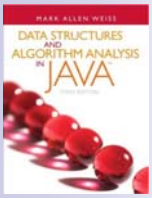
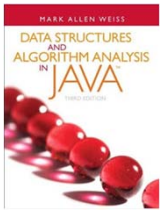


FIU FLORIDA INTERNATIONAL UNIVERSITY

Some Problems on Graphs (III)

Dr. Antonio L. Bajuelos

FIU School of Computing & Information Sciences



COP-3530 - Data Structures

Module #7: Some Problems on Graphs (part III)

Outline:

- **Shortest-Path Problem:**
  - **Dijkstra's Algorithm**

Single-Source Shortest-Path Problem.

- Let a **weighted graph** (i.e. associated with each edge (v_i,v_j) is a cost $c_{i,j}$ to traverse the edge).
- The **cost of a path** $v_1v_2 \ldots v_N$ is $\sum_{i=1}^{N-1} c_{i,i+1}$. This is referred to as the **weighted path length**.
- For the **unweighted path length** is merely the number of edges on the path, namely, N – 1.

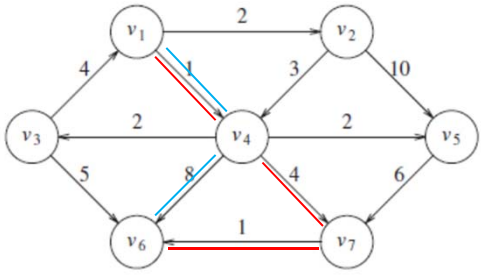
Single-Source Shortest-Path Problem.

Given a **weighted graph**, G = (V,E), and a distinguished vertex, s, find the **shortest weighted path** from s to every other vertex in G.

3

Single-Source Shortest-Path Problem.

- Example:

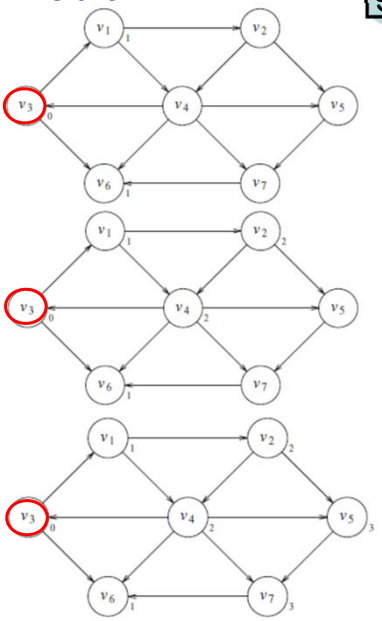


- The **shortest weighted path** from v_1 to v_6 has a cost of 6 and goes from v_1 to v_4 to v_7 to v_6 (red path).
- The **shortest unweighted path** between these vertices is 2 (blue path).
- Note: For the **shortest unweighted path** we are only interested in the number of edges contained on the path. This is clearly a special case of the **weighted shortest-path problem**, since we could assign all edges a weight of 1.

4

Shortest Unweighted Path Problem.

- Suppose we choose start vertex, *s*, to be v_3. The shortest path from *s* to v_3 is then a path of length 0.
- Then we can start looking for all vertices that are a distance 1 away from *s* (by looking at the vertices that are adjacent to *s*).
- Next: find vertices whose shortest path from *s* is exactly 2, (by finding all the vertices adjacent to v_1 and v_6), whose shortest paths are not already known. This search tells us that the shortest path (from *s*) to v_2 and v_4 is 2.
- etc..



Shortest-Path Problem. Some Applications

- Driving directions
- Cheap flight itineraries
- Network routing
- Critical paths in project management

6

Shortest Weighted Path. Dijkstra's Algorithm

- **Inventor: Edsger Dijkstra** (1930-2002), one of the founders of Computer Science; this is just one of his many contributions.

- **The Problem:**
  - Given a **digraph with non-negative edge weights**, G=(V,E), and a distinguished source vertex s in V.
  - Determine the distance and a **shortest path** from the source vertex to every vertex in the digraph.

- **Question:**
  - How do you design an **efficient algorithm** for this problem?

Edsger Dijkstra:

http://amturing.acm.org/award_winners/dijkstra_1053701.cfm

7

Dijkstra's Algorithm

- **Basic Ideas:**
  - Each vertex is marked as either ***known*** or ***unknown***. A tentative distance d_v is kept for each vertex.
  - d_v turns out to be the shortest path length from *s* to *v* using only *known* vertices as intermediates.
  - We record p_v, which is the last vertex to cause a change to d_v.
  - Its is a **greedy algorithm** i.e., at each stage, **Dijkstra's algorithm** selects a vertex *v*, which has the smallest d_v among all the *unknown* vertices, and declares that the shortest path from *s* to *v* is *known.*

