第 3 节 枚举算法

1.【NOIP2014】数字删除

下面程序的功能是将字符串中的数字字符删除后输出。请填空。

```
#include<iostream>
using namespace std;
int delnum(char *s){
    int i,j;
    j=0;
    for (i=0;s[i]!='\0';i++)
        if (s[i]<'0'__①__s[i]>'9'){
            s[j]=s[i];
            __②__;
        }
    return__③__;
}
const int SIZE=30;
int main(){
    char s[SIZE];
    int len,i;
    cin.getline(s,sizeof(s));
    len=delnum(s);
    for (i=0;i<len;i++)cout<<__④__;
    cout<<endl;
    return 0;
}
```

● **选择题**

(1)①处应填(A)。

A. &&　　　　B. &&s[i]>='a'&&s[i]<='z'&&

C. ||　　　　D. &&s[i]>='A'&&s[i]<='Z'&&

(2)②处应填(C)。

A. i++　　　　B. i= j

C. ++j　　　　D. j=i

(3)③处应填(A)。

A. j　　　　B. i

C. s[j]　　　　D. s[i]

(4)④处应填(B)。

A. i　　　　B. s[i]

C. (!s[i])　　　　D. (s[i]==1)

2.【NOIP2012】坐标统计

输入 n 个整点在平面上的坐标。对于每个点,可以控制所有位于它左下方的点(即 x、y 坐标都比它小),它可以控制的点的数目称为"战斗力"。依次输出每个点的战斗力,最后输出战斗力最高的点的编号(如果若干个点的战斗力并列最高,输出其中最大的编号)。

```
#include<iostream>
using namespace std;
const int SIZE=100;
int x[SIZE],y[SIZE],f[SIZE];
int n,i,j,max_f,ans;
int main(){
    cin>>n;
    for (i=1;i<=n;i++)
        cin>>x[i]>>y[i];
    max_f=0;
    for (i=1;i<=n;i++){
        f[i]= ___①___;
        for (j=1;j<=n;j++){
            if (x[j]<x[i]&&___②___)
              ___③___;
        }
        if (___④___){
            max_f=f[i];
            ___⑤___;
        }
    }
    for (i=1;i<=n;i++)
        cout<<f[i]<<endl;
    cout<<ans<<endl;
}
```

○选择题

(1)①处应填(B)。

A. i B. 0

C. x[i] D. y[i]

(2)②处应填(B)。

A. y[i]<y[j] B. y[j]<y[i]

C. y[i]<=y[j] D. y[j]<=y[i]

(3)③处应填(C)。

A. f[i]=x[i] B. f[i]=0

C. f[i]++ D. f[i]=y[i]

(4)④处应填(A)。

A. f[i]>max_f B. f[i]<max_f

C. f[i]>=max_f D. f[i]<=max_f

(5)⑤处应填(A)。

A. ans=i B. ans<max_f

C. ans=f[i] D. ans=0

3. 【NOIP2011】子矩阵

输入一个 n1 * m1 的矩阵 a 和一个 n2 * m2 的矩阵 b，问 a 中是否存在子矩阵和 b 相等。若存在，输出所有子矩阵左上角的坐标，若不存在输出“There is no answer”。

```
#include<iostream>
using namespace std;
const int SIZE=50;
int n1,m1,n2,m2,a[SIZE][SIZE],b[SIZE][SIZE];
int main(){
    int i,j,k1,k2;
    bool good,haveAns;
    cin>>n1>>m1;
    for (i=1;i<=n1;i++)
        for (j=1;j<=m1;j++)
            cin>>a[i][j];
    cin>>n2>>m2;
    for (i=1;i<=n2;i++)
        for (j=1;j<=m2;j++)
            __①__;
    haveAns=false;
    for (i=1;i<=n1-n2+1;i++)
        for (j=1;j<=__②__;j++){
            __③__;
                for (k1=1;k1<=n2;k1++)
                    for (k2=1;k2<=__④__;k2++){
                        if (a[i+k1-1][j+k2-1]!=b[k1][k2])
                            good=false;
                    }
                if (good){
                    cout<<i<<' '<<j<<endl;
                    __⑤__;
                }
        }
    if (!haveAns)
        cout<<"Thereisnoanswer"<<endl;
    return 0;
}
```

