信息学竞赛中构造题的常用解题方法

西南大学附属中学校 蒋凌宇

摘 要

构造题是近年的算法竞赛中常见的一种题目类型。本文对构造题中常见的几个解题方法进行了介绍，包括抽屉原理的运用、DFS 树的运用、递归法的运用等，并给出了例题和讲解。

1 引言

在近年的算法竞赛中，构造题的出现越来越频繁。不同于传统的计数、最优化等问题，构造题只要求选手给出一组满足约束条件的解，而不需要统计解的数量，或是寻找一组“最优”的解。然而，由于其模型繁多，涉及图论、数论、字符串等各领域，且常常难以发现，要解决起来并不容易。

本文对构造题中较常出现的一些解题思路进行了介绍，并给出了例题和讲解，希望对读者有所启发，在解决构造题时能更加得心应手。

2 抽屉原理

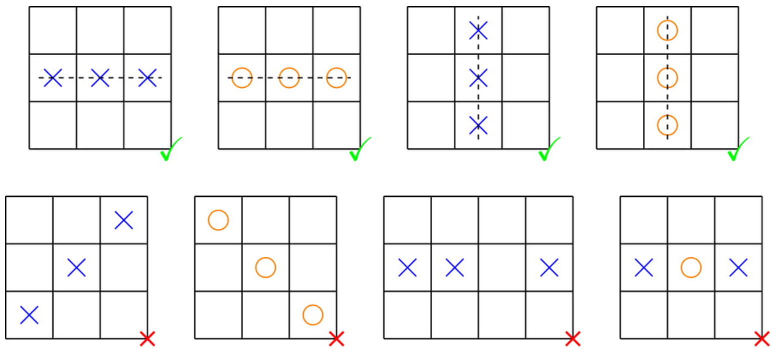
抽屉原理，或称为鸽巢原理，是组合数学中一个非常重要的原理。通常的表述是，若将 n 件物品放入 k 个抽屉，则其中一定有一个抽屉包含至少 $\left\lceil \frac{n}{k} \right\rceil$ 件物品，也一定有一个抽屉包含至多 $\left\lfloor \frac{n}{k} \right\rfloor$ 件物品。

在一些构造题中，常常会要求构造一个权值至少为（或不超过）某一个数的方案。很多时候，可以考虑找出若干个可行的方案，使得它们的权值之和是定值。假设找出了 k 个可行方案，其总权值和为 n，由抽屉原理，这些方案中最小的权值一定不超过 $\left\lfloor \frac{n}{k} \right\rfloor$，最大的权值至少为 $\left\lceil \frac{n}{k} \right\rceil$。

2.1 Errich-Tac-Toe[1]

2.1.1 题目大意

给定一张 n 行 n 列的棋盘，每个格子可能是空的或包含一个标志，标志有 X 和 O 两种。如果有三个相同的标志排列在一行或一列上的三个连续的位置，则称这个棋盘是一个**胜局**，否则称其为**平局**。



例如，上图第一行的局面都是胜局，而第二行的局面都是平局。

在一次操作中，你可以将一个 X 改成 O，或将一个 O 改成 X。

设棋盘中标志的总数为 k，你需要用不超过 $\left\lfloor \frac{k}{3} \right\rfloor$ 次操作把给定的局面变成平局。

2.1.2 数据范围

$1 \le n \le 300$。

2.1.3 解题过程

不妨将行列都用 $0, 1, \ldots, n-1$ 编号，将第 r 行第 c 列的格子记为 (r, c)。

我们将所有格子分成 3 类，其中第 i $(0 \le i < 3)$ 类包含所有满足 $r + c \equiv i \mod 3$ 的格子 (r, c)。不难发现，在一行或一列上的连续三个格子包含第 $0, 1, 2$ 类格子各一个。

由此，不难想到以下的几种操作方案：

- 将第 0 类格子上的 X 都改成 O，将第 1 类格子上的 O 都改成 X。
- 将第 1 类格子上的 X 都改成 O，将第 2 类格子上的 O 都改成 X。
- 将第 2 类格子上的 X 都改成 O，将第 0 类格子上的 O 都改成 X。

[1] https://codeforces.com/contest/1450/problem/C2

显然这三种操作方案都能使得局面变成平局，而它们的操作次数的总和恰好是棋盘中标志的总数 k，因此其中操作次数最少的方案的操作次数一定不超过 $\left\lfloor \frac{k}{3} \right\rfloor$。

2.2 Mine Sweeper II[2]

2.2.1 题目大意

扫雷地图是一张 n 行 m 列的网格，其中每个格子是地雷或空地。每个空地会显示一个数字代表与它相邻的雷的数量（两个格子相邻当且仅当它们共用一个顶点或一条边，不在边界上的格子与恰好 8 个格子相邻）。

