YDSP Senior 官方详解

快速对答案：BBDAD DCACA BDBDC, FTTBBA, TTFTBAC, TTFBBC, BCDBC, ACDBAC

1. 单项选择题

2. 考纲中写的是“模意义下的逆元”，所以加法逆元能考。注意到模 7 意义下，减去 4 相当于加上 3。
3. A 拼写错误；B 中程序不能用自己的头文件；C 代码不能超过 100KB；D 只不过复杂度高了些，可能有部分分。题干说有程序拿分，所以选 D。

5. ST 表要求同一个数多次参与运算，结果不变，而异或不满足这个性质。
6. C++ 中，0 开头的数表示八进制，而 101 是十进制。
7. 不难想到，如果让 5 个点形成完全图，最后一个点先往五个点各连一条出边（此时有 25 条边）是最坏情况，再连一条边就会强联通。

9. 考虑 Dijkstra 的过程，我们说优先队列优化 Dijkstra 是 $O(|E|\log|E|)$ 的，那是因为最坏情况下，每一条边都会更新一次优先队列。但是在这里，我们发现每次更新队列，至少都会让所有距离和减少 1，而所有距离和只有 $O(n^3)$，因此入优先队列不超过 $O(n^3)$ 次，所以最坏时间复杂度为 $O(n^4)$。
10. 这个算法虽然使用了位运算，但是每次变化 x,y 都只往上跳 1 级祖先，所以还是暴力。

12. 把叶子变成 3 孩子结点，叶子增加 2 个；变成 4 孩子结点，叶子增加 3 个。因此替换 1000 次一共增加了 2022 个叶子，设 3 孩子结点有 x 个，则 $2x+3(1000-x)=2022$。
13. 不难发现，`cnt++` 当且仅当找到一组 (i,j,k,l)，且我们有不等式 $1\le k<j<i<i+l\le 100$。所以本题答案为 $\binom{100}{4}$（或写作 C_{100}^4）。
14. 把矩阵看作邻接矩阵，则 $A_{i,j}^{2023}\neq 0$ 表示存在有 2023 条边的 $i\to j$ 的路径。$1,4,5,6$ 互通，$2,3$ 互通但是只能 $2\to 3,3\to 2$。
15. 如果会 Zeller 公式就是好的，没背过建议最后再做。哪怕浙江你都不差这两分。

2. 阅读程序

第 1 题

本程序要解决的问题是，你有一个算式 $a_1+a_2+\ldots+a_n$，你可以把三个 $+$ 改成 $\bmod$，求得数的最小值。

`res[i]` 记录的是连续 i 个 $\bmod$ 放在一起可以让答案减少多少。

1. 你可能觉得，$a<2,b<2,c<2$ 的情况不可能在排序时排到前面，然而 n 够小时，这种非法情形也可能被排进前八。
2. 在所有合法情形中，`a[0]` 完全没被用到，爱等多少等多少。
3. 是的，在求 `res` 数组是已经错误。
4. $7\bmod 2023\bmod 4\bmod 2=1$。
5. 最优答案是 $10+20\bmod 15\bmod 20+15\bmod 20$，所以是块 2。
6. 对于一个 j，其让四个 i 非法，所以 i 需要枚举前五大才能保证有合法解；j 同理。

第 2 题

读入程序后，先讨论 tar 是否大于 2 倍所有数字和，如果不大于则在 a 复制三次后的序列里用双指针找区间和为 tar 的子串；否则枚举 l，去掉前缀后对数列和求余并查找后缀。

所以这个程序解决的问题就是 a 复制无限次后能否找到和 tar 的子串；全程在用双指针，所以时间复杂度是线性的。

1. a_i 本身确实没有溢出，且很快我们就只用 `pre` 和 `suf` 数组了，所以没有问题。
2. 因为前 $n/3$（原来的 n）个数前缀和已经求好了，所以没有关系。
3. 不可以，求余优先级更高。

第 3 题

本程序要解决的问题是，给出 s, m，求 $[0, s]$ 内有多少个整数，各个数位和是 m 的倍数。

1. 哪怕你没看出这题要干什么，也可以注意到 $f(n, 0)$ 总是正数——永远从 0 转移即可。如果看出这题要干什么，那么你知道 0 是任何数的倍数。
2. `dval` 最大 10^{18}，而 `10*dval` 最大 $10^{19} < 2^{64}$，没有溢出；然后两个代码都表示取这一位的数码，功能也相同。
3. 虽然这里 f_k 总是由 f_{k-1} 得到，但是后面程序要用到 f。
4. 没啥技巧的模拟。
5. 数位和是 9 的倍数当且仅当本身是 9 的倍数，所以直接写上 $\lfloor s/9 \rfloor + 1$ 即可。
6. 大意就是如何 handle 负数求余。A 中 `m<<20` 应加上括号；B 有得到 m 的风险；D 中不能保证 $i < m$，所以可能需要扣除不止一个 m。

3. 完善程序

第 1 题

1. 根据第 $32 \sim 35$ 行，`pure` 用于数没有问号的字符串方案数。已经排序好后，`cnt` 记录了所有相等的 p 的个数，接下来要算这些相等 p 的贡献，选 B。
2. 根据 `s[i]` 是 char 类型，或者根据 `pure` 中第 13 行需要把 `s[i]` 左移一遍，知道这里不应左移，只需要重新编号。
3. 这里需要这个问号遍历空格和二十六个字母。字母已经在第 2 空重新编号为 $0 \sim 25$，而 `pure` 第 13 行告诉我们空格是 -1，所以你应该填 `i-2`。
4. 这里要对子串去重。对于不跨过问号的子串，我们重复统计了 27 次，需要扣除 `cf` 次。在定义 `cf` 时已经记录了问号左边的子串数量，下一步是把问号右边平移到开头再调用一次 `pure`。右边有 `n-idx` 个字符，字符的偏移量是 `idx`。
5. 同上。

第 2 题

长代码考虑分块，先理解整体功能，后考虑具体实现。

- build 将 In_l 到 In_r 内元素建立表达式树，返回根，左孩子和右孩子分别记录在 L 和 R 数组内。
- calc 函数是给出表达式树，算出结果。
- main 读入字符串后，判定格式是否正确，对每个左括号找到其右括号记在 `brac` 内。

1. 括号是为了表达优先级的，这里它的使命结束了，直接左右各丢掉一个字符即可。
2. 情况开始复杂了。第 $28 \sim 41$ 行的目的是找到最后一步算谁，用 x 记录。首先括号肯定不是最后一步；然后有加减法的记录在 `rt`，有乘除法记录在 `ls`，有乘方的，从第 41 行推，应该是 `rs`。最左边的乘方是最

后算的，所以要 `!rs`。这么看来，其实 `rs` 记录当前的 k 即可。

3. 现在 $x = 0$，也就是一个运算符都没有检测到，是纯数字。根据 `calc` 里面第 54 行，出现 `?` 表示有数字，调用的是 `val`，所以这里也要读取 `val[x]`。
4. 什么时候求余要返回错误？当除数为 0。
5. 现在 `In[i-1]` 是运算符，`In[i]` 是 `-`，那么按理后面要跟数字，所以如果发现后面跟的不是数字，就要报错。
6. 上面是检测括号匹配的，那么括号不匹配就要报错。下溢报错前面已经出现过，所以只要栈里有东西报错就可以了。