## 第6节 搜索算法

1.【NOIP2009】国王放置

在n*m的棋盘上放置k个国王，要求k个国王互相不攻击，有多少种不同的放置方法。假设国王放置在第(x,y)格，国王攻击的区域是(x−1,y−1),(x−1,y),(x−1,y+1),(x,y−1),(x,y+1),(x+1,y−1),(x+1,y),(x+1,y+1)。读入三个数n,m,k，输出答案。题目利用回溯法求解。棋盘行标号为0～n−1，列标号为0～m−1。

```
#include<iostream>
using namespace std;
int n,m,k,ans;
int hash[5][5];
void work(int x,int y,int tot){
    int i,j;
    if (tot==k){
        ans++;
        return;
    }
    do{
        while (hash[x][y]){
            y++;
            if (y==m){
                x++;
                y= ①  ;
            }
            if (x==n)
                return;
        }
        for (i=x-1;i<=x+1;i++)
            if (i>=0&&i<n)
                for (j=y-1;j<=y+1;j++)
                    if (j>=0&&j<m)
                        ②  ;
        ③  ;
        for (i=x-1;i<=x+1;i++)
            if (i>=0&&i<n)
                for (j=y-1;j<=y+1;j++)
                    if (j>=0&&j<m)
                        ④  ;
        y++;
        if (y==m){
            x++;
            y=0;
```

回溯

```
        }
        if (x==n)
            return;
    }
    while (1);
}
int main(){
    cin>>n>>m>>k;
    ans=0;
    memset(hash,0,sizeof(hash));
    ⑤ ;
    cout<<ans<<endl;
    return 0;
}
```

○选择题

(1)①处应填(B)。

A. 1　　B. 0

C. x　　D. m

(2)②处应填(C)。

A. hash[i][j]++　　B. hash[i][j]--

C. hash[i][j]=0　　D. hash[i][j]=1

(3)③处应填(D)。

A. work(x,y,tot++)　　B. work(x,y,++tot)

C. work(x,y,tot+1)　　D. work(x,y,tot)

(4)④处应填(C)。

A. hash[i][j]++　　B. hash[i][j]--

C. hash[i][j]=0　　D. hash[i][j]=1

(5)⑤处应填(A)。

A. work(0,0,0)　　B. work(0,0,1)

C. work(1,1,1)　　D. work(n,m,k)

2.【NOIP2009】寻找等差数列

有一些长度相等的等差数列(数列中每个数都为0～59的整数),设长度均为L,将等差数列中的所有数打乱顺序放在一起。现在给你这些打乱后的数,问原先L最大可能为多大?先读入一个数n(1≤n≤60),再读入n个数,代表打乱后的数。输出等差数列最大可能长度L。

```
#include<iostream>
#include<cstring>
using namespace std;
int hash[60];
int n,x,ans,maxnum;
int work(int now){
    int first,second,delta,i;
    int ok;
    while (  ①  &&!hash[now])
        ++now;
    if (now>maxnum)
        return 1;
    first=now;
    for (second=first;second<=maxnum;second++)
        if (hash[second]){
            delta=  ②  ;
            if (first+delta*  ③  >maxnum)
                break;
            if (delta==0)
                ok=(  ④  );
            else {
                ok=1;
                for (i=0;i<ans;i++)
                    ok=  ⑤  &&(hash[first+delta*i]);
            }
            if (ok){
                for (i=0;i<ans;i++)
                    hash[first+delta*i]--;
                if (work(first))
                    return 1;
                for (i=0;i<ans;i++)
```

```
                hash[first+delta*i]++;
            }
        }
        return 0;
}
int main(){
    int i;
    memset(hash,0,sizeof(hash));
    cin>>n;
    maxnum=0;
    for (i=0;i<n;i++){
        cin>>x;
        hash[x]++;
        if (x>maxnum)
            maxnum=x;
    }
    for (ans=n;ans>=1;ans--)
    if (n% ans==0&& ⑥ ){
        cout<<ans<<endl;
        break;
    }
    return 0;
}
```

