程序设计实习

郭炜　微博 http://weibo.com/guoweiofpku
http://blog.sina.com.cn/u/3266490431
刘家瑛 微博 http://weibo.com/pkuliujiaying

动态规划

例题：灌溉草场

灌溉草场(POJ2373)

在一片草场上:有一条长度为L（1 <= L <= 1,000,000，L为偶数)的线段。 John的N（1 <= N <= 1000）头奶牛都沿着草场上这条线段吃草，每头牛的活动范围是一个开区间(S,E)，S，E都是整数。不同奶牛的活动范围可以有重叠。

John要在这条线段上安装喷水头灌溉草场。每个喷水头的喷洒半径可以随意调节，调节范围是 [A　B](1 <= A <= B <= 1000)，A,B都是整数。要求

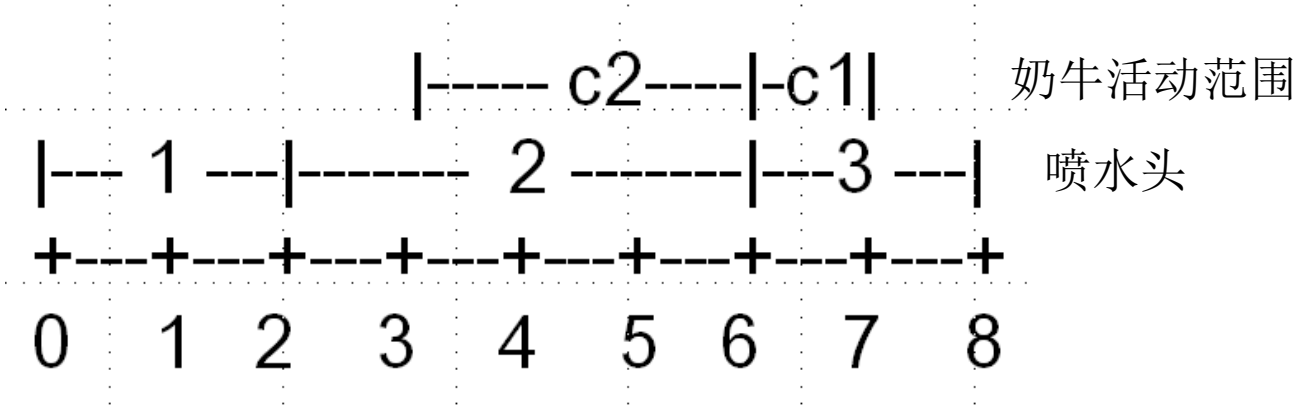
线段上的每个整点恰好位于一个喷水头的喷洒范围内

每头奶牛的活动范围要位于一个喷水头的喷洒范围内

任何喷水头的喷洒范围不可越过线段的两端(左端是0,右端是L ）

请问， John 最少需要安装多少个喷水头。

灌溉草场(POJ2373)



在位置2和6，喷水头的喷洒范围不算重叠

- **输入**

  第1行：整数N、L。

  第2行：整数A、B。

  第3到N+2行：每行两个整数S、E（0 <= S < E <= L），表示某头牛活动范围的起点和终点在线段上的坐标(即到线段起点的距离)。

- **输出：**最少需要安装的多少个喷水头；若没有符合要求的喷水头安装方案，则输出-1。

输入样例

```
2 8
1 2
6 7
3 6
```

输出样例

```
3
```

问题分析

- 从线段的起点向终点安装喷水头，令f(X)表示：所安装喷水头的喷洒范围恰好覆盖直线上的区间[0　X]时，最少需要多少个喷水头
- 显然，X应满足下列条件
  - X为偶数
  - X所在位置不会出现奶牛，即X不属于任何一个(S,E)
  - $X \geq 2A$
  - 当$X>2B$时，存在$Y \in [X-2B　X-2A]$且Y满足上述三个条件，使得$f(X)=f(Y)+1$

解题思路

- 递推计算f(X)
  - f(X) = ∝ ： X 是奇数
  - f(X) = ∝ ： X < 2A
  - f(X) = ∝ ：X处可能有奶牛出没
  - f(X)=1： 2A≤X≤2B 、且X位于任何奶牛的活动范围之外
  - f(X)=1+min{f(Y): Y∈[X-2B　X-2A]、Y位于任何奶牛的活动范围之外}： X>2B

