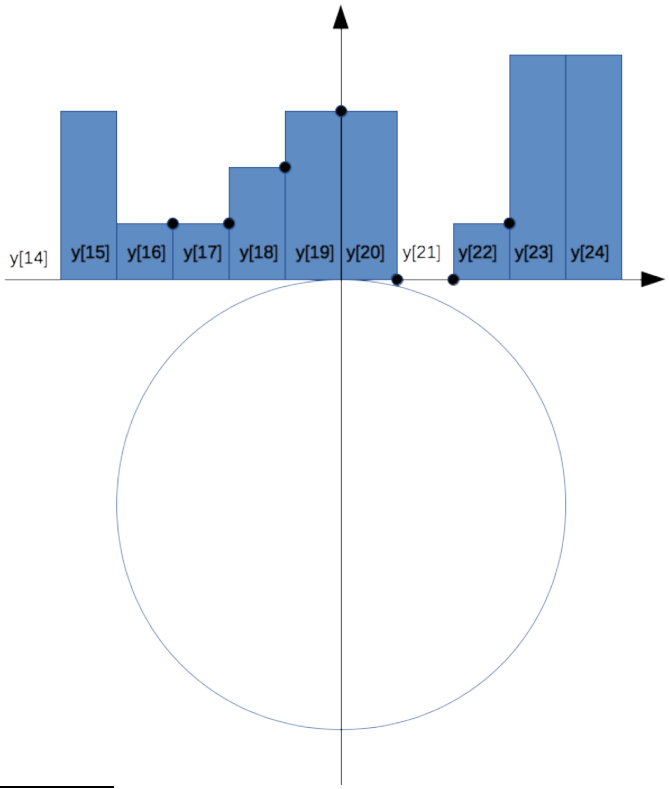
ACM ICPC2014 国内予選　問題 C バンパイア

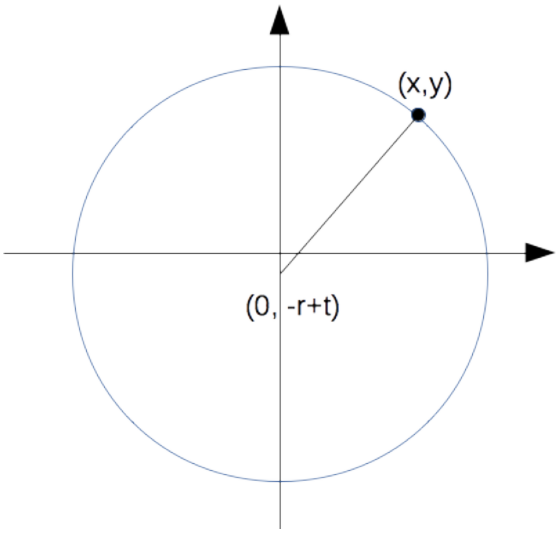
【考察】

- シルエットのx座標は-20~20の位置にあるので、大きさ 40 の配列yを用い、x座標が$[x, x+1]$の場所のシルエットの高さを$y[x+20]$に格納するものとする（$-20 \leq x \leq 20$）。
- シルエットのx座標は重なることもあるので、$y[x+20]$にはその位置のシルエットの高さの最大値を入れる。
- このため、シルエットごとに、シルエットの左端xlから右端xrの位置のシルエットの高さに基づき$[xl, xr]$に対応する配列yの要素の値を、全シルエットの最大値になるように必要に応じて更新する。
- 太陽はy軸上を上に移動し、シルエットの左右の端のx座標は整数値なので、$[-r+1, r-1]$の範囲でx座標が整数の垂直線とシルエットの輪郭との交点（各垂直線に対して最も低い位置の交点）と太陽の輪郭が接する位置を考えれば良い（図の黒丸の点）[1]。これらの中でもっとも太陽の位置が低いものが、太陽の光が遮られている最後の瞬間を表す。



[1] $x = \pm r$の垂直線は太陽に接するので、考慮する必要は無い。また、黒丸のy座標は、両隣の領域のシルエットの高さの低い方の値となる。

- t秒後に太陽の輪郭が点(x, y)に接するとする。この時、太陽の中心は$(0, -r+t)$であり、$r^2 = x^2 + (y-(-r+t))^2$が成立するので、最初に接する時刻は$t = y + r - \sqrt{r^2 - x^2}$となる。



【プログラム例】

```
// ACM-ICPC2014 Domestic Problem C バンパイア
//   http://icpc.iisf.or.jp/past-icpc/domestic2014/#section_C
//   Filename = pc.c
//   Compile: cc pc.c -lm
//   Execution: ./a.out < C0 > C0.out
//   Check:          cc -o check check.c
//                   ./check C0.ans C0.out

#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#define MAXX 20             // x 座標の絶対値の最大値
#define MAXH 20             // 高さの最大値
#define SIZE (MAXX*2)
#define MAX(x,y) ((x)>(y)?(x):(y))
#define MIN(x,y) ((x)<(y)?(x):(y))
int y[SIZE];                // [x,x+1]の高さ = y[20+x]

int main()
{
  int r;  // 太陽の半径 1 <= r <= 20
  int n;  // 建物の影の個数 0 <= n <= 20
  int xl; // 建物の影の左端の x 座標
  int xr; // 建物の影の右端の x 座標
  int h;  // 建物の影の高さ
  int i;
```

```
    int x;
    double t, mint;
    while(1){
      scanf("%d%d", &r, &n);                    // 半径 r と影の個数 n の入力
      if(r==0 && n==0) break;                   // 両方 0 なら終了
      for(i=0; i<SIZE; i++) y[i] = 0;           // 影の高さの配列を 0 に初期化
      for(i=0; i<n; i++){
        scanf("%d%d%d", &xl, &xr, &h);
        for(x=xl; x < xr; x++){                 // [xl,xr]の位置の影の高さを
          y[x+20] = MAX(h, y[x+20]);  // その位置の影の高さの最大値にセット
        }
      }
      mint = MAXH;
      for(x = -r+1; x <= r-1; x++){             // 各位置 x = -r+1 ～ r-1 に対して
        h = MIN(y[x+19],y[x+20]);               // 両隣の低い方の影の高さを求め
        t = h + r - sqrt(r*r - x*x);        // その位置に太陽がかかる時刻を求める
        mint = MIN(mint, t);                // 各位置に対するその時刻の最小値を求める
      }
      printf("%lf¥n", mint);
    }
}
```