在一次操作中，你可以将一个地雷改成空地，或将空地改成地雷。

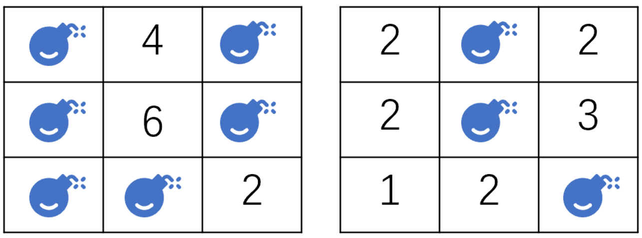
给定两张扫雷地图 A，B，你需要对 A 进行不超过 $\left\lfloor \frac{nm}{2} \right\rfloor$ 次操作，使得 A 所有空地上的数字之和等于 B 所有空地上的数字之和。

2.2.2 数据范围

$1 \leq n, m \leq 1000$。

2.2.3 解题过程

注意到一张地图所有空地上的数字之和等于相邻的（地雷，空地）的对数。这就意味着，如果将一张地图的所有地雷改成空地，所有空地改成地雷，其所有空地上的数字和不变。如下图所示。



由此，显然有以下两种方案：

- 将 A 改成 B。
- 将 A 改成与 B 恰好相反，即若 B 的某个格子是地雷，则 A 对应的格子是空地，反之亦然。

[2]The 2020 ICPC Asia Shanghai Regional Contest, Problem B, https://codeforces.com/gym/102900/problem/B

由于每个格子只会在恰好一种方案中被修改，这两种方案的操作次数之和应为 nm。因此取其中较少的一种，操作次数不超过 $\left\lfloor \frac{nm}{2} \right\rfloor$。

3 DFS 树

在解决一些图上的构造问题时，DFS 树往往有非常大的帮助。

一张图的 **DFS 树**是在对其进行深度优先遍历时，所形成的树结构。建立了 DFS 树后，图上的边可以分成四类：

- **树边**即每个点到其所有孩子结点的边，也即每个点第一次被访问时经过的边。
- **前向边**是每个点到其后代的边，不包括树边。
- **后向边**是每个点到其祖先的边。
- 其余边称为**横叉边**。

其中，前向边、后向边、横叉边统称为**非树边**。

在构造题中，通常我们用到的是无向图的 DFS 树。如果我们将每条边按照第一次经过时的方向进行定向，则无向图的 DFS 树满足所有非树边都是后向边。这个性质在解题过程中有非常大的作用。

3.1 Ehab's Last Corollary[3]

3.1.1 题目大意

给定一张 n 个点 m 条边的无向连通图，以及一个整数 k，你需要

- 找到一个**恰好** $\left\lceil \frac{k}{2} \right\rceil$ 个点的独立集，
- 或者找到一个长度**不超过** k 的**简单**环。

3.1.2 数据范围

$3 \le k \le n \le 10^5, n-1 \le m \le 2 \cdot 10^5$。

[3]https://codeforces.com/contest/1364/problem/D

3.1.3 解题过程

记 $l = \lceil \frac{k}{2} \rceil$。

建出图的 DFS 树，考虑每条非树边 (u, v) (正如上文所说，它一定是后向边)，如果 $|\mathrm{dep}_u - \mathrm{dep}_v| < k$，则取 (u, v) 加上 v 到 u 的树上路径即为一个长度不超过 k 的简单环。

否则，考虑两种情况：

- 若 $m = n - 1$，即图是一棵树。把所有点按照深度的奇偶性分成两个集合，取其中较大的一个集合，即为大小至少为 $\lceil \frac{n}{2} \rceil \geq l$ 的独立集，取其中任意 l 个点即为所求。
- 若 $m > n - 1$，这时 DFS 树上存在非树边，但不满足 $|\mathrm{dep}_u - \mathrm{dep}_v| < k$，意味着 DFS 树的深度至少为 k，且任意一对深度差在 $[2, k)$ 中的点都不存在边相连。设深度最大的点为 x，取 x 以及 x 的 $2, 4, \ldots, 2l - 2$ 级祖先，即为所求的独立集。

至此，我们仅用一次 DFS，在 $O(n + m)$ 的时间内解决了此题。

3.2 景点划分[4]

3.2.1 题目大意

给定一张 n 个点 m 条边的无向连通图，以及三个整数 a, b, c，满足 $a + b + c = n$。你需要将 n 个顶点分成三个集合 A, B, C，大小分别为 a, b, c，使得其中至少两个集合是连通的 (集合中的任意两个点能只经过该集合内的点互相到达)。有可能无解。

