随机化算法总结

刘勤

lqgy2001@gmail.com

2007 年 5 月 25 日

目 录

Chapter 1

引论

定义1. 随机化算法是这样一种算法，在算法中使用了随机函数，且随机函数的返回值直接或间接地影响了算法的执行流程或执行结果。

所以如果一个算法是随机化算法，则它执行的流程或结果就会受其中使用的随机函数的影响。我们按影响的性质和程度分三种情况：

1. 随机不影响执行结果。这时，随机必然影响了执行的流程，其效应多表现为算法的时间效率的波动。

2. 随机影响执行结果的正确性。在这种情况中，原问题要求我们求出某个可行解，或者原问题为判定性问题，随机的效应表现为执行得到正确解的概率。

3. 随机影响执行结果的优劣。这时，随机的效应表现为实际执行结果与理论上的最优解或期望结果的差异。

那么随机化算法和“运气”的关系如何呢？由于随机化算法的随机种子不同，每次运行的正确性、时间效率都有可能不同，而一个好的随机算法应该保证算法的稳定性。算法的稳定性是评价一个随机化算法的重要指标。

Chapter 2

常见算法

2.1 纯随机化算法

问题1. *Troublemakers*
来源: *UVa 10982*

问题描述：

Every school class has its troublemakers — those kids who can make the teacher's life miserable. On his own, a troublemaker is manageable, but when you put certain pairs of troublemakers together in the same room, teaching a class becomes very hard. There are n kids in Mrs. Shaida's math class, and there are m pairs of troublemakers among them. The situation has gotten so bad that Mrs. Shaida has decided to split the class into two classes. Help her do it in such a way that the number of troublemaker pairs is reduced by at least a half.

输入数据：

The first line of input gives the number of cases, N. N test cases follow. Each one starts with a line containing n ($0 \leq n \leq 100$) and m ($0 < m < 5000$). The next m lines will contain a pair of integers u and v meaning that when kids u and v are in the same room, they make a troublemaker pair. Kids are numbered from 1 to n.

输出数据：

For each test case, output one line containing "Case #x:" followed by L — the number of kids who will be moved to a different class (in a different room). The next line should list those kids. The total number of troublemaker pairs in the two rooms must be at most $\frac{m}{2}$. If that is impossible, print "Impossible." instead of L and an empty line afterwards.

样例输入：

```
2
4 3
1 2
2 3
3 4
4 6
1 2
1 3
1 4
2 3
2 4
3 4
```

样例输出：

```
Case #1: 3
1 3 4
Case #2: 2
1 2
```

问题分析：

题目大意是在图 G 中寻找一个 边数 $\geq \frac{m}{2}$ 的割，n 为点数，m 为边数。

首先介绍贪心算法 1：

证明. 让 E_i 表示所有的 $Edge(i,j),(j<i)$ 。显然 $E_1 \dots E_n$ 的交为空，而并为 E(所有边) 。在第 i 步，算法检查 E_i 中的边，a,b 分别表示集合 A,B 中连接 i 的边的数量（ $[A,B]$ 是图 G 的一个割)。当 $a \leq b$ 时，点 i 加入集合 A

Algorithm 1 Troublemakers 贪心算法

```
1: A ← {}
2: B ← {}
3: for i = 1 to n do
4:     a ← 0
5:     b ← 0
6:     for j = 1 to i − 1 do
7:         if there is a edge(i, j) then
8:             if j ∈ A then
9:                 a++
10:            else
11:                b++
12:            end if
13:        end if
14:    end for
15:    if a ≤ b then
16:        B ← A + i
17:    else
18:        B ← B + i
19:    end if
20: end for
```

，否则加入集合 B 。所以每一步都恰好有 $\max\{a,b\}$ 的边加入割中。而显然地 $\max\{a,b\} \geq \frac{|E_i|}{2}$ 。所以最后割的数量大等于 $\sum_{i=1}^{n} \frac{|E_i|}{2} = \frac{m}{2}$ ，满足了题目的要求。 □

