浅谈随机化在信息学竞赛中的应用

广东省韶关市第一中学 刘家骅

引言

信息学竞赛的题目日新月异

新型算法层出不穷

随机化算法作为一种新兴算法

犹如新生的太阳在信息学竞赛的
广阔天空上焕发光芒

简单问题的另类算法

例题 Geometrical dreams(Ural1046)

- 有一个多边形 $A_1A_2...A_N$ ，在每条边 A_iA_{i+1} 上向多边形外做一个等腰三角形 $A_iM_iA_{i+1}$ 使得角 $A_iM_iA_{i+1}=\alpha_i$
- 由 α_i 组成的集合满足其任何非空子集的角度和不是 360 度的倍数
- 给出 $N(3 \le N \le 50)$ ，所有 M_i 的坐标和 α_i ，写一个程序，输出多边行的顶点的坐标

例题 Geometrical dreams(Ural1046)

- 一般方法——简单解方程
- 有没有其他方法呢？
- 让我们换一个角度思考问题
- 只要确定了一个顶点的坐标，多边形的其他顶点的坐标就能够通过简单计算得到，那么问题就转化为确定多边形的一个顶点的坐标

例题 Geometrical dreams(Ural1046)

确定一个顶点的位置？

随机化

当第一个顶点的位置被暂时确定

通过计算能够得到第 N+1 个顶点的位置

当第 N+1 个顶点越接近第 1 个顶点

这个顶点的位置就越接近其实际位置

例题 Geometrical dreams(Ural1046)

- 我们每次可以在当前最优位置旁边随机化确定第一个顶点的位置，然后计算此时第 N+1 个顶点与第 1 个顶点的距离
- 如果这个距离比当前最优距离更小，那么我们就用这个位置更新当前最优位置
- 显然，当第 1 个顶点与第 N+1 个顶点重合时，此时第 1 个顶点的位置即为其实际位置

例题 Geometrical dreams(Ural1046)

