ACM/ICPC 天津赛区 复旦大学命题 题解

City

题目大意：在一棵树结构中，移动一个边，使树结构仍然成立，同时使最远的两个点之间的距离最小.

此题相对比较简单. 首先枚举每个边 k, 记其边权为 W, 删除这个边，树结构被分成两个独立的连通分量 S 和 T.

对两个连通分量分别做两次 DFS.

第一次 求出以各点 u 为根，最长的路径长度和次长路径长度，以及 最长路径上对应的 u 的邻接点 (即此时 u 的孩子).

第二次 对每个点，求出在其独立的连通分量中，**过**每一点 u 的最长路径长度 f(u), 和**以**点 u **为起点**的最长路径长度 g(u). 若记 U 中 g(u) 最小的点 u 为 a, V 中 g(u) 最小的点 u 为 b, 用边 k 连接 (a, b), 过此边的最长路径长度 L = (g(a) + g(b) + W), 为所有移动边 k 的方案中新产生的最长路径最短的方案. 此时，移动边 k 得到的新树中的最长路径为 move(k) = max(max(f(u)), L).

输出最小的 move(k) 即为题目所求答案.

Farm

首先，题目中要求每次偷的菜的价值严格小于以前所有的，所以首先按菜的价格由小到大排序.

要让每两次刷新之间积累的怒气值最大值最小，很容易想到二分这个上界，接下来就是判断可行性了.

很直观的想法就是用动态规划. 我们用 F[i][j] 表示第 i 次刷新，偷到了 j 棵菜之后，在满足二分上界的条件下最少花费的时间. 转移很好写：F[i][j] = min(f[i-1][k] + cost(k+1,j)) + r，其中 k+1 到 j 这些菜的怒气值之和不超过上界, cost(k+1,j)为这次偷菜所用时间.

这样 DP 的时间复杂度是 n*n*m 的，显然不能满足要求. 但是这个 DP 方程满足凸完全单调性.

我们重写 cost 函数：令 Dsum[i] 表示 d1+d2+d3…+di, g[i] 表示 Dsum[1]+Dsum[2]+…+Dsum[i]，那么很显然，cost(k+1,j) = (j-k) * Dsum[j] – (g[j-1] – g[k-1]).

假如存在某一个 x，对于 i < j, F[t][i] + cost(i+1,x) > F[t][j] + cost(j+1,x)，那么显然，对于之后任意 y > x，都有 F[t][i] + cost(i+1,y) > F[t][j] + cost(j+1,y)，于是 i 这个状态就没有用了.

化简上面的不等式，得:

[(f[k – 1][i] + g[i – 1]) – (f[k – 1][j] + g[j – 1])] / (u – v) < dsum[x]

即上不等式成立时, i 应该由 j 替代.

我们可以用单调队列维护这个转移，再注意一些边界问题即可.

时间复杂度 O(nmloga), a 为怒气值之和.

GRAPH AND QUERIES

题目按顺序想比较复杂，但如果逆序处理请求就非常容易了.
先删除所有要求删除的边，并等效的修改所有要求改变边权的请求.
用并查集记录结点间的连通关系，用 treap 之类的平衡二叉树维护结点间点权之间的大小关系.
逆序处理每一个请求:

1. 删边操作等价于添加一条新边，如果待添加的边可以加入 (即两个顶点之前不连通)，将结点数量较少的连通集中的点当作新结点插入结点数量较多的连通集构建的平衡二叉树，可以使合并操作的时间复杂度降低到 O(nlognlogn).
2. 修改点权要先从平衡树中删除该点，修改边权，再重新插入原平衡树中.
3. 询问操作直接在平衡树中查找即可，注意无解的特殊情况.

H

计算几何题. 平面中有若干个圆, 求出从点 S 到点 T 的路径 L, 使 L 只通过圆的圆周, 同时 L 上权值最大的点与最小的点之间的权值之差最小. 点的权值的定义为落在该点上的圆的数目.

题目要求非常简单, 首先预处理出由顶点和边构成的图(注意这里顶点和边都是带权的), 然后枚举所求值并检验就可以了. 但是实现需要相当仔细, 代码较长, 容易出错.

1. 两两枚举平面上的圆, 求出平面上所有圆的交点.
2. 枚举每一个交点, 求出其权值, 并记录落在该点上的所有圆的序号.
3. 枚举每一个圆, 记其圆心为 C, 求出所有落在该圆圆周上的交点, 以点 C 为参考点, 将这些交点按极角排序. 在两两相邻的交点之间连一条边, 边权为同时落在这两个交点上的圆的个数, 即两点之间圆周上每一点的"权值".
4. 拆点为边, 将图中每一个交点拆为两个顶点, 并在两点之间连一条边, 边权为该交点的权值.
5. 枚举路径 L 上点的权值的上限和下限, 用搜索算法验证是否存在满足条件的路径, 并记录最佳答案即可.

Truth

用最小费用流算法即可解决.
首先离散名次.
建图, 将源点连向每个人, 容量为 1; 每个名次区间连向汇点, 容量为区间长度; 每个人连向他所在的各个区间, 容量分别为 1.
在该图中求出最大流, 即为最多说实话的人的数目.
为了保证字典序最大, 给源点流向人的边加上费用, 第 i 个人的费用为 2^(n-i), 这样如果取了第 i 个人, 后面字典序再大, 费用和也小于单选第 i 个人的费用. 这样就能保证取到字典序最大.
对该图求最小费用最大流, 从源出发满流的边所指向的结点集即为字典序最大的方案.