在比赛时我没有想到以上的贪心方法。所以我使用了随机化算法。注意到 $\frac{m}{2}$ 这个宽松的条件，我猜想这道题的解可能很丰富。所以我认为随机地把点分成两类，这样形成的割边有很大的机会大等于 $\frac{m}{2}$ 。因此这道题的随机化算法就是每步随机的把点分成两类，如果所形成的割边大等于 $\frac{m}{2}$ 就停止，否则重复以上步骤直到找到解为止（根据上面的证明，本题不存在无解情况）。 □

问题2. *坦克 (tank)*

来源: *CTSC 2007*

问题描述：

在一个平面上上有 N 个坦克（可以认为是点）。同时还有 M 个栅栏一样的障碍物，每个都可以看作是一条线段，这些线段不会相交，也不会有公共端点。你要用一个激光发射器摧毁所有坦克。

激光发射器可以被放在任何位置（但不能放在障碍物或坦克上），一旦放置后就不能再移动。它可以向任何方向发射激光，激光沿直线运动，不能横穿栅栏（但是可以紧贴着栅栏经过）。当激光碰到栅栏的时候会发生反射，此时可以把栅栏看作镜子，反射规律和镜面反射相同。

任何被激光打到（即在激光的运动路径上）的坦克都会被立即摧毁，我们称从发射源开始，沿着这条激光运动直到目标坦克的路径为激光攻击路径。但激光在每次被栅栏反射的时候都会有能量的衰减，如果被反射的次数超过 K 次，激光就不会再具有摧毁敌方克的能力了。

希望你能够找到一个合适的位置安放激光反射器来摧毁所有的坦克，并且设计摧毁每个敌方坦克需要的激光方向。同时，他希望在满足摧毁所有敌方坦克的前提下，最长的激光攻击路径最短。

本题是提交答案式题目。

问题分析：

一道激光朝某个方向发射后，又经过 K 次反射后的路线无疑是难以计算的。而题目又要求最长的激光攻击路径最短。这使得我们想到无疑走直

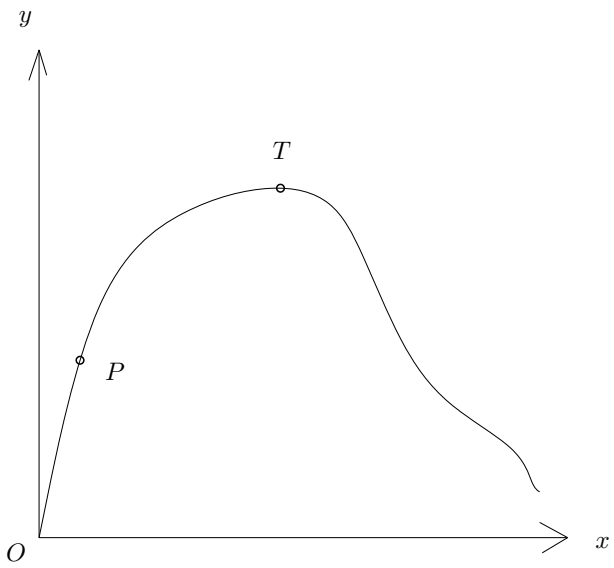

Figure 2.1: 牛顿爬山法示意图

线是最好的选择，但是由于我们很难求出在哪些区域放置激光发射器可以使激光一次击中目标坦克，所以只好使用随机化算法。

随机找一个点放置激光发射器，判断这个点到每个坦克之间是否有障碍物，如果有则继续随机，否则更新最优值。在重复一定次数后停止。

虽然这道题用随机化算法只能拿到 70 分左右，但还是很有价值的。 □

以上两种算法似乎都很简单，然而考试的时候用出来的人却属于凤毛麟角。一方面是现在人们对随机化算法的了解不多，另一方面说明了随机化算法也有一定的规律可循。

2.2 牛顿爬山法

图 2.1 为牛顿爬山法的原理。设爬山者在 P 点，为了爬上山峰，他可以向左或向右移动。爬山法要求每次移动之前计算新位置与当前位置的差（即改进量），一般选择改进量大的方向前进。由于有了这样的启发信息，一般很快就可以找到一个最值。

