DSA OJ 生存指南


数据结构
30240184
CST数据结构（2024春）
邓俊辉
2024.01.01-2024.08.01
Next DDL: 2030.07.05

前言：

本篇指南是主要针对选**邓公**课的同学，由于不确定其他老师的 OJ 或者期末考是否跟邓公的一样，所以不保证此份文档能够帮助到选择其他老师的同学。另外，OJ 的题可能会有小部分变动，所以可能有些写在内文的题出现消失的情况，请见谅。

做好心理建设：

首先邓公的数据结构是贵系大一大二培养方案课程中最具难度的课程，4 个学分，任务量也算非常大，班上大卷怪巨多（秋季），但邓公给分一般还是很不错的。 邓公上课讲解非常细致，会深入探讨数据结构的接口、作用和执行流程，并配上生动地举例及自身的见解使同学加深对数据结构和算法的印象和理解。

然而，OJ 上的题大多数需要透过同学之间讨论或者上网查阅资料、代码，以及熟读邓公的 PPT 才能够写出，和上课教受的内容差异不小。 另外，期末考主要考察对于数据结构的算法和应用及其延伸，也考得非常细致，有些题目甚至要记忆 PPT 上的英文专有名词才读懂; 题型包含**是非题、计算题和论证题**等等。建议平时养成**专心上课并熟读 PPT 的习惯，并阅读邓公所编的数据结构习题解析**。 最近期末考仍有越来越难的趋势，甚至连送分题都没出现（班平均大概都**30 多分**），对于不会的理论证明题，仍要以自己理解的方式将**答案硬凑出甚至虾写也行，千万不要留白**。 这边只能说**上课、OJ 和期末考**是三个回事。

期末考主要考的内容：

首先包含是非题，考察包含对于任意数据结构操作的时间复杂度和空复杂度（之后简称时空复杂度）、操作步骤等等等.....，计算证明大题考察包含主定理，逆波兰中缀表达式，KMP 算法，B 树、红黑树或伸展树的操作，快速排序（轴点分析）、计数排序、桶排序、散列表，图论最小生成树、双连通分量，完全二叉堆、左式堆，AVL 树、KD 树等计算证明.....，即对于任意树的所有操作步骤和时间、空间复杂度都要记熟。 计算题常考二项分布或几何分布的概率或期望值计算，有时会需要手写代码，或者证明树的势能等等; 压轴题一般为算法构造题，需要构造符合题意和时空复杂度的算法解题，并证明该算法的正确性。

0J 生存指南：

虽然上面讲了一大堆期末考的东西，但期末考的内容太多太杂太难，不好把握，唯有**把握 0J 的分数比较容易在总评拿高分**。 这篇主要还是 0J 生存指南，主要提及哪些 0J 应该写，**哪些应该避开不要踩雷**，并提供解题思路跟各题难度评价，可供读者参考。

0J 题目分配：

刚开学会比较轻松，有个运用二维动态规划（之后简称 DP）来解题的 LAB0，但这个是不算分的，不用花太多时间在这上面，**但还是要好好做**，而越后面出现的题则是越来越难。 一般来说每四周会是一个循环，总共 4 次作业，每次作业可以写四周，而每次作业大概有 4 题，所以平均一周做一题是刚好的。 每次作业都是一回 PA 和一回 LAB，PA 是需要从多数题目中选出符合提交规定的题目数来撰写，LAB 则需要手动实验代码正确性或者编写数据生成器，并撰写实验报告。

4 次 LAB（LAB0~LAB3），其中 LAB0 不算分，每次一道题。4 次 PA（PA1~PA4），PA1 須選 2 道題，PA2~PA4 選 3 道題做。每道題占學期總成績 3%，總共 11（PA）+3（LAB）=14 道題，**佔學期總成績 42%**，期末考佔 58%。LAB 白盒測試佔 100%，PA 的黑盒測試佔 80%，白盒佔 20%，通常 REPORT 好好寫白盒都能拿到 100 分。

