#34: Floyd-Warshall's Algorithm

May 3, 2021 · *G Carl Evans*

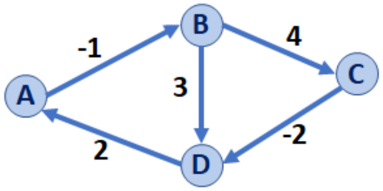
From Friday:

- 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 not 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

```
FloydWarshall(G):
  Input: G, Graph;
  Output: d, an adjacency matrix of distances between
             All vertex pairs

  Let d be an adj. matrix (2d array) initialized to +inf
  foreach (Vertex v : G):
    d[v][v] = 0
  foreach (Edge (u, v) : G):
    d[u][v] = cost(u, v)

  foreach (Vertex k : G):
    foreach (Vertex u : G):
      foreach (Vertex v : G):
        if d[u, v] > d[u, k] + d[k, v]:
          d[u, v] = d[u, k] + d[k, v]

  return d
```

CS 225 – Things To Be Doing:

1. Final Project due May 12th.
2. Rejoice that there is no Final Exam