- 虽然这样的方法显然在任何方面都要比前面提到的普通做法要复杂
- 对于解决这道题目没有太大的意义
- 但是，它提供给我们一种崭新的思路——

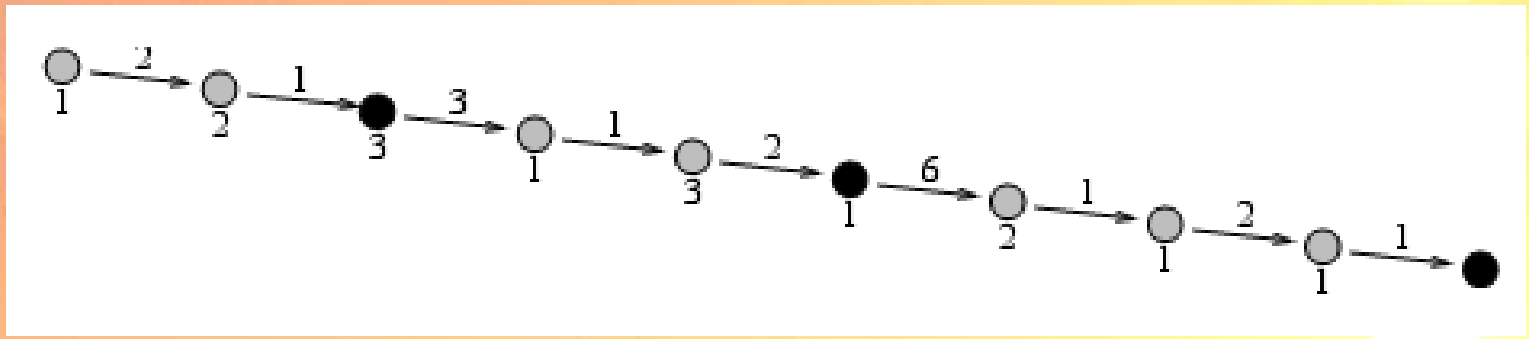
随机化

•随机化算法对于这样的题目没有优势，但它在很多问题上都能得到运用，下面，我们一起来进一步领略随机化算法的魅力吧

小试牛刀

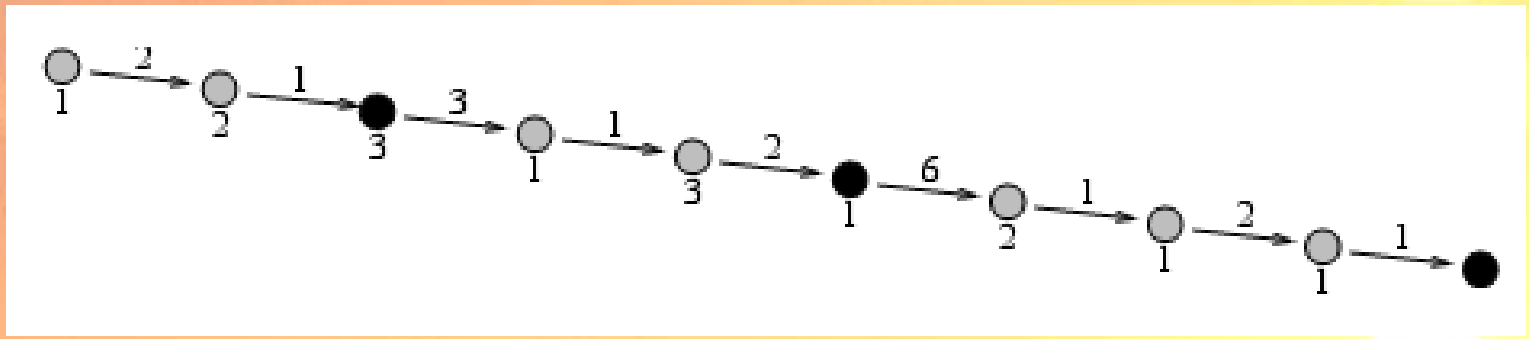
例题： Two sawmills(CEOI2004)

- 从山顶到山脚的路上有 n 棵老树，现在政府决定砍掉它们，为了不浪费木材，每一棵树都会被转运到锯木场
- 树只能往一个方向运输——向下



例题： Two sawmills(CEOI2004)

- 在路的尽头有一个锯木场。两个额外的锯木场可以在路上任意一棵老树位置上
- 你必须选择在哪里建造，使得运输的费用达到最少
- 运输费用是一分每米每千克木材



例题： Two sawmills(CEOI2004)

- 这道题目的标准算法将数据转化为图象，用栈进行处理求出两个矩形的最大覆盖面积，时间复杂度为 O(N)。
- 但是，这种算法对能力要求不小，不太容易想到。
- 我们看下随机化算法在这题上的表现

例题： Two sawmills(CEOI2004)

- 首先最容易想到的随机化当然就是直接随机寻找两个点，计算出以这两个点为锯木场时的总运费，多次随机后将总费用最小的输出
- 我们可以进行预处理，将计算的时间复杂度降为 O(1)，那么在时限内我们可以随机几百万次甚至几千万次，但是相对于总状态的四亿来说，寻找到最优解的几率不是很大

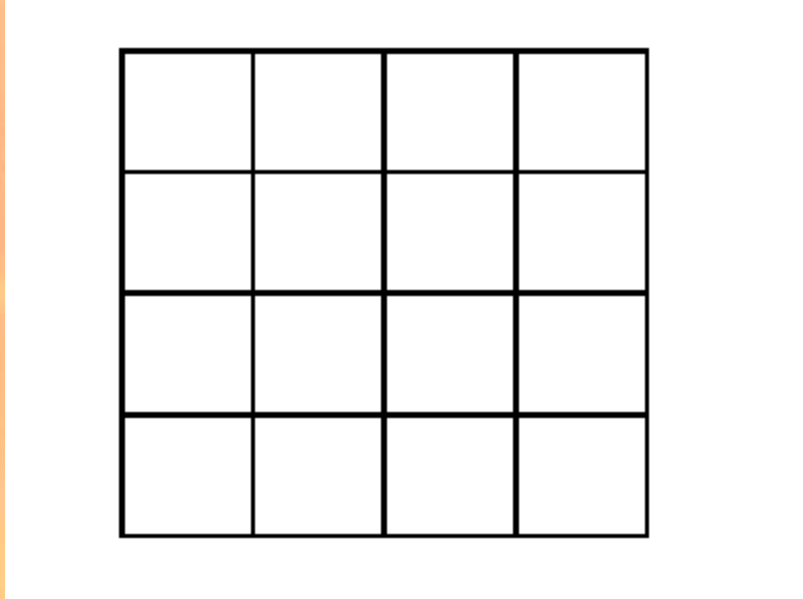
例题： Two sawmills(CEOI2004)