爬山法的缺点是可能陷入局部最优解，而无法找到全局最优解。这时我们可以使用禁忌搜索，来改进算法。简单说就是如果迭代了一定次数后爬山

法无法找到满足的条件的最优解则重新寻找其它的初始点，参看《禁忌搜索》。另一种改进就是下文所提到的模拟退火算法。

问题3. *Graceful*
来源: *Amber's Contest*

问题描述：

优美标号的概念是由 Rosa[1] 于 1967 年提出的。下面给出优美标号的定义。

定义2. *(Golomb, 1972)*: 具有 n 个点的图 G 的一个优美标号是一个定义在结点上的函数: $f : V(G) \to \{0, \ldots, n-1\}$，它使得不同的顶点被标记为不同的整数，且满足 $\{|f(u) - f(v)| : (u, v) \in E(G)\} = \{1, \ldots, n-1\}$。如果 G 有一个优美标号，则该图是优美的。

猜想1. *(优美树猜想，Ringel-KotzigConjecture，Ringel1964)*: 任意树都是优美的，即任意树都存在一个优美标号。[2]

过了 40 多年，优美树猜想还是没有得到彻底的证明 Michael Horton 在 2003 年写出了一篇总结性的论文 Graceful Trees: Statistics and Algorithms 其中总结了各种各样的已经被构造证明是优美的树其中最有意思的是点数小于等于 27 的树都有优美标号的证明。

但是固执的 Amber 下意识里认为这个命题是错的，你的任务就是尽快让 Amber 打消这个想法：给出一棵含 n 个结点的树，尽量快地构造出该树的一个优美标号由于规模很小，所以论文的证明很简单，相信聪明的你一定会想出来一个简单的构造方法的。

输入数据：

本题有多组数据，第一行包含一个正整数 $t(1 \le t \le 10)$，表示数据组数，下面共描述了 t 组数据。

[1]Rosa, A. 1967, On certain valuations of the vertices of a graph, in Theory of Graphs(International Symposium, Rome, July 1966), Gordon and Breach,NewYork,pp. 349-355.

[2]Ringel,G. 1964, Problem25, in Theoryof Graphsandits Applications, ProceedingsSymposiumSmolenice, Prague.

第一行包含一个整数 $n(1 \leq n \leq 10^6)$ 接下来 $n-1$ 行，每行包含两个正整数 $(u_i, v_i)(1 \leq u_i, v_i \leq n, u_i \neq v_i)$，表示结点 u_i 和结点 v_i 相连。

输出数据：

对于每组数据，打印 n 行。第 i 行包含一个整数 l_i，表示结点 i 的标号为 l_i。

问题分析：

本题是一道提交答案式问题，本论文只讨论可以用随机化算法解决的部分问题，即 $n \leq 160$ 的部分。

本题的题目描述十分冗长，邪恶的 Amber 想把人往构造的路上引，其实只有 $n \geq 10000$ 的毛毛虫数据，才能用 CEOI 2000 The Caterpillar 的方法构造，而且跟题目中的构造证明无关。

对于 $n \leq 160$ 的数据只要随机化就可以了，然而当时考场上有不少人想到了用随机化，但是却没有能够出解，这是为什么呢？因为他们都使用了纯随机算法，没有一点方向性，而这道题的解并没有题目 1 那么多，自然是几乎不可能直接随机出解。正确的解法应该是加上了爬山法和竞技搜索的随机化算法。

首先随机一个初始状态，然后尝试所有可以交换的节点，选择一个改进量最大的状态更新，直到找到一个解。由于找解的速度与初始解的优劣有关，所以在尝试了一定次数仍未出解的情况下要重新随机初始解。 □

2.3 模拟退火算法

模拟退火算法来源于固体退火原理。将固体加温至温度充分高，再让其徐徐冷却。加温时，固体内部粒子随升温变为无序状，内能增大；而徐徐冷却时粒子渐趋有序，在每个温度都达到平衡态，最后在常温时达到基态，内能减为最小。

根据 Metropolis 准则，粒子在温度 T 时趋于平衡的概率为

$$e - \frac{\Delta E(\text{内能改变量})}{k(\text{Boltzmann 常数}) * T}.$$

我们也可以把模拟退火算法看作是牛顿爬山法的一种改进，即在牛顿爬山法的基础上以一定的概率回退。

模拟退火算法描述如算法 2 所示。

Algorithm 2 模拟退火算法伪代码

初始化：初始温度 T（足够大），初始解 S，L（每个温度的迭代次数）
while $T > 0$ **do**
    for $k = 1$ to L **do**
        产生新解 S'
        计算增量 $\Delta t' := C(S') - C(S)$，其中 $C(S)$ 为评价函数
        if $\Delta t' < 0$ **then**
            接受新解 S' 作为当前解
        else
            以概率 $\exp(-\Delta t'/T)$ 接受 S'
        end if
        如果满足终止条件则终止
    end for
    温度 T 减小
end while