解题思路

f(X)=1+min{f(Y): Y∈[X-2B　X-2A]、Y位于任何奶牛的活动范围之外}：　X>2B

- 对每个X求f(X)，都要遍历区间 [X-2B, X -2A]去寻找其中最小的f(Y),则时间复杂度为：L * B = 1000000 * 1000，太慢
- 快速找到[X-2B　X-2A]中使得f(Y)最小的元素是问题求解速度的关键 。

解题思路

- 可以使用优先队列priority_queue！（multiset也可以，比priority_queue慢一点）！

- 求F(X)时，若坐标属于[X-2B, X-2A]的二元组(i,F(i))都保存在一个priority_queue中，并根据F(i)值排序，则队头的元素就能确保是F(i)值最小的。

解题思路

- 在求 X点的F(x)时，必须确保队列中包含所有属于 [X-2B,X-2A]的点。而且，不允许出现坐标大于X-2A的点，因为这样的点对求F(X)无用，如果这样的点出现在队头，因其对求后续点的F值有用，故不能抛弃之，于是算法就无法继续了。

- 队列中可以出现坐标小于 X-2B 的点。这样的点若出现在队头，则直接将其抛弃。

- 求出X点的F值后，将(X-2A+2，F(X-2A+2))放入队列，为求F(X+2)作准备
- 队列里只要存坐标为偶数的点即可

```
#include <iostream>
#include <cstring>
#include <queue>
using namespace std;
const int INFINITE = 1<<31;
const int MAXL = 1000010;
const int MAXN = 1010;
int F[MAXL];  // F[L]就是答案
int cowThere[MAXL]; //cowThere[i]为1表示点i有奶牛
int N,L,A,B;
struct Fx {
        int x;       int f;
        bool operator<(const Fx & a) const {   return f > a.f;   }
        Fx(int xx=0,int ff=0):x(xx),f(ff) { }
};// 在优先队列里，f值越小的越优先
priority_queue<Fx> qFx;
```

```
int main()
{
        cin >> N >> L;
        cin >> A >> B;
        A <<= 1; B <<= 1; //A,B的定义变为覆盖的直径
        memset(cowThere,0,sizeof(cowThere));
        for( int i = 0;i < N; ++i ) {
                int s,e;
                cin >> s >> e;
                ++cowThere[s+1]; //从s+1起进入一个奶牛区
                --cowThere[e]; //从e起退出一个奶牛区
        }
        int  inCows = 0; //表示当前点位于多少头奶牛的活动范围之内
        for( int i = 0;i <= L ; i ++) { //算出每个点是否有奶牛
                F[i] = INFINITE;
                inCows += cowThere[i];
                cowThere[i] = inCows > 0;
        }
```

```
for( int i = A;i <= B ; i += 2 ) //初始化队列
        if(! cowThere[i] ) {
                F[i] = 1;
                if( i <= B + 2 - A ) //在求F[i]的时候，要确保队列里的点x，x <= i - A
                        qFx.push(Fx(i,1));
        }
for( int i = B + 2 ; i <= L; i += 2 ) {
        if( !cowThere[i] ) {   Fx fx;
                while(!qFx.empty()) {
                        fx = qFx.top();
                        if( fx.x < i - B )
                                qFx.pop();
                        else
                                break;
                }
                if ( ! qFx.empty() )
                        F[i] = fx.f + 1;
        }
```

```
                if( F[i- A + 2] != INFINITE) {//队列中增加一个+1可达下个点的点
                        qFx.push(Fx(i- A + 2, F[i- A + 2]));
                }
        }
        if( F[L] == INFINITE )
                cout  << -1 <<endl;
        else
                cout << F[L] << endl;
        return 0;
} // 复杂度：O)(nlogn)
```

手工实现优先队列的方法

- 如果一个队列满足以下条件：

1) 开始为空
2) 每在队尾加入一个元素a之前，都从现有队尾往前删除元素，一直删到碰到小于 a的元素为止，然后再加入a

- 那么队列就是递增的，当然队头的元素，一定是队列中最小的