最优化选讲：凸优化与模拟退火

Ebola

Institute of Mathematics,
Zhejiang University.

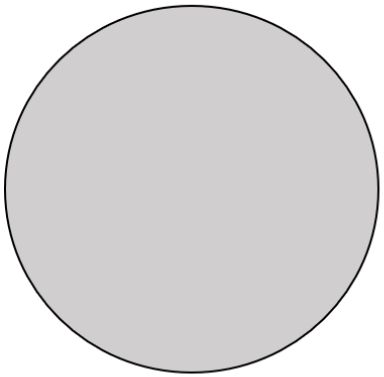
Jan, 2024

问题形式

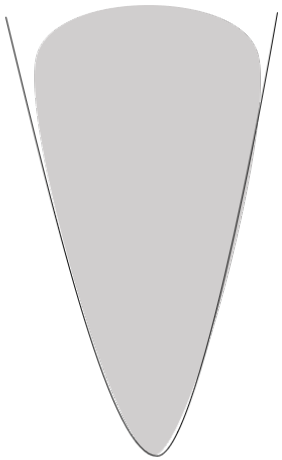
给定一个函数 $f(x)$，求 $f(x)$ 在 $[l, r]$ 中的最小值。

凸集

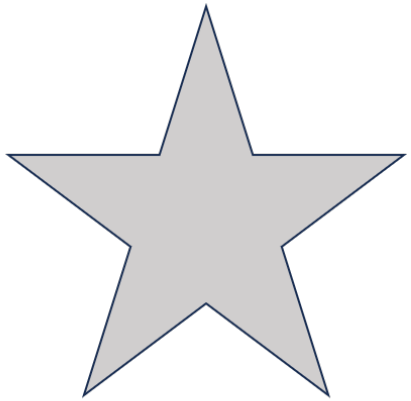
在一个平面上，如果一个区域中任意两点之间的连线都落在这个区域内，我们就说这个区域是“凸集”。



(a) 圆形是凸集



(b) 这条抛物线上方的区域是凸集

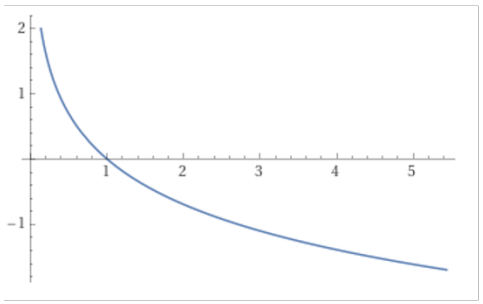


(c) 五角星不是凸集

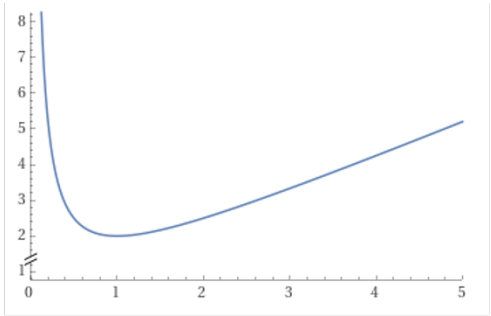
凸函数

函数图像 $y = f(x)$ 将平面直角坐标系分割成上下两个部分，如果上半部分是凸集，那么我们就说 $f(x)$ 是凸函数。

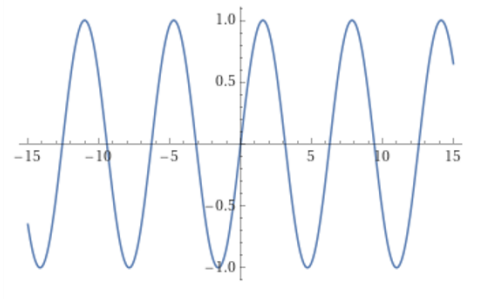
- 抛物线 $f(x) = x^2$ 是凸函数；
- 对勾函数 $f(x) = x + \frac{1}{x}\ (x > 0)$ 是凸函数；
- 负对数 $f(x) = -\ln x\ (x > 0)$ 是凸函数；
- 三角函数 $f(x) = \sin x$ 不是凸函数。