3.2.2 数据范围

$3 \leq n \leq 10^5, 2 \leq m \leq 2 \cdot 10^5$。

3.2.3 解题过程

不妨设 $a \leq b \leq c$，则只需要集合 A, B 连通即可。假设是 A, C 连通，我们可以通过将 C 中的一些顶点加入 B 中，使得 C 仍然连通且大小变成 b，因此仍然是合法的解。B, C 连通的情况同理。

这时我们遇到了一些困难，因为有可能无解，而题目中并未给出、我们也并未发现有解的条件，非常难以下手。因此我们不妨先来考察图是一棵树的情况。

当图是一棵树时，集合 A, B 都是其中的子树，因此一定存在一条边，使得 A, B 处于边的两侧。显然，我们只需要找到一条边，使得其两侧的较小和较大的子树大小分别不小于 a,

[4] IOI2019 第一试第二题, https://loj.ac/p/3176

b 即可。注意到一条边两侧较大的子树一定包含重心，我们可以考虑对重心进行一些分析。如果删去重心后最大的连通块大小小于 a，则显然无解。否则，设这个连通块的大小为 x，由重心的性质显然有 $x \le n/2$，因此删去这棵子树后还剩 $n - x \ge n/2$ 个点。又由于 $b \le n/2$ (因为 $b \le c$)，因此这棵子树与重心之间的边就是我们要找的边。

回到一般的情况，我们建立图的 DFS 树。找到 DFS 树的重心，设为 u，记 u 上方的子树为 T，u 下方的子树为 $S_1, S_2, \ldots S_k$。考虑几种情况：

- 如果 T 或某个 S_i 的大小不小于 a，则我们可以用和树一样的方法构造一组解。
- 如果 T 和所有 S_i 的大小都小于 a，我们就需要考虑无向图 DFS 树的性质。不同的 S_i 之间是没有边相连的，同时有一些 S_i 与 T 相连。如果所有与 T 相连的 S_i 加上 T 的大小之和小于 a，则一定是无解的，因为这表示集合 A, B 都必须包含重心 u。从 T 开始，我们依次加入与 T 相连的 S_i，直到其大小不小于 a。设得到的点集为 X，则 X 是连通的，我们可以在其中选出 A。同时由于 T 和所有 S_i 的大小之和都小于 a，X 的大小不超过 $2a$。而 $2a + b \le a + b + c = n$，因此我们在删除 X 之后，剩余的点数至少为 $n - 2a \ge b$，我们可以在其中选出集合 B。

至此，我们在 $O(n + m)$ 的时间内完成了构造。

4 递归法

在一些构造题中，对于不同的输入，问题的结构有很大的相似性。在很多时候，这往往意味着我们的构造也具有很大的相似性，或是具有周期性。这时，我们往往可以通过递归的方式，对子问题进行构造，并在子问题的构造的基础上进行一些小的调整，来得到原问题的构造。

需要指出的是，递归可以作为一种思想，但在实际解题过程中可能有代码、时空复杂度高的缺点，需要选手灵活运用。

4.1 Baggage[5]

4.1.1 题目大意

有 $2n$ 个包裹，其中有 n 个 A 类包裹，和 n 个 B 类包裹，初始时它们的排列如下：

$$\text{B A B A B A} \ldots \text{B A}$$

[5] 2014 ACM-ICPC World Finals, Problem A, https://codeforces.com/gym/101221/problem/A

这些包裹占据了编号为 1 到 $2n$ 的格子，同时还有编号为 $-2n+1$ 到 0 的 $2n$ 个空格子可供使用。现在要将这些包裹重新排列，使得它们形如

$$\mathrm{A\ A\ \ldots\ A\ B\ \ldots\ B\ B}$$

即，这些包裹占据了相邻的 $2n$ 个格子 (不一定是 1 到 $2n$)，且所有的 A 类包裹在所有的 B 类包裹的左边。

排列过程由若干次操作组成，在每一次操作中，可以选择相邻的两个包裹 (不能只选择一个)，并将它们移动至某两个相邻的空格中。

给定 n，找到一个最短的操作序列。

4.1.2 数据范围

$3 \le n \le 100$。

4.1.3 解题过程

与通常的构造题不同，本题要求的是最短的操作序列，看起来难以下手。但经过一些尝试，或是对 n 较小的情况进行搜索，会发现它们的最短操作序列的长度都是 n。

事实上，证明操作次数不少于 n 是容易的：考虑有多少对相邻的包裹的类型相同，设这个个数为 d。初始时 $d=0$，而在结束局面中 $d=2n-2$。在一次操作过程中，取出包裹时不会 d 不会增加，而在放回包裹时，假设放的位置是 t 和 $t+1$，则只可能增加 $(t-1,t)$ 和 $(t+1,t+2)$ 这两对相邻的包裹。同时，容易发现第一次操作至多使得 d 增加 1，因此总的操作次数不少于 $1+\lceil(2n-3)/2\rceil=n$。

