A

考虑三个人 i,j,k 选定的时候，它们的顺序按 b 从大到小排肯定不劣（实际上只要不是最大的在中间肯定都不劣）。于是我们把 b 从大到小排，然后枚举中间的元素 j。维护前缀的最小 a_i+b_i，与后缀的最小 a_i，然后更新答案。

B

设图 G_i 表示原树只保留边权 $\geq i$ 的边之后所构成的森林。

$F(x,y)$ 等于 x 到 y 路径上的边权最小值 $\Rightarrow F(x,y)=\sum_{i=1}^{20}[x,y\text{ 在图 }G_i\text{ 中联通 }]$。

那么 $\sum F(x,y)=\sum_{i=1}^{20}\sum_{\text{连通块 }P\in G_i}\binom{|P|}{2}$。

我们枚举 i，然后对于需要断掉的一条边 (u,v)，看它的边权是否 $\geq i$，也就是断掉以后是否会对 G_i 产生影响。边权 $\geq i$ 则说明 u,v 在 G_i 的一个连通块内，断掉之后该连通块将被一分为二（v 子树内和子树外），否则将没有任何影响。同时我们只关心连通块大小，所以在枚举 i 之后容易做一遍或两遍 DFS 求出每条边新增的贡献。

时间复杂度 $\mathcal{O}(20n)$。由于 DFS 常数较大，建议预处理一遍 DFS/BFS 序，每次只要 `for` 循环就可以了。具体可以看代码。

C

对于 $Q=1$ 我们观察发现一个人的策略一定是不断地挪到另一个人的后面然后再跳到后面的一个数。设 f_i 表示目前的人在 i，另外的那个人在 $i+1$，之后目前的人能获得的最大分数，则有

$$\begin{aligned}f_i&=\max_{j>i+1}a_j-f_{j-1}-(s_{j-1}-s_i)\\&=s_i+\max_{j>i+1}a_j-f_{j-1}-s_{j-1}\end{aligned}$$

维护后缀最大值即可。

对于 $Q>1$。我们发现如果 $a_n>\sum a_i$ 那必然是先手直接去拿 x 然后后手拿剩下的。现在考虑 $a_n<\sum a_n$。

我们把这个 f 改的再好看一点。这个 f_i 和 f_{i-1} 中有很多部分很相似。进一步，我们有 $f_i=\max(f_{i+1}-a_{i+1},-f_{i+1}+a_{i+1})=|f_{i+1}-a_{i+1}|$。于是我们就可以舍掉 DP，直接从右往左扫一遍，初始值 $x=a_n$，从 $n-1$ 到 3 不断做 $x:=|x-a_i|$。最后再 $+a_1-a_2$。我们先不管这个 $+a_1-a_2$。

设一个 DP $g(i,j)$ 表示考虑 $[1,i]$，从 i 开始往前扫，并且目前初始值为 j 的最终结果，于是我们有 $g(i,j)=g(i+1,|a_i-j|)$。很显然，这个 DP 的形式大概就是把 DP 数组向右移 a_i 位，然后再把原先的 $g[1,a_i]$ 给翻转一下插到前面。可以用一个 deque 或者链表什么的维护整体 DP。复杂度 $O(n+Q+\sum a)$。

D

取1号点为根，将树上的所有边都定向成从儿子指向父亲的有向边。在逻辑上新建0号点以及一条从1号点指向0号点的边。这样由于树的性质，当某个点集只有一条指向点集外的边时，这个点集就构成一个子树。

到点标号的数轴上考虑问题，因为题目要求子树的标号是连续区间，问题也就变成了要将[1, n]这个区间切成k个子区间，满足每一个子区间都有且只有一条指向区间外边的方案数。

用 f[i][j] 表示当前位于 i 号点，已经划出了 j 个区间的方案数。可以写出一个比较暴力的转移方程：

f[i][j] = sigma{f[k][j − 1] (k + 1 <= i 且 区间(k, i]仅有一条指向外部的边)}

考虑根据区间上边的分布来做一些优化。将从较小编号的点指向较大编号点的边称为后向边，否则称为前向边。

先考虑所有可能跨越右边界i指向外侧的后向边， 把所有出发点小于等于i的后向边按出发点标号大到小排序为 x1, x2, ...

则对于 k >= x1 的转移点，区间 (k, i] 没有指向区间外的后向边, x1 > k >= x2 时 区间 (k, i] 恰有一条指向区间外的后向边。

接着考虑所有可能跨越左边界k+1指向外侧的前向边，任意一条起始点小于等于i的前向边 x –> y (i >= x > y) 都会使 x > k >= y 的区间 (k, i] 拥有一条指向外部的前向边。

最终能转移到i的转移点k共有以下两种:

1. 位于 [x1, i] 且仅有一条后向边的转移点k
2. 位于 [x2, x1) 且没有后向边的转移点k

只要能对拥有0或1条前向边的下标k做快速的区间求和就可以高效的完成dp转移。

位置k对应的前向边数可以随着i的扩大时时维护，让每一条前向边 x –> y (x > y) 在 i = x 时加入，每次加入都是将区间 [y, i − 1] 内转移点的前向边数加一，对当前位置i而言也就是一段后缀的处理。

由于前向边数只增，且某个位置的前向边数大于1后，就不用再维护，所以处理区间加时，可以暴力的遍历每一个点，由于最多被处理两次，整体而言总数不超过2n次。又由于每次操作都是删除末尾/末尾添加，所以只需要对0/1条前向边分别开一个数组就可以O(1)的维护好它们的前缀和，支持快速的区间求和。

取值区间 x1, x2 的维护比较简单，可以用一个堆，随着i的扩大时时维护，让每一条边 x –> y (x < y) 在 i=x 时入堆 i=y 时退出。查询时取堆中的最大值和次大值即可。

时间复杂度O(nk + n * log(n))。