(a) 负对数函数



(b) 对勾函数

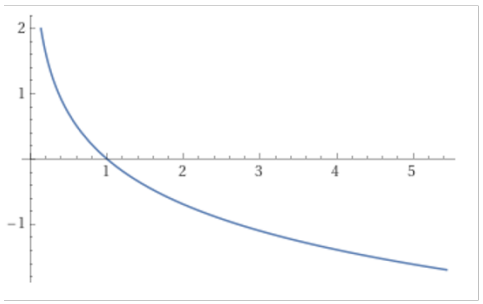


(c) 三角函数

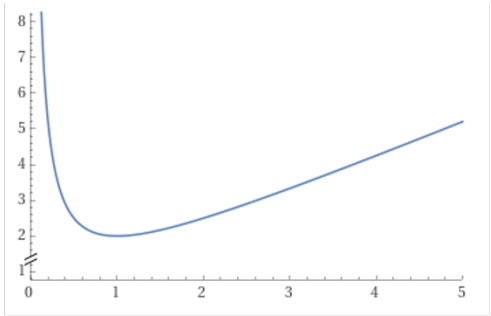
凸函数

函数图像 $y = f(x)$ 将平面直角坐标系分割成上下两个部分，如果上半部分是凸集，那么我们就说 $f(x)$ 是凸函数。

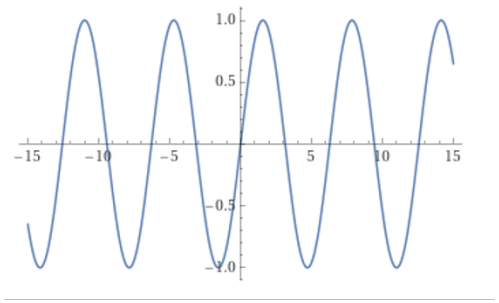
- 抛物线 $f(x) = x^2$ 是凸函数；
- 对勾函数 $f(x) = x + \frac{1}{x}\ (x > 0)$ 是凸函数；
- 负对数 $f(x) = -\ln x\ (x > 0)$ 是凸函数；
- 三角函数 $f(x) = \sin x$ 不是凸函数。



(a) 负对数函数



(b) 对勾函数



(c) 三角函数

如果 $[l, r]$ 在凸函数 $f(x)$ 的定义域内，那么 $f(x)$ 在 $[l, r]$ 中存在唯一的最小值点。

求凸函数的最小值

给定一个凸函数 $f(x)$，求 $f(x)$ 在 $[l, r]$ 中的最小值。

求凸函数的最小值

给定一个凸函数 $f(x)$，求 $f(x)$ 在 $[l, r]$ 中的最小值。

三分法。

```
1  double search(double l, double r)
2  {
3      double mid1 = l + (r-l) / 3;
4      double mid2 = r - (r-l) / 3;
5      while(mid2-mid1 > 1e-9)
6      {
7          if(f(mid1) > f(mid2)) l = mid1;
8          else r = mid2;
9          mid1 = l + (r-l) / 3;
10         mid2 = r - (r-l) / 3;
11     }
12     return l;
13 }
```

求凸函数的最小值

给定一个凸函数 $f(x)$，求 $f(x)$ 在 $[l,r]$ 中的最小值。

三分法。

```
1  double search(double l, double r)
2  {
3      double mid1 = l + (r-l) / 3;
4      double mid2 = r - (r-l) / 3;
5      while(mid2-mid1 > 1e-9)
6      {
7          if(f(mid1) > f(mid2)) l = mid1;
8          else r = mid2;
9          mid1 = l + (r-l) / 3;
10         mid2 = r - (r-l) / 3;
11     }
12     return l;
13 }
```

