参考`OI-wiki`上的总结：https://oi-wiki.org/basic/greedy/

贪心算法（greedy algorithm），是用计算机来模拟一个“贪心”的人做出决策的过程。这个人十分贪婪，每一步行动总是按某种指标选取最优的操作。而且他目光短浅，总是只看眼前，并不考虑以后可能造成的影响。

可想而知，并不是所有的时候贪心法都能获得最优解，所以一般使用贪心法的时候，都要确保自己能证明其正确性。

贪心思想

1. 相邻交换法：如果交换方案中任意两个元素/相邻的两个元素后，答案不会变得更好，那么可以推定目前的解已经是最优解了。
2. 最优解性质：对于最优性问题，一个好的思路是**逆向思维**，思考最优解满足何种性质（即最优解的必要条件，例如单调性、连续性、不交叉等等），由这些性质切入往往可以简化题目。
3. cut&paste：构造一个最优解，将当前贪心得到的这一部分解“复制粘贴”到最优解中，若最优解不会变差，则可证明贪心不会丢失最优解；换句话说，肯定有某些最优解包含当前贪心得到的这一部分解。例如MST、哈夫曼编码。
4. 最优性&可行性：若能同时说明某个解的**最优性&可行性**，则能够说明其就是最优解。
5. 归纳法：先算得出边界情况（例如 $n = 1$）的最优解 F_1，然后再证明：对于每个 n，F_{n+1} 都可以由 F_n 推导出结果。
6. 证明在任意局面下，做出局部最优决策之后，在可达解集中包含了做出其他任何决策后的可达解集（不会丢失任意解），即这个局部最优策略提供的可能性包含其他所有策略提供的可能性。
7. 在设计贪心算法时，**手玩样例**的能力往往至关重要。
8. 能贪心的问题必须要具有**最优子结构**，这一点和DP相同。所不同的是DP需要计算所有子问题的解然后进行比较、选择其中最优的，而贪心则是**按照某种准则直接进行决策**。这提示我们对于有决策的题目，都可以首先设计DP解法，发现复杂度过高且不能优化之后，尝试贪心解法。

常见模型

最优性&可行性

往往需要一定的观察力和经验。

【U86308】 01翻转(flip)

https://www.luogu.com.cn/problem/U86308

排序解法

最优解序列满足**按某种属性**有序这一特点，那么直接按这个属性排序后做即可。**排序准测**往往需要一定的推导。

一个好的方法是是逻辑运算拆 max/min

【P1080】 国王游戏

https://www.luogu.com.cn/blog/BeWild/p1080-guo-wang-you-hu

【CF1043E】 Train Hard, Win Easy

https://www.luogu.com.cn/blog/BeWild/cf1043e-train-hard-win-easy

参考：【相邻交换法】https://www.luogu.com.cn/blog/dengyaotriangle/xiang-lin-jiao-huan-fa

取当前最优

每次进行当前最优的决策。这个在大部分情况下是错误的，但是若**决策收益只会变差不会变好**，那么就是正确的，因为当前最优之后一直都是最优。

可以建模成一棵带点权的树，满足每个点权值小于父节点权值，取包含根节点的一个特定大小联通块、使其点权和最大；贪心策略为：每次取候选结点集合中的最大值即可。

经典的【 n 个班排队模型】本质也是每个班自身的单调性。

【U130967】冰红茶(tea)

经典的【冰红茶模型】

【P1315】 观光公交

https://www.luogu.com.cn/blog/BeWild/p1315-guan-guang-gong-jiao

【CF912D】 Fishes

https://www.luogu.com.cn/blog/BeWild/cf912d-fishes

双列匹配

给两个序列 $a[1...n]$ 和 $b[1...n]$ 进行两两匹配，使得某种指标最优，最基本的就是**火柴排队**模型（排序不等式）。匹配过程经常使用双指针来实现。

【P5021】 赛道修建

https://www.luogu.com.cn/blog/BeWild/p5021-sai-dao-xiu-jian

【P1084】 疫情控制

https://www.luogu.com.cn/blog/BeWild/p1084-yi-qing-kong-zhi

贪心+反悔

无论当前的选项是否最优都接受，然后进行比较，如果选择之后不是最优了，则反悔，舍弃掉这个选项；否则，正式接受。如此往复。

反悔特性往往由**优先队列（或者单调队列）**来支持。

【P3545】 [POI2012]HUR-Warehouse Store

此题可以假设已经卖掉的商品商家可以“反悔”，即强制顾客退货。这样就可以贪心，若当前顾客为 $b[i]$ ，能卖就卖，不能卖时检查是否满足过 $b[j] > b[i]$ 的顾客，退货然后进行1次替换。需要一个支持 `max()`，`insert()`，`pop_max()` 的数据结构，堆即可。

