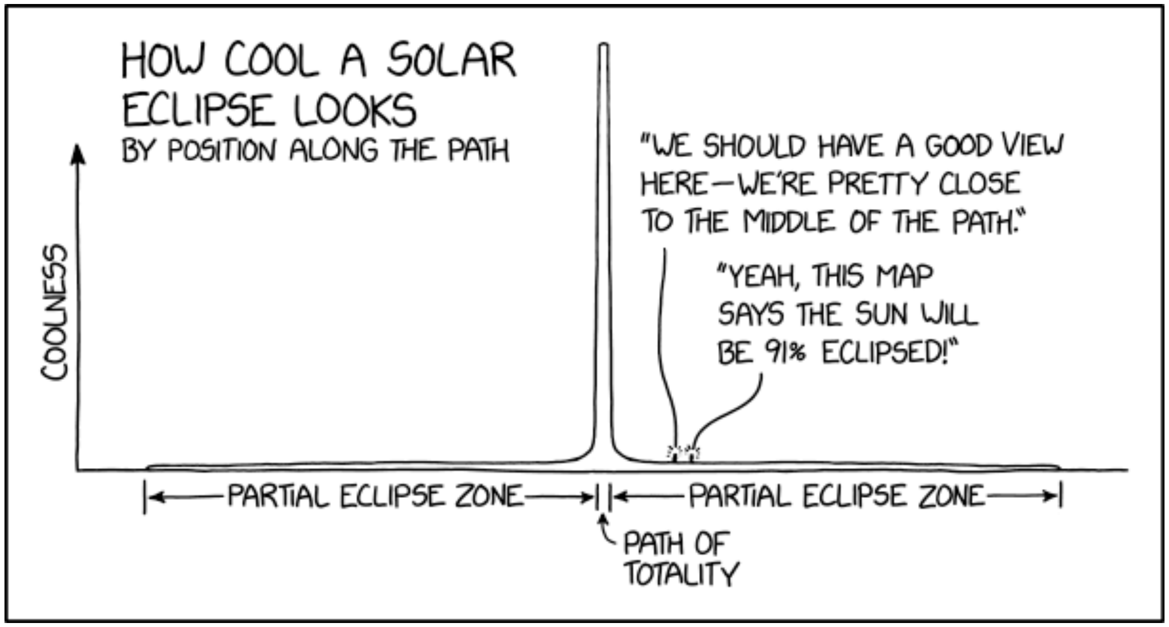
CS 225

Data Structures

April 10 – Floyd-Warshall's Algorithm

G Carl Evans

Monday


HOW COOL A SOLAR ECLIPSE LOOKS
BY POSITION ALONG THE PATH
COOLNESS
"WE SHOULD HAVE A GOOD VIEW HERE—WE'RE PRETTY CLOSE TO THE MIDDLE OF THE PATH."
"YEAH, THIS MAP SAYS THE SUN WILL BE 91% ECLIPSED!"
PARTIAL ECLIPSE ZONE
PARTIAL ECLIPSE ZONE
PATH OF TOTALITY

Exam 3 Review

You are working on an editor and have been asked to implement an undo feature. This feature should store all the edits that have been made to the file since the last time it was saved. You want to be able to save edits and remove edits if they have been undone. You can assume that there is some object that holds a single edit and you have to store that in your data structure.

Exam 3 Review

You are working for a financial system and are building a system to handle transactions that are happening. Transactions will all have a unique id number that will be used to track them. Users will be able to add, query the status of, and delete transactions. You can assume that the system you're running on has enough memory to store all transactions in memory.

Exam 3 Review

You are working on a small embedded system and have only a fixed amount of memory to work with. You need to build a system that buffers requests for information in case more come in than can be handled in a given time. The requests need to be handled in order that they are received. If more requests are received than you have space for you can ignore them.

Exam 3 Review

You are working on a storage system for a new social media site where you will store users' video game clips. You will need to handle the storage of 100s of clips for millions of users. The users will be able to add and search for clips but not remove them. You will not be able to fit everything in memory.

Exam 3 Review

You are working on a logging system which logs transactions by time stamp for a business. You will need to handle adding transactions as quickly as possible on average this is the most important constraint. You will also need to allow people to look up the transactions between two times.

Floyd-Warshall Algorithm

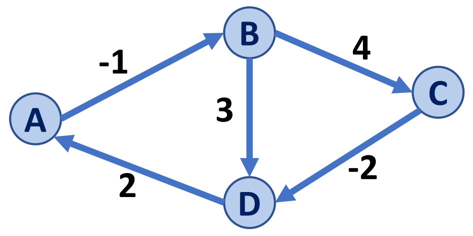
Floyd-Warshall's Algorithm is an alterative to Dijkstra in the presence of negative-weight edges (not negative weight cycles).