习题：【P1883】函数

求凸函数的最小值

一种卡常到极致的方法：0.618 法。令：

$$c = \frac{\sqrt{5}-1}{2} (\approx 0.618),$$

可以验证，$1 - c^2 = c$. 在区间 $[l, r]$ 中，我们取

$$mid_1 = r - c(r-l), \quad mid_2 = l + c(r-l).$$

如果 $f(mid_1) > f(mid_2)$，就令 $l = mid_1$；否则 $r = mid_2$。

求凸函数的最小值

一种卡常到极致的方法：0.618 法。令：

$$c = \frac{\sqrt{5}-1}{2} (\approx 0.618),$$

可以验证，$1 - c^2 = c$. 在区间 $[l, r]$ 中，我们取

$$mid_1 = r - c(r-l), \quad mid_2 = l + c(r-l).$$

如果 $f(mid_1) > f(mid_2)$，就令 $l = mid_1$；否则 $r = mid_2$。
现在假设 $l = mid_1$ （另一种情况是类似的），那么下一步，我们有：

$$mid_1' = r - c(r - mid_1), \quad mid_2' = mid_1 + c(r - mid_1).$$

求凸函数的最小值

一种卡常到极致的方法：0.618 法。令：

$$c = \frac{\sqrt{5}-1}{2} (\approx 0.618),$$

可以验证，$1 - c^2 = c$. 在区间 $[l, r]$ 中，我们取

$$mid_1 = r - c(r-l), \quad mid_2 = l + c(r-l).$$

如果 $f(mid_1) > f(mid_2)$，就令 $l = mid_1$；否则 $r = mid_2$。
现在假设 $l = mid_1$ （另一种情况是类似的），那么下一步，我们有：

$$mid_1' = r - c(r - mid_1), \quad mid_2' = mid_1 + c(r - mid_1).$$

可以发现：

$$\begin{aligned} mid_1' &= r - c(r - mid_1) = r - c(r - (r - c(r-l))) = r - c^2(r-l) \\ &= r - (1-c)(r-l) = l + c(r-l) = mid_2. \end{aligned}$$

所以不需要重复计算 $f(mid_1')$，只需要计算 $f(mid_2')$ 即可。

求凸函数的最小值

一种卡常到极致的方法：0.618 法。令：

$$c = \frac{\sqrt{5}-1}{2} (\approx 0.618),$$

可以验证，$1 - c^2 = c$. 在区间 $[l, r]$ 中，我们取

$$mid_1 = r - c(r-l), \quad mid_2 = l + c(r-l).$$

如果 $f(mid_1) > f(mid_2)$，就令 $l = mid_1$；否则 $r = mid_2$。
现在假设 $l = mid_1$ （另一种情况是类似的），那么下一步，我们有：

$$mid_1' = r - c(r - mid_1), \quad mid_2' = mid_1 + c(r - mid_1).$$

可以发现：

$$\begin{aligned} mid_1' &= r - c(r - mid_1) = r - c(r - (r - c(r-l))) = r - c^2(r-l) \\ &= r - (1-c)(r-l) = l + c(r-l) = mid_2. \end{aligned}$$

所以不需要重复计算 $f(mid_1')$，只需要计算 $f(mid_2')$ 即可。

- 三分法每次将区间缩小到 $\frac{2}{3}$ 倍，每次循环需要算两个点的函数值；
- 0.618 法每次将区间缩小到 0.618 倍，每次循环只要算一个点的函数值。

当函数值的计算复杂度为 $O(n)$ 时，0.618 法能带来巨大优势。

求凸函数的最小值

0.618 法代码

```
1  double search(double l, double r){
2      const double c = (sqrt(5)-1)/2;
3      double f1 = f(r - c * (r-l));
4      double f2 = f(l + c * (r-l));
5      while(r-l > 2e-8){
6          if(f1 > f2){
7              l = r - c * (r-l);
8              f1 = f2;
9              f2 = f(l + c * (r-l));
10         } else {
11             r = l + c * (r-l);
12             f2 = f1;
13             f1 = f(r - c * (r-l));
14         }
15     }
16     return (l+r)/2;
17 }
```

高维凸优化

以二维为例，给定一个函数 $f(x, y)$，定义域是 x, y 为任意实数，求它的最小值。

高维凸优化

以二维为例，给定一个函数 $f(x,y)$，定义域是 x,y 为任意实数，求它的最小值。

选择一个方向 (p,q)，令 $g(t)=f(x+tp,y+tq)$，然后用三分（或 0.618）法求 $g(t)$ 的最小值。然后换一个方向重复上述过程，直到充分接近最小值点为止。

