right] = temp;

left++;

right--;

}

else if (flower[left] == 1)

{

left++;

}

else if (flower[right] == 3)

{

right--;

}

}

//打印出来检查效果

for (int i = 0; i < n; i++)

{

printf( "%d" ,flower[i]);

}

return 0;

}

**作业一**

#include<vector>

#include <iostream>

#include<stdio.h>

using namespace std;

int flag(char a); //字母在字典中顺序

int compare(vector<char>& member1, vector<char>& member2); //比较位置，用于排序

int main()

{

vector<vector<char>> member;

member = { {'P','A','B'},{'5','C'},{'P','A','B','C'},{'C','X','Y'},{'C','R','S','I'},{'7' }, { 'B','8','9','9' }, { 'B','9' }};

vector<char> temp;

//冒泡排序

for (int i = 7; i >=0; i--)

{

for (int j = 0; j < i; j++)

{

if (compare(member[j], member[j + 1]) == 0)

{

temp = member[j];

member[j] = member[j + 1];

member[j + 1] = temp;

}

}

}

//另一种排序算法

/\*for (int i = 0; i < 8; i++)

{

for (int j = i+1; j < 8 ; j++)

{

if (compare(member[i], member[j]) == 0)

{

temp = member[i];

member[i] = member[j];

member[j] = temp;

}

}

}\*/

//输出检验排序是否成功

for (int k = 0; k < 8; k++)

{

for (int i = 0; i < member[k].size(); i++)

{

printf("%c", member[k][i]);

}

printf(" ");

}

return 0;

}

int flag(char a)

{

vector<char> stand(36);

stand = { 'A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P','Q','R','S','T','U','V','W','X','Y','Z','0','1','2','3','4','5','6','7','8','9' };

for (int i = 0; i < 36; i++)

{

if (a == stand[i])

return i;

}

}

int compare(vector<char>& member1, vector<char>& member2)

{

//字符串长度内比较

for (int i = 0; i < member1.size(),i<member2.size(); i++)

{

if (flag(member1[i]) <flag(member2[i]))

return 1;

else if (flag(member1[i]) >flag( member2[i]))

return 0;

else;

}

//字符串长度内都相同，需要比较长短

if (member1.size() < member2.size())

return 1;

else if (member1.size() > member2.size())

return 0;

}

**作业二**

#include<vector>

#include <iostream>

#include<stdio.h>

using namespace std;

int max(vector<int>& nums);

int main()

{

vector<int> nums(100);

nums = {20,21,23,45,234,546,31,2,3,4,56,3,3,5,6,2,3,4,5,5,5,5,5,2,2234,5,5,13,4,6,6,62,4756,24,243,53,23,3244,2,2,3,42,12,23,24,223,34,12,3,4,23,45,23,34,23,3,34,234,234,234,22,3,4,3,2,32,34,3,242,4,2,423,2,42,3,2,234,423,4,2,4,2,78,65,67,46,7,87,9,78,67,789,78,68,89,80,68,78,79,798};

int maxflag = max(nums);

printf("最大元素值为%d ,在数组中下标为 %d ", nums[maxflag], maxflag);

}

int max(vector<int>& nums)

{

int flag = 0;

for (int i = 0; i < 100; i++)

{

if (nums[i] > nums[flag])

{

flag = i;

}

}

return flag;

}

可能采取的数据结构有链表和线性数组。如果是用链表存储的这100个元素，则要将上面max函数中交换用于标志最大位置的flag换为指针，交换下标改为交换指针。

**作业三**

此题可转化为图模型，选手为点，Pi胜Pj即为Pi指向Pj的一条边，求题目中要求的序列也就是求一条有向路。

//此题可转化为图模型，选手为点，Pi胜Pj即为Pi指向Pj的一条边，求题目中要求的序列也就是求一条有向路。

//

//测试用例1：

// 请输入选手数量n(2到100)

//4

//请输入n \* n的矩阵

//aij 为【1】表明i选手赢了j选手，aij为【0】表明i选手输给了j选手

//(对角线填【2】，表示自己未与自己比赛)

//2 1 0 0

//0 2 1 1

//1 0 2 0

//1 0 1 2

//4 -> 1 -> 2 -> 3

//

//测试用例2：

//请输入选手数量n(2到100)

//4

//请输入n \* n的矩阵

//aij 为【1】表明i选手赢了j选手，aij为【0】表明i选手输给了j选手

//(对角线填【2】，表示自己未与自己比赛)

//2 0 0 0

//1 2 0 0

//1 1 2 0

//1 1 1 2

//4 -> 3 -> 2 -> 1

#include<stdio.h>

#include <iostream>

#include<vector>

int win = 0; //标志是否找到

int loc = -1; //用于排序后存入数组

using namespace std;

int go(int start, int size, vector<vector<int>>& race, vector<int>& order); //递归程序用于排序

int test(int test\_size, vector<int>& order, vector<vector<int>> race); //检验找到的排序是否正确

int main()

{

int n;

vector<vector<int>> race(100); //储存比赛结果

vector<int> order; //用于储存排序后的选手顺序

order.resize(100);

cout << "请输入选手数量n(2到100)" << endl;

cin >> n;

cout << "请输入n\*n的矩阵\naij 为【1】表明i选手赢了j选手，aij为【0】表明i选手输给了j选手\n(对角线填【2】，表示自己未与自己比赛)" << endl;

for (int i = 0; i < n; i++)

{

race[i].resize(100);

order[i] = -1;

for (int j = 0; j < n; j++)

{

cin >> race[i][j];

}

}

//存入比赛结果

for (int i = 0; i < n; i++)

{

loc = -1;

if (go(i, n, race, order) == 3) //找到会返回3，接下来打印

{

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

{

printf("%d -> ", order[i] + 1);

}

printf("%d", order[n - 1] + 1);

break;

}

}

return 0;

}

int go(int start, int size, vector<vector<int>>& race, vector<int>& order)

{

loc++;

order[loc] = start; //把找到的存入数组

if (test(size, order, race) == 1)

{

return 3;

} //找完了返回3

else if (win != 1) //如果没找到，继续递归

{

for (int j = 0; j < size; j++) //对下一个可能排序的选手遍历

{

if (race[start][j] == 1)

{

go(j, size, race, order);

}

if (test(size, order, race) == 1)

{

return 3;

}

else if (test(size, order, race) == 2) //测试结果为2说明排序是一个未包含所有选手的圈，需跳出，否则会陷入循环

{

break;

}

}

loc--; //排序失败时说明最新存入数组的选手不对，位置后退一位

return 0;

}

}

int test(int test\_size, vector<int>& flag, vector<vector<int>> race)

{

for (int i = 0; i < test\_size; i++)

{

if (flag[i] == -1) //说明排序中断，没有把所有选手都排上

{

return 0;

}

}

for (int i = 0; i < test\_size - 1; i++) //说明得到的序列不满足胜负关系要求

{

if (race[flag[i]][flag[i + 1]] != 1)

{

return 0;

}

}

for (int i = 0; i < test\_size; i++) //有选手被重复排序，陷入了圈循环

{

for (int j = i + 1; j < test\_size; j++)

{

if (flag[i] == flag[j])

{

return 2;

}

}

}

return 1;

}