愉快的复习【8.18】

很快就要期末考啦哈哈哈～

现在来复习总结一下整个暑假我们学了些什么叭～

贪心

贪心是什么？**局部最优解带来全局最优解**

对于一些题目，我们做一个**局部最优操作**之后，剩下的问题规模会变小

对于剩下的问题同样做**局部最优操作**，继续缩小问题的规模，直到可以一下子解决

通过一系列的局部最优来解题的思路，就叫贪心思想

不过有些题可以贪心，有些题贪心会 WA

证明一道题用贪心是对的比较难，不过证明一道题贪心会 WA 只需要找个反例即可

所以看到一道题要不要用贪心，先思考一下你的贪心思路存不存在反例

然后？就贪心试试看叭～

需要注意的是，如果这题贪心是对的，贪心做法会给出一个最优解法，而题目**可能会存在其它**最优解法，但是其它的最优解法的结果都**不会优于**贪心的结果

例题 and 思路：

https://www.luogu.com.cn/problem/P5019

铺设道路：我们的局部最优解为：从左数第一个需要填充的坑开始填，如果这时右边有连续的k个坑也需要填，则顺带把后面的坑也填上。

搜索

搜索本来是图的一种遍历方式：**根据联通关系，从一个点往另一个点搜索**

搜索有两种常见的方式：

深度优先搜索（DFS）：从点 A 搜索到点 B 时，即将起点改为点 B，根据和点 B 的联通关系进行下一个搜索

宽度优先搜索（BFS）：从点 A 开始搜索，与它联通的有点 B、C、D，那么先搜索完点 B、C、D，再将起点改为点 B（或点 C、D），根据新的联通关系进行下一个搜索

DFS 板子：

```
void dfs(int x) {
    for (each direction) {
        if (meet the conditions) dfs(next node)
    }
}
```

BFS 板子：

```
while (q.size()) {
    int top = q.front();
    q.pop()
    for (each direction) {
        if (meet the conditions) q.push(next node)
    }
}
```

例题 and 思路：

https://www.luogu.com.cn/problem/P1451

对于每一个联通块，我们采用两种遍历方式都可以完全遍历该联通块

directions：上下左右，conditions：数字1-9

模拟

什么是模拟？**直接按照题意编写**

模拟一般适用于签到题，或者一道题的某一个小部分

只要你感觉直接模拟不会超时，那就大胆的模拟吧～

例题 and 思路：

https://www.luogu.com.cn/problem/P1042

直接模拟的话时间复杂度为 $O(n)$，n 为字母个数（<=25*2500）

常规来看只要复杂度里面的数算出来不超过10^8，都不会超时滴

so 大胆的模拟～

排序

排序的话只需掌握 sort() 函数即可

主要是 cmp() 比较函数的编写

结构体比较、字符串非字典序比较、各种另类比较...

当然你也需要看的出这是道排序题

例题 and 思路：

https://www.luogu.com.cn/problem/P1093

题意已说明按什么方式排序，由于和多个分数和学号有关，这边建议采用结构体排序

暴力

暴力就是枚举所有可能的情况，然后找出要求的解

严格意义来说呢，这算个啥算法～

暴力有个明显的优势：永远不会 WA（但是可能会 TLE）

因此也需要预先算好时间复杂度，感觉不会 TLE 才能暴力哦～

例题 and 思路：

https://www.luogu.com.cn/problem/P3392

最原始的暴力：先枚举白色行、蓝色行的个数（剩下的就是红色行），之后枚举每一个位置，统计需要改变的格子数。这样是 $O(mn^3)$ 的，由于 $n, m \leq 50$，因此也不会超时～

虽然可以通过预处理优化成 $O(nm + n^3)$ 甚至 $O(nm + n^2)$ 但没必要～

回溯

回溯是配合 DFS 的一种技术/思想

在深搜的时候可能会改变某些状态

比如说搜索到了某个位置，把它的访问数组 visit[i] = 1;

那么回溯的思想为：在一步深搜结束后，把访问数组 visit[i] 重置

因此来确保这个方向的深搜过程改变的某些状态

在结束这个方向的深搜之后会**重置为原来的样子**

即不影响下个方向的深搜

回溯本身不难，不过要掌握什么时候要用回溯，什么时候不用

例题 and 思路：

https://www.luogu.com.cn/problem/P1605

求迷宫方案总数，因此需要确保某个方向搜索后不影响下个方向的搜索，因此需要回溯

递归/递推

这两个都是协助 DP 的工具

一个是从后往前，一个是从前往后

鲁迅说过：DP = 递推 + 记忆化搜索（递归的优化版本）

例题 and 思路：

https://www.luogu.com.cn/problem/P2437

观察知，编号为 n 的蜂巢可以从编号为 n-1 和编号为 n-2 的蜂巢爬过来，因此 a[n] = a[n-1]+a[n-2]。（其实就是个斐波那契数列），递归递推都很好写（需要注意的是这道题需要高精，所以当初没出这道题给你们）

