天才黑客 (hack) 解题报告

quailty

2017 年 5 月 10 日

天才黑客 (hack.c/cpp/pas)

给定一个 n 个点 m 条边的无向图，每条边上有一个权值和一个字符串，一条路径的长度是边上的权值和再加上相邻两条边上的字符串的 LCP 长度 $+1$ 之和，再给定一棵 k 个节点的字典树，无向图每条边上的字符串是字典树上根到某个点路径上的字符顺次拼接构成的，求出从 1 号点出发到其他每个点的最短路径长度，10 组数据，其中不超过 2 组大数据。

25 分做法

- **对于第 1 和第 2 个测试点，** $n \leq 5000$，$m \leq 5000$，$k \leq 20000$。

25 分做法

- 对于第 1 和第 2 个测试点，$n \leq 5000$，$m \leq 5000$，$k \leq 20000$。
- 每条边拆成正反两条，dis_i 表示最后一条边是第 i 条边的最短路，转移枚举第 i 条边的终点连出去的下一条边。

25 分做法

- 对于第 1 和第 2 个测试点，$n \leq 5000$，$m \leq 5000$，$k \leq 20000$。
- 每条边拆成正反两条，dis_i 表示最后一条边是第 i 条边的最短路，转移枚举第 i 条边的终点连出去的下一条边。
- Dijkstra 转移，需要计算两条边上的字符串的 LCP 长度，也就是求出字典树上两个点的 LCA。

25 分做法

- 对于第 1 和第 2 个测试点，$n \leq 5000$，$m \leq 5000$，$k \leq 20000$。
- 每条边拆成正反两条，dis_i 表示最后一条边是第 i 条边的最短路，转移枚举第 i 条边的终点连出去的下一条边。
- Dijkstra 转移，需要计算两条边上的字符串的 LCP 长度，也就是求出字典树上两个点的 LCA。
- 最后枚举每条边，更新这条边的终点的答案。

25 分做法

- 对于第 1 和第 2 个测试点，$n \leq 5000$，$m \leq 5000$，$k \leq 20000$。
- 每条边拆成正反两条，dis_i 表示最后一条边是第 i 条边的最短路，转移枚举第 i 条边的终点连出去的下一条边。
- Dijkstra 转移，需要计算两条边上的字符串的 LCP 长度，也就是求出字典树上两个点的 LCA。
- 最后枚举每条边，更新这条边的终点的答案。
- 复杂度 $O(n + m^2 + k \log k)$。

70 分做法

- **对于第 6 到第 14 个测试点**，$n \leq 50000$，$m \leq 50000$，$k \leq 20000$，$nk \leq 200000$。

70 分做法

- 对于第 6 到第 14 个测试点，$n \leq 50000$，$m \leq 50000$，$k \leq 20000$，$nk \leq 200000$。
- 每个点 i 拆成 in 和 out 两组点，每组 k 个点（和字典树上的点一一对应），表示进入点 i 所走的边上的字符串和离开点 i 所走的边上的字符串。

70 分做法

- 对于第 6 到第 14 个测试点，$n \leq 50000$，$m \leq 50000$，$k \leq 20000$，$nk \leq 200000$。
- 每个点 i 拆成 in 和 out 两组点，每组 k 个点（和字典树上的点一一对应），表示进入点 i 所走的边上的字符串和离开点 i 所走的边上的字符串。
- 从 i 的 in 中的 p 号点走到 i 的 out 中的 q 号点时，会产生 $dep_{LCA(p,q)}$ 的代价，这里认为 $dep_{root} = 0$，$LCA(p, q)$ 表示 p 和 q 在字典树上的 LCA。

70 分做法

- 对于第 6 到第 14 个测试点，$n \leq 50000$，$m \leq 50000$，$k \leq 20000$，$nk \leq 200000$。
- 每个点 i 拆成 in 和 out 两组点，每组 k 个点（和字典树上的点一一对应），表示进入点 i 所走的边上的字符串和离开点 i 所走的边上的字符串。
- 从 i 的 in 中的 p 号点走到 i 的 out 中的 q 号点时，会产生 $dep_{LCA(p,q)}$ 的代价，这里认为 $dep_{root} = 0$，$LCA(p, q)$ 表示 p 和 q 在字典树上的 LCA。
- 每条边 (a, b, c, d) 相当于从 a 的 out 中的 d 号点花费 c 的代价可以跳到 b 的 in 中的 d 号点。