有没有更好的方法呢？

- 我们刚才是用随机化算法直接出解，准确性不太好
- 为了增加准确性，那么我们尝试一下用随机化来缩小区域范围

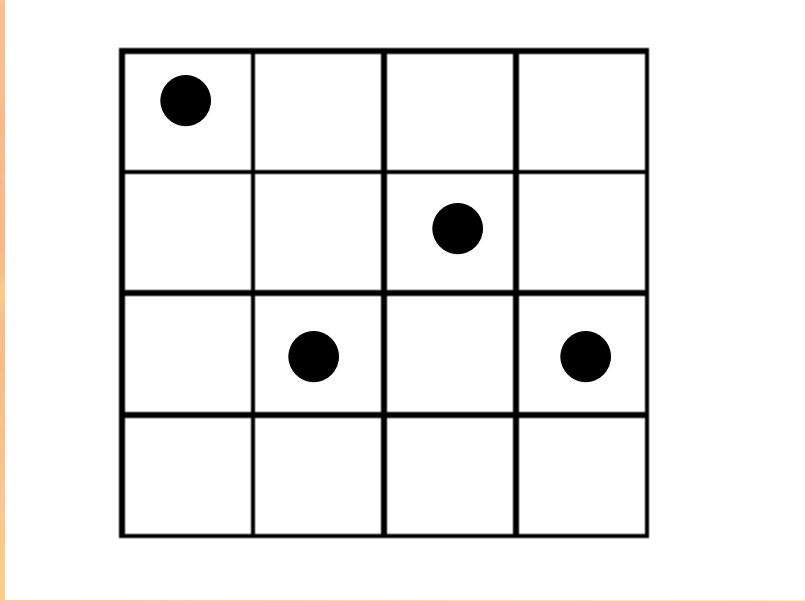
例题： Two sawmills(CEOI2004)

- 我们建立一个矩阵 P ， P[X,Y] 表示第一个锯木场建立在 X ，第二个锯木场建立在 Y 时的总运费



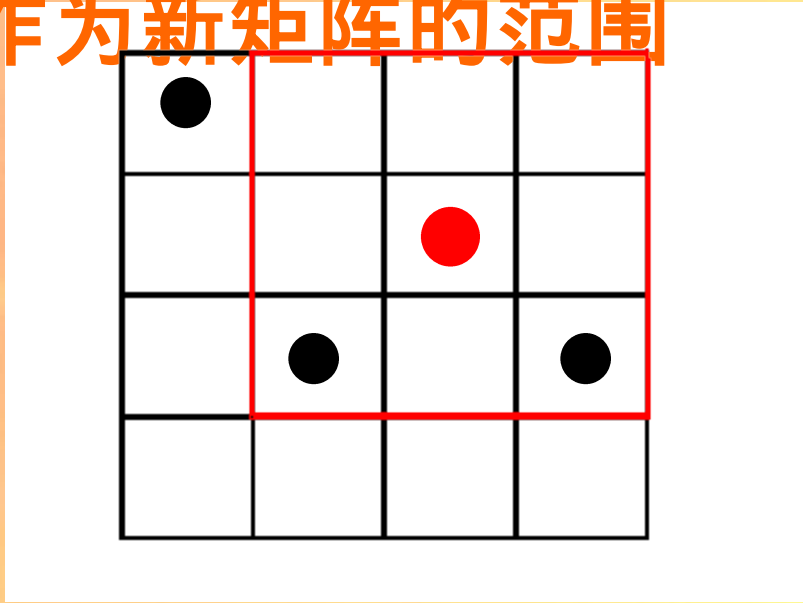
例题： Two sawmills(CEOI2004)