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

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

Floyd-Warshall Algorithm

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

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

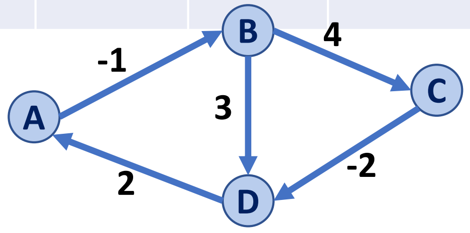
	A	B	C	D
A				
B				
C				
D				



Floyd-Warshall Algorithm

```
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]
```

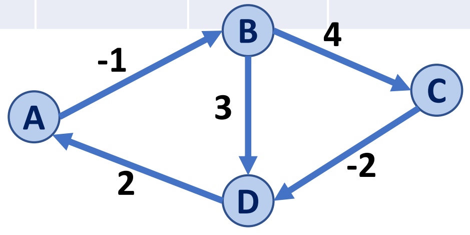
	A	B	C	D
A	0	-1	∞	∞
B	∞	0	4	3
C	∞	∞	0	-2
D	2	∞	∞	0



Floyd-Warshall Algorithm

```
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]
```

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



Let us consider k=A:

$B \rightarrow C$ **4** vs. $B \rightarrow A \rightarrow C$ **+∞**

$B \rightarrow D$ **3** vs. $B \rightarrow A \rightarrow D$ **+∞**

$C \rightarrow B$ **+∞** vs. $C \rightarrow A \rightarrow B$ **+∞**

$C \rightarrow D$ **-2** vs. $C \rightarrow A \rightarrow D$ **+∞**

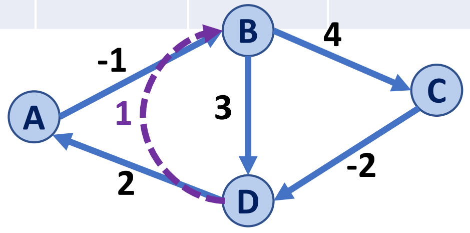
$D \rightarrow B$ **+∞** vs. $D \rightarrow A \rightarrow B$

$D \rightarrow C$ **+∞** vs. $D \rightarrow A \rightarrow C$

Floyd-Warshall Algorithm

```
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]
```

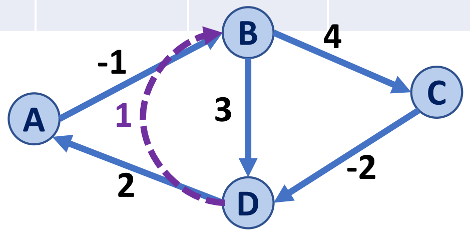
	A	B	C	D
A	0	-1	∞	∞
B	∞	0	4	3
C	∞	∞	0	-2
D	2	1	∞	0



Floyd-Warshall Algorithm

```
  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]
```

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



Let us consider k=B:

$A \rightarrow C$ vs. $A \rightarrow B \rightarrow C$

$A \rightarrow D$ vs. $A \rightarrow B \rightarrow D$

$C \rightarrow A$ vs. $C \rightarrow B \rightarrow A$

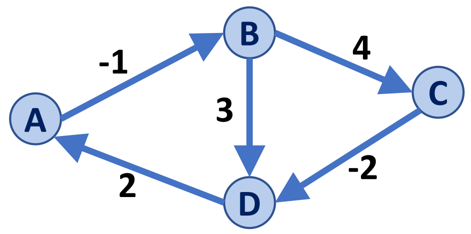
$C \rightarrow D$ vs. $C \rightarrow B \rightarrow D$

$D \rightarrow A$ vs. $D \rightarrow B \rightarrow A$

$D \rightarrow C$ vs. $D \rightarrow B \rightarrow C$

Floyd-Warshall Algorithm

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

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



	A	B	C	D
A				
B				
C				
D				

	A	B	C	D
A				
B				
C				
D				

Floyd-Warshall Algorithm

Running Time?

```
FloydWarshall(G):
  Let d be a adj. matrix 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 w : G):
        if d[u, v] > d[u, w] + d[w, v]:
          d[u, v] = d[u, w] + d[w, v]
```

Deterministic Data Structures

List

ADT

- Insert
- Remove
- Access
- IsEmpty

Trees

ADT?

- Insert
- Find
- Traversal

- Binary
- Binary Search
- Balanced Binary Search
- Btree

Special Trees

- kDTree
- Huffman Tree
- Heap
- Up Trees

Graphs

- Edge List
- Adjacency Matrix
- Adjacency List

Graph Algorithms

- Traversal
- MST
- Shortest Path