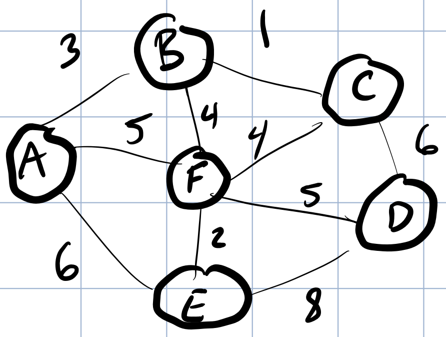
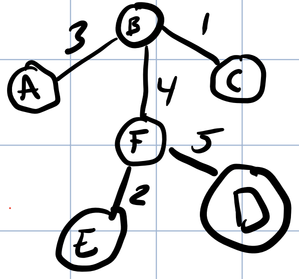
PRIMS

Minimum spanning Tree. (MST)





Greedy w/ respect to vertices. Trys to add vertices.

can get to B from A at a cost of 3

1 isolated vertex is a tree.

vertices = # edges +1.

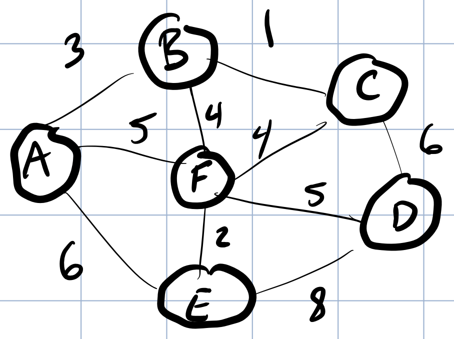
Pick Rand Start		A	B	C	D	E	F
		0	∞	∞	∞	∞	∞
A	0	X	3 A	∞	∞	6 A	5 A
B	3 A	X	X	1 B	∞	6 A	4 B < 5 A
C	1 B	X	X	X	6 C	6 A	4 B
F	4 B	X	X	X	5 F < 6 C	2 F < 6 A	X
E	2 F	X	X	X	8 E ≮ 5 F	X	X
D	5 F	X	X	X	X	X	X

1. How can I get to a vertice from this point.
2. a. If I can't, inherit from previous value.
   b. If I can already, see if mine is more efficient & keep cheapest.
3. Repeat for each in Row.
4. Start next Row with cheapest value.

Adding another vertex with an edge preserves a tree.

Add vertices one at a time until connected into a tree.

Dijkstras Algo - Shortest Path



Pick Start Point

S	A	B	C	D	E	F
	∞ -1	∞ -1	∞ -1	∞ -1	∞ -1	∞ -1
A $\boxed{0}$ -1	X	$\boxed{0}$ +3 3 A < ∞	$\boxed{0}$ ∞	$\boxed{0}$ ∞	$\boxed{0}$ +6 6 A < ∞	$\boxed{0}$ +5 5 A < ∞
B 3 A	X	X	$\boxed{3}$ +1 4 B < ∞	$\boxed{3}$ ∞	$\boxed{3}$ 6 A	$\boxed{3}$ +4 7 B < 5 A
C 4 B	X	X	X	$\boxed{4}$ +6 10 C < ∞	$\boxed{4}$ 6 A	$\boxed{4}$ +4 8 C < 5 A
F 5 A	X	X	X	$\boxed{5}$ +5 10 F < 10 C	$\boxed{5}$ +2 7 F < 6 A	X
E 6 A	X	X	X	$\boxed{6}$ +8 14 E < 10 C	X	X
D 10 C	X	X	X	X	X	X

Adjacency Matrix

	A	B	C	D	E	F
A	-1	3	-1	-1	6	5
B	3	-1	1	-1	-1	4
C	-1	1	-1	6	-1	4
D	-1	-1	6	-1	8	5
E	6	-1	-1	8	-1	2
F	5	4	4	5	2	—

Pre: Make Adj Matrix

1. Take start node value, add in node cost & compare to previous.
2. start with Cheapest for next rows
3. Repeat

1. Must be a directed Graph.
2. Shows All shortest paths From a single start node to All possible destinations.

- Single Source → All Destinations

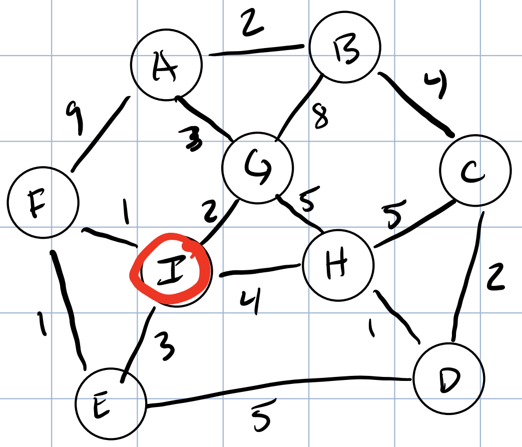
Work: # Vertices^2 → V^2

↪ All Pairs → # vertices^3 → V^3

Any SRC → Any Dest = V · Table

FLOYDS

All Source to all dest. Lowest Cost.



	A	B	C	D	E	F	G	H	I
A	0	2	6	8	10	9	3	8	5
B	2	0	4	6	11	11	5	7	7
C	6	4	0	2	7	8	8	3	7
D	8	6	2	0	5	6	6	1	5
E	10	11	7	5	0	1	11	6	2
F	9	11	8	6	1	0	12	7	1
G	3	5	8	6	11	12	0	5	2
H	8	7	3	1	6	7	5	0	4
I	5	7	7	5	2	1	2	4	0

1. Start w/ a road block & fill out table w/ shortest Path from any start → destination.
2. Lift Road block & update every Row.

To Check if efficient: start → Road block → End. Is it cheaper? → check Table. $BE = BI + IE$ → update if cheaper

Continued

FLOYD Continued

Start out w/ Road Block in every city. $\longrightarrow$ vertices connecting set connections to 1 or ∞

Recalculate graph after removing each Road Block.

↳ Start table will look like adjacency table.