Announcement

UCF High School Programming Tournament
March 12, 2018 (Monday)
Need Volunteers! (to be hosts)
S.zielinski@72 gmail.com
Steve Zielinski

Schedule

3/6 - Dijkstra's ALG * * *
3/8 - Network Flow (Prog Due)
SPRING BREAK
3/20 - TRAVIS (Exam 2 Review)
3/22 - Exam #2 (TAs will give)

Dijkstra's Algs

3/6/18 (2)

Problem: Single Source Shortest Path Problem on a DIRECTED WEIGHTED graph.

***

Only works on graphs with non-negative edge weights!!!

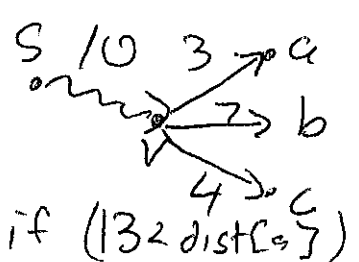
How it works

1) Distance array (store estimates from source to each other vertex)
Initialize 0 (source), ∞ (everywhere else)

0	∞	∞	∞	∞	∞
0	1	2	3	4	5

2) Create a set S of vertices to which we already know the shortest distance.
Set S = ∅. (In code, boolean array...)

3) While (S != V) {



(a) find vertex v such that v ∉ S and distance[v] is minimum.
(b) Add v to S. used[v] = true;
(c) for each edge from v to w (for all vertices connected to v)
dist[w] = min (dist[w], dist[v] + adj[v][w])

}

3/6/18 (3)

4) (12 pts) Dijkstra's Algorithm Trace

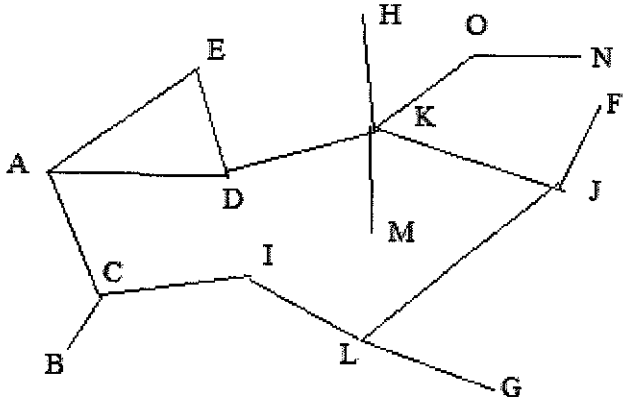
Trace through Dijkstra's algorithm on the graph described below using A as the source vertex. The last line of your chart should display the shortest distances from A to all of the vertices in the graph. In order to receive credit, you must properly fill in the chart, which shows incremental updates to the distance array. (Note: Each edge shown below is a directed edge.)

AB 12	BG 4	CG 4	EB 1	FE 1
AC 16	BC 1	DC 8	EG 8	FG 10
AD 5	CE 10	DF 4	FC 4	

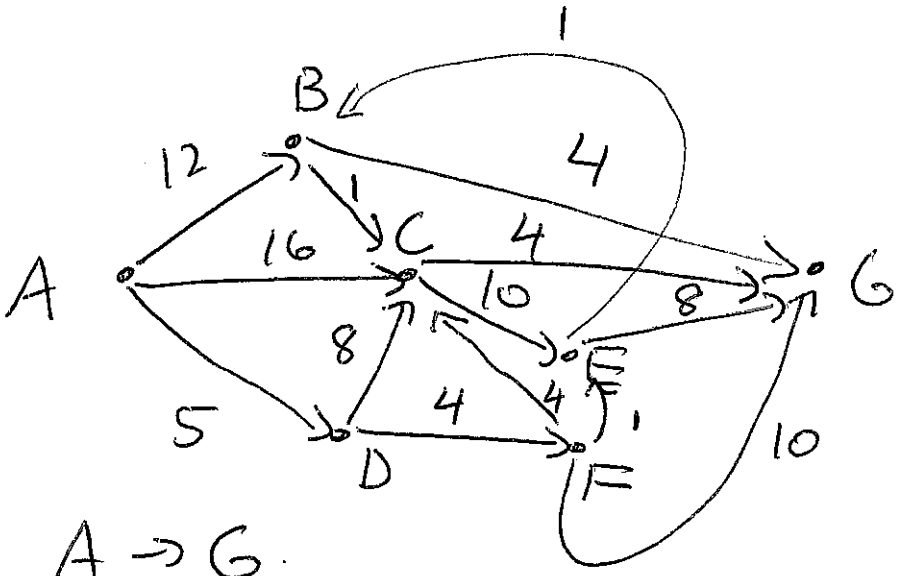
Add to S	A	B	C	D	E	F	G
A	0/A	12/A	16/A	5/A	∞	∞	∞
D	0/A	12/A	13/D	5/A	∞	9/D	∞
F	0/A	12/A	13/D	5/A	10/F	9/D	19/F
E	0/A	11/E	13/D	5/A	10/F	9/D	18/E
B	0/A	11/E	12/B	5/A	10/F	9/D	15/B
C	0/A	11/E	12/B	5/A	10/F	9/D	15/B
G							

5) (10 pts) Graphs – Breadth First Search

Show the order in which the vertices get visited during a breadth first search of the graph shown below, starting at vertex A. When enqueing multiple unvisited neighbors of a current node during the algorithm, always enqueue the neighbors in alphabetical order.

