# P1433 吃奶酪

## 题目描述

房间里放着n块奶酪。一只小老鼠要把它们都吃掉，问至少要跑多少距离？老鼠一开始在(0,0)点处。

## 输入输出格式

输入格式：

第一行一个数n (n<=15)

接下来每行2个实数，表示第i块奶酪的坐标。

两点之间的距离公式=sqrt((x1-x2)*(x1-x2)+(y1-y2)*(y1-y2))

输出格式：

一个数，表示要跑的最少距离，保留2位小数。

## 输入输出样例

输入样例#1：

复制

```
4
1 1
1 -1
-1 1
-1 -1
```

输出样例#1：

复制

```
7.41
```

## 思路

钱嘉欢 20:48:54 先输入，然后初始化和在map数组里存好任意两个位置的距离。搜索时，如果现在走过的路程已经大于最少路程，就直接返回，如果走过的点等于n，就比一下大小，答案如果比现在这个路程长度，答案就变成现在的路程长度，不然不变。 如果走过的点没有到达n，就用回溯继续下去

来日可追 20:46:51 dfs里面的三个参数是什么意思

钱嘉欢 20:50:08 dfs里分别是经过几个点、现在在那个点上和走过的路程长度

# 代码

```cpp
//钱嘉欢
#include<iostream>
#include<cstdio>
#include<cmath>
using namespace std;
int n;
double x[20],y[20],map[20][20],ans;
bool v[20];
void dfs(int jg,int xz,double lc)
{
    if(ans<lc) return ;
    if(jg==n)
    {
        if(ans>lc) ans=lc;
        return ;
    }
    for(int i=1;i<=n;i++)
    {
        if(v[i]==false)
        {
            v[i]=true;
            dfs(jg+1,i,lc+map[xz][i]);
            v[i]=false;
        }
    }
}
int main()
{
    cin>>n;
    x[0]=0;
    y[0]=0;
    ans=100000;
    for(int i=1;i<=n;i++)
        cin>>x[i]>>y[i];
    for(int i=0;i<=n;i++)
        for(int j=0;j<=n;j++)
            map[i][j]=sqrt((x[i]-x[j])*(x[i]-x[j])+(y[i]-y[j])*(y[i]-y[j]));
    dfs(0,0,0.0);
    printf("%0.2f",ans);
    return 0;
}
```

# 代码2

```cpp
//周子涵
#include<bits/stdc++.h>
using namespace std;
double ans=1111111000;//答案
int n,nr;
bool b[20];
double ax[20],ay[20],now;//坐标和现在距离
double jl[20][20];//距离
void dfs(int nr,int d){
```

```
    if(now>ans) return;//如果现在距离小于答案 return
    if(nr==n){//判断奶酪是否吃完了
        if(now<=ans) ans=now;//判断现在距离是否比ans小
        return;
    }
    for(int i=1;i<=n;i++){//一个点一个点的搜索
        if(b[i]==false){ //是否走过
            if(jl[d][i]!=0){//判断两点是否有距离
                b[i]=true;
                now+=jl[d][i];
                dfs(nr+1,i);
                b[i]=false;
                now-=jl[d][i];
            }else if(jl[i][d]!=0){//同上
                b[i]=true;
                now+=jl[i][d];
                dfs(nr+1,i);
                b[i]=false;
                now-=jl[i][d];
            }else{//如果没有，只能老实求
                b[i]=true;
                jl[i][d]=sqrt((ax[i]-ax[d])*(ax[i]-ax[d])+(ay[i]-ay[d])*(ay[i]-ay[d]));
                now+=jl[i][d];
                dfs(nr+1,i);
                b[i]=false;
                now-=jl[i][d];
            }//如同记忆化
        }
    }
    return;
}
int main(){
    cin>>n;
    if(n==0){//等于0输出return
        cout<<0.00;
        return 0;
    }
    for(int i=1;i<=n;i++){//输入坐标
        cin>>ax[i];
        cin>>ay[i];
    }
    b[0]=true;//初始化
    dfs(0,0);
    printf("%0.2lf",ans);//输出
    return 0;
}

```

# 代码3

```
#include<cmath>
#include<cstdio>
#include<cstring>
#include<iostream>
using namespace std;
long long n;
double dx[20],dy[20],jl=2147483647.00,dis[20][20]/*<-类似打表。*/;
bool vis[20];//前面是否用过。
void dfs(long long walked/*已经走过。*/,long long now/*现在身处。*/,double meter/*走过路程。
*/){
```

```
    if(meter>=jl){
        return ;//大了直接跳掉。
    }
    if(walked==n){
        if(meter<jl){
            jl=meter;
        }//作比较。
    }
    for(int i=1;i<=n;i++){//枚举所有点。
        if(vis[i]==true){
            vis[i]=false;//把它用掉，标记成已拜访。
            dfs(walked+1,i,meter+dis[now][i]/*<-走过这个点，加上这个点所需的米数。*/);
            vis[i]=true;//因为后面还需要用，所以要重新变成没拜访过。
        }
    }
    return ;
}
int main(){
    scanf("%lld",&n);
    for(int i=1;i<=n;i++){
        scanf("%lf%lf",&dx[i],&dy[i]);
    }
    memset(vis,true,sizeof(vis));//都没拜访过。
    for(int i=0;i<=n;i++){
        for(int j=0;j<=n;j++){
            dis[i][j]=sqrt((dx[i]-dx[j])*(dx[i]-dx[j])+(dy[i]-dy[j])*(dy[i]-dy[j]));//套
洛谷公式。
        }
    }
    dfs(0,0,0.00);
    printf("%0.2lf\n",jl);
    return 0;
}
/*李明烨  21:32:26
反思：1.C++自带函数是被包装起来的不能改变，在那种活题里会TLE，但是自己的dfs可以随意改动优化；
         2.有时候long long也不是很好，递归的时候最好用int；
收获：1.两点距离公式；
2.懂得了什么时候可以用C++函数，什么时候必须自己写
3.数据类型在不同“场合”，要选择不同的类型
追加一条反思:平时应该多看一些编程类的书籍
*/

```