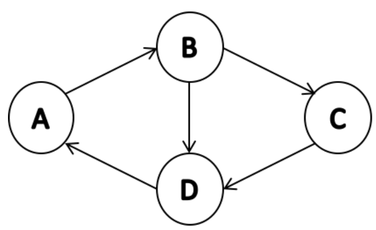
CS 225 Spring 2019 :: TA Lecture Notes
4/26 Floyd-Warshall's Algorithm

By Wenjie

Floyd-Warshall's Algorithm solves the problem Dijkstra's algorithm has with **negative** edges (not negative cycles, because those are mathematically undefined).

- Algorithm setup:
  - Maintain a table (matrix) that has the shortest known paths between vertices.
  - Initialize the table with three possible values:
    - self edges to 0
    - edges by edge weights
    - unknown paths to infinity

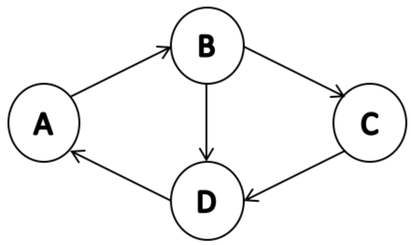


	A	B	C	D
A	0	-1	∞	∞
B	∞	0	4	3
C	∞	∞	0	-2
D	2	∞	∞	0

- Algorithm logic:
  - Consider adding every vertex to optimize the existing path:
    - $A \rightarrow B \quad -1$ vs. $A \rightarrow C \rightarrow B \quad \infty$
      $A \rightarrow D \rightarrow B \quad \infty$
    - $A \rightarrow C \quad \infty$ vs. $A \rightarrow B \rightarrow C \quad -1+3=2 \Rightarrow UPDATE$
      $A \rightarrow D \rightarrow C \quad \infty$
    - $A \rightarrow D \quad \infty$ vs. $A \rightarrow B \rightarrow D \quad -1+3=2 \Rightarrow UPDATE$
      $A \rightarrow C \rightarrow D \quad 3+(-2)=1 \Rightarrow UPDATE$

CS 225 Spring 2019 :: TA Lecture Notes
4/26 Floyd-Warshall's Algorithm

By Wenjie



	A	B	C	D
A	0	-1	2	1
B	∞	0	4	3
C	∞	∞	0	-2
D	2	∞	∞	0

- Now, do the same with the rest of the vertices (B, C, and D). At the end of the algorithm, we will have shortest paths for all pairs.

- Running time:
    - $O(n^3)$
    - With Dijkstra's algorithm we assumed optimality → once we find a path from A to B we do not try to find another path from A to B with shorter distance.
    - On the other hand Floyd-Warshall's algorithm explores all possible paths to determine the shortest path. If we explored all possible paths with Dijkstra's algorithm, the running time would have been much worse than n^3 .