PA 有分 33%的题跟 26%的题，33%的题通常颇具难度，需要花上数小时撰写，甚至几天，而 26%的题相对简单，大概都是能在 1 小时内做出来的，**可以考量自身能力选择**。

注意事项 ：

1.可以和同學相互交流想法或者解题思路，但是不要直接交流代碼，**畢竟 DSA 課程代碼查重非常嚴格**。

2.写 33%难度的题可**适量使用九成测**，因为五成测的测试数据可能不足以验证代码的时空复杂度正确性，但九成测有次数限制，并且每次使用罚 1 分。

3.尽量平时就去思考 OJ 的题目，不要等到 DDL 前几天才做，毕竟不少题目是颇具难度的，若超过 DDL 后提交必然会被扣分。

4.DSA 几乎所有题目都**不能用 STL 库**，因此排序算法和需要用到的数据结构都需要自行手打。

5.每次 PA 的题目有分组（Group），每个组最多选一道题做就好。

6.33%的题若拿 80 分以上则可抵 26%的题，若拿不到 80 分，则提交 26%拿满的题即可。 而 33%的题每题会比 26%的题多拿学期总成绩 0.63 分，可和期末考一个是非题（1 分），占学期总成绩 0.58%做对比 XD。

7.本篇指南只转发给特定同学，请勿刻意流传。

提醒：

以下大部分 OJ 题会有根据 codeforce 大致难度的评分（主观评判），以及大概平均所要花费的时间（审题+写代码+除错的综合时间），有些题的顺序或者名称可能不一样，请见谅。 接着开始进入 OJ 题，作为参考的标准，大一秋季程设课中导弹拦截（n 到 1e5、最长共同子序列+dp+upper_bound）的难度在 codeforce 中为 **1500 分**左右。 **（大部分题目主要还是以让同学思考为准，不会给予太多的提示，更后面的部分会有每题稍微详细一点的作法）**

（感谢由 ：）学姊、小七学长、Christopher、H_Q_Y、Te 大佬们提供的建议和题解）

LAB0：（必做）

难度：800 分，平均时间约 2 小时

提示： 二维 DP、写白盒报告、除错

PA1:

1-1 Gift（建议做，**和下题选其中一题即可**）

难度： 1500 分，平均时间约 2 小时

提示： Set、折半枚举、二分搜索

1-2 Graphics（建议做，**和上题选其中一题即可**）

难度： 1400 分，平均时间约 2 小时

提示： 依斜率排序、二分搜索

1-3 Filename（建议做）

难度： 1600 分，平均时间约 3 小时

提示：LCS 最常共同子序列、DP

1-4 二維偏序（非常不建议做，难度太高）

难度： 2400+ 分，平均时间 4+小时

提示：CDQ 分治、树状树组 BIT

LAB1: Zuma (必做)

难度： 1200 分，平均时间约 8 小时

提示:

1. RE 2. RE 3. TLE （可以请大老们试试看能不能让他递归过深造成 RE）

4. WA 5. WA 6. WA 7. WA 8. WA 9. RE 10. WA

PA2:

2-1-1 Risk (建议做)

难度： 1600 分，平均时间约 3 小时

提示： 单调队列、Queue、Stack、前缀和

2-2 Polynomial (建议做)

难度： 1800 分，平均时间约 7 小时

提示: Stack、中綴表達式轉逆波蘭

2-4 Triangulation (建议做)

难度： 1300 分，平均时间约 3 小时 （没有题目叙述难度为 2400+）

提示: Stack、凸包

2-3 build (非常不建议做)

