SE284: Introduction to Graph Algorithms

Katerina Taškova

Email: katerina.taskova@auckland.ac.nz
Room: 303S.483

Outline

Weighted (di)graphs, Dijkstra's algorithm

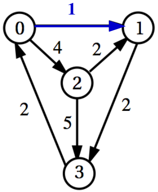
Lecture Notes 29, Textbook 6.1-3

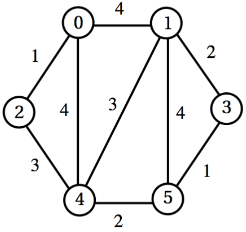
Acknowledgment for slide content: Michael Dinneen, Simone Linz

Weighted (di)graphs

- Very common in applications, also called “networks". Optimization problems on networks are important in operations research.
- Each arc carries a real number “weight", usually positive, can be $+\infty$. Weight typically represents cost, distance, time.
- Representation: weighted adjacency matrix or double adjacency list.
- Standard problems concern finding a minimum or maximum weight path between given nodes (covered here), spanning tree (covered here), cycle or tour (e.g travel salesman problem), matching, flow, etc.

Computer representations of weighted digraphs



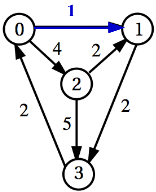


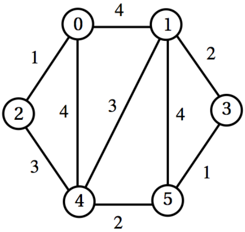
Cost Matrices:

$$\begin{bmatrix} 0 & 1 & 4 & 0 \\ 0 & 0 & 0 & 2 \\ 0 & 2 & 0 & 5 \\ 2 & 0 & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 4 & 1 & 0 & 4 & 0 \\ 4 & 0 & 0 & 2 & 3 & 4 \\ 1 & 0 & 0 & 0 & 3 & 0 \\ 0 & 2 & 0 & 0 & 0 & 1 \\ 4 & 3 & 3 & 0 & 0 & 2 \\ 0 & 4 & 0 & 1 & 2 & 0 \end{bmatrix}$$

Computer representations of weighted digraphs





Weighted (Double) Adjacency Lists:

1	1	2	4
3	2		
1	2	3	5
0	2		

1	4	2	1	4	4		
0	4	3	2	4	3	5	4
0	1	4	3				
1	2	5	1				
0	4	1	3	2	3	5	2
1	4	3	1	4	2		

Computer representations of weighted digraphs – example

Example 29.3. Draw the weighted graph given by the weighted matrix below.

$$\begin{bmatrix} 0 & 3 & 4 & 0 \\ 3 & 0 & 1 & 3 \\ 4 & 1 & 0 & 2 \\ 0 & 3 & 2 & 0 \end{bmatrix}$$

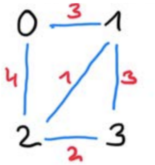
Draw the weighted digraph given by the weighted list representation below.

$$\begin{array}{c|cccc} 0 & 1 & 3 & 2 & 4 \\ 1 & 0 & 2 & 3 & 2 \\ 2 & 1 & 3 & & \\ 3 & 2 & 1 & & \end{array}$$

Computer representations of weighted digraphs – example

Example 29.3. Draw the weighted graph given by the weighted matrix below.

$$\begin{array}{c}0\\1\\2\\3\end{array}\begin{bmatrix}0 & 3 & 4 & 0\\3 & 0 & 1 & 3\\4 & 1 & 0 & 2\\0 & 3 & 2 & 0\end{bmatrix}$$



Draw the weighted digraph given by the weighted list representation below.

0	1	3	2	4
1	0	2	3	2
2	1	3		
3	2	1		

Paths/Distances – revisited

Definition

For a digraph (V, E) with edge weights $\{c(u, v) \mid (u, v) \in E\}$ we say that the distance $d(u, v)$ between two vertices u and v of V is the minimum cost of a path between u and v.

