[报告]poj 2331 Water pipe

[Source]

http://poj.org/problem?id=2331

[Description]

在二维平面只有整数坐标的坐标系里，给出两个点，然后给出最多 k（k<=4）种长度的线段，以及每种长度的线段的数量（小于等于 10），规定线段只能垂直或者水平的摆放，两段线段之间连接的方式只能是首尾相接，线段之间可以交叉重合，求把给出的那两点给连接起来最少要几根线段。

[Solution]

这题等效于求一个从一个点指向另一个点的向量。基本思路是先把 x 轴方向的分量给凑齐，再凑齐 y 轴方向的分量。下面有几种方法实现这个思路：

1. 枚举法，注意到数据范围，所有线段在同一个方向上最多只能组成 21^4 个不同的长度。所以可以把这长度都枚举出来，然后挑出恰好等于 x 轴分量的那些组合，以及恰好等于 y 轴分量的那些组合。然后再把这两类组合再组合，选出最优。
2. 用 IDA*进行搜索，先搜 x 轴的分量，再搜 y 轴的分量。

[Code]

1. 枚举法

```
#include<iostream>
#include<stdio.h>
#include<string.h>
#include<algorithm>
using namespace std;
const int inf= 10000;
class S{
public:
    int len;
    int num[5];
};
S que[2][22*22*22*22+10];
S x[22*22*22*22+10],y[22*22*22*22+10];
int c[10],len[10];
void init(int k)
{
    for(int i=1;i<=k;i++)
        cin>>len[i];
```

```
    for(int i=1;i<=k;i++)
        cin>>c[i];
}
void cal(int k,int dx,int dy)
{
    int cur,pre,i,j,tmp,q;

    for(i=1;i<=k;i++){
        cur=i%2;
        pre=!cur;
        if(i==1){
            que[pre][0].len=1;
            for(j=1;j<=k;j++)
                que[pre][1].num[j]=0;
            que[pre][1].len=0;
        }
        que[cur][0].len=0;
        for(j=1;j<=que[pre][0].len;j++){
            for(k=-c[i];k<=c[i];k++){
                que[cur][0].len++;
                tmp=que[cur][0].len;
                que[cur][tmp].len=que[pre][j].len+k*len[i];
                que[cur][tmp].num[i]=abs(k);
                for(q=1;q<i;q++)
                    que[cur][tmp].num[q]=que[pre][j].num[q];
            }
        }
    }
    x[0].len=y[0].len=0;
    for(i=1;i<=que[cur][0].len;i++){
        if(que[cur][i].len==dx){
            x[0].len++;
            x[x[0].len]=que[cur][i];
        }
        if(que[cur][i].len==dy){
            y[0].len++;
            y[y[0].len]=que[cur][i];
        }
    }
}
bool check(int nx,int ny,int k)
{
    int i;
```

```
    for(i=1;i<=k;i++)
        if(x[nx].num[i]+y[ny].num[i]>c[i])
            return false;
    return true;
}
int tot(int nx,int ny,int k)
{
    int t=0;

    for(int i=1;i<=k;i++)
        t+=x[nx].num[i]+y[ny].num[i];
    return t;
}
int work(int k)
{
    int mm=inf,i,j,p,q,tmp;

    for(i=1;i<=x[0].len;i++){
        for(j=1;j<=y[0].len;j++){
            if(check(i,j,k))
                mm=min(tot(i,j,k),mm);
        }
    }
    if(mm==inf)
        mm=-1;
    return mm;
}
int main()
{
    int x1,y1,x2,y2,k,ans;

    while(scanf("%d %d %d %d %d",&x1,&y1,&x2,&y2,&k)==5){
        init(k);
        cal(k,x2-x1,y2-y1);
        cout<<work(k)<<endl;
    }
    return 0;
}
```

2. IDA*

```
# include <cstdio>
# include <cstring>
# include <queue>
using namespace std;
```

```
int c[5],l[5],n,res=0xfffffff;
int x1,y1,x2,y2;
int referx[1001],refery[1001];
queue<int> q;
bool solve(int pos,int len,bool type)
{
   if(!type)
   {
        if(referx[pos]==-1||len+referx[pos]>res) return false;
        if(pos==x2)
        {
            return solve(y1,len,1);
        }
        for(int i=1;i<=n;i++)
          if(c[i])
          {
            if(x2<pos)
            {
             if(pos-l[i]>=1)
             {
                c[i]--;
                if(solve(pos-l[i],len+1,0)) return true;;
                c[i]++;
             }
             if(pos+l[i]<=1000)
             {
                c[i]--;
                if(solve(pos+l[i],len+1,0)) return true;
                c[i]++;
             }
            }
            else
            {
             if(pos+l[i]<=1000)
             {
                c[i]--;
                if(solve(pos+l[i],len+1,0)) return true;
                c[i]++;
             }
             if(pos-l[i]>=1)
             {
                c[i]--;
                if(solve(pos-l[i],len+1,0)) return true;;
                c[i]++;
```

```
          }
         }
       }
}
else
{
     if(refery[pos]==-1||len+refery[pos]>res) return false;
     if(pos==y2)
     {
         return true;
     }
     for(int i=1;i<=n;i++)
       if(c[i])
       {
         if(y2<pos)
         {
          if(pos-l[i]>=1)
          {
             c[i]--;
             if(solve(pos-l[i],len+1,1)) return true;
             c[i]++;
          }
          if(pos+l[i]<=1000)
          {
             c[i]--;
             if(solve(pos+l[i],len+1,1)) return true;
             c[i]++;
          }
         }
         else
         {
          if(pos+l[i]<=1000)
          {
             c[i]--;
             if(solve(pos+l[i],len+1,1)) return true;
             c[i]++;
          }
          if(pos-l[i]>=1)
          {
             c[i]--;
             if(solve(pos-l[i],len+1,1)) return true;
             c[i]++;
          }
         }
```

```
            }
        }
        return false;
}
void cal(int refer[],int pos)
{
    refer[pos]=0;
    q.push(pos);
    while(!q.empty())
    {
        pos=q.front();
        q.pop();
        for(int i=1;i<=n;i++)
        {
          if(pos-l[i]>=1&&refer[pos-l[i]]==-1)
           {
               refer[pos-l[i]]=refer[pos]+1;
               q.push(pos-l[i]);
           }
          if(pos+l[i]<=1000&&refer[pos+l[i]]==-1)
           {
               refer[pos+l[i]]=refer[pos]+1;
               q.push(pos+l[i]);
           }
        }
    }
}
int main()
{
    int total=0;
    scanf("%d%d%d%d%d",&x1,&y1,&x2,&y2,&n);
    for(int i=1;i<=n;i++)
        scanf("%d",l+i);
    for(int i=1;i<=n;i++)
    {
        scanf("%d",c+i);
        total+=c[i];
    }
    memset(referx,-1,sizeof(referx));
    memset(refery,-1,sizeof(refery));
    cal(referx,x2);
    cal(refery,y2);
    for(res=0;res<=total;res++)
        if(solve(x1,0,0)) break;
```

```
    if(res==total+1) printf("-1\n");
    else printf("%d\n",res);
 //   system("pause");
    return 0;
}
```