难度： 2000+分，平均时间 10+小时

提示: 多叉樹、後綴和、鏈表

2-6 scheduler（补 33%的题，可做）

难度： 1100 分，平均时间约 1 小时

提示： 链表数组

LAB2 Hashfun（必做）

难度： 1100 分 平均时间 7 小时

提示： 数据分析、多项式哈希、数据生成、线性试探、双向平方试探、公共溢出

PA3： （四次 PA 中最难的一次）

3-1-1 Match（建议做）

难度： 2400 分，平均时间约 15 小时

提示： treap、splay tree、hash、旋转操作、常数优化

3-2-2 Kidd（建议做）

难度： 1600 分，平均时间约 7 小时

提示： KD tree、归并排序、搜索优化

3-3-2 Nearest neighbor（建议做）

难度： 1900 分，平均时间约 8 小时

提示： KD tree、归并排序、搜索优化、常数优化

3-1-2 Feature（非常不建议做）**（这题用 splay tree 只会有 80 分）**

难度： 2500 分，平均时间约 16 小时

提示： 双 splay tree、权值线段树、常数优化

3-3-1 Temperature （不建议做）

难度： 2100 分，平均时间约 14 小时

提示： KD tree、搜索优化、常数优化

3-4 History（补 33%的题，可做）

难度： 900 分，平均时间约 30 分钟

提示: MAD 法、散列表

3-5 Huffman（补 33%的题，可做）

难度： 1100 分，平均时间约 1 小时

提示： Huffman tree、插入排序

LAB3: (必须做)

难度： 1600 分，平均时间约 12 小时

提示： splay tree、avl tree、数据生成

PA4:

4-2 Game（建议做）

难度： 1600 分，平均时间约 3 小时

提示： dijkstra 算法、二叉堆、关联矩阵、排序、DP

4-4 Component（建议做）

难度： 1700 分，平均时间约 4 小时

提示： 左式堆合并、并查集、最大堆

4-6 Sort（建议做）

难度： 2000 分，平均时间约 4 小时

提示： 四路归并

4-1 Melody（不建议做）

难度： 2000+分

提示：字符串后缀自动机

4-5 Chrompoly（非常不建议做）

难度： 2200+分

提示： 色数多项式计算、递归

4-7 Virus（补 33%的题，可做）

难度： 1200 分，平均时间约 1 小时

提示符： BFS、Queue

以下为题目较为详细的提示（主要是个人写的 report，请勿直接抄袭）

（有些题目可以上 Wiki io 参考概念还有代码，或者翻阅邓公 ppt 以及网站都有伪代码或者试例代码）

LAB0

关于二维动态规划的问题。 这题主要是教同学怎么写白盒报告，以及如何运用调适器帮代码 debug，~~由于不算分所以也不用太认真写~~，但还是尽量按照题目叙述完成比较好。

PA1:

1-1 Gift

n 到 40 的情况需要尝试折半枚举，本次运用递归方式求所有可能性。 分成两半后每部分约 1e6 的数据量将其中一半数据排序后，用另一半资料逐个 upper_bound 寻找有多少个可能性组合并将算出来的数值加总即为答案。

1-2 Graphics

输入所有 x 坐标和 y 坐标后对他们进行 mergesort，接着一一对应，即不会有线段交叉的情形发生。 接着对每次查询的坐标进行二分搜，若点在该线段的右边（带入该线段的方程式大于 0）表示该点和原点连成的线段能够和更多线段交叉，则往右搜，反之则往左搜，若该点已在某条线段上，则返回此线段位于第几条，而当左边界和右边界下标差值不大于 1 时做最后检查，返回该点和原点连线最终可和多少条线段相交。

1-3 Filename

在考虑 K 只有到 100 后，也就是修改距离最多不超过 100，可以发现只要注意字符串 A 的第 i 个字元左右检查 100 位就好了，因此复杂度可以降至 O（N*K），但仍然要注意边界条件、访问的字符串下标，以及每行数组向左为移的个数，最后使用滚动数组。

1-4 二维偏序

难度太高了，不建议非信竞的选手做，想挑战的同学可以上网或者 wiki io 查询 CDQ 分治，是个非常高端的算法技巧。

(由 H_Q_Y 大師提供的方法):

由于题目结果仅与a,b的相对大小有关，故对$\{b_i\}$进行离散化，使$\{b_i\}$的取值落在区间$[1,n]$中，并保持原来的大小关系. 再以a_i为关键字，将三元组列$\{(a_i,b_i,c_i)\}$从小到大排序. 得到新三元组列(x_i,y_i,v_i).