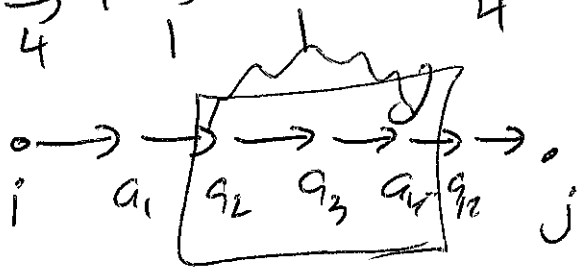


____ , ____ , ____ , ____ , ____ , ____ , ____ , ____ , ____ , ____ , ____ , ____ , ____ , ____ , ____ , ____



$A \to G$

$A \xrightarrow[5]{} D \xrightarrow[4]{} F \xrightarrow[1]{} E \to B \xrightarrow[4]{} G$



All subpaths in any shortest path is ALSO a shortest path

3/6/18

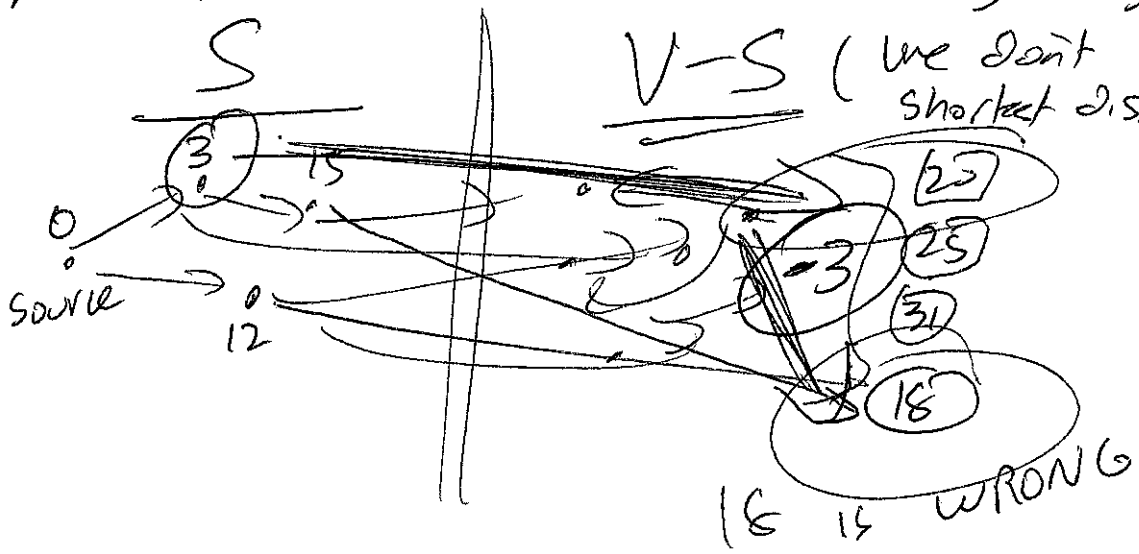
Prove Dst it works

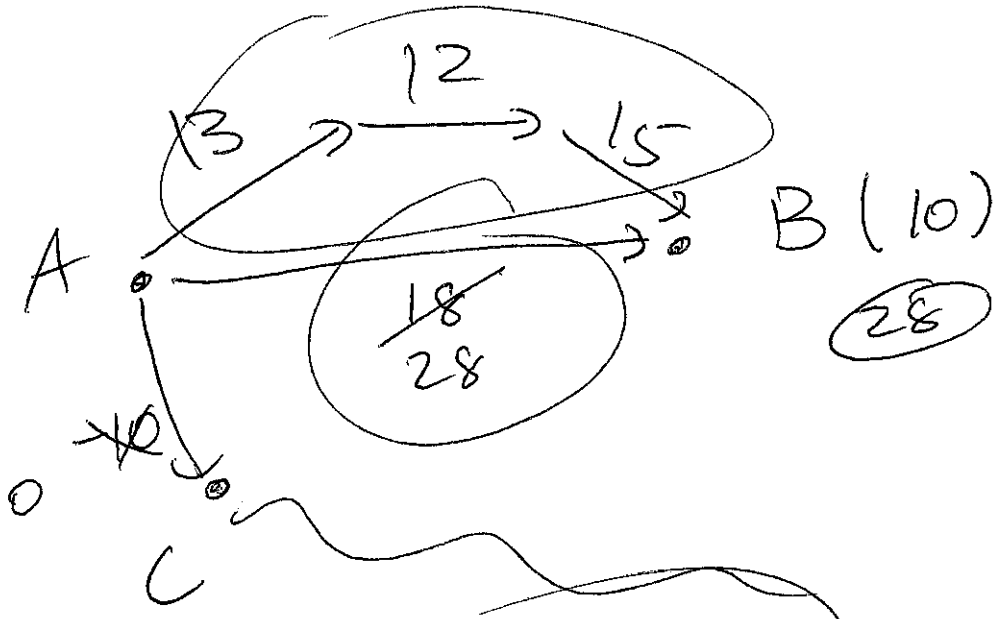
Key fact - NO EDGE WEIGHTS that are NEGATIVE

Pf by contradiction

assume the algorithm erroneously adds an item to set S.

When we add ~~s~~ source, we are safe.
Since there are no neg edge weights
impossible a→a (a to a) w/ neg weight.





NEG EDGE WEIGHTS

Bellman - FORD

Slow $V * V$ (double loop)

FAST $E \lg E$ $E \leq V^2$

Outer loop

poll PQ

$\log V^2$

$= 2 \log V$

$= O(\lg V)$

Run Time of efficient Dijkstra's

$E \lg E \leq O(E \lg V)$