括号计数 (bracket)

本意是考枚举/搜索的，直接搜索每个字符的可能性并 $O(|S|)$ 检查，时间复杂度 $O(8^{|S|})$，得分：70pts（当然，已经确定的不要再搜/枚举了，要不然你就 40pts 了）。

加一些剪枝，得分：70~90pts（90pts 是给恰好 $O(ans)$ 的，也即 $O(\text{Catalan}(\frac{|S|}{2})4^{|S|})$）。

正解是给区间 dp 的，设 $f_{l,r}$ 表示区间 $[l,r]$ 是合法的方案数，转移如下：

$$f_{l,r}=\sum_{i=l+1}^{r}\text{count}(S_l,S_i)f_{l+1,i-1}\times f_{i+1,r}$$

$$f_{i+1,i}=1$$

其中 $\text{count}(p,q)$ 表示 p,q 有几种能够匹配的可能。

时间复杂度：$O(|S|^3)$，开 long long 即可，如果考虑高精乘法的复杂度为 $O(nm)$，那么复杂度为 $O(|S|^5)$。

不降序列 (sequence)

$l_i = r_i$ 的是送分的，爱写不写。

$-10 \le l_i \le r_i \le 10$ 的，可以写一个 dp：

> *设 $f_{i,j}$ 表示 $a_i = j$，以 i 为末尾的可能的最长不降连续子序列，转移可以用前缀 $\min$ 优化到 $O(nV)$，或者 $O(nV^2)$ 可能也能通过。*

考虑如何判断一个区间 $[p,q]$ 是否可行，从前往后确定 a_i，a_i 贪心选择最小的一个不小于 a_{i-1} 和 l_i 的，即 $a_i = \max(a_{i-1}, l_i)$，再 检查 $a_i \le r_i$ 即可，时间复杂度：$O(n^3)$，轻松优化到 $O(n^2)$。

然后发现这个东西可以双指针，再把 $[p,q]$ 的要求重写一下：

$$\forall p \le i \le j \le q, l_i \le r_j$$

- 考虑 p 加一：如果 $[p,q]$ 满足，那么 $[p+1,q]$ 也一定满足；
- 考虑 q 加一：判断 $r_q \ge \max_{i=p}^{q}\{l_i\}$ 即可，使用倍增/线段树询问区间 $\max$ 即可，时间复杂度：$O(n \log n)$。

但实际上可以优化到 $O(n)$，由于有单调性的限制，所以可以使用单调队列维护 $\max_{i=p}^{q}\{l_i\}$：如果 $x < y$ 且 $l_x \le l_y$，那么 l_x 就不需要记录了，维护一个单调递减的 l 序列，每次弹出最前面的直到开头的 $l \le r_q$，更新答案即可，时空复杂度：$O(n)$。

最优方案 (plan)

分析性质，发现所有边（对应到一条路径）一定互相不交（边不交），而且最后每条边一定被一条路径覆盖一次，比较抽象，有点难描述。

并且最后不可能出现 $u \to v, v \to w$ 的两条边，一定可以调整成 $u \to w$ 的一条边，显然更优。

那么一个子树，向上能够连向祖先的点，就只有恰好一个，那么就可以设计 dp，$f_{u,i}$ 表示 u 子树中，剩下 i 需要向上连，子树剩下部分的最优值，那么转移为（找到 u 的某个儿子 v 使得 i 在 v 的子树中）：

$$f_{u,i} = f_{v,i} + \sum_{w \in \text{son}(u) \setminus \{v\}} \max_{j \in \text{sub}(w)} \{f_{w,j} + a_u a_j\}$$

$$f_{u,u} = \sum_{w \in \text{son}(u)} \max_{j \in \text{sub}(w)} \{f_{w,j} + a_u a_j\}$$

$O(n^2)$ 轻松转移，进一步优化，需要首先掌握李超线段树和线段树合并。

对于一个 j，就可以看成一条直线 $a_j x + f_{w,j}$，那么每次去查询 f_w 的若干直线中，$x = a_u$ 处取值的最大值即可。

子树之间的影响只有对于所有直线整体加一个常数，然后再把 u 的所有子树的直线合并到 u 的线段树中，最后插入 u 代表的直线即可，直接使用李超线段树的合并即可。

时间复杂度：$O(n \log n)$，空间复杂度：$O(n)$，偏难的一道题，对数据结构的要求较高。

建筑游戏 (construct)

首先，容易发现这是一道 dp 题，设计状态 f_i 表示用 $1 \sim i$ 的木棍，最多获得多少积分。

然后只需要枚举区间 $[i+1,j]$，在 $[i+1,j]$ 中枚举三条边 a,b,c 计算面积 S，用 f_i+16S^2 更新 f_j 即可。时间复杂度：$O(n^5)$，常数非常小，可以通过 sub1。

考虑优化这个 dp，容易发现，选择三条边搭成三角形后，i,j 的枚举是无意义的，所以枚举 $[i+1,j]$ 后，强制要求 $i+1,j$ 都要选中，在 $(i+1,j)$ 中再枚举另一条边 k 即可。另外，需要增加 $f_{i+1} \leftarrow f_i$ 的转移。时间复杂度：$O(n^3)$，常数较小，可以通过 sub2,3。

接着，考虑三条边 $\sqrt{a},\sqrt{b},\sqrt{c}$ 搭成三角形的面积是 $2(ab+bc+ca)-(a^2+b^2+c^2)$，故当 a,b 确定时，将面积写成关于 c 的二次函数 $c^2+2(a+b)c-(a-b)^2$。所以，当 $|c-(a+b)|$ 越小，面积越大。

所以，优化上一个做法时，中间的 k 无需枚举，在固定 i，向右枚举 j 的时候用 set 维护出可能的 a_k，再用 lower_bound 求出 $a_{i+1}+a_j$ 的前驱后继计算贡献即可。时间复杂度：$O(n^2\log n)$。

这个做法可以使用并查集优化到 $O(n^2\alpha(n))$，但是与正解并无多少关系，此处不赘述。

接下来考虑正解，使用分治优化，用 $f_{l-1\sim mid-1}$ 更新 $f_{mid+1\sim r}$。

我们考虑两根木棍在 $[l,mid]$，一根木棍在 $[mid+1,r]$ 的情况，编号分别为 $i,j,k(l\le i<j\le mid<k\le r)$。

那么考虑对 i 找出最优的 j:

- 若 $a_j\le a_i$，那么一定选择最大的 a_j；
- 若 $a_j>a_i$，那么只需要取出一个 $>a_i$ 最小的 a_j 即可。

第一点是显然的，我们考虑第二点：根据之前的结论，我们应该选择 a_j 最接近 a_i+a_k 的 j，那么若存在 $a_x>a_j$ 且更接近 a_i+a_k，那么选择 (x,j,k) 会优于 (x,i,k)，所以并不需要考虑 (x,i,k)。

于是，对于每个 i，用 set 找到两种可能的 j。对于 k，和 a_i,a_j 有关的项可以写成一条直线 $x\times a_k+y$ 的形式，所以可以预处理凸包然后查询单点最大值即可，std 中使用了没有细节的李超线段树。

另外，一根在 $[l,mid]$，两根在 $[mid+1,r]$ 的情况同理，此处不赘述。

至此时间复杂度做到 $O(n\log n\log V)/O(n\log^2 n)$，空间复杂度为 $O(n)$。