The cost (or weight) of a walk/path $v_0, v_1, \ldots, v_k$ is $d(v_0, v_k) = \sum_{i=0}^{k-1} c(v_i, v_{i+1})$.

If a path/walk from u to v does not exist, then d is undefined $(+\infty)$.

Definition

The diameter of a digraph $G = (V, E)$ is the maximum of $d(u, v)$ over all pairs $u, v \in V$. If the digraph is not strongly connected, the diameter of G is not defined $(+\infty)$.

Note: there are analogous definitions for graphs.

Paths/Distances – revisited

Definition

The eccentricity of a node u in V is the maximum of $d(u, v)$ over all $v \in V$.

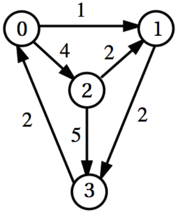
If there exist v such that a path from u to v does not exist, then the eccentricity of u is undefined ($+\infty$).

Definition

The radius of a digraph $G = (V, E)$ is the minimum eccentricity of nodes in V.

Note: there are analogous definitions for graphs.

Diameter/Radius - example

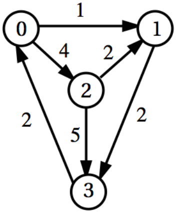


weighted adjacency matrix:

$$\begin{bmatrix} 0 & 1 & 4 & 0 \\ 0 & 0 & 0 & 2 \\ 0 & 2 & 0 & 5 \\ 2 & 0 & 0 & 0 \end{bmatrix}$$

We need to calculate the distance matrix first.

Diameter/Radius - example



4 = 0 to 2
8 = 1 to 2
1st 6 = 2 to 0
2nd 6 = 3 to 2

weighted adjacency matrix:

$$\begin{bmatrix} 0 & 1 & 4 & 0 \\ 0 & 0 & 0 & 2 \\ 0 & 2 & 0 & 5 \\ 2 & 0 & 0 & 0 \end{bmatrix}$$

distance matrix:

$$\begin{bmatrix} 0 & 1 & 4 & 3 \\ 4 & 0 & 8 & 2 \\ \mathbf{6} & 2 & 0 & \mathbf{4} \\ 2 & 3 & 6 & 0 \end{bmatrix}$$

Hence, the diameter is 8, and the radius is $\mathrm{min}\{4, 8, 6, 6\} = 4$
$d(2, 3) = c(2, 1) + c(1, 3)$ (not $c(2, 3)$)
$d(2, 0) = c(2, 1) + c(1, 3) + c(3, 0)$ (not $c(2, 3) + c(3, 0)$)

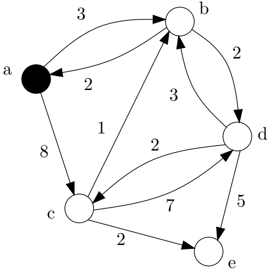
Single-source shortest path problem

- Given an originating node v, find shortest (minimum weight) path to each other node. If all weights are equal then BFS works, otherwise not.
- Several algorithms are known; we present one, Dijkstra's algorithm. An example of a greedy algorithm; locally best choice is globally best. Doesn't work if weights can be negative.
- Maintain list S of visited nodes (say using a priority queue). Choose closest unvisited node u that is on a path with internal nodes in S. Update distances (of remaining unvisited nodes) from source in case adding u has established shorter paths. Repeat.
- Complexity depends on data structures used, especially for priority queue; $O(m + n \log n)$ is possible.