8

Dijkstra's Algorithm

- **Pseudocode:**

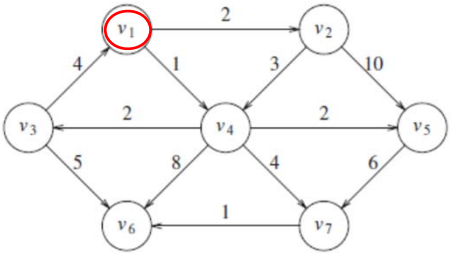
1. For each node **v**, set **v.dist =** ∞ and **v.known = false**
2. Set **source.dist = 0**
3. While there are unknown nodes in the graph
   a) Select the unknown node **v** with lowest distance
   b) Mark **v** as known
   c) For each edge **(v,u)** with weight **w**,

```
c1 = v.dist + w // distance of best path through v to u
c2 = u.cost     // distance of best path to u previously known
if(c1 < c2)     // if the path through v is better /shorter
 {
   u.dist = c1
    u.path = v  // for computing actual paths
 }
```

9

Dijkstra's Algorithm

- **Example:**



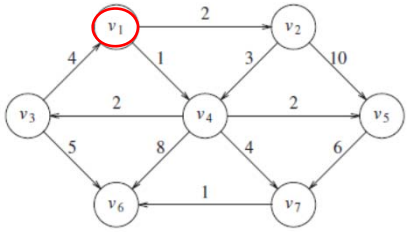
Initial Configuration

v	*known*	d_v	p_v
v_1	F	0	0
v_2	F	∞	0
v_3	F	∞	0
v_4	F	∞	0
v_5	F	∞	0
v_6	F	∞	0
v_7	F	∞	0

10

Dijkstra's Algorithm

- **Example:**



Initial Configuration

v	known	d_v	p_v
v_1	F	0	0
v_2	F	∞	0
v_3	F	∞	0
v_4	F	∞	0
v_5	F	∞	0
v_6	F	∞	0
v_7	F	∞	0

After v_1 is declared known

v	known	d_v	p_v
v_1	T	0	0
v_2	F	2	v_1
v_3	F	∞	0
v_4	F	1	v_1
v_5	F	∞	0
v_6	F	∞	0
v_7	F	∞	0

After v_4 is declared known

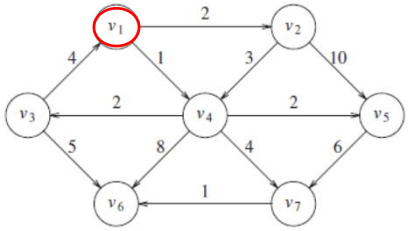
v	known	d_v	p_v
v_1	T	0	0
v_2	F	2	v_1
v_3	F	3	v_4
v_4	T	1	v_1
v_5	F	3	v_4
v_6	F	9	v_4
v_7	F	5	v_4

Dijkstra's Algorithm

- **Example:**

After v_4 is declared known

v	known	d_v	p_v
v_1	T	0	0
v_2	F	2	v_1
v_3	F	3	v_4
v_4	T	1	v_1
v_5	F	3	v_4
v_6	F	9	v_4
v_7	F	5	v_4



After v_2 is declared known

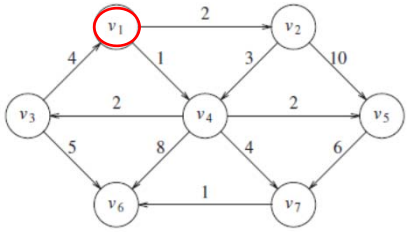
v	known	d_v	p_v
v_1	T	0	0
v_2	T	2	v_1
v_3	F	3	v_4
v_4	T	1	v_1
v_5	F	3	v_4
v_6	F	9	v_4
v_7	F	5	v_4

After v_5 and then v_3 are declared known

v	known	d_v	p_v
v_1	T	0	0
v_2	T	2	v_1
v_3	T	3	v_4
v_4	T	1	v_1
v_5	T	3	v_4
v_6	F	8	v_3
v_7	F	5	v_4

Dijkstra's Algorithm

- **Example:**



After v_5 and then v_3 are declared known

v	known	d_v	p_v
v_1	T	0	0
v_2	T	2	v_1
v_3	T	3	v_4
v_4	T	1	v_1
v_5	T	3	v_4
v_6	F	8	v_3
v_7	F	5	v_4

After v_7 is declared known

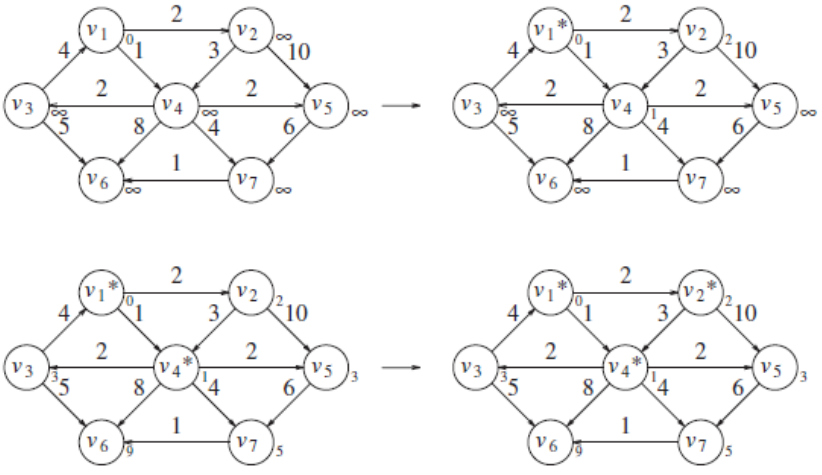
v	known	d_v	p_v
v_1	T	0	0
v_2	T	2	v_1
v_3	T	3	v_4
v_4	T	1	v_1
v_5	T	3	v_4
v_6	F	6	v_7
v_7	T	5	v_4