设$f(i)$表示从$[1,i]$中取若干个下标$1\leqslant j_1<\cdots<j_{k-1}<j_k=i$，使得$a_{j_1}\leqslant\cdots\leqslant a_{j_k},b_{j_1}\leqslant\cdots\leqslant b_{j_k}$时$c_{j_1}+\cdots+c_{j_k}$的最大值. 考虑状态转移，$f(i)=\max\limits_{1\leqslant l<i,b_l\leqslant b_i}f(l)+c_i$

使用树状数组维护数组$\{g(t)=\max\limits_{b_l=t}f(l)\}_{t\in[1,n]}$的前缀最大值，则每次状态转移前询问值域$[1,b_i]$中的前缀最大值，状态转移后更新单点处的可能最大值.

LAB1 Zuma

比较麻烦的 debug 题，有 10 份代码要 debug 并且生成测例需要花上不少时间。

这边只能跟同学说明是那些错误，剩下的操作和生成测例需要各位自行探索。

PA2:

2-1-1 Risk

基本想法是开一个结构体，其为包含 queue 跟 stack 功能的数据结构 qu，纪录天数下标，且下标对应的人数为非严格递减。 操作时从左到右扫描人数数组，由于最多向前考察的天数下标是递增的，可以将 qu 前面没用的下标 pop，而后端则是将比当下人数小的下标丢弃，最后将当下人数下标入栈，并将当下最多人数纪录于前缀和数组中。 最后查询时直接将查询数值用前缀和数组相减即为答案。

2-2 Polynomial

可以參考鄧公講義去寫運算符比較函數

(由 H_Q_Y 大師提供的方法):

建立一个多项式类，然后用栈处理多项式的运算。 栈中的运算符优先级单调不降，遇到新的运算符时，考虑原栈顶的运算符与之比较，若新运算符优先级更高，则直接入栈，否则将原栈顶的运算符取出进行计算，再重复这个过程，这样就可以处理一个新加入的运算符。 若加入了一个整数或者 x，则无需进行计算，直接入栈即可。

2-4 Triangulation

看懂题目的输入操作，是按照多边形点的顺序绕一圈输入的，因此还需要找到 x 最小的点和 x 最大的点去进行严格分类和区分每个点是属于上链还是下链。

本题可以直接照着题目叙述中的提示运用单调栈去做，但要先对所有点进行分类，且之后要对 X 坐标进行排序。需要先自行运用数组构造维护 stack 的数据结构，将每个点排好序，并区分成上链或者下链，。 在执行操作时，从左到右扫一遍，若操作点和栈顶点在不同链上则直接对栈中所有点进行剖分，反之若操作点和栈顶点在同一链上则尝试将栈中前两个元素和操作点配对，直到不能形成三角形为止（三角形必须在多边形内部形成），再将操作点入栈（栈中形成优链）。

2-3 build

千万别写

(由 H_Q_Y 大師提供的方法):

使用双向链表存储一个节点的所有子节点，这样寻找特定的节点花费的时间是 cost 的常数倍。 维护每个节点的子树大小 sz，在删除的时候，该节点原父节点到根节点路径上的所有节点子树大小均减少被删除子树的大小； 在添加的时候，该节点新父节点到根节点路径上的所有节点子树大小均增加新增的子树的大小。

维护每个结点的高度 h，由于结点的高度等于其所有子节点高度最大值+1，故利用子节点序列中高度 h 的后缀最大值 suf，可以使 suf 的维护花费时间是 cost 的整数倍而 h 的维护花费时间是修改节点的深度。

2-6 scheduler (26%)

根据题目要求，我们会需要在队尾或队头进行插入，且在队中进行删除，那可以建立一个类似链表的结构体数组，每个编号（下标）的节点存前节点和后节点的编号，若没有前节点和后节点则为-1。 另外用 fr 和 ba 分别存当下队列的头节点编号和尾节点编号，并根据题叙对四种字符串分别进行判断和操作即可。 需要注意的是有时候前节点或者后节点为-1，或者需要修改为-1，则需要特别做处理。