高维凸优化

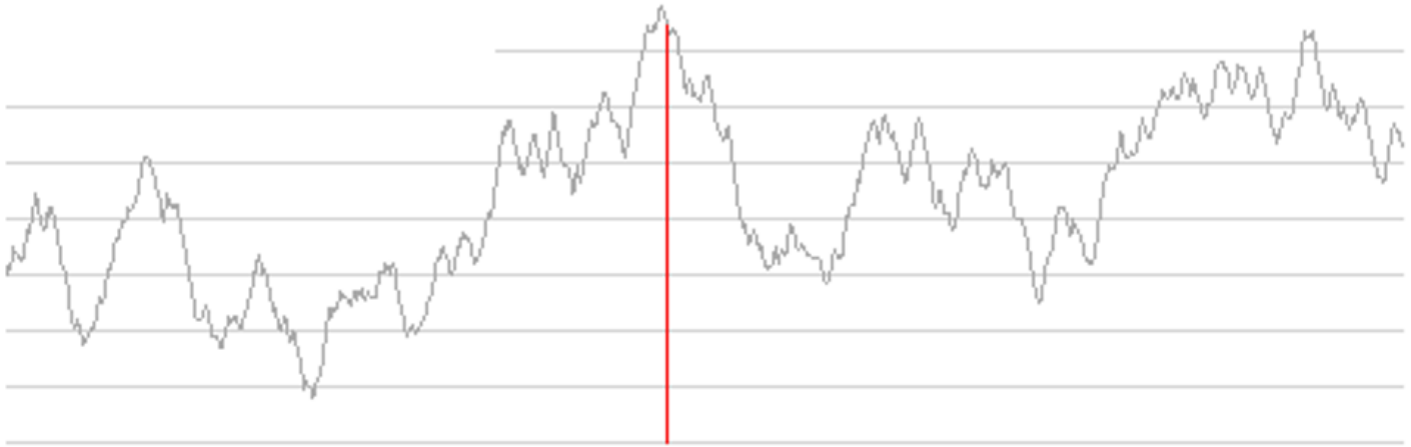
以二维为例，给定一个函数 $f(x,y)$，定义域是 x,y 为任意实数，求它的最小值。

选择一个方向 (p,q)，令 $g(t)=f(x+tp,y+tq)$，然后用三分（或 0.618）法求 $g(t)$ 的最小值。然后换一个方向重复上述过程，直到充分接近最小值点为止。

- 最速下降法：每次选择负梯度方向，即 $p=-\frac{\partial f}{\partial x}(x,y), q=-\frac{\partial f}{\partial y}(x,y)$;
- 共轭梯度法;
- ……