◎选择题

(1)①处应填(A)。

A. cin>>b[i][j]　　　　B. cin>>a[i][j]

C. cin>>b[n1][m1]　　　　D. cin>>b[n2][m2]

(2)②处应填(B)。

A. m1　　　　B. m1-m2+1

C. m1-1　　　　D. m1+1

(3)③处应填(D)。

A. good=0　　　　B. good='1'

C. good=false　　　　D. good=1

(4)④处应填(B)。

A. k1+1　　　　B. m2

C. m2-1　　　　D. k1

(5)⑤处应填(C)。

A. break　　　　B. return

C. haveAns=true　　　　D. haveAns=false

4.【NOIP2015】打印月历

输入月份 m(1≤m≤12),按一定格式打印 2015 年第 m 月的月历。

例如,2020 年 1 月的月历打印效果如下(第一列为周日):

```
S    M    T    W    T    F    S
               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
26   27   28   29   30   31
```

```
#include<iostream>
using namespace std;
const int dayNum[]={-1,31, ___①___ ,31,30,31,30,31,31,30,31,30,31};
int m,offset,i;
int main(){
    cin>>m;
    cout<<"S\tM\tT\tW\tT\tF\tS"<<endl; //'\t'为 TAB 制表符
    offset=3;
    for (i=1;i<m;i++)
        offset=(offset+ ___②___ )%7;
    for (i=0;i<offset;i++)
        cout<<'\t';
    for (i=1;i<= ___③___ ;i++){
        cout<<i;
        if (i==dayNum[m] ___④___ ___⑤___ %7==0)
            cout<<endl;
        else
            cout<<'\t';
    }
    return 0;
}
```

●选择题

(1)①处应填(A)。

A. 28　　B. 29

C. 30　　D. 31

(2)②处应填(C)。

A. dayNum[0]　　B. dayNum[i-1]

C. dayNum[i]　　D. dayNum[3]

(3)③处应填(A)。

A. dayNum[m]　　B. dayNum[m*m]

C. m　　D. m*m

(4)④处应填(B)。

A. &&　　B. ||

C. !　　D. ==

(5)⑤处应填(D)。

A. dayNum[offset]　　B. !

C. (offset+dayNum[0])　　D. (offset+i)

5.【NOIP2012】排列数

输入两个正整数 n,m(1≤n≤20,1≤m≤n),在 1～n 中任取 m 个数,按字典序从小到大输出所有这样的排列。

例如:

输入:

```
3 2
```

输出:

```
1 2
1 3
2 1
2 3
3 1
3 2
```

```
#include<iostream>
#include<cstring>
using namespace std;
const int SIZE=25;
bool used[SIZE];
int data[SIZE];
int n,m,i,j,k;
bool flag;
int main(){
    cin>>n>>m;
    memset(used,false,sizeof(used));
    for (i=1;i<=m;i++){
        data[i]=i;
        used[i]=true;
    }
    flag=true;
    while (flag){
        for (i=1;i<=m-1;i++)cout<<data[i]<<"";
        cout<<data[m]<<endl;
        flag=___①___;
        for (i=m;i>=1;i--){
            ___②___;
            for (j=data[i]+1;j<=n;j++)
                if (!used[j]){
                    used[j]=true;
                    data[i]=___③___;
                    flag=true;
                    break;
                }
            if (flag){
                for (k=i+1;k<=m;k++)
                    for (j=1;j<=___④___;j++)
                        if (!used[j]){
                            data[k]=j;
                            used[j]=true;
                            break;
                        }
                ___⑤___;
            }
        }
    }
```

(1)①处应填(A)。

A. 0　　B. 1

C. !flag　　D. used[m]

(2)②处应填(A)。

A. used[data[i]]=false　　B. used[i]=false

C. used[data[i]]=true　　D. used[i]=true

(3)③处应填(C)。

A. !flag　　B. data[j]

C. j　　D. flag

(4)④处应填(C)。

A. n+m　　B. k

C. n　　D. m

(5)⑤处应填(D)。

A. return 0　　B. continue

C. !flag　　D. break

链表

1.【NOIP2018】简单链表

对于一个1到n的排列P(即1到n中每一个数在P中恰好出现了一次),令q_i为第i个位置之后第一个比P_i值更大的位置,如果不存在这样的位置,则$q_i=n+1$。