- 一开始时，矩阵的边长为 N
- 我们随机寻找一定数量的点，计算出它们的值



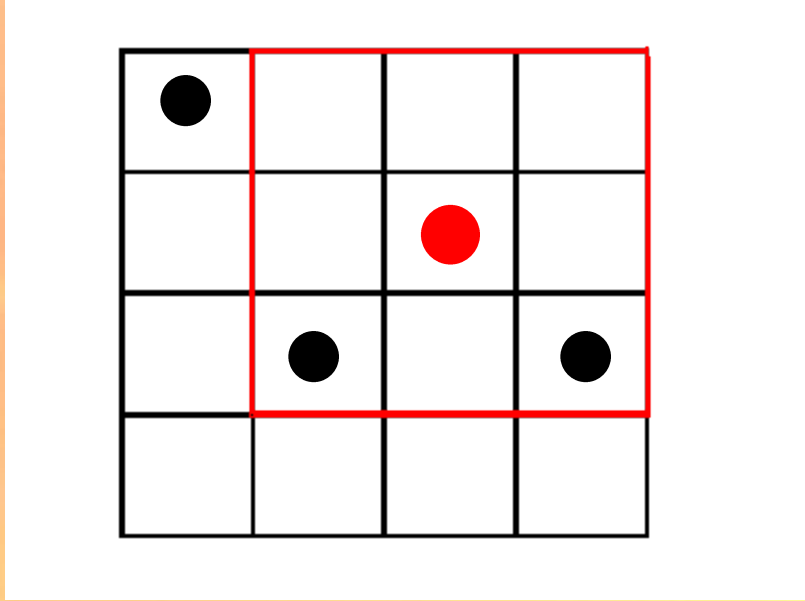
例题： Two sawmills(CEOI2004)

- 选取值最小的点
- 以这个点为新矩阵的中心，以现在矩阵的边长的固定比例长度作为新矩形的边长 (如图中取 3/4) ，从原来的矩阵中取出一块作为新矩阵的范围



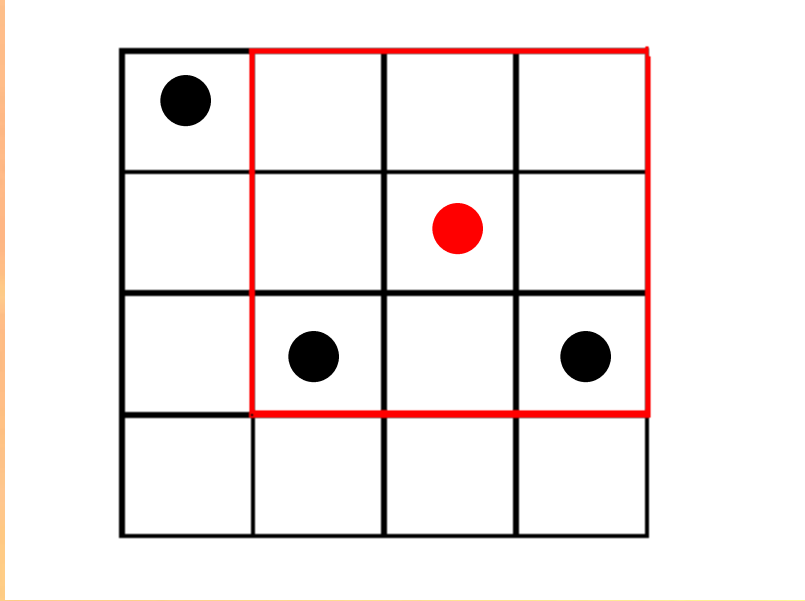
例题： Two sawmills(CEOI2004)