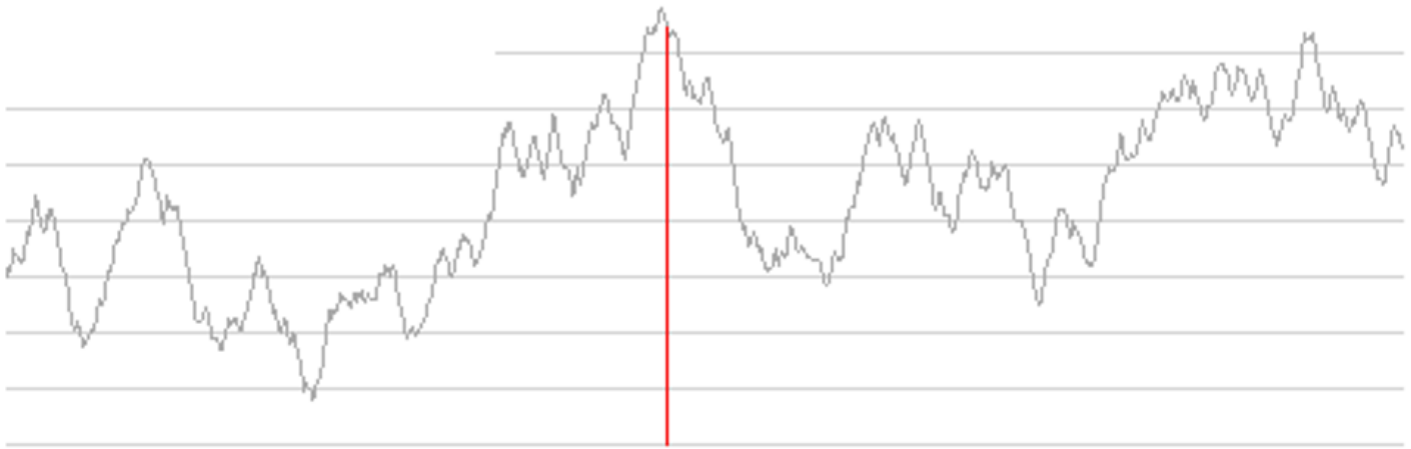
一维非凸优化

我们来看下面这个函数



一维非凸优化

我们来看下面这个函数



太复杂了，完全用不了三分法。

模拟退火

模拟退火的基本想法如下：

- 假设当前位置是 x，“温度”是 T，在 $[x-T, x+T]$ 中随机选取一个新位置 x_{new}；

模拟退火

模拟退火的基本想法如下：

- 假设当前位置是 x，“温度”是 T，在 $[x-T, x+T]$ 中随机选取一个新位置 x_{new}；
- 如果新位置更小，就把 x 更新为 x_{new}；否则“**抛硬币**”来决定是否把 x 更新为 x_{new}；

模拟退火

模拟退火的基本想法如下：

- 假设当前位置是 x，“温度”是 T，在 $[x-T, x+T]$ 中随机选取一个新位置 x_{new}；
- 如果新位置更小，就把 x 更新为 x_{new}；否则“**抛硬币**”来决定是否把 x 更新为 x_{new}；
- 逐步缩小 T，最后收敛到某个位置。

模拟退火

模拟退火的基本想法如下：

- 假设当前位置是 x，“温度”是 T，在 $[x-T, x+T]$ 中随机选取一个新位置 x_{new}；
- 如果新位置更小，就把 x 更新为 x_{new}；否则“**抛硬币**”来决定是否把 x 更新为 x_{new}；
- 逐步缩小 T，最后收敛到某个位置。

“抛硬币”是为了防止程序在陷入局部最优点之后不愿意出来寻找更优点。

模拟退火

模拟退火的基本想法如下：

- 假设当前位置是 x，“温度”是 T，在 $[x-T, x+T]$ 中随机选取一个新位置 x_{new}；
- 如果新位置更小，就把 x 更新为 x_{new}；否则“**抛硬币**”来决定是否把 x 更新为 x_{new}；
- 逐步缩小 T，最后收敛到某个位置。

“抛硬币”是为了防止程序在陷入局部最优点之后不愿意出来寻找更优点。这个硬币是一枚特殊的硬币，它有 p 的概率出现正面，$1-p$ 的概率出现反面，其中

$$p = e^{-\frac{f(x_{\text{new}})-f(x)}{T}},$$

抛到正面就更新，反面就不更新。

模拟退火代码

```
1  double f(double x){return ...}
2  double RAND(){return (double)rand() / RAND_MAX;}
3  double accept(double dta,double tem){
4      return dta < 0 || RAND() < exp(-dta/tem);
5  }
6  // tem: 初温; delta: 降温系数; end: 终温;
7  void anneal(double pnt, double tem, double delta, double end)
8  {
9      double ans = f(pnt);
10     while(tem > end)
11     {
12         double nxt = pnt + tem * (RAND()*2-1);
13         double nxtans = f(nxt);
14         if(accept(nxtans-ans, tem)) pnt = nxt;
15         tem *= delta;
16     }
17 }
```

$$\text{循环次数} = \log_{\text{降温系数}} \frac{\text{终温}}{\text{初温}}.$$

模拟退火的调参

模拟退火的三个参数一般按如下方式选择：

- 初温是可行解空间的尺寸；
- 终温是精度要求的 0.1 倍；
- 降温系数要保证程序运行时间足够久，但又不能超时。

模拟退火的技术

模拟退火有以下优化技巧：

- **记录最优解**：用一个全局变量保存每一次计算中的最小值；
- **卡时**：重复退火，直到即将 TLE 为止；
- **局部迭代**：在退火结束后把温度固定为终温，重复循环若干次。

