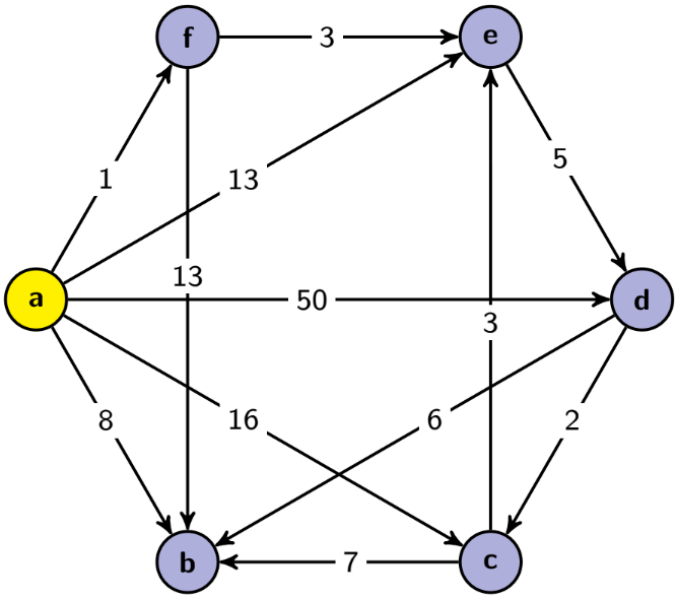
CSE 332: Data Structures and Parallelism

Section 7: Dijkstra

0. Velociraptors

Consider the following graph:



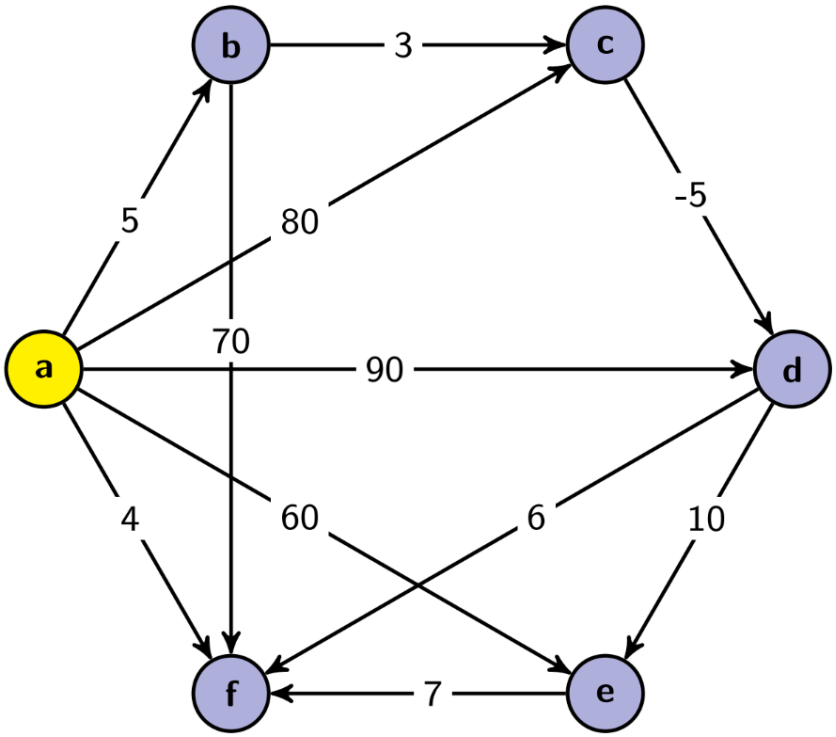
Suppose that you are at **a** and you are planning your escape from a bunch of hungry velociraptors (edge weights represent the expected number of velociraptors you will meet on this path). Run Dijkstra's Algorithm to **find the lengths of the shortest paths** (fewest number of velociraptors met) from **a** to each of the other vertices. Remember to store the path variable and **list the order vertices are added to the known set**.

Vertex	Visited?	Cost of Path	Pred
a	True	0	
b	True	~~∞~~ 8	a
c	True	~~∞~~ ~~16~~ 11	~~a~~ d
d	True	~~∞~~ ~~50~~ 9	~~a~~ e
e	True	~~∞~~ ~~13~~ 4	~~a~~ f
f	True	~~∞~~ 1	a

Order added to known set: a, f, e, b, d, c

1. Better Find the Shortest Path Before It Catches You!

Consider the following graph:



a) Use Dijkstra's Algorithm to find the lengths of the shortest paths from a to each of the other vertices. Show your work at every step.

Vertex	Visited?	Cost of Path	Pred
a	True	0	
b	True	~~∞~~ 05	a
c	True	~~∞~~ ~~80~~ 08	~~a~~ b
d	True	~~∞~~ ~~90~~ 03	~~a~~ c
e	True	~~∞~~ ~~60~~ 13	~~a~~ d
f	True	~~∞~~ 04	a

Order added to known set: a, f, b, c, d, e

b) Are any of the lengths you computed using Dijkstra's Algorithm in part (a) incorrect? Why or why not?

In this case, no. In general, Dijkstra's Algorithm does not necessarily work correctly with negative edge weights, but here, it actually returns the right result.

c) Explain how you would use Dijkstra's Algorithm to recover the actual paths (rather than just the lengths).

Keep an extra dictionary which maps nodes to their predecessors. (A predecessor is the node we took an edge from to get to the node.) Then, walk from the target vertex back toward the source vertex using the predecessor map.