- 然后继续在新矩阵中重复这样的操作，直至新矩阵足够小时，我们即可枚举新矩阵上的每一个点，取其中最小值作为答案。



例题： Two sawmills(CEOI2004)

- 我们惊喜地发现，这种随机化算法对于测试数据能够全部通过！



例题： Two sawmills(CEOI2004)

通过这道题目可以看出：

- 随机化算法的灵活多变使得它的具有更为广阔的运用范围面
- 而这样的多变性也使得我们需要灵活恰当地运用随机化算法才能发挥出它的优势
- 随机化算法并不只是简单地随便乱来，使用随机化算法的时候与其他算法一样值得细细斟酌，需要匠心独运

让我们来看看随机化算法在实际比赛中的运用解析

实战解析

例题：小 H 的聚会 (NOI2005)

- 题目大意是给出由 N 个点、 M 条边组成的图，求最大生成树
- 在图 1 所示例子中黑色边组成的树即为最优方案

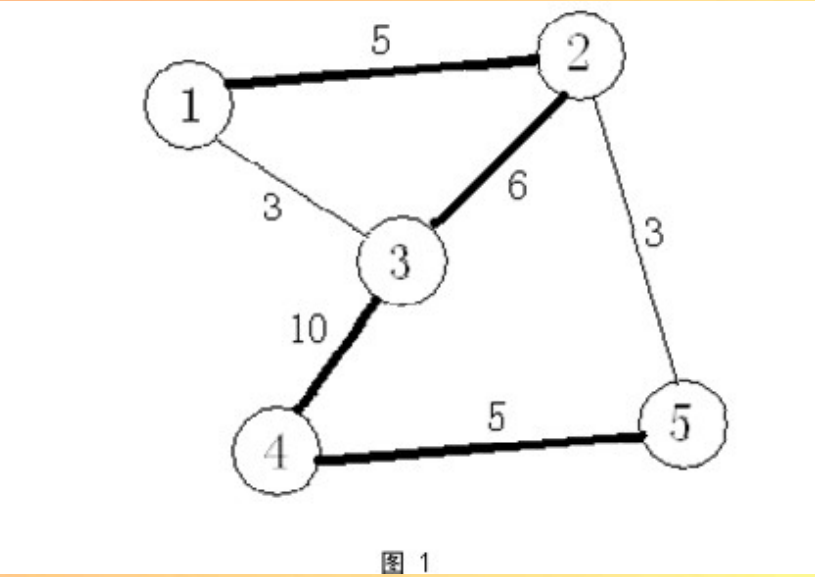


图 1

例题：小 H 的聚会 (NOI2005)

- 但是，为了减轻每个顶点的负担，题目设定了每个顶点的最大连接边数 ki
- 还是用图 1 的例子，若我们为 1 至 5 每个点分别加上了 k_i = 1, 1, 4, 2, 2 的限制，则上述方案就不能满足要求了。此时的最优方案如图 2 所示

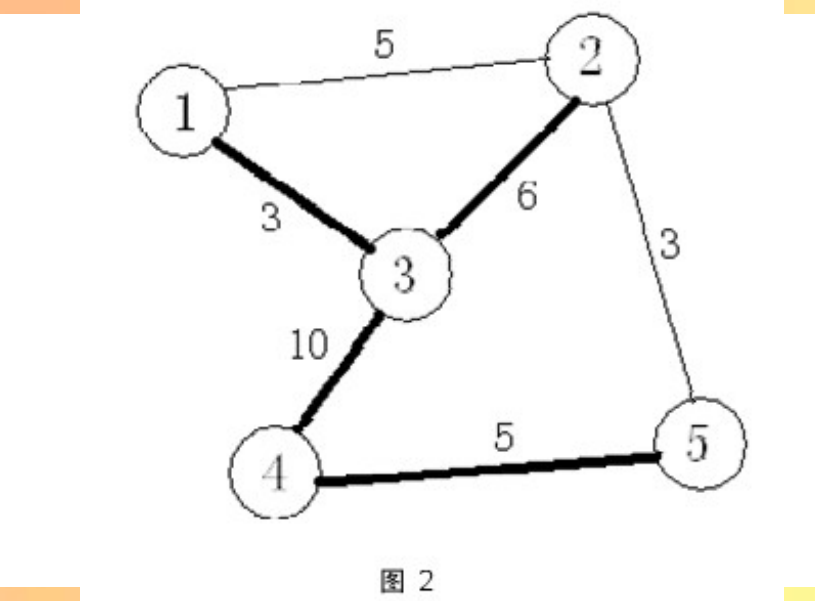


图 2

例题：小 H 的聚会 (NOI2005)