实用的模拟退火板子

```
double f(double x){
    double res = ...;
    if(res < ans) ans = res, ansx = x; // 记录最优解
    return res;
}
double RAND(){return (double)rand() / RAND_MAX;}
double accept(double dta,double tem){
    return dta < 0 || RAND() < exp(-dta/tem);
}
// tem: 初温; delta: 降温系数; end: 终温;
void anneal(double pnt, double tem, double delta, double end){
    double local_ans = f(pnt);
    while(tem > end){
        double nxt = pnt + tem * (RAND() * 2 - 1);
        double nxtans = f(nxt);
        if(accept(nxtans-local_ans, tem)) pnt = nxt;
        tem *= delta;
    }
    for(int i = 1; i <= 1000; i++)  // 局部迭代1000次
        f(ansx + tem * (RAND() * 2 - 1));
}
bool TLE(){ // 卡时 0.9 秒
    return (double)(clock() - begint) / CLOCKS_PER_SEC > 0.9;
}
```

P1337 [JSOI2004] 平衡点

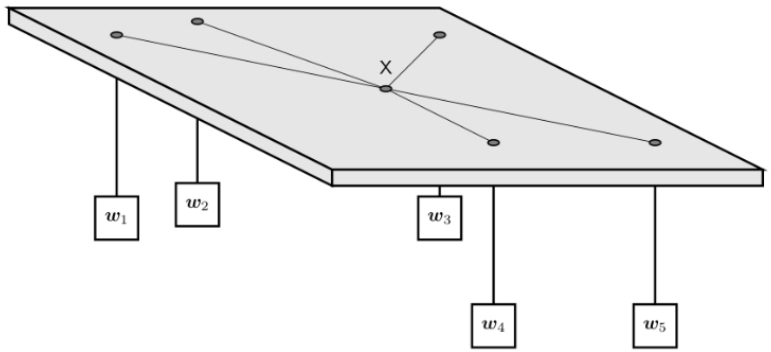
题目描述

复制Markdown 展开

如图，有 n 个重物，每个重物系在一条足够长的绳子上。

每条绳子自上而下穿过桌面上的洞，然后系在一起。图中 x 处就是公共的绳结。假设绳子是完全弹性的（即不会造成能量损失），桌子足够高（重物不会垂到地上），且忽略所有的摩擦，求绳结 x 最终平衡于何处。

注意：桌面上的洞都比绳结 x 小得多，所以即使某个重物特别重，绳结 x 也不可能穿过桌面上的洞掉下来，最多是卡在某个洞口处。



输入格式

文件的第一行为一个正整数 n（$1 \le n \le 1000$），表示重物和洞的数目。

接下来的 n 行，每行是 3 个整数 x_i, y_i, w_i，分别表示第 i 个洞的坐标以及第 i 个重物的重量。（$-10000 \le x_i, y_i \le 10000, 0 < w_i \le 1000$）

输出格式

你的程序必须输出两个浮点数（保留小数点后三位），分别表示处于最终平衡状态时绳结 x 的横坐标和纵坐标。两个数以一个空格隔开。

输入输出样例

输入 #1 复制

```
3
0 0 1
0 2 1
1 1 1
```

输出 #1 复制

```
0.577 1.000
```

P1337 [JSOI2004] 平衡点

看上去是要求“合力为零”的点，但模拟退火能做的事情是“求最小值”，如何转化？

P1337 [JSOI2004] 平衡点

看上去是要求“合力为零”的点，但模拟退火能做的事情是“求最小值”，如何转化？
实际上，我们可以求“势能最小”的点。第 i 个重物提供的势能是 w_iL_i，其中 L_i 是绳结离第 i 个洞的距离。假设绳结的位置是 (x,y)，我们要求势能函数

$$F(x,y)=\sum_{i=1}^{n} w_i\sqrt{(x-x_i)^2+(y-y_i)^2}$$

的最小值。

P1337 [JSOI2004] 平衡点

看上去是要求“合力为零”的点，但模拟退火能做的事情是“求最小值”，如何转化？
实际上，我们可以求“势能最小”的点。第 i 个重物提供的势能是 w_iL_i，其中 L_i 是绳结离第 i 个洞的距离。假设绳结的位置是 (x,y)，我们要求势能函数