70 分做法

- 对于第 6 到第 14 个测试点，$n \leq 50000$，$m \leq 50000$，$k \leq 20000$，$nk \leq 200000$。
- 每个点 i 拆成 in 和 out 两组点，每组 k 个点（和字典树上的点一一对应），表示进入点 i 所走的边上的字符串和离开点 i 所走的边上的字符串。
- 从 i 的 in 中的 p 号点走到 i 的 out 中的 q 号点时，会产生 $dep_{LCA(p,q)}$ 的代价，这里认为 $dep_{root} = 0$，$LCA(p, q)$ 表示 p 和 q 在字典树上的 LCA。
- 每条边 (a, b, c, d) 相当于从 a 的 out 中的 d 号点花费 c 的代价可以跳到 b 的 in 中的 d 号点。
- 问题转化成求从 1 的 in 中的 1 号点出发走到其它点的 in 中的任意一个点的最短路。

70 分做法

- 如果每个 i 的 in 中的每个点和 out 中的每个点直接两两连边，边数会达到 $O(nk^2 + m)$，这是不能接受的。

70 分做法

- 如果每个 i 的 in 中的每个点和 out 中的每个点直接两两连边，边数会达到 $O(nk^2 + m)$，这是不能接受的。
- 考虑枚举作为 LCA 的点 u，从 in 中的 u 号点花费 dep_u 的代价可以跳到 out 中 u 子树内的点。

70 分做法

- 如果每个 i 的 in 中的每个点和 out 中的每个点直接两两连边，边数会达到 $O(nk^2 + m)$，这是不能接受的。
- 考虑枚举作为 LCA 的点 u，从 in 中的 u 号点花费 dep_u 的代价可以跳到 out 中 u 子树内的点。
- 再枚举一个 u 的儿子 v，从 in 中的 v 子树内的点花费 dep_u 的代价可以跳到 out 中的 u 子树内但不在 v 子树内的点。

70 分做法

- 考虑 Dijkstra 的过程，每次会从未被访问过的点之中选出最短路最小的点，将其标为已访问并更新其相邻点的最短路。

70 分做法

- 考虑 Dijkstra 的过程，每次会从未被访问过的点之中选出最短路最小的点，将其标为已访问并更新其相邻点的最短路。
- 可以关注到如果选出的点在某条边的起点集合内，那么更新完终点集合内的点的最短路之后这条边就没有用了。

70 分做法

- 考虑 Dijkstra 的过程，每次会从未被访问过的点之中选出最短路最小的点，将其标为已访问并更新其相邻点的最短路。
- 可以关注到如果选出的点在某条边的起点集合内，那么更新完终点集合内的点的最短路之后这条边就没有用了。
- 以后选出的起点集合内的点的最短路不会比第一次选出的小，没有必要再用这条边更新终点集合内的点的最短路。

70 分做法

- 求出 DFS 序后每棵子树对应一段连续区间，线段树维护起点到每个点的最短路。

70 分做法

- 求出 DFS 序后每棵子树对应一段连续区间，线段树维护起点到每个点的最短路。
- 每次取出当前未被访问过的点中选出最短路最小的点，将其标为已访问（在线段树上赋值为 INF）。

70 分做法

- 求出 DFS 序后每棵子树对应一段连续区间，线段树维护起点到每个点的最短路。
- 每次取出当前未被访问过的点中选出最短路最小的点，将其标为已访问（在线段树上赋值为 INF）。
- 枚举起点集合包含当前点的边。

70 分做法

- 求出 DFS 序后每棵子树对应一段连续区间，线段树维护起点到每个点的最短路。
- 每次取出当前未被访问过的点中选出最短路最小的点，将其标为已访问（在线段树上赋值为 INF）。
- 枚举起点集合包含当前点的边。
- 对于 m 条边中起点是当前点的边以及从当前点出发跳到另一棵子树内的边，直接枚举即可。

70 分做法

- 求出 DFS 序后每棵子树对应一段连续区间，线段树维护起点到每个点的最短路。
- 每次取出当前未被访问过的点中选出最短路最小的点，将其标为已访问（在线段树上赋值为 INF）。
- 枚举起点集合包含当前点的边。
- 对于 m 条边中起点是当前点的边以及从当前点出发跳到另一棵子树内的边，直接枚举即可。
- 对于从一棵子树内跳到另一棵子树外的边，从当前点出发沿着字典树暴力往上跳，如果遇到已经跳过的点，那么从这个点往上一直到根都是跳过的，直接退出即可。

