图论算法选讲-解题报告

徐沐杰

南京大学

2023 年 7 月 11 日

追忆

- 任选一个点的 DFS 序，查看其最后一个点，这必然是它的一个叶子节点。于是它所在的 DFS 序的下一个点必然是它的父亲。

追忆

- 任选一个点的 DFS 序，查看其最后一个点，这必然是它的一个叶子节点。于是它所在的 DFS 序的下一个点必然是它的父亲。
- 于是我们知道了它和它的父亲之间有一条边。我们把它从所有 DFS 序中删去，再观察下一个叶子节点，重复以上流程即可。

追忆

- 任选一个点的 DFS 序，查看其最后一个点，这必然是它的一个叶子节点。于是它所在的 DFS 序的下一个点必然是它的父亲。
- 于是我们知道了它和它的父亲之间有一条边。我们把它从所有 DFS 序中删去，再观察下一个叶子节点，重复以上流程即可。
- 复杂度 $O(n^2)$。

军争

- 离线从后往前处理询问。

军争

- 离线从后往前处理询问。
- 发现删点变为加点，考虑维护一个支持增加点的最短路径。

军争

- 离线从后往前处理询问。
- 发现删点变为加点，考虑维护一个支持增加点的最短路径。
- 发现这就是 Floyd 换了个点的枚举顺序罢了，复杂度 $O(n^3 + q)$。

旅程

- 做法很多，这里只给出一种可行的解法。

旅程

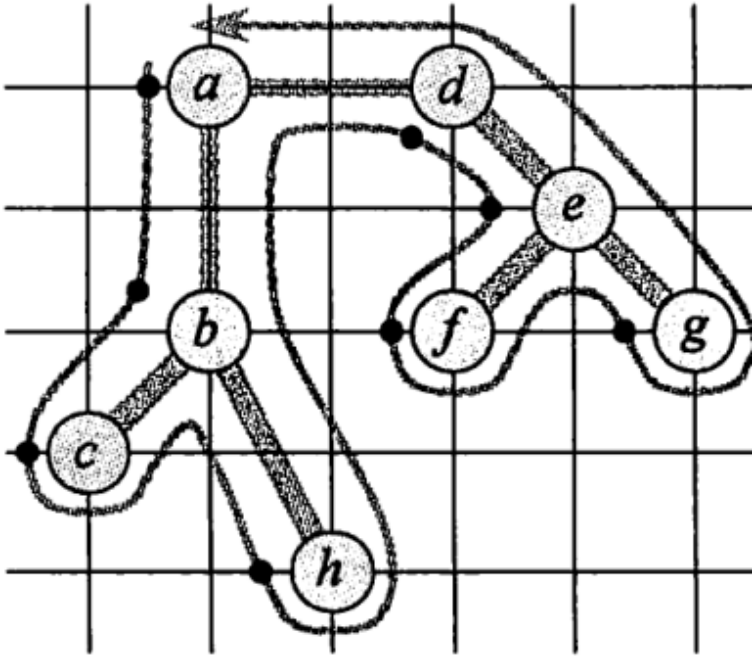
- 做法很多，这里只给出一种可行的解法。
- 考虑计算出平面上点的最小生成树，然后按照生成树的 dfs 序逐一访问点。使用 Prim 计算最小生成树，复杂度 $O(n^2)$。

旅程

- 做法很多，这里只给出一种可行的解法。
- 考虑计算出平面上点的最小生成树，然后按照生成树的 dfs 序逐一访问点。使用 Prim 计算最小生成树，复杂度 $O(n^2)$。
- 因为最优解的代价肯定大于等于任一生成树，且按照 dfs 序逐一访问点的代价总和是不会超过生成树代价的两倍的。

旅程

- 做法很多，这里只给出一种可行的解法。
- 考虑计算出平面上点的最小生成树，然后按照生成树的 dfs 序逐一访问点。使用 Prim 计算最小生成树，复杂度 $O(n^2)$。
- 因为最优解的代价肯定大于等于任一生成树，且按照 dfs 序逐一访问点的代价总和是不会超过生成树代价的两倍的。
- 因此这个估计解的代价不可能大于 TSP 的代价的两倍。



相聚

- 25pts Floyd 模板题。

相聚

- 25pts Floyd 模板题。
- 10pts (bonus) 输出权值最短的一条边。

相聚

- 25pts Floyd 模板题。
- 10pts (bonus) 输出权值最短的一条边。
- 50pts (bonus) 对称有向图事实上就是无向图。

相聚

- 25pts Floyd 模板题。
- 10pts (bonus) 输出权值最短的一条边。
- 50pts (bonus) 对称有向图事实上就是无向图。
- 考虑使用一个超级源点向所有关键点连长度为 0 的单向边。

相聚

- 25pts Floyd 模板题。
- 10pts (bonus) 输出权值最短的一条边。
- 50pts (bonus) 对称有向图事实上就是无向图。
- 考虑使用一个超级源点向所有关键点连长度为 0 的单向边。
- 以超级源点为起点进行 Dijkstra 算法，当有任意两个关键点延伸出的最短路相交时，我们就找到了一条连通两个关键点的道路。

相聚

- 25pts Floyd 模板题。
- 10pts (bonus) 输出权值最短的一条边。
- 50pts (bonus) 对称有向图事实上就是无向图。
- 考虑使用一个超级源点向所有关键点连长度为 0 的单向边。
- 以超级源点为起点进行 Dijkstra 算法，当有任意两个关键点延伸出的最短路相交时，我们就找到了一条连通两个关键点的道路。
- 操作上除了记录点的最短距离还要记录来源关键点。