$$F(x,y)=\sum_{i=1}^{n}w_i\sqrt{(x-x_i)^2+(y-y_i)^2}$$

的最小值。
模拟退火即可，如果没有一次 AC 可以尝试调参。

P5544 [JSOI2016] 炸弹攻击 1

题目背景

复制Markdown 展开

JYY 最近迷上了一款塔防游戏，在游戏里 JYY 除了建设建筑，还可以使用炸弹对屏幕上的敌人进行范围杀伤。

题目描述

游戏地图可以简单认为是一个二维平面。JYY 建造了N个建筑，每个建筑都是一个圆，其中第i个建筑的圆心位于(x_i, y_i)且半径为r_i。地图上一共有M个敌人，一个敌人可以近似看成一个平面上的点，其中第i个敌人位于(p_i, q_i)。JYY 可以使用一枚可以设置半径的炸弹，可以设置一个不超过R的范围，然后选择平面上的一个点引爆，范围内的所有敌人全部消灭。

当然，由于炸弹威力巨大，如果爆炸范围接触到 JYY 的建筑，那么 JYY 的建筑也会受到损伤。（注：如果炸弹的爆炸范围仅接触到了 JYY 建筑的边界，则不会对 JYY 的建筑造成损伤；如果敌人出现在了爆炸范围的边界，则该敌人被消灭）JYY 可以自由控制炸弹的爆炸地点和爆炸半径。作为一个保守的玩家，他希望在保证自己建筑毫发无损的情况下，消灭尽量多的敌人。

输入格式

第一行包含三个非负整数，分别为N, M, R；

接下来N行，每行三个整数，其中第i行为x_i, y_i, r_i，表示第i个建筑的位置和半径。数据保证所有建筑不相交（但是有可能边界接触）；

接下来M行，每行两个整数，其中第i行为p_i, q_i，表示第i个敌人的位置。

输出格式

输出一行一个整数，表示 JYY 最多可以消灭的敌人数量。

P5544 [JSOI2016] 炸弹攻击 1

位置确定之后，把建筑遍历一遍就知道最大的可爆炸半径是多少，再枚举一遍敌人即可算出被消灭的人数。

P5544 [JSOI2016] 炸弹攻击 1

位置确定之后，把建筑遍历一遍就知道最大的可爆炸半径是多少，再枚举一遍敌人即可算出被消灭的人数。用模拟退火来确定位置即可。

开始!

到此为止，我们介绍的都是模拟退火的正确用法，这里的“正确”是指：从概率论的角度可以证明这么做的正确性。

开始!

到此为止，我们介绍的都是模拟退火的正确用法，这里的“正确”是指：从概率论的角度可以证明这么做的正确性。
接下来，我们将介绍模拟退火的一些“错误”用法，它可以帮助你在毫无思路的时候“骗取”更多的分数。

旅行商问题

在一个带边权的完全无向图中，找一个权值和最小的哈密顿回路。
“哈密顿回路”是指经过每个点恰好一次的环。这是经典的 NPC 问题，

目前为止没有多项式复杂度的正确算法。

旅行商问题

可以使用模拟退火。先随机产生一个排列，然后每次迭代随机交换两个点的位置，再根据退火的准则来决定是否接受新的解。

```
1  void anneal(vector<int> pnt, double tem, double delta, double end){
2      double local_ans = f(pnt);
3      vector<int> nxt;
4      while(tem > end){
5          nxt = pnt;
6          swap(nxt[rand()%n], nxt[rand()%n]);
7          double nxtans = f(nxt);
8          if(accept(nxtans-local_ans, tem)) pnt = nxt;
9          tem *= delta;
10     }
11 }
```