DP

什么是 DP？~~动态规划~~ 状态转移

DP 怎么做？**将一个问题拆成几个子问题，分别求解这些子问题，即可推断出大问题的解**

因此 DP 题的做题步骤一般是：

- 定义子状态
- 寻找状态之间的转移方式
- 解决最小的子状态
- 从子状态通过转移方式逐步推导解出最终状态

例题 and 思路：

https://www.luogu.com.cn/problem/P1002

二维 DP：定义 dp[i][j] 为卒走到这个点的方案数，因为卒可以往右往下走，因此走到某个点的方案数只和走到它左边点的方案数和走到它上面点的方案数有关，因此状态转移方程为 dp[i][j] = dp[i-1][j]+dp[i][j-1]

https://www.luogu.com.cn/problem/P1359

一维 DP：定义 dp[i] 为来到出租站 i 的最小租金，可以看出 dp[i] = min(dp[1]+r[1][i],dp[2]+r[2][i],...,dp[i-1]+r[i-1][i])，同时这个也是状态转移方程

https://www.luogu.com.cn/problem/P1048

01 背包：背包问题的话首先找出每个物品的体积和价值，然后套 01 背包模版就可以啦

数论

数论的小知识点太多啦

各种位运算（&,|,^,<<,>>,~）：a&b <= min(a,b)，a|b >= max(a,b)，a^a = 0，1<<n = 2^n，~a = -a-1...

加法乘法原理还知道是啥嘛～

取余一般简单题很少见，可以先不管

排列组合怎么算还记得嘛：A(n,m) = n!/(n-m)!，C(n,m) = n!/(m!*(n-m)!)，C(n,0)+C(n,1)+...+C(n,n) = 2^n，还有组合数和杨辉三角的关系～

还有最大公约数最小公倍数一些零零散散的性质呀：a*b = gcd(a,b)*lcm(a,b)

还有素数的判断、筛法、因数的分解之类的～

数论也常常和贪心结合出题

比如说一种贪心算法的结果需要数论的知识去支撑计算

例题 and 思路：

https://www.luogu.com.cn/problem/P1372

贪心选法+数论计算结果～

树

树是一种特殊的图，有许多特殊的性质～（目前大多讨论有根树）

有根、有父子关系、n-1条边、无环、两点之间距离唯一

树的算法也是非常多～

最重要的是递归（三种遍历）、DFS、BFS（层次遍历）

例题 and 思路：

https://www.luogu.com.cn/problem/P1030

知道中序排列和另外任意一个，求另一个排列是树的基础知识哦

图

图是个什么东西呢？点和联通关系

需要掌握的知识点有：图的构建、搜索/遍历（包括联通性、有无环）、最短路（Dijkstra、Floyd）、最小生成树（Kruskal）、拓扑排序（看到算法笔试题有一道拓扑排序的～）

一般树和图的编程题做法都比较“模拟”，构建树/图，然后套各种算法板子就可以啦

例题 and 思路：

https://www.luogu.com.cn/problem/P5318

这道题就考了有向图的构建、搜索/遍历～

脑筋急转弯

脑筋急转弯是什么？一种看得出编程题考什么、突破口在哪的能力

可以发现的是，数据结构机考并没有考到什么硬算法（比如说最短路这些）

而更多的是从看上去复杂的题目描述中找出解题的关键

姐姐还记得第三题是个排队夹娃娃的（题目还提示你也许需要排序）

但其实就是道简单的贪心题

例题 and 思路：

https://www.luogu.com.cn/problem/CF1339A

题目写的那么复杂，实际上就是 cin >> n; cout << n << endl;

https://www.luogu.com.cn/problem/CF1355A

找规律题，计算到一定程度 a_n 就不再变啦

这两天的任务

有时候脑筋急转弯是很重要滴

既帮你看出题意，又帮你节省很多发呆的时间

当然这也是贪心算法的关键：找出合适的贪心思路

（可以提前透露的是，20号的测试也是以脑筋急转弯/贪心题居多，硬算法题较少

https://www.luogu.com.cn/problem/CF1368A

https://www.luogu.com.cn/problem/CF1370B

https://www.luogu.com.cn/problem/CF1365A

https://www.luogu.com.cn/problem/CF1369B