举例来说,如果n=5且P为15423,则q为26656。

下列程序读入了排列P,使用双向链表求解了答案。试补全程序。数据范围$1\leqslant n\leqslant 10^5$。

```
#include<iostream>
using namespace std;
const int N=100010;
int n;
int L[N],R[N],a[N];
int main()
{
    cin>>n;
    for (int i=1;i<=n;++i){
        int x;
        cin>>x;
        __①__;
    }
    for (int i=1;i<=n;++i){
        R[i]=__②__;
        L[i]=i-1;
    }
    for (int i=1;i<=n;++i){
        L[__③__]=L[a[i]];
        R[L[a[i]]]=R[__④__];
    }
    for (int i=1;i<=n;++i){
        cout<<__⑤__<<" ";
    }
    cout<<endl;
    return 0;
}
```

●选择题

(1)①处应填(A)。

A. a[i]=x　　　　B. a[x]=i

C. x=a[i]　　　　D. a[x]=*i

(2)②处应填(A)。

A. i+1　　　　B. i*2

C. n-i　　　　D. i+1

(3)③处应填(　　)。

A. R[a[i]]　　B. L[a[i]]

C. L[R[i]]　　D. R[L[i]]

(4)④处应填(　　)。

A. a[L[i]]　　B. a[i]

C. R[L[i]]　　D. L[i]

(5)⑤处应填(　　)。

A. L[i]　　B. a[i]

C. R[i]　　D. R[L[i]]

2.【NOIP2016】交朋友

根据社会学研究表明,人们都喜欢找和自己身高相近的人做朋友。现在有 n 名身高两两不相同的同学依次走入教室,调查人员想预测每个人在走入教室的瞬间最想和已经进入教室的哪个人做朋友。当有两名同学和这名同学的身高差一样时,这名同学会更想和高的那个人做朋友。比如一名身高为 1.80 米的同学进入教室时,有一名身高为 1.79 米的同学和一名身高为 1.81 米的同学在教室里,那么这名身高为 1.80 米的同学会更想和身高为 1.81 米的同学做朋友。对于第一个走入教室的同学我们不作预测。

由于我们知道所有人的身高和走进教室的次序,所以我们可以采用离线的做法来解决这样的问题,用排序加链表的方式帮助每一个人找到在他之前进入教室的并且和他身高最相近的人。

```
#include<iostream>
using namespace std;
#define MAXN 200000
#define infinity 2147483647
int answer[MAXN],height[MAXN],previous[MAXN],next[MAXN];
int rank[MAXN];
int n;
void sort (int l,int r){
    int x=height[rank[(l+r)/2]],i=l,j=r,temp;
        while (i<=j){
        while (height[rank[i]]<x)i++;
        while (height[rank[j]]>x)j--;
        if (__①__){
            temp=rank[i];rank[i]=rank[j];rank[j]=temp;
            i++;j--;
        }
    }
    if (i<r)sort(i,r);
    if (l<j)sort(l,j);
}
int main(){
    cin>>n;
    int i,higher,shorter;
    for (i=1;i<=n;i++){
        cin>>height[i];
        rank[i]=i;
    }
    sort(1,n);
    for (i=1;i<=n;i++){
        previous[rank[i]]=rank[i-1];
        __②__;
    }
    for (i=n;i>=2;i--){
        higher=shorter=infinity;
        if (previous[i]!=0)
            shorter=height[i]-height[previous[i]];
        if (next[i]!=0)
            __③__;
        if (__④__)
            answer[i]=previous[i];
        else
            answer[i]=next[i];
        next[previous[i]]=next[i];
        __⑤__;
    }
    for (i=2;i<=n;i++)
        cout<<i<<":"<<answer[i];
    return 0;
}
```

●选择题

(1)①处应填(　　)。

A. `i<j`　　B. `i>j`

C. `i<=j`　　D. `i>=j`

(2)②处应填(　　)。

A. `next[rank[i-1]]=rank[i]`　　B. `next[rank[i+1]]=rank[i]`

C. `next[i]=rank[i+1]`　　D. `next[rank[i]]=rank[i+1]`

(3)③处应填(　　)。

A. `higher=height[next[i]]-height[i]`

B. `higher=height[i]-height[previous[i]]`

C. `higher=max(higher,height[next[i]]-height[i])`

D. `higher=min(higher,height[next[i]]-height[i])`

(4)④处应填(　　)。

A. `shorter<=higher`　　B. `shorter<higher`

C. `answer[i]>previous[i];`　　D. `answer[i]<previous[i];`

(5)⑤处应填(　　)。

A. `next[previous[i]]=next[i]`　　B. `previous[next[i]]=previous[i]`

C. `previous[next[i]]=next[i]`　　D. `next[next[i]]=next[i]`