模拟退火算法的一个著名应用是 TSP 问题，就不在此赘述了。

2.4 遗传算法

在遗传算法里，优化问题的解被称为个体，它表示为一个参数列表，叫做染色体或者基因串。染色体一般被表达为简单的字符串或数字串，不过也有其他的表示方法适用。一开始，算法随机生成一定数量的个体，有时候操作者也可以对这个随机产生过程进行干预，播下已经部分优化的种子。在每一代中，每一个个体都被评价，并通过计算适应度函数得到一个适应度数值。种群中的个体被按照适应度排序，适应度高的在前面。这里的“高”是相对于初始的种群的低适应度来说的。

下一步是产生下一代个体并组成种群。这个过程是通过选择和繁殖完成的，其中繁殖包括杂交和突变。选择则是根据新个体的适应度进行的，适应度越高，被选择的机会越高，而适应度低的，被选择的机会就低。初始的

数据可以通过这样的选择过程组成一个相对优化的群体。之后，被选择的个体进入杂交过程。一般的遗传算法都有一个杂交率的参数，范围一般是 $0.6 \sim 1$，这个杂交率反映两个被选中的个体进行杂交的概率。例如，杂交率为 0.8，则 80% 的“夫妻”会生育后代。每两个个体通过杂交产生两个新个体，代替原来的“老”个体，而不杂交的个体则保持不变。杂交父母的染色体相互交错，从而产生两个新的染色体，第一个个体前半段是父亲的染色体，后半段是母亲的，第二个个体则正好相反。不过这里的半段不是真正的一半，这个位置叫做杂交点，也是随机产生的，可以是染色体的任意位置。再下一步是突变，通过突变产生新的“子”个体。一般遗传算法都有一个固定的突变常数，通常是 0.01 或者更小，这代表突变发生的概率。根据这个概率，新个体的染色体随机的突变，通常就是改变染色体的一个字节（ 0 变到 1，或者 1 变到 0 ）。

经过这一系列的过程（选择、杂交和突变），产生的新一代个体不同于初始的一代，并一代一代向增加整体适应度的方向发展，因为最好的个体总是更多的被选择去产生下一代，而适应度低的个体逐渐被淘汰掉。这样的过程不断的重复：每个个体被评价，计算出适应度，两个个体杂交，然后突变，产生第三代。周而复始，直到终止条件满足为止。一般终止条件有以下几种：

- 进化次数限制；
- 计算耗费的资源限制（例如计算时间、计算占用的内存等）；
- 一个个体已经满足最小值的条件，即最小值已经找到；
- 适应度已经达到饱和，继续进化不会造成适应度更好的个体；
- 人为干预；
- 以及以上两种或更多种的组合；

2.5 概率算法简介

详见《概率算法简介》和《理论计算机初步：概率算法和近似算法》。

Chapter 3

随机化贪心调整

随机贪心，就是用随机与贪心结合，在算法的每一步，都尽量使决策取得优，但不一定是最优决策。通过多次运行，使得算法能取得一个较优的解。

随机化贪心算法的基本思想是：设置贪心程度 $rate\%(rate \in [0, 100])$，选一种较好的贪心标准为基础，每次求局部最优解的过程改为每次求在该贪心标准下贪心程度不小于 $rate\%$ 的某个局部较优解。这一修改可由以下伪代码 3 描述：

Algorithm 3 随机化贪心算法

```
1: for A = 局部最优解 to 局部最差解 do
2:     if Random(100) ≤ rate then ▷ Random(100) 产生 [1, 100] 的随机数
3:         return A
4:     end if
5:     return 局部最差解
6: end for
```

对于目前任一种贪心标准，都存在不符合它的输入，也就是说，存在输入使算法在不是每次都选局部最优解的情况下，得到的解比每次都选局部最优解所得到的解更优，而上述随机化贪心算法能覆盖这种情况。同时上述随机化贪心算法在 $rate = 100$ 时得到的结果就是原来未加修改的贪心算法的结果，所以上述随机化的算法至少不比非随机化的差。