Dijkstra's algorithm

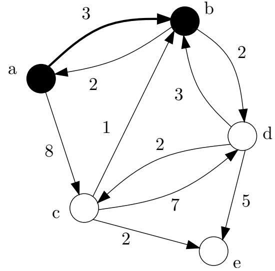
```
function DIJKSTRA(weighted digraph (G, c); node s ∈ V(G))
    array colour[0..n − 1], dist[0..n − 1]
    for u ∈ V(G) do
        dist[u] ← c[s, u]; colour[u] ← WHITE
    dist[s] ← 0; colour[s] ← BLACK
    while there is a white node do
        find a white node u so that dist[u] is minimum
        colour[u] ← BLACK
        for x ∈ V(G) do
            if colour[x] = WHITE then
                dist[x] ← min{dist[x], dist[u] + c[u, x]}
    return dist
```

Illustrating Dijkstra's algorithm



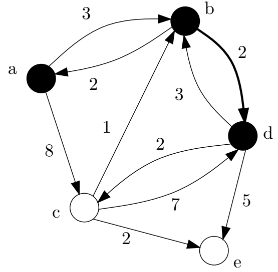
BLACK	$dist[x]$
	a,b,c,d,e
a	$0, 3, 8, \infty, \infty$
a,b	$0, 3, 8, 3+2=5, \infty$
a,b,d	$0, 3, 3+2+2=7, 5, 3+2+5=10$
a,b,c,d	$0, 3, 7, 5, 7+2=9$
$V(G)$	

Illustrating Dijkstra's algorithm



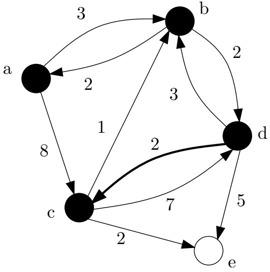
BLACK	*dist*[*x*]
	a,b,c,d,e
a	$0, 3, 8, \infty, \infty$
a,b	$0, 3, 8, 3+2=5, \infty$
a,b,d	$0, 3, 3+2+2=7, 5, 3+2+5=10$
a,b,c,d	$0, 3, 7, 5, 7+2=9$
$V(G)$	

Illustrating Dijkstra's algorithm



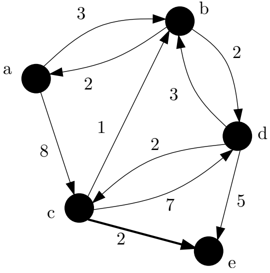
BLACK	$dist[x]$
	a,b,c,d,e
a	$0, 3, 8, \infty, \infty$
a,b	$0, 3, 8, 3+2=5, \infty$
a,b,d	$0, 3, 3+2+2=7, 5, 3+2+5=10$
a,b,c,d	$0, 3, 7, 5, 7+2=9$
$V(G)$	

Illustrating Dijkstra's algorithm



BLACK	$dist[x]$
	a,b,c,d,e
a	$0, 3, 8, \infty, \infty$
a,b	$0, 3, 8, 3+2=5, \infty$
a,b,d	$0, 3, 3+2+2=7, 5, 3+2+5=10$
a,b,c,d	$0, 3, 7, 5, 7+2=9$
$V(G)$	

Illustrating Dijkstra's algorithm

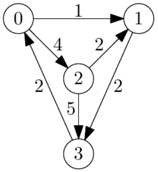


BLACK	$dist[x]$
	a,b,c,d,e
a	$0, 3, 8, \infty, \infty$
a,b	$0, 3, 8, 3+2=5, \infty$
a,b,d	$0, 3, 3+2+2=7, 5, 3+2+5=10$
a,b,c,d	$0, 3, 7, 5, 7+2=9$
$V(G)$	

Dijkstra's algorithm – examples

Example 29.11.

An application of Dijkstra's algorithm on the digraph below for each starting vertex s. Complete the table for the starting vertex 2.