After v_6 is declared known

v	known	d_v	p_v
v_1	T	0	0
v_2	T	2	v_1
v_3	T	3	v_4
v_4	T	1	v_1
v_5	T	3	v_4
v_6	T	6	v_7
v_7	T	5	v_4

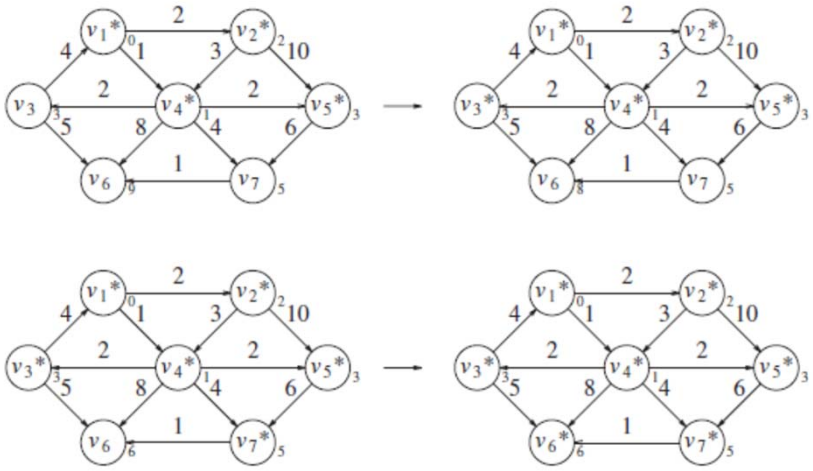
Dijkstra's Algorithm

- **Example: Stages of Dijkstra's Algorithm**



Dijkstra's Algorithm

- **Example: Stages of Dijkstra's Algorithm**



Dijkstra's Algorithm

- **Efficiency, first approach**

Dijkstra(Graph G, Node start)

```
for each node:
  x.dist=infinity, x.known=false, start.dist = 0
```
O(|V|)

```
while(not all nodes are known) {
  b = find unknown node with smallest dist
  b.known = true
```
O(|V|²)

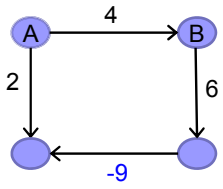
```
  for each edge (b,a) in G
   if(!a.known)
     if(b.dist + weight((b,a)) < a.dist){
       a.dist = b.dist + weight((b,a))
       a.path = b
      }
}
```
O(|E|)

Total Running Time Complexity: O(|V|²)

Homework: How to improve the **O(|V|²)** of the **Dijkstra's Algorithm**?
Hint: use a MinHeap to store the vertices for which shortest distance is not finalized yet.
Recommended Link: https://www.youtube.com/watch?v=zXfDYaahsNA

Dijkstra's Algorithm

- Shortest paths with **negative weights**: **failed examples!**



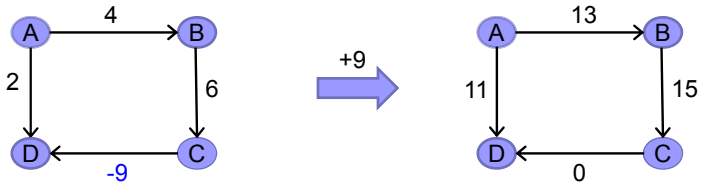
Dijkstra's Algorithm

Dijkstra Solution:
Shortest path (A,D) = [A,D] = 2

Correct solution:
Shortest path (A,D) = [A,B,C,D] = 1 **!!!**

Dijkstra's Algorithm

- **Re-weighting**. Adding a constant to every edge weight also doesn't work.



Dijkstra's Algorithm

A	F 0	T 0	T 0	T 0	T 0
B	F ∞	F 13	F 13	**T 13**	**T 13**
C	F ∞	F ∞	F ∞	F 28	**T 28**
D	F ∞	F 11	**T 11**	**T 11**	**T 11**

Dijkstra Solution:
Shortest path (A,D) = [A,D] = 11

Correct solution (in the original graph):
Shortest path (A,D) = [A,B,C,D] = 1 **!!!**

Bad news: Need a different algorithm!!!

Dijkstra's Algorithm

- For a full version of the Dikstra's algorithm implementation please consult:

  https://users.cs.fiu.edu/~weiss/dsj2/code/Graph.java

  (**Author: Mark Weiss**)

19