上述思想较简单，所以实现起来不困难。而且贪心算法都有很高的时间效率，多次贪心消耗的时间也不会很长。在实现中，我们还加了些其它的优化，如对重复项只计算一次。

问题4. *Matrix*

来源: *CTSC 2007*

问题描述：

给定一个整数 D，n 行 m 列的实数矩阵 A，其第 i 行第 j 列的元素是 $a_{i,j}$，且 $0 \leq a_{i,j} \leq D(1 \leq i \leq n, 1 \leq j \leq m)$。希望你能够由此提供一个 n 行 m 列的 01 矩阵。

对于给定的 A 矩阵和你提供的 B 矩阵，可以求出

$$p1 = \max \begin{cases} \max\limits_{1 \leqslant j \leqslant m} \{|b_{i,j} - a_{i,j}/D|\} \\ \max\limits_{1 \leqslant i \leqslant m} \{|b_{i,j} - a_{i,j}/D|\} \end{cases};$$

$$p2 = \max_{1<i \leqslant n, 1<j \leqslant m} \{|b_{i,j} + b_{i-1,j} + b_{i,j-1} + b_{i-1,j-1} - (a_{i,j} + a_{i-1,j} + a_{i,j-1} + a_{i-1,j-1})/D|\}$$

在不同的测试数据中，我们希望提供的 B 矩阵能使 $p1$ 或者 $p2$ 尽量小。

问题分析：

标准算法是构造的，比较复杂，不介绍了。

对于第一问，题目要求 $p1 \leq 1$，这等价于限制了 B 矩阵中每一行和每一列的 1 的数量，这样可以用有上下界的网络流来做。另一种方法是贪心，统计 B 矩阵中每一行和每一列需要的 1 的数量，那么我只要记录当前每一行还剩多少个 1，然后一列一列地看过来。我把所有行的 1 按照从大到小排序，然后按这个顺序在这列上添加 1。

对于第二问，题目要求 $p2 \leq 1.5$，有一种贪心是举行的从右上对于每四个格子按照左下、左上、右下、右上的顺序调整，直到满足要求。一种类似想法的随机化算法是对任何不满足条件的四个格子随机产生对应 B 矩阵中的数，直到所有格子满足要求。

第二问的随机化算法非常简单。考场上一些人虽然想到了随机化算法，却没有想到这种微调的方法，导致效果差了很多。这说明根据题目的特性，从细微的地方逐步接近解是很重要的。纯粹的对整个 B 矩阵进行随机很难得到好的解。

ps. 感谢袁洋提供的两种贪心。 □

Chapter 4

总结

本文介绍了牛顿爬山法、模拟退火算法、遗传算法和几种概率算法，这几种算法的精髓都是在随机的过程中不断的调整，以期不断的接近目标。实践中，随机化算法的设计没有公式可套，所以有时不一定要使用特定的随机化算法，只要贪心调整就可以得到好的效果。

随机化算法主要受限于正确性、稳定性和时间效率。如果一个问题对算法不强求 100% 的稳定，即对于同样的输入，不必每次运行的情况都相同（当然这种不稳定性不能太大），同时作为补偿的，又要求算法在其它某些方面有较好的性能 (如出解迅速)，而这些性能是一般非随机化算法无法达到的，那么此时，随机化算法可能就是能有效解决问题的候选算法之一。否则，随机化算法便不适用。

当一个随机化算法适用于解决某个问题，且该算法有较高的稳定性，同时它在其它某些方面有突出表现 (如速度快，代码短等)，能比一般非随机化算法做得更出色，那么这个随机化算法就是一个行之有效的算法。

参 考 文 献

[ZYJ01] 周咏基。论随机化算法的原理与设计，1999。

[HWD01] 胡伟栋。浅析非完美算法在信息学竞赛中的应用，2005。

[LZ01] 李世炳，鄒忠毅。簡介導引模擬退火法及其應用。（物理雙月刊（廿四卷二期）2002 年 4 月）

[TWB01] 唐文斌。浅谈“调整”思想在信息学竞赛中的应用，2006。