LAB2 Hashfun

用乘积哈希和多项式哈希去跑公共溢出、线性试探跟双向平方试探

需要用到 OOP 策略模式概念

PA3:

3-1-1 Match

(由 Te 大佬提供解答)

是一个不断修改字串和判断相同的问题,我们以每一个字符作为一个 Node,建立 splaytree,每一个 Node 代表着该子树的中序通历的字串,实现 splaytree 的插入与删除,我们满足了第一第二个操作.对于 flip 操作,我们先建立哨兵 a 在两端,然后在 flip 时将被 “flip 区间向外走一个位置的节点分别 splay 上顶和次顶,那么就有一棵子树就是被 splay 子树.我们使用 bool lazy_flip 来懒惰标记 flip 操作,在深入遍历时再 exec_flip。

对于制等操作,我们对每棵子树维护一个 hash， 该 hash 是将字串视为 26 进制并取模的结果,考虑到会 flip,我们同时维护逆序的 hash,判等是类似地 splay 出相应子树,然后判 hash。

3-2-2 Kidd

本题明显可用线段树去维护区间总和。 由于坐标可以大到 2^31，因此可对所有点坐标离散化，归并排序后记录于 po 数组中，并在两实点之间插入虚节点，代表着一段区间，否则实节点直接表示一个点。 Build 树时维护好每个线段树节点的实际左界跟右界（线段树左子树下标即为 2*cid，右子树为 2*cid+1）； 修改时维护区间和，将覆盖到的区间之区间和加上当前区间的长度，并维护懒惰标记; 若覆盖不完全则将之下推到左子树跟右子树，并继续递归修改。 查询时也使用递归，将覆盖的区间和加总并回传，并在有标记之处也执行下推。

3-3-2 Nearest Neighbor

本题可维护多维的 KDtree 并在每个节点存入中间点的位置，每次用归并排序查找该维度中间节点的位置，若左界和右界的下标差距在 22 以内则停止递归（递归到底可能影响之后查询效率），并记录叶节点的左界和右界，每个节点维护每个维度坐标的最大值和最小值。查询时先和该节点所存的坐标更新距离最小值，根据和子节点超立方体的最近距离（判断函数），判断优先往左子树或右子树递归查询，并更新答案，若另一颗树最近距离比答案小，则继续查询更新答案。

（可参考 wiki io 上 kd tree 寫法）

3-1-2 Feature （这题用 splay tree 只会有 80 分）

建立两棵 splay tree（**splay tree 可以参考邓公的讲义去写**），一棵维护没有加绝对值的权重，以及左右子树大小，另一棵维护有加绝对值的权重，并且维护左右子数大小及其线性组合的总和。 初始化时将节点按顺序插入两棵树; 修改权重时先将该节点转到根部后移除修改，之后再重新插入（两棵树都是）; 修该特

征值累积找到第 k 大权重将之转到根部后，直接修改特征值，最后查询总和时也是累积找到绝对值第 k 大权重，将停止的点转回根部，并将总和全数加总回传。

(由 H_Q_Y 大師提供的權值線段樹方法):

以$|w|$为值域，建立一棵权值线段树，线段树的节点维护区间$[l,r]$的信息，$size$表示一共有$size$个(w,x)二元对满足$|w| \in [l,r]$，$size_p$表示这些二元对中$w>0$的一共有$size_p$个，s表示这些二元对的特征值之和（即$\sum w_i x_i$），同时在叶子节点维护当前特征的x值. 进行操作1时，用数组维护第i个特征当前的w值，然后在线段树中删除对应的二元对的贡献，再增加二元对(w',x)的贡献. 进行操作2时，借助$size_p$和$size$的值对权重第k大的二元对进行定位（可以利用大小关系进行分类讨论：绝对值大的正数w >绝对值小的正数w >绝对值小的负数w >绝对值大的负数w），然后修改相应的x值，更新到根节点路径上的s值. 进行操作3时，借助$size$确定每个区间是否对结果产生贡献，递归计算绝对值前k大的特征.

