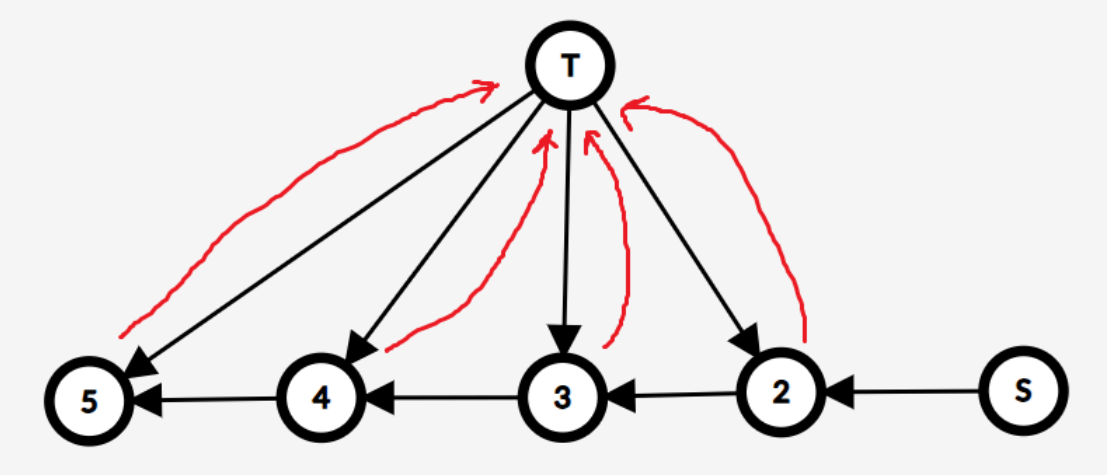
travel

如果只爬一次山, 必定是一段前缀使用抬高自己, 剩下的后缀的使用降低下一个, 使得后缀全部相同.

多次爬山时, 一定不会对这个后缀再降低, 否则会浪费前面使用的抬高操作. 所以枚举这个分界点, 然后用前缀和计算答案即可.

tree



枚举 T 的入边(红色), 每一条入边会贡献 2^k 到答案.

bracket

引理: $k+1$ 的答案必定是在 k 基础上再把一个位置变为 0. 证明通过反证法加调整即可.

首先考虑 $k=0$ 的答案: 使用堆优化贪心即可得到.

考虑 $k=1$ 的答案: 假设翻转了位置 i, 要么括号序列不变; 要么存在另一个 $j \neq i$, i,j 同时翻转. 此时)(始终可以变成 (), () 变成)(需要中间前缀和仍 ≥ 0.

考虑使用线段树维护修改收益. 注意到前缀和仍 ≥ 0 相当于中间没有出现过全局最小值. 每个节点维护修改(左右括号)(是否免费)(是否包含当前节点的全局最小值)(左右侧) 共 2^4 个信息, 同时维护 () ->)(,)(-> () 分别是否包含当前节点的全局最小值共 4 个信息. 最后和不翻转取 max.

注意合并的时候细节很多.