相聚

- 25pts Floyd 模板题。
- 10pts (bonus) 输出权值最短的一条边。
- 50pts (bonus) 对称有向图事实上就是无向图。
- 考虑使用一个超级源点向所有关键点连长度为 0 的单向边。
- 以超级源点为起点进行 Dijkstra 算法，当有任意两个关键点延伸出的最短路相交时，我们就找到了一条连通两个关键点的道路。
- 操作上除了记录点的最短距离还要记录来源关键点。
- 只需枚举两个关键点延伸出的路径交汇的边 (u, v, w)，计算 $d(S, u) + w + d(S, v)$ 即可。当然其中 u, v 的来源关键点需要不同。

相聚

- 100pts 仍用超级源点，并维护一个「关键点次短路」，对于每个点不仅记录下其最短距离，还记录下次短距离/来源点。其中，次短距离是除最短距离来源点而外的最短路径长。

相聚

- 100pts 仍用超级源点，并维护一个「关键点次短路」，对于每个点不仅记录下其最短距离，还记录下次短距离/来源点。其中，次短距离是除最短距离来源点而外的最短路径长。
- 与基于 Dij 的次短路类似，一个点最多会被松弛两次。

相聚

- 100pts 仍用超级源点，并维护一个「关键点次短路」，对于每个点不仅记录下其最短距离，还记录下次短距离/来源点。其中，次短距离是除最短距离来源点而外的最短路径长。
- 与基于 Dij 的次短路类似，一个点最多会被松弛两次。
- 还可以使用另外一种次短路方案来类推。

相聚

- 100pts 仍用超级源点，并维护一个「关键点次短路」，对于每个点不仅记录下其最短距离，还记录下次短距离/来源点。其中，次短距离是除最短距离来源点而外的最短路径长。
- 与基于 Dij 的次短路类似，一个点最多会被松弛两次。
- 还可以使用另外一种次短路方案来类推。
- 考虑任意一条边 (u, v, w)，如果最近关键点对的最短路径上含有它，那么这个距离应该是 $d(S, u) + w + d(v, S)$。且其中的两个距离应该来自于不同的关键点。

相聚

- 100pts 仍用超级源点，并维护一个「关键点次短路」，对于每个点不仅记录下其最短距离，还记录下次短距离/来源点。其中，次短距离是除最短距离来源点而外的最短路径长。
- 与基于 Dij 的次短路类似，一个点最多会被松弛两次。
- 还可以使用另外一种次短路方案来类推。
- 考虑任意一条边 (u, v, w)，如果最近关键点对的最短路径上含有它，那么这个距离应该是 $d(S, u) + w + d(v, S)$。且其中的两个距离应该来自于不同的关键点。
- 发现这个思路和 85 分做法已经非常相似了。只是 $d(v, S)$ 不容易求。

相聚

- 100pts 仍用超级源点，并维护一个「关键点次短路」，对于每个点不仅记录下其最短距离，还记录下次短距离/来源点。其中，次短距离是除最短距离来源点而外的最短路径长。
- 与基于 Dij 的次短路类似，一个点最多会被松弛两次。
- 还可以使用另外一种次短路方案来类推。
- 考虑任意一条边 (u, v, w)，如果最近关键点对的最短路径上含有它，那么这个距离应该是 $d(S, u) + w + d(v, S)$。且其中的两个距离应该来自于不同的关键点。
- 发现这个思路和 85 分做法已经非常相似了。只是 $d(v, S)$ 不容易求。
- 观察发现只需要把除超级源点外的边取反就可以计算出 $d(v, S)$ 了。

相聚

- 100pts 仍用超级源点，并维护一个「关键点次短路」，对于每个点不仅记录下其最短距离，还记录下次短距离/来源点。其中，次短距离是除最短距离来源点而外的最短路径长。
- 与基于 Dij 的次短路类似，一个点最多会被松弛两次。
- 还可以使用另外一种次短路方案来类推。
- 考虑任意一条边 (u, v, w)，如果最近关键点对的最短路径上含有它，那么这个距离应该是 $d(S, u) + w + d(v, S)$。且其中的两个距离应该来自于不同的关键点。
- 发现这个思路和 85 分做法已经非常相似了。只是 $d(v, S)$ 不容易求。
- 观察发现只需要把除超级源点外的边取反就可以计算出 $d(v, S)$ 了。
- 和无向图做法一样，枚举交汇边界的边即可。

相聚

- 100pts 仍用超级源点，并维护一个「关键点次短路」，对于每个点不仅记录下其最短距离，还记录下次短距离/来源点。其中，次短距离是除最短距离来源点而外的最短路径长。
- 与基于 Dij 的次短路类似，一个点最多会被松弛两次。
- 还可以使用另外一种次短路方案来类推。
- 考虑任意一条边 (u, v, w)，如果最近关键点对的最短路径上含有它，那么这个距离应该是 $d(S, u) + w + d(v, S)$。且其中的两个距离应该来自于不同的关键点。
- 发现这个思路和 85 分做法已经非常相似了。只是 $d(v, S)$ 不容易求。
- 观察发现只需要把除超级源点外的边取反就可以计算出 $d(v, S)$ 了。
- 和无向图做法一样，枚举交汇边界的边即可。
- 复杂度均为 $O(m \log n)$。

提问 || 完结

Any Question?

提问 || 完结

Any Question?

Thank you for listening!