MINIMUM SPANNING TREE : Minimum cost tree which connects all the vertices in a graph

KRUSKAL'S ALGORITHM

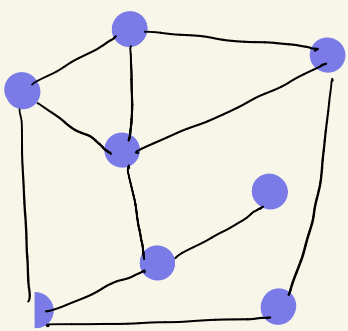
KEY CONCEPT: Repeatedly add the next lightest edge if this doesn't produce a cycle.

PRIM'S ALGORITHM

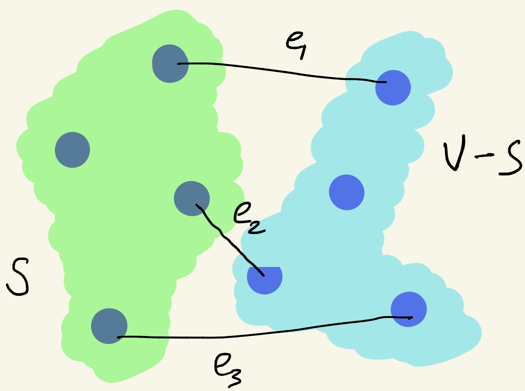
KEY CONCEPT: Repeatedly attach a new vertex to the current tree by a lightest edge

CUT PROPERTY

Let S be any subset of vertices and let e be the min cost edge with exactly one endpoint in S. Then Minimum spanning tree of the graph contains e.



G → Graph.



$e = \min(e_1, e_2, e_3)$

$\Rightarrow e \in MST(G)$

KRUSKAL'S ALGORITHM

$\forall\ u \in V$:

MakeSet(u)

$X \leftarrow$ empty set

Sort all edges E by weight

$\forall\ \{u, v\} \in E$ in non-decreasing order

if Find(u) $\neq$ Find(v)

add $\{u, v\}$ to X

Union (u, v)

return (X)

checks whether a cycle is forming

Joins connected components

IMPLEMENTATION

SIMPLE ARRAY

① QuickFind
② weighted Quick Union

	Find	Union
QuickFind	$O(1)$	$O(N)$
weighted Quick Union	$O(\log N)$	$O(\log N)$

RUNNING TIME ANALYSIS

① Sorting edges $= O(|E| \log |V|)$

② Processing edges =

$2|E|\, T(\text{find}) + |V| \cdot T(\text{Union})$

$= O(|E| \log |V|)$

Running Time $= O(|E| \log |V|)$

PRIM'S ALGORITHM

$\forall\, u \in V:$

$\quad cost[u] \leftarrow \infty$

$\quad parent[u] \leftarrow None$

Pick initial vertex u_0

$cost[u] \leftarrow 0$

$PrioQ \leftarrow MakeQueue(V)$

while PrioQ is not empty :-

$\quad v \leftarrow ExtractMin(PrioQ)$

$\quad \forall\, \{v, z\} \in E:$

$\quad\quad$ if $z \in PrioQ$ and $cost[z] > w(v, z)$:

$\quad\quad\quad cost[z] \leftarrow (v, z)$

$\quad\quad\quad parent[z] \leftarrow v$

$\quad\quad\quad ChangePriority(PrioQ, z, cost[z])$

IMPLEMENTATION

PrioQ → Priority Queue

1. SIMPLE ARRAY
2. BINARY HEAP BASED

	EXTRACT MIN	CHANGE PRIORITY
SIMPLE ARRAY	$O(V)$	$O(1)$
BINARY HEAP	$\log\lvert V\rvert$	$\log\lvert V\rvert$

RUNNING TIME ANALYSIS

$$|V| \cdot T(\text{EXTRACT MIN}) + |E| \cdot T(\text{Change Priority})$$

ARRAY: $O(|V|^2)$ BINARY HEAP: $O(|E| \log |V|)$