3-3-1 Temperature

(由 H_Q_Y 大師提供的方法):

使用线段树套平衡树. 先对 x 坐标进行离散化，建立线段树. 然后线段树的每一个节点处均建立一棵以 y 坐标为关键字的平衡树. 在插入时，先以链表的形式存储线段树节点对应矩形块中的所有点，然后用二分法建立静态的平衡树（直接构建最优形态，后续无需维护平衡）. 在查询时，对 x 坐标在线段树中查询，在查询区间中的线段树节点再对 y 坐标进行平衡树查询,得出矩形中的最大值和最小值.

3-4 History (26%)

MAD 法映射盡量找質數+查找

3-5 Huffman (26%)

本题字串长度虽然达到 50 万，但只要根据每个字母出现的次数做排序，可以知道如何将所有字母合并成 huffman 树。 总共只有 26 种字母，因此使用选择排序的方式每次选出组内出现次数最少的两个字（最小的两个数），合并之后再放

回去和其他数字一同排序，并顺便由叶子开始往上维护 huffman 树，直到最后剩一个字母。 最后对整个二叉树做递归探索，将空数组传入，往左则补 0，右则补 1，到树叶则记录整个字串的内容。 最后对出现过的字母进行输出字符串即可。

LAB3 BBST

就把 Match 那题改一下（splay tree），再写 avl tree，**模板可以参考邓公的讲义去写**，测试样例其中一个用随机生成的，另一个不断查询相同的点（这样 splay tree 的时间会比较少（局部访问 splay tree 时间复杂度比较优秀） ），区分两种树。

PA4:

4-2 Game

本题需要用到 dijkstra 最短路径算法+动态规划。 首先自行创建 vector 类当作储存邻接矩阵的数据结构，接着构造二叉堆 priority_queue 方便之后进行 dijkstra 的优先级搜索，重载写出需要用到的成员函数即可。 接着对整张图进行 PFS，即 dijkstra 算法，迭代直到 pq 内的元素为空，更新初始节点到每个节点的最短距离，并将之存入 sd 表中。 对 sd 表中的距离进行排序并存好下标，接着直接进行动态规划，寻找是否有节点到它的后继加上两个之间的距离等于后继节点最短距离表（sd 表）中储存的数字，若等于则在 dp 表中加上原节点行走路线的次数（即 dp[j]+=dp[i]，其中 i 为前驱节点，j 是后继），最后输出 dp[n] 即可。

4-4 Component

本题可以运用并查集+左式堆完成。 在 u，v 之间连一条边，表示他们在相同集合里，并查集运用搜寻和合并函数维护每个点之间的联通关系，以及每个左式堆之间的合并，也维护好每个堆的节点数量不超过 k 的最小左式堆（大于 k 用函数

删掉），以及每个堆的老大是几号节点。 **左式堆可以参考邓公的讲义写**。

4-6 Sort

本题可用四路归并法，才能在规定比较次数内完成。 将数组不断切分，当长度小于等于 3 时做完排序后回传，若不是则将数组均匀拆分成四段继续递归。 都递归完后先进行四路合并，每次比较三个数，最大的数下次不用参与比较，最小的则将之存入 sav 数组并往后推进。 当剩下三个数组有数字时，就直接进行三个数字比较并依序归入 sav 数组；剩两个数组有数字，就传两个数字比较（剩下那个数字随机挑），并依序归入 sav 数组，剩一个数组有数字就直接归入 sav 数组。

4-1 Melody

待补充

4-5 Chromploy

待补充

4-7 Virus （26%）

本题可接运用图论的 BFS 解决。 先自行构造一个队列储存坐标跟时间，将病毒的初始坐标都放入队列，接着直接对队列进行迭代，直到队列为空；跑到邻格子（上下左右）的前提是那格还没有被拜访过，visit 数组存的是到那格的最少时间，若 visit[i][j]=0 表示当格本来就是病毒或者还没拜访过。 最后将非病毒每个格子答案全部加起来即可。