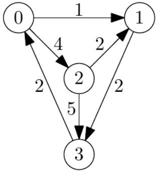
The table illustrates that the distance vector is updated at most $n-1$ times (only before a new vertex is selected and added to S). Thus we could have omitted the lines with $S = \{0, 1, 2, 3\}$.

current $S \subseteq V$	**distance vector** `dist`
$\{0\}$	$0, 1, 4, \infty$
$\{0, 1\}$	$0, 1, 4, 3$
$\{0, 1, 3\}$	$0, 1, 4, 3$
$\{0, 1, 2, 3\}$	$0, 1, 4, 3$
$\{1\}$	$\infty, 0, \infty, 2$
$\{1, 3\}$	$4, 0, \infty, 2$
$\{0, 1, 3\}$	$4, 0, 8, 2$
$\{0, 1, 2, 3\}$	$4, 0, 8, 2$
$\{2\}$	
$\{\quad\}$	
$\{\quad\quad\}$	
$\{0, 1, 2, 3\}$	
$\{3\}$	$2, \infty, \infty, 0$
$\{0, 3\}$	$2, 3, 6, 0$
$\{0, 1, 3\}$	$2, 3, 6, 0$
$\{0, 1, 2, 3\}$	$2, 3, 6, 0$

Dijkstra's algorithm – examples

Example 29.11.

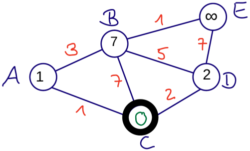
An application of Dijkstra's algorithm on the digraph below for each starting vertex s. Complete the table for the starting vertex 2.



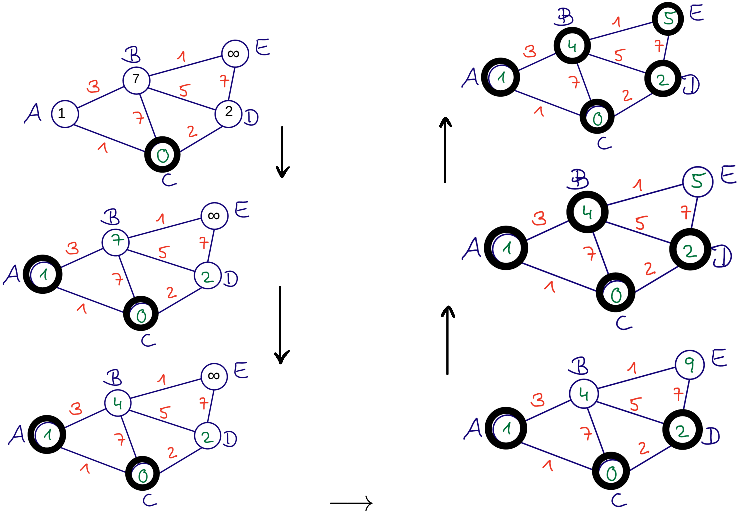
The table illustrates that the distance vector is updated at most $n-1$ times (only before a new vertex is selected and added to S). Thus we could have omitted the lines with $S = \{0, 1, 2, 3\}$.

current $S \subseteq V$	**distance vector** `dist`
$\{0\}$	$0, 1, 4, \infty$
$\{0, 1\}$	$0, 1, 4, 3$
$\{0, 1, 3\}$	$0, 1, 4, 3$
$\{0, 1, 2, 3\}$	$0, 1, 4, 3$
$\{1\}$	$\infty, 0, \infty, 2$
$\{1, 3\}$	$4, 0, \infty, 2$
$\{0, 1, 3\}$	$4, 0, 8, 2$
$\{0, 1, 2, 3\}$	$4, 0, 8, 2$
$\{2\}$	∞, **2**, 0, 5
{ **1**, 2 }	∞, 2, 0, 2+2=**4**
{1, 2, **3**}	4+2=**6**, 2, 0, 4
{**0**, 1, 2, 3}	6, 2, 0, 4
$\{3\}$	$2, \infty, \infty, 0$
$\{0, 3\}$	$2, 3, 6, 0$
$\{0, 1, 3\}$	$2, 3, 6, 0$
$\{0, 1, 2, 3\}$	$2, 3, 6, 0$

Dijkstra's algorithm – examples



Dijkstra’s algorithm – examples



Thank you!