70 分做法

- 求出 DFS 序后每棵子树对应一段连续区间，线段树维护起点到每个点的最短路。
- 每次取出当前未被访问过的点中选出最短路最小的点，将其标为已访问（在线段树上赋值为 INF）。
- 枚举起点集合包含当前点的边。
- 对于 m 条边中起点是当前点的边以及从当前点出发跳到另一棵子树内的边，直接枚举即可。
- 对于从一棵子树内跳到另一棵子树外的边，从当前点出发沿着字典树暴力往上跳，如果遇到已经跳过的点，那么从这个点往上一直到根都是跳过的，直接退出即可。
- 复杂度 $O((nk+m)\log(nk))$。

70 分做法

- 求出 DFS 序后每棵子树对应一段连续区间，线段树维护起点到每个点的最短路。
- 每次取出当前未被访问过的点中选出最短路最小的点，将其标为已访问（在线段树上赋值为 INF）。
- 枚举起点集合包含当前点的边。
- 对于 m 条边中起点是当前点的边以及从当前点出发跳到另一棵子树内的边，直接枚举即可。
- 对于从一棵子树内跳到另一棵子树外的边，从当前点出发沿着字典树暴力往上跳，如果遇到已经跳过的点，那么从这个点往上一直到根都是跳过的，直接退出即可。
- 复杂度 $O((nk+m)\log(nk))$。
- 需要特判小数据，否则会 TLE。

100 分做法

- **对于第 15 到第 20 个测试点**，$n \leq 50000$，$m \leq 50000$，$k \leq 20000$。

100 分做法

- 对于第 15 到第 20 个测试点，$n \leq 50000$，$m \leq 50000$，$k \leq 20000$。
- 每个点 i 拆成 in 和 out 两组点，每组 k 个点（和字典树上的点一一对应），表示进入点 i 所走的边上的字符串和离开点 i 所走的边上的字符串。

100 分做法

- 对于第 15 到第 20 个测试点，$n \leq 50000$，$m \leq 50000$，$k \leq 20000$。
- 每个点 i 拆成 in 和 out 两组点，每组 k 个点（和字典树上的点一一对应），表示进入点 i 所走的边上的字符串和离开点 i 所走的边上的字符串。
- 如果存在一条边 (a, b, c, d)，这条边相当于连接了 a 的 out 中的 d 号点以及 b 的 in 中的 d 号点，边权为 c，则认为 a 的 in 和 out 中的 d 号点以及 b 的 in 和 out 中的 d 号点都是关键点。

100 分做法

- 对于第 15 到第 20 个测试点，$n \leq 50000$，$m \leq 50000$，$k \leq 20000$。
- 每个点 i 拆成 in 和 out 两组点，每组 k 个点（和字典树上的点一一对应），表示进入点 i 所走的边上的字符串和离开点 i 所走的边上的字符串。
- 如果存在一条边 (a, b, c, d)，这条边相当于连接了 a 的 out 中的 d 号点以及 b 的 in 中的 d 号点，边权为 c，则认为 a 的 in 和 out 中的 d 号点以及 b 的 in 和 out 中的 d 号点都是关键点。
- 只需要保留关键点及其两两 LCA，也就是对每个 in 和 out 求个虚树，最后只会剩下 $O(n+m)$ 个点。

100 分做法

- 对于第 15 到第 20 个测试点，$n \leq 50000$，$m \leq 50000$，$k \leq 20000$。
- 每个点 i 拆成 in 和 out 两组点，每组 k 个点（和字典树上的点一一对应），表示进入点 i 所走的边上的字符串和离开点 i 所走的边上的字符串。
- 如果存在一条边 (a, b, c, d)，这条边相当于连接了 a 的 out 中的 d 号点以及 b 的 in 中的 d 号点，边权为 c，则认为 a 的 in 和 out 中的 d 号点以及 b 的 in 和 out 中的 d 号点都是关键点。
- 只需要保留关键点及其两两 LCA，也就是对每个 in 和 out 求个虚树，最后只会剩下 $O(n+m)$ 个点。
- 复杂度 $O((n+m)\log(n+m)+k\log k)$。

Thank you!