接下来，我们就要尝试对所有 n 构造长度为 n 的操作序列。对于 n 较小 ($n \le 7$) 的情况，我们可以直接利用搜索求出操作序列。经过观察发现，在 $n>3$ 的情形中，我们都是将这些包裹从编号为 1 到 $2n$ 的格子移至编号为 -1 到 $2n-2$ 的格子。

因此，我们可以定义函数 $\mathrm{solve}(n,x)$ 表示将包裹从编号为 $x+1$ 到 $x+2n$ 的格子 (这些包裹形如 $\mathrm{B\ A\ B\ A\ \ldots\ B\ A}$) 移至编号 $x-1$ 到 $x+2n-2$ 的格子，并排列成形如 $\mathrm{A\ A\ \ldots\ A\ B\ \ldots\ B\ B}$。

通过尝试，或是对 n 稍大一些的情形的观察，对于 $n \ge 8$ 的情况，我们可以构造出如下的操作序列 (其中 _ 表示空格子)：

$$\mathrm{\_\ \_\ B\ A\ B\ A\ B\ A\ B\ A\ \ldots\ B\ A\ B\ {\color{red}A\ B}\ A}$$

$$\mathrm{{\color{blue}A\ B}\ B\ A\ {\color{red}B\ A}\ B\ A\ B\ A\ \ldots\ B\ A\ B\ \_\ \_\ A}$$

$$\mathrm{A\ B\ B\ A\ {\color{red}\_\ \_\ B\ A\ B\ A\ \ldots\ B\ A}\ B\ {\color{blue}B\ A}\ A}$$

在这里，我们发现最后一行的红色部分正好符合 solve 函数的输入，因此我们调用 $\mathrm{solve}(n-4,x+4)$ 对其进行递归求解。

$$\text{A B B A A A A} \ldots \text{B B B \_ \_ B B A A}$$

$$\text{A \_ \_ A A A A} \ldots \text{B B B B B B B A A}$$

$$\text{A A A A A A A} \ldots \text{B B B B B B B \_ \_}$$

至此，我们便完成了长度为 n 的操作序列的构造，解决了此题。

4.2 Strange Housing[6]

4.2.1 题目大意

给定一张 n 个点 m 条边的无向图，你需要选择一个点集 S，满足：

- 一条边 (u, v) 是**开启**的当且仅当 $u \in S$ 或 $v \in S$，则任意一对点都能只经过开启的边互相到达。
- 不存在一条边 (u, v) 满足 $u \in S$ 且 $v \in S$。

有可能无解。

4.2.2 数据范围

$2 \leq n \leq 3 \cdot 10^5, 0 \leq m \leq 3 \cdot 10^5$。

4.2.3 解题过程

显然如果原图不连通则一定无解。

我们不妨猜测当原图连通时一定有解，考虑归纳证明。当 $n = 1$ 时，显然 $\varnothing$ 是合法的解。假设 $n = k - 1$ 时一定有解，考虑 $n = k$ 的情况。我们考虑删除一个非割点的点，容易发现这样的点是一定存在的。设这个点是 v，则 $G \setminus \{v\}$ 的点数为 $k - 1$，且是连通图，由归纳假设，我们可以为其找到一组解 S'。接下来，考虑两种情况：

- 若 G 中至少有一个与 v 相邻的点属于 S'，则令 $S = S'$ 即为一组合法的解。
- 否则，G 中与 v 相邻的所有点都不属于 S'，则令 $S = S' \cup \{v\}$ 即为一组合法的解。这是因为 G 是连通图，因此至少有一个点与 v 相邻。

至此，我们证明了有解当且仅当给定的图是连通图。然而，每次寻找一个非割点的复杂度太高，不能承受。事实上，观察归纳的过程，我们只需按照任意一个 DFS 序依次加入点即可。时间复杂度为 $O(n + m)$。

[6]https://codeforces.com/contest/1470/problem/D

5 总结

构造题的考察越来越频繁，但对许多选手来说，解决一道构造题并不容易。本文介绍了笔者在解题过程中总结的几个较常用的解题思路，希望能够让大家有所启发。

本文介绍的仅仅是构造题中的冰山一角，希望读者在训练过程中，也对构造题的解法进行归纳、总结，并分享给大家。同时也希望更多有趣的构造题能出现在算法竞赛中。

致谢

感谢中国计算机学会提供学习与交流的机会。

感谢国家集训队教练高闻远的指导与帮助。

感谢父母对我多年来的培养与关心。

感谢西南大学附属中学校潘玉斌教练的指导与帮助。

感谢给予我鼓励与帮助的老师以及同学。