CS 225

#41: Floyd-Warshall's Algorithm

December 1, 2021 · *G Carl Evans*

From Friday:

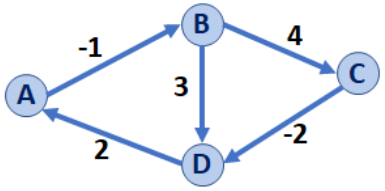
- Graphs with a negative-weight **cycle** have no finite shortest path. *(We can always take the cycle one more time to get an even shorter path!)*
- Graphs with a negative-weight **edge without a negative-weight cycle** DO have a finite shortest path!

Floyd-Warshall Algorithm

Floyd-Warshall's Algorithm is an alternative to Dijkstra in the presence of negative-weight edges (but <u>not</u> negative weight cycles).

Algorithm Design:

- **Goal:** Find the shortest path from vertex **u** to **v**.
- **Setup:** Create an n×n matrix that maintains the best known path between every pair of vertices:
  - Initialize (u, u) to 0.
  - Initialize all edges present on the graph to their edge weight.
  - Initialize all other edges to +infinity.



	A	B	C	D
A				
B				
C				
D				

- For every vertex **k**, consider which of the following are shorter:
  - path(u, v) - or -
  - path(u, **k**) + path(**k**, v)

Big Idea: _____

- Store intermediate results to improve build towards an optimal solution.
- Example application of memorization and **dynamic programming (DP)** – more in CS 374!

Running Time:

Pseudocode for Floyd-Warshall's Algorithm

```
1  FloydWarshall(G):
2    Input: G, Graph;
3    Output: d, an adjacency matrix of distances between
4              All vertex pairs
5
6    Let d be an adj. matrix (2d array) initialized to +inf
7    foreach (Vertex v : G):
8      d[v][v] = 0
9    foreach (Edge (u, v) : G):
10     d[u][v] = cost(u, v)
11
12   foreach (Vertex k : G):
13     foreach (Vertex u : G):
14       foreach (Vertex v : G):
15         if d[u, v] > d[u, k] + d[k, v]:
16           d[u, v] = d[u, k] + d[k, v]
17
18   return d
```

CS 225 – Things To Be Doing:

1. Last PotD released.
2. Final Project Midpoint due Friday
3. Rejoice that there is no Final Exam