绝区零 Zenless Zone Zero 模拟赛 题解

绝 (zenless.cpp)

贪心地考虑我们要扔掉什么数据包。问题的一个本质是，在空间很小的情况下我们不希望一个数据包占据通道太长的时间。所以容易想到先假设通道的容量是正无穷，每次操作尽量先选择加入时间较早的数据包，预处理出每个数据包在这种假设下离开通道的时间，然后在构造时每次贪心丢弃离开时间最靠后的数据包。这个算法的最优性证明略显繁琐，但是本质不难，留作练习。

本题存在许多不同可行的贪心算法，你同样可以倒序考虑操作序列，然后用线段树维护贪心选取留下的数据包的过程。同时也有一些不正确的贪心，大样例的强度一般，需要选手进行适当的对拍。

时间复杂度 $O(n\log n)$。

区 (zone.cpp)

对于包含绝对值的 $\max$ 问题，通常会将 $|x|$ 拆成 $\max(x,-x)$，从而想到将每个机器拆成 $f(i,j)+cj$ 和 $f(i,j)-cj$ 两个参数，可惜在本题中这个技巧行不通。主要会遇到的问题是，这样计算容易将机器和自己之间的代价当成 $2f(i,j)$ 而非 0，在一些数据上会得到错误的结果。

考虑更换建模使得更好的区分机器。可行的方案是采取图论建模。对于档位，我们将其看成一个数轴，相邻的数之间连边，边权为 c。对于机器，我们对其建立一个点，向其选择的档位连边，代价为对应的 f 值。一种合法的选择档位的方案可以看做这张图的一个生成树，而危险值对应了这张图的直径。

对于直径相关的问题，容易想到考虑生成树的直径中心。这个直径中心可能在数轴上，也可能在某个机器和数轴的连边上。可以进行分类讨论。对于后者，直接枚举连边，使用预处理后求出最小直径是容易的。对于前者，同样可以进行一些预处理后枚举直径中心在数轴上的位置进行计算。

时间复杂度 $O(nk)$。部分 $O(nk^2)$ 和 $O(nk\log n)$ 的实现也可以通过。

零 (zero.cpp)

首先考虑子任务 4。容易发现在其对应的特殊性质条件下，一定只会使用操作 1 和 4。考察最终结果串中最后一个字符 `A` 是在哪一步操作中被生成的。容易发现，它一定是先进行了一个操作 1，然后紧接着立刻进行一个操作 4 得到的。将这两个操作看成一组，将它们插入前 $b-1$ 个 `A` 的操作序列中的方案为 $2b-1$。如此迭代下去，~~方案数是桂花树~~，方案数是 $(2b-1)!!$，其中 $!!$ 代表双阶乘。~~不知道参加了去年 NOI 的选手有没有看出来这个，出题人很良心只给了很少的部分分。~~

对于子任务 5，其同样只会使用操作 1 和 4。相比子任务 4 的区别是，对于最终的字符串中的 b 个字符，只有 $b-a$ 个字符是新生成的。容易发现这 $b-a$ 个字符可以在最终结果中处于任何位置集合，容易得到答案是 $\binom{b}{a}(2b-2a-1)!!$。

对于子任务 6，在操作 1 和 4 的基础上，出现了可能删除字符 `A` 的 2 和 3 两种操作。考虑用一个组合意义去刻画子任务 4 和 5，来方便我们对做法进行修改。容易想到的一个建模是把操作 1 和 4 转成括号序列。更进一步的想法是把括号序列看成一棵树的 DFS 序，然后考虑那棵树。在确定树的形态之后，还要确定每个结点生成的括号在最终字符串中处于什么位置。它需要满足一个拓扑序。对所有树的拓扑序数量计数并求和可以得到与上文相同的结果。

在子任务 6 中，这棵树上的结点不一定在最终字符串中存在一个对应的字符：它可能在后续的过程中被删掉了。那么我们可以给每个结点定义两种状态：存在和删除。处于删除状态的结点，其子树中的所有节点一定也处于删除状态。一个结点被删除的时刻对应了树上的一条边，表示我们在进行 DFS 过程的时候，在离开那条边下的子树时删除了该节点。一个结点被删除的边的 DFS 序一定晚于其所有儿子。

考虑固定树的形态，计数选择若干个结点处于删除状态并钦定它们对应的边的方案数。一个结论是：这个方案数只和总节点数，以及处于删除状态的节点数有关，和树的形态以及确定的拓扑序无关。对于这点的证明，可以采用归纳。

于是我们解决了子任务 6。对于原问题，我们可以采用类似子任务 4 到子任务 5 的推广。假设初始串中的每个字符 `A` 都被保留到了最终串中，那么答案显然是 $\binom{b}{a}$ 乘上 $a = 0$ 的情形。若恰好有一个字符被删除，则对应到了 $b - a + 2$ 个字符，且有 1 个被删除的情形。对于有更多字符被删除的情况其实类似。

仔细推式子后方案数的形式非常简洁。进行上文所述枚举，在预处理组合数的情况下，容易计算答案。

时间复杂度 $O(n)$。