最大团问题

求无向图的最大团。“团”是指：在图里选出若干个点，这些点两两之间都有连边。“最大团”就是最大的团。

最大团问题

还是模拟退火。每次迭代往当前团中随机添加一个新的点，然后将原团中与新点不相连的点移出，得到一个新团。将新团的大小与原团的大小比较，根据退火准则决定是否接受新解。

P3959 [NOIP2017] 宝藏

题目描述

参与考古挖掘的小明得到了一份藏宝图，藏宝图上标出了 n 个深埋在地下的宝藏屋， 也给出了这 n 个宝藏屋之间可供开发的 m 条道路和它们的长度。

小明决心亲自前往挖掘所有宝藏屋中的宝藏。但是，每个宝藏屋距离地面都很远，也就是说，从地面打通一条到某个宝藏屋的道路是很困难的，而开发宝藏屋之间的道路则相对容易很多。

小明的决心感动了考古挖掘的赞助商，赞助商决定免费赞助他打通一条从地面到某个宝藏屋的通道，通往哪个宝藏屋则由小明来决定。

在此基础上，小明还需要考虑如何开凿宝藏屋之间的道路。已经开凿出的道路可以 任意通行不消耗代价。每开凿出一条新道路，小明就会与考古队一起挖掘出由该条道路所能到达的宝藏屋的宝藏。另外，小明不想开发无用道路，即两个已经被挖掘过的宝藏屋之间的道路无需再开发。

新开发一条道路的代价是 $\mathrm{L} \times \mathrm{K}$。其中 L 代表这条道路的长度，K 代表从赞助商帮你打通的宝藏屋到这条道路起点的宝藏屋所经过的宝藏屋的数量（包括赞助商帮你打通的宝藏屋和这条道路起点的宝藏屋）。

请你编写程序为小明选定由赞助商打通的宝藏屋和之后开凿的道路，使得工程总代价最小，并输出这个最小值。

输入格式

第一行两个用空格分离的正整数 n, m，代表宝藏屋的个数和道路数。

接下来 m 行，每行三个用空格分离的正整数，分别是由一条道路连接的两个宝藏屋的编号（编号为 $1-n$），和这条道路的长度 v。

P3959 [NOIP2017] 宝藏

如果已经知道了宝藏屋之间的树形结构，计算总代价显然是很简单的。
我们考虑用模拟退火来确定这个树形结构。

P3959 [NOIP2017] 宝藏

如果已经知道了宝藏屋之间的树形结构，计算总代价显然是很简单的。
我们考虑用模拟退火来确定这个树形结构。

```
1  void anneal(int root,double tem,double delta,double end){
2      int res=calc(root);
3      while(tem>end){
4          int x=rand()%n+1,y=rand()%n+1;
5          while(x==y||x==root) x=rand()%n+1,y=rand()%n+1;
6          int pre=fa[x];fa[x]=y;  // 随机更换某个节点的父亲
7          if(!check()){fa[x]=pre;continue;}  // 若更换后不再是树形结构，重新随机
8          LL nxt=calc(root);
9          if(accept(nxt-res,tem)) res=nxt;  // 用退火准则来决定是否接受新解
10         else fa[x]=pre;
11         tem*=delta;
12     }
13 }
```

P3878 [TJOI2010] 分金币

题目描述

复制Markdown 展开

现在有 n 枚金币，它们可能会有不同的价值，第 i 枚金币的价值为 v_i。

现在要把它们分成两部分，要求这两部分金币数目之差不超过 1，问这样分成的两部分金币的价值之差最小是多少?

输入格式

本题单个测试点内有多组测试数据。

输入的第一行是一个正整数 T，表示该测试点内数据组数。

对于每组测试数据的格式为:

每组测试数据占两行。

第一行是一个整数 n，表示金币的个数。

第二行有 n 个整数，第 i 个整数表示第 i 个金币的价值 v_i。

输出格式

对于每组数据输出一行一个整数表示答案。

P3878 [TJOI2010] 分金币

第一组金币的数量是 $n_1 = \left\lfloor \frac{n}{2} \right\rfloor$，第二组金币的数量是 $n_2 = n - n_1$。

P3878 [TJOI2010] 分金币

第一组金币的数量是 $n_1 = \left\lfloor \frac{n}{2} \right\rfloor$，第二组金币的数量是 $n_2 = n - n_1$。

模拟退火，每次迭代随机交换第一组中的某个金币和第二组中的某个金币，然后把新的价值差和原来的价值差比较，根据退火准则决定是否接受新解。

Thank You