Tutorial 5

BFS：从 Breadth-First 到 Best-First

BestFS skeleton

在图中贪心搜索某种最优结构时（MST，最短路径）时的一类共性的方法：BestFS，其源于广度优先搜索。

在 BestFS 中，与图遍历类似，每个节点不可逆地在以下三种状态中转换（3F）：

- Fresh：贪心搜索尚未涉及
- Fringe：进行贪心选择的候选节点
- Finished：已完成贪心搜索的节点

BestFS 的搜索过程也与 BFS 相似，从某个节点开始，处理完其之后将其所有邻居添加到候选点集合，进入 Fringe 状态。随着贪心选择，Fringe 中不断有节点被选出，处理完成进入 Finish 状态，其邻居根据状态的不同一一被更新

- 邻居为 Fresh：节点进入候选节点，变为 Fringe，与图遍历中遇到白色节点相似
- 邻居为 Fringe：检查其贪心指标是否随节点被选入而更新

在上述过程中，所有节点都经历 Fresh $\to$ Fringe $\to$ Finished 的不可逆的转变

算法推进由一个调度器完成，在 BestFS 中使用的调度器为优先级队列，优先级即为贪心指标。

BestFS 的代价

BestFS 的代价主要取决于优先级队列。从点的视角来看，每个节点进入一次队列（INSERT），离开一次队列（EXTRACT-MIN），从边的视角来看，对于每条边都有可能更新其指向的顶点的优先级（DECREASE-KEY），故

$$T(n,m)=O(n\times C_{\text{EXTRACT-MIN}}+n\times C_{\text{INSERT}}+m\times C_{\text{DECREASE-KEY}})$$

根据优先级队列实现的不同，对应的操作代价也不同

实现	INSERT	EXTRACT-MIN	DECREASE-KEY
数组	O(1)	O(n)	O(1)
堆	O(log n)	O(log n)	O(log n)

常见误区

堆实现优先级队列时，堆操作代价究竟是 $O(\log m)$ 还是 $O(\log n)$?

$O(\log n)$ ，因为在队列中调度的是顶点而非边

堆实现优先级队列时，DECREASE-KEY 的代价是 $O(\log n)$ 还是 $O(n)$?

$O(\log n)$ ，具体实现时可使用一个 hash table 存储 value-index 对（value 需要 unique）

MCE

MCE skeleton

在求解 MST 的过程中，Prim 和 Kruskal 可看作一个更抽象的框架的实例化，这个框架称为 MCE (Minimum-weight Cut-crossing Edge) 框架。这个框架基于切（Cut）的概念

> **定义 10.3** 切
>
> 给定连通无向图 $G=(V,E)$ ，如果非空点集 V_1,V_2 满足
>
> $$V_1 \cup V_2 = V, V_1 \cap V_2 = \varnothing$$
>
> 则 V_1,V_2 构成 G 的一个切

显然对 G 的任意某个切，可以将边划分为跨越切的边和切内部的边，图的连通性保证了跨越切的边的存在，在其中权值最小（不一定唯一）的边即为 MCE

MCE 与 MST 的本质相关联

> **定理 10.6** 对某条边 e ，若存在某个切使得 e 成为该切的 MCE，则 e 必然属于某个 MST

证明使用归谬法即可

用 MCE 求解具体问题

图中最小的边是否一定在 MST?

一定，一定存在一个 cut 使其为 MCE

图中第二小的边是否一定在 MST?

一定，一定存在一个 cut 使其为 MCE

图中第三小的边是否一定在 MST?

不一定，可使用对手策略使任意 cut 的 MCE 均不是第三小的边（hint：三条边成环，第二小和第一小的边一定跨越所有可能的 cut)

MCE 视角的 Prim 与 Kruskal

Prim

对于 Prim 算法的执行阶段 $T^{(k-1)}$ ，将其中节点记为 V_1 ，令 $V_2 = V \setminus V_1$ ，则 V_1, V_2 形成一个切，对于跨越切的边在 V_2 中的端点，这些节点即是 Fringe。Prim 的贪心选择可认为是选择了一条 MCE。由于算法选择的 MCE 总是一端在 $T^{(k-1)}$ 一端在 Fringe，故不会成环，最终所有 MCE 形成图的 MST

Kruskal

从 MCE 角度来看，Kruskal 每次选择的边均是 MCE，即可构造出一个对应的 cut，将所选边的两个顶点和与其相连的顶点分别分配至两个点集，其余顶点任意分配即可，只需保证互相连通的顶点分配在同一点集。这样构造出来的 cut 保证所选边是其 MCE（权值比其小的边使其成为切内部的边），Kruskal 使用并查集保证选择的边不成环，故最终选择的所有 MCE 构成 MST

Further reading

基于精化 (renement) 关系理解算法的正确性：以MST、共识算法为例

优化问题

Path 与 MST 的不同

Path 和 MST 的贪心原则不同，也导致了其对负权边的容忍程度不同

DAG

在 DAG 中可以解决一些一般图中无法解决的问题

- 存在负权边的 SSSP
- 最长路径
- 路径权乘积最大

最大相容任务集合

贪心解决方案：贪心指标为结束时间最早，证明可见课本 P.153

DP 方案：子问题 S_{ij} 为任务 i 结束后任务 j 开始前的任务，其中最大相容个数