- 这是一道提交答案题，题目数据分为三种类型
- 第一种类型包括第 1-3 个数据，每个点的度限制均为 N-1 ，等同于没有限制，直接用最小生成树算法即可解决
- 第二种类型包括第 4-6 个数据，其中 N-1 个点的度限制为 N-1 ，只有一个点有度限制，可以使用最小度限制生成树算法解决。

例题：小 H 的聚会 (NOI2005)

- 题目数据的第三种类型包括第 7-10 个数据，数据中所有点都有度限制，没有太好的准确算法
- 而这个题目又是不要求最优解的开放性题目
- 这种问题实在是随机化算法自由翱翔的广阔天空。
- 我们可以采取随机化算法结合不同算法解决这个题目。

例题：小 H 的聚会 (NOI2005)

方法一：基于 Kruskal 的随机化算法

- 我们用 Kruskal 算法计算出符合点的度限制的一棵生成树
- 由于有了度限制，所以这样得出的生成树不一定是最优的
- 我们可以在按边权大小排序后，再随机打乱部分边的顺序，再用 Kruskal 算法，以求得到更优的解。

例题：小 H 的聚会 (NOI2005)

方法一：基于 Kruskal 的随机化算法

- 在每次得到这样一棵较大的生成树后，我们再随机化选取一条不在生成树上的边，加入生成树上得到一个圈
- 再在这个圈上试图删除一条边使得生成树仍然满足度限制并且结果更优
- 如此重复多次，以得到不能再通过这种方法得到更优解的极优解。

例题：小 H 的聚会 (NOI2005)

方法一：基于 Kruskal 的随机化算法

- 多次使用 Kruskal 算法后输出计算出的最优解

例题：小 H 的聚会 (NOI2005)

方法二：基于 Prim 的随机化算法

- 先用 Prim 算法计算出一个不考虑度限制的最大生成树
- 在这棵生成树上，每次随机选取一个不满足度限制的点，并试图寻找不在生成树上边替代在生成树内并且连接到这个点的边以降低这个点的度

例题：小 H 的聚会 (NOI2005)

方法二：基于 Prim 的随机化算法

- 多次重复直至将这棵树修改到满足度限制或者无法再用这样的方法降低这棵树不满足度限制的点的度的时候停止
- 重新在原最大生成树上重复上述操作多次，输出计算出的最优解

例题：小 H 的聚会 (NOI2005)

两种算法的比较

		算法一		算法二	
数据	参考答案	稳定输出[1]	得分	稳定输出	得分
7	64034	62712	7	63856	9
8	570138	565529	8	563403	8
9	1021302	1021385	10	1021461	10
10	1041394	1041604	10	1042087	10

例题：小 H 的聚会 (NOI2005)

通过上面例题可以看出：

使用随机化算法在开放性题目能够得到很好的结果

使用随机化算法结合不同的算法会得到不同的结果

在解题时运用恰当的算法配合随机化算法能提高程序的准确性

总结

从上述的题目中可以看出，随机化算法具有灵活多变的特点，应用随机化算法应该注意几点：

- 针对不同情况灵活运用随机化算法，充分发挥其灵活多变的特点
- 注意应用随机化算法与恰当的算法结合以激发更大的活力
- 注意随机决策和调整的效率，充分利用题目所给出的限制时间

总结

最后，让我们展开创造性思维的翅膀，在信息学的广阔蓝天上自由翱翔，划出一道亮丽的风景


谢谢