思路:首先在经过每个加油站时，都要确保油箱是满的。我们将不同加油站的油看成是一份加入到油箱中。利用双端队列维护油箱，双端队列的前端装价格低的油，后端装价格高的油。

在从加油站i-1驶向加油站i的过程中，优先使用队列前端的油，并进行价格的累加。每经过一个加油站，将该加油站的油价和队列中的油价进行对比，弹出队列中比该加油站价格高的油，然后用该加油站的油填满油箱剩余部分。

以上贪心策略保证了价格低的油可以最先被使用。另外需要注意的一点是，double类型的数字之间，由于精度问题，需要引入eps进行比较。

注意以上策略可以处理油站所在的位置以及cmax,d,davg均不是int类型的情况。

如果以上变量均是int类型，可以采用以下思路:计算出cmax升汽油可行驶的公里数，将汽油价格油低到高进行排列，使用价格低的汽油对路径进行覆盖(覆盖单位是1公里，这也就是这种方法为什么需要油站所在的位置以及cmax,d,davg均是int类型的原因)

#include <bits/stdc++.h>

using namespace std;

struct station{

double pi;

int di;

bool operator < (const station& sta) const{

return pi<sta.pi;

}

}sta[30001];

int main()

{

int Cmax,D,Davg,N;

while(cin>>Cmax>>D>>Davg>>N){

for(int i=0;i<N;i++){

cin>>sta[i].pi>>sta[i].di;

}

sort(sta,sta+N);

int Smax = Cmax\*Davg;

double sum=0;

int flag[30001]={0};

for(int i=0;i<N;i++){

int tmp = (sta[i].di+Smax)<D?Smax:(D-sta[i].di);

int cnt=0;

for(int j=sta[i].di;j<sta[i].di+tmp;j++){

if(flag[j]==0){

flag[j]=1;

cnt++;

}

}

sum+=cnt\*sta[i].pi/Davg;

}

int i;

for(i=0;i<D;i++){

if(flag[i]==0){

printf("The maximum travel distance = %.2f\n",float(i));

break;

}

}

if(i==D){

printf("%.2f\n",sum);

}

}

return 0;

}