Minimum Spanning Tree (prims and kruskals)

08 April 2025 00:23

uses greedy approach

for acyclic graph T,

$$w(T) = \sum_{u,v \in T} w(u,v)$$ is minimized.

	Binary Heap	Fibonacci Heap
Kruskal's →	$O(E \lg V)$	
Prims →	$O(E \lg V)$	$O(E + V \log V)$ better if $\lvert V\rvert >> \lvert E\rvert$

growing of min spanning tree.

$G(V,E)$ with $f(w) : E \to R$

* Generic Min spanning tree which grows one edge at a time.

$A \to$ subset of MST.

* we add edge $e(u,v)$ such that A maintains the invariant $(A + (u,v) \subseteq MST)$

$\underbrace{\text{add edge } e(u,v)}_{\text{safe edge.}}$

GENERIC_MST(G, w)

1.) $A = \varphi$
2.) while A does not form a spanning tree
3.) find $e(u,v)$ that is safe
4.) $A = A \cup \{u,v\}$
5.) return A.

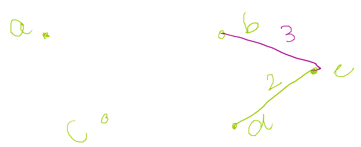
* start empty
* while not complete ↓ add safe edge.

CUT$(S, V-S)$ → an edge crosses the cut if one endpoint in S and another in $V-S$

The cut respects the set A. if no edge in a crosses the cut

Light edge → edge in a cut with min weight. can take any if tie.

KRUSKALS → set A is a forest, always adds minimum weight edge to the graph that connects two distinct components



PRIMS :→ set A is a single tree, adds min edge (u,v), u in A and v not in A.

KRUSKALS :→ greedy, at each step adds edge of minimum weight.

Uses disjoint set structure. union by Rank + path compression.

MST_KRUSKALS (G, W)

1.) $A = \varphi$
2.) for each vertex $v \in G.V$
3.) MAKE_SET(V) $\rightarrow O(1) * V = O(V)$
4.) sort edges of G.E into ascending order by weight w. $\rightarrow O(E \log E)$
5.) for each edge(u,v) in sorted order $\rightarrow O(E)$
6.) if FIND_SET(U) ≠ FIND_SET(V)
7.) $A = A \cup \{u,v\}$
8.) UNION(u,v)
9.) return A.

$O((V+E)\,\alpha(V))$

∴ G is connected $|E| \geq |V|-1$

$\Rightarrow O(E\,\alpha(V))$ since $\alpha(V) = O(\lg V) = O(\lg E)$

we have $O(E \lg V)$ ← Kruskal's algo.

PRIMS ALGO

edges in set A always forms a single tree.

start → φ

end → $V \in A \;\forall\; V \in G.V$

* arbitrary root

PRIMS_ALGO(G, W)

1.) for each v in G.V
2.) v.key = ∞
3.) v.π = NULL

$O(V)$ (1–5)

4.) r.key = 0
5.) Q = G.V
6.) while Q ≠ φ → |V|
7.) u = EXTRACT_MIN(Q) → $O(\lg V)$
8.) for all $v \in G.adj[u]$ — $O(E)$
9.) if $v \in Q$ and $w(u,v) < v.key$
10.) $v.key = w[u,v]$ → decreaseKey $O(\log V)$
11.) $v.\pi = u$.

∴ Total T.C = $O(V \log V + E \log V)$
$= O(E \log V)$

almost same as of Kruskal's.

using Fibonacci Heap
$= O(E + V \log V)$