①选择题

(1)①处应填(　　)。

A. now<maxnum　　B. now>=maxnum

C. now<=maxnum　　D. now>maxnum

(2)②处应填(　　)。

A. second-first　　B. second

C. first-second　　D. second-ans

(3)③处应填(　　)。

A. ans-1　　B. ans-delta

C. ans　　D. ans-first

(4)④处应填(　　)。

A. hash[first]>ans　　B. hash[first]>=ans

C. hash[second]<=ans　　D. hash[second]<ans

(5)⑤处应填(　　)。

A. 0　　B. (!ok)

C. 1　　D. ok

(6)⑥处应填(　　)。

A. work(0)　　B. work(0)<=0

C. work(ans)　　D. (!work(ans))

3.【NOIP2010】过河问题

在一个月黑风高的夜晚，有一群人在河的右岸，想通过唯一的一根独木桥走到河的左岸。在伸手不见五指的黑夜里，过桥时必须借助灯光来照明，不幸的是，他们只有一盏灯。另外，独木桥上最多能承受两个人同时经过，否则将会坍塌。每个人单独过独木桥都需要一定的时间，不同的人需要的时间可能不同。两个人一起过独木桥时，由于只有一盏灯，所以需要的时间是较慢的那个人单独过桥所花费的时间。现在输入 N(2≤N<1000)和这 N 个人单独过桥需要的时间，请计算总共最少需要多少时间，他们才能全部到达河左岸。

例如，有 3 个人甲、乙、丙，他们单独过桥的时间分别为 1、2、4，则总共最少需要的时间为 7。具体方法是：甲、乙一起过桥到河的左岸，甲单独回到河的右岸将灯带回，然后甲，丙在一起过桥到河的左岸，总时间为 2+1+4=7。

```
#include<iostream>
#include<cstring>
using namespace std;
const int SIZE=100;
const int INFINITY=10000;
const bool LEFT=true;
const bool RIGHT=false;
const bool LEFT_TO_RIGHT=true;
const bool RIGHT_TO_LEFT=false;
int n,hour[SIZE];
bool pos[SIZE];
int max(int a,int b){
```

```
    if (a>b)
        return a;
    else
        return b;
}
int go (bool stage)
{
    int i,j,num,tmp,ans;
    if (stage==RIGHT_TO_LEFT)
    {
        num=0;
        ans=0;
        for (i=1;i<=n;i++)
          if (pos[i]==RIGHT)
          {
              num++;
              if (hour[i]>ans)
                  ans=hour[i];
          }
        if (  ①  )
            return ans;
        ans=INFINITY;
        for (i=1;i<=n-1;i++)
          if (pos[i]==RIGHT)
              for (j=i+1;j<=n;j++)
                  if (pos[j]==RIGHT){
                      pos[i]=LEFT;
                      pos[j]=LEFT;
                      tmp=max(hour[i],hour[j])+  ②  ;
                      if (tmp<ans)
                          ans=tmp;
                      pos[i]=RIGHT;
                      pos[j]=RIGHT;
                  }
        return ans;
    }
    if (stage==LEFT_TO_RIGHT)
    {
        ans=INFINITY;
        for (i=1;i<=n;i++)
            if (  ③  )
            {
```

```
                pos[i]=RIGHT;
                tmp=  ④  ;
                if (tmp<ans)
                    ans=tmp;
                ⑤;
            }
        return ans;
    }
    return 0;
}
int main()
{
    int i;
    cin>>n;
    for (i=1;i<=n;i++){
        cin>>hour[i];
        pos[i]=RIGHT;
    }
    cout<<go(RIGHT_TO_LEFT)<<endl;
    return 0;
}
```

**●选择题**

(1)①处应填(　　)。

A. num<=0　　B. num<=1

C. num<=2　　D. num<=3

(2)②处应填(　　)。

A. go(LEFT)　　B. go(RIGHT)

C. go(LEFT_TO_RIGHT)　　D. go(RIGHT_TO_LEFT)

(3)③处应填(　　)。

A. pos[i]=LEFT　　B. pos[i]=RIGHT

C. pos[i]=LEFT_TO_RIGHT　　D. pos[i]=RIGHT_TO_LEFT

(4)④处应填(　　)。

A. hour[i]+go(RIGHT_TO_LEFT)　　B. hour[i]-go(RIGHT_TO_LEFT)

C. hour[i]+go(LEFT_TO_RIGHT)　　D. hour[i]-go(LEFT_TO_RIGHT)

(5)⑤处应填(　　)。

A. pos[i]=LEFT　　B. pos[i]=RIGHT

C. pos[i]=LEFT_TO_RIGHT　　D. pos[i]=RIGHT_TO_LEFT