CS 225

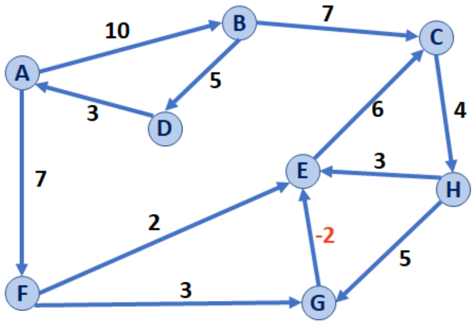
#42: Floyd-Warshall's Algorithm

April 29, 2019 · *Fagen-Ulmschneider, Zilles*

From Friday:

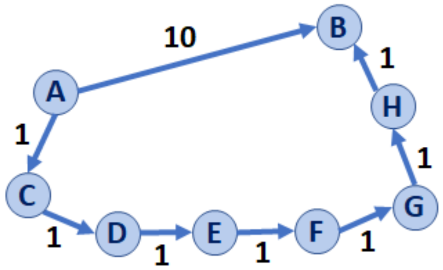
- Graphs with a negative-weight **cycle** have no finite shortest path. *(We can always take the cycle one more time to get an even shorter path!)*
- Graphs with a negative-weight **edge without a negative-weight cycle** DO have a finite shortest path!
- Does Dijkstra's algorithm find it?

Dijkstra: What if we have a minimum-weight edge, without having a negative-weight cycle?



...what assumption does Dijkstra's algorithm make?

Q: Can we transform the graph by adding **+k** to every edge?



Dijkstra: What is the running time?

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).

Intuition:

Consider a graph G with vertices V numbered 1 through N.

Consider the function shortestPath(i, j, k) that returns the shortest possible path from i to j using only vertices from the set {1,2, … ,k}.

Clearly, shortestPath(i, j, N) returns ____________________

For each pair of vertices, the shortestPath(i, j, k) could be either

(1) a path that **doesn't** go through k (only uses vertices in the set {1, …, k-1}.)

(2) a path that **does** go through k (from i to k and then from k to j, both only using intermediate vertices in {1, …, k-1}

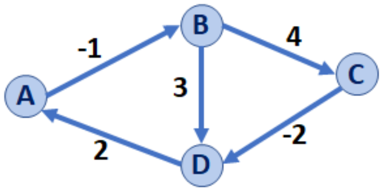
If w(i,j) is the weight of the edge between vertices i and j, we can recursively define shortestPath (i,j,k) as:

// base case
shortestPath(i, j, 0) =

// recursive
shortestPath(i, j, k) = min(

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)

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

  return d
```

Big Idea: ____________________

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

Overview of Graphs:

Implementations: Edge List, Adjacency Matrix, Adjacency List
Traversals: Breadth First, Depth First
Minimum Spanning Tree (MST): Kruskal's, Prim's Algorithm
Shortest Path:

- Dijkstra's Algorithm *(Single Source)*
- Floyd-Warshall's Algorithm *(All Pairs)*

Maximum Flow

- Ford-Fulkerson (DFS paths) Algorithm
- Edmonds-Karp (BFS paths) Algorithm

End of Semester :(

"Pre-Final" Grade Update

- As soon as possible after the MP7 deadline, we'll provide a "Pre-Final" grade update in Compass 2g with all grades except for your final exam.

End of Semester Grade Review

- Did we miss something that impacts your final grade? I want to be absolutely sure you get the grade you earned!
- After final grades are posted, I will provide a Google Sheet that allows you to submit a **Grade Review** if you believe the grade review will change your final letter grade.
  - You will have the chance to justify why you received an incorrect grade and how it impacts your letter grade in the course.
  - Instructions on Piazza at the same time as that the final grades are posted.

CS 225 – Things To Be Doing:

1. MP7 due tonight (April 29)
2. **In-lecture review w/ TAs** on Wednesday, May 1st
3. Final Exam starts Thursday, May 2nd