算法笔记

1.The Legend of Freya the Frog

题意：

从$(0,0)$到(x,y)，每步走d个距离，其中$x,y\in\mathbb{N},k\in\mathbb{N}^+,0\leq d\leq k$，求最小步数。

思路：

可以按照$\lceil\frac{y}{k}\rceil$和$\lceil\frac{x}{k}\rceil$的大小进行分类。如果$\lceil\frac{y}{k}\rceil\geq\lceil\frac{x}{k}\rceil$，那么向右需要多走$\lceil\frac{y}{k}\rceil-\lceil\frac{x}{k}\rceil$的"空步"，共需走$2\lceil\frac{y}{k}\rceil$步；否则，需要走$\lceil\frac{x}{k}\rceil-\lceil\frac{y}{k}\rceil-1$的"空步"，共需走$2\lceil\frac{x}{k}\rceil-1$步。

代码：

```python
import math

t=int(input())
for _ in range(t):
    x,y,k=map(int,input().split())
    if math.ceil(y/k)>=math.ceil(x/k):
        print(2*math.ceil(y/k))
    else:
        print(2*math.ceil(x/k)-1)
```

2.Kousuke's Assignment

题意：

给定一个序列，求出其中不重叠的和为0的最大子序列的个数。

思路：

求不重叠的子序列是一个典型的dp问题。我们需要引入前缀和序列p，即$p[i]=\sum_{k=1}^{i}a[k]$，其中序列a为给定序列。那么如果找到两个前缀和$p[i]$和$p[j]$，使得$p[i]=p[j]$，那么i,j之间的子序列和为0。可以按照这个思路进行计算。

代码：

```python
def solve(a):
    n=len(a)
    #计算前缀和
    p=[0]*(n+1)
    for i in range(1,n+1):
        p[i]=p[i-1]+a[i-1]

    position={} #字典用于记录每个前缀和最后出现的位置，如果有相同的就更新最新位置。
    position[0]=0
    dp=[0]*(n+1)  #dp序列用于记录前a中前i个数出现的和为0的子序列个数。

    for j in range(1,n+1):
        if p[j] in position:
            k=position[p[j]]
            dp[j]=max(dp[j-1],dp[k]+1)
        else:
```

```
            dp[j]=dp[j-1]
        position[p[j]]=j

    return dp[n]

t=int(input())
for _ in range(t):
    n=int(input())
    a=list(map(int,input().split()))
    print(solve(a))
```

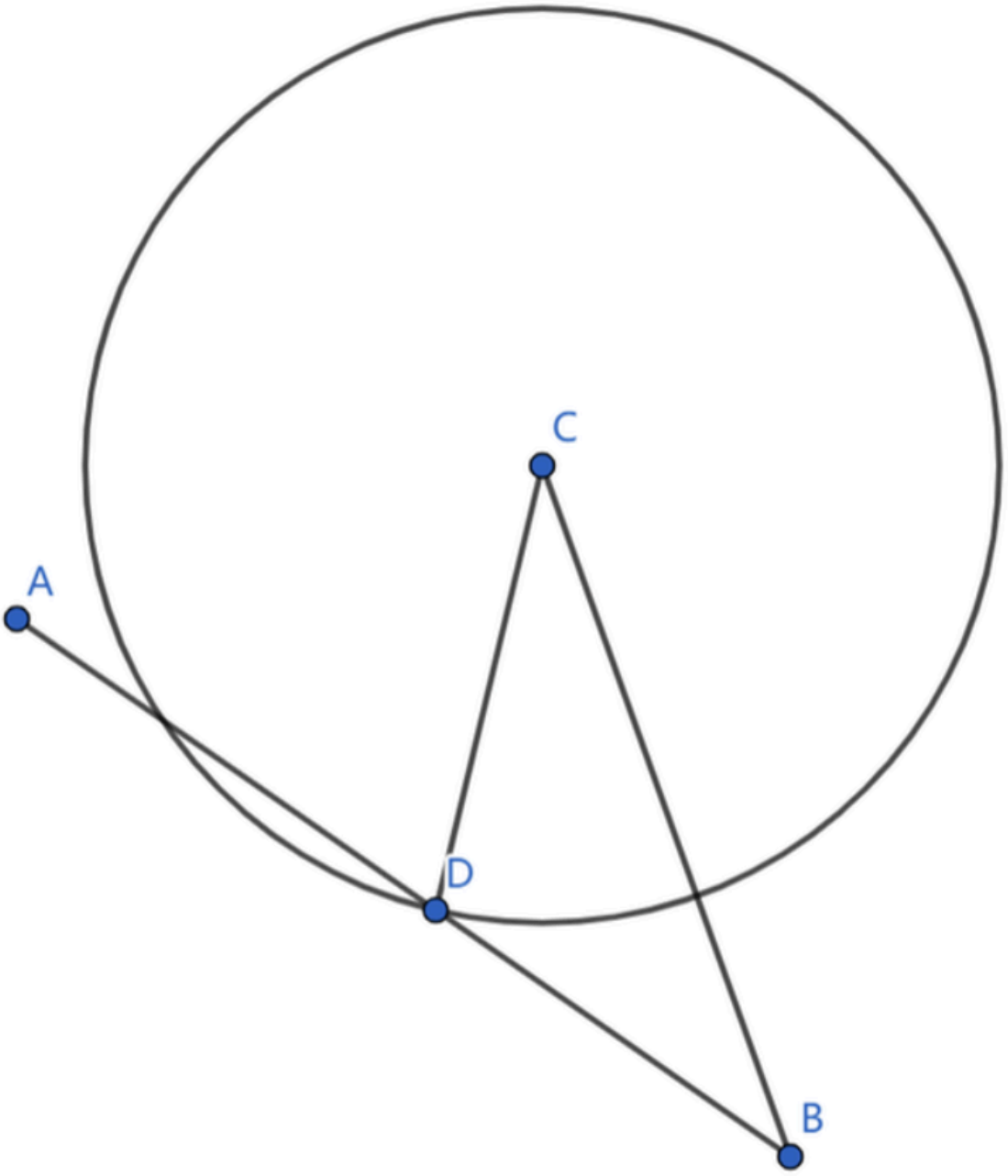
3.Black Circles

题意：

从(x_1,y_1)到(x_2,y_2)，速度为1。同时有i个雷达分布在(x_i,y_i)，以1的速度扩大扫描范围。判断从起到到终点是否会被探测到。

思路：

这是一个巧妙的几何问题。容易想到的最快路径是直线到达，下面进行尝试。



假设我们需要从A到B，C为雷达所在位置，那么不被探测到的条件就是我们先到达B，也即$AB < CB$。下面给出证明：

假设我们在D点被探测到，那么就有$AD = CD$，由三角不等式：$CB - BD \leq CD$，也即$CB \leq BD + AD = AB$。那么它的逆否命题：我们能不被探测地到达终点，如果$CB > AB$便也是真命题。证明完毕！

代码：

```
def solve(position,x1,x2,y1,y2):
    n=len(position)
    for i in range(n):
        if (position[i][0]-x2)**2+(position[i][1]-y2)**2<=(x1-x2)**2+(y1-y2)**2:
            return "no"
    return "yes"

t=int(input())
for _ in range(t):
```

```
    n=int(input())
    position=[]
    for _ in range(n):
        x,y=map(int,input().split())
        position.append((x,y))
    x1,y1,x2,y2=map(int,input().split())
    print(solve(position,x1,x2,y1,y2))
```