【P2209】[USACO13OPEN]Fuel Economy S

把油箱中的油想象成离散、一份一份的，不同价格的油不会混在一起。接着假设之前加的油，可以在之后的加油站“退货”（前提是不能消耗掉）。

这样就可以贪心：

1. 路程中需要消耗燃油时，先消耗低价油；
2. 到加油站 $c[i]$ 时，将 $c[j] > c[i]$ 的高价油退货，加满 $c[i]$ 油；

用一个单调递增的队列可以支持上述操作。

注意无解的判断：

1. 2个加油站间距 > 油箱容量 M ；
2. 第一个加油站距离 $x[1]$ > 初始油量 B ；

总复杂度线性。

【P1484】 种树

蓝书上有详细讲解。【0x17节】例题：数据备份。

【CF865D】 Buy Low Sell High

贪心反悔，不会损失决策所以是正确的。

【P4053】 [JSOI2007] 建筑抢修

【P3826】[NOI2017] 蔬菜

【P2949】 [USACO09OPEN]Work Scheduling G

https://www.luogu.com.cn/blog/BeWild/p3826-noi2017-shu-cai

上述三道题目有3种经典做法：

- 贪心反悔：按时间从前往后模拟，使用数据结构存放已经选择的元素集合，若当前元素选不了则进行反悔；利用决策包含性（不损失决策）证明正确性
- 时光倒流：按时间从后前模拟，使用数据结构存放当前能选择的元素，贪心选当前最优的元素

- 贪心覆盖：按收益从大到小决策，每次尽量往后放

注意这个问题具有时间上的最优子结构，前 t 天的最优解排序之后就可以得出前 $1\sim t-1$ 天的最优解。

时光倒流

有些有后效性的问题，如果倒着考虑就能够消除后效性。比较难的题目还需要使用数据结构维护候选集合和快速决策。

【P1954】[NOI2010] 航空管制

对于第一问：建反图倒着拓扑排序，每次弹 t 最大的结点（或者在 t 时刻再加入，那么可以弹任意一个结点）。

对于第二问：拓扑过程中让 i 号结点在堆中保留尽量长的时间，除非 `q.size()=1` 或者下一个弹不了否则不弹即可。

作业题解

【CF1132D】 Stressful Training

https://www.luogu.com.cn/blog/BeWild/cf1132d-stressful-training

注意本题利用了值域比较小从而可以去log的常见套路。

【CF1183G】 Candy Box (hard version)

见wucstdio的题解：https://www.luogu.com.cn/problem/solution/CF1183G

【CF1060D】 Social Circles

考虑什么样的人会坐在相邻位置，一定是一个人的左手与另一个人的右手相差较小时较好，因为这样子重复利用的椅子数量更多。那么我们可以由此获得贪心策略：对左手与右手分别排序，然后求$\sum\limits_{i=1}^{n}max(l_i,r_i)$。

【CF1151D】Stas and the Queue at the Buffet

看起来是相邻交换法，其实只用考虑第 j 个人的贡献：

- $a(j-1)+b(N-j)=j(a-b)+(bN-a)$

发现 $(bN-a)$ 这项和位置 j 无关，那么根据排序不等式，把 $a-b$ 大的放在 j 小的问题上即可。

【CF1148D】 Dirty Deeds Done Dirt Cheap

将所有整数对 (a,b) 分成2类：

1. $a[i]>b[i]$

2. $a[i] < b[i]$

观察到答案序列中仅包含某一类数对，那么分别考虑这两类整数对。基于所有数互不相同，有关键性质：

- 任意数量的同一类数对，都可以形成答案序列

例如考虑所有 $a[i] < b[i]$ 整数对，按 $a[i]$ 从小->大排序后依次相连即可满足条件。

这样只要统计哪类比较多即可。

【CF1117C】 Magic Ship

考虑二答案，当天数 t 固定时，由风造成总位移 (x, y) 也确定了。相当于先假设船不移动，那么 t 天之后会被风吹到 (x, y) 的位置，此时看和终点 $(x2, y2)$ 间的曼哈顿距离是否小于天数即可。

为了快速计算 (x, y)，需要预处理2个方向上的位移前缀和。因为要取模，从`0`计数比较方便。

另外1个需要注意的问题是二分范围，由于1个循环中船到终点的曼哈顿距离有可能只减少 2，所以`rbound`要设成`1e14`。

【CF1114E】 Arithmetic Progression

首先用`30`次询问二分求出数列末项 $a[n]$。

接着注意到等差数列的性质：

- 构造数组 $b[i] = a[n] - a[i]$
- 公差 $d = gcd(b[1], b[2], \ldots, b[n])$

接下来进行`30`次随机询问，每次求出 $b[i] = a[n] - a[i]$，和 d 取gcd，可以证明这样做的错误概率在`1e-9`量级。

证明 by xuanquang1999：https://codeforces.com/blog/entry/65136

【CF989D】 A Shade of Moonlight

https://www.luogu.com.cn/blog/BeWild/cf989d-a-shade-of-moonlight