Graph Primitives

Dijkstra's Algorithm: The Basics

Design and Analysis of Algorithms I

Single-Source Shortest Paths

Input: directed graph G=(V, E). (m=|E|, n=|V|)
- each edge has non negative length l_e
- source vertex s

**Length of path
= sum of edge lengths**

Output: for each $v \in V$, compute
   L(v) := length of a shortest s-v path in G

Path length = 6

Assumption:
1. [for convenience] $\forall v \in V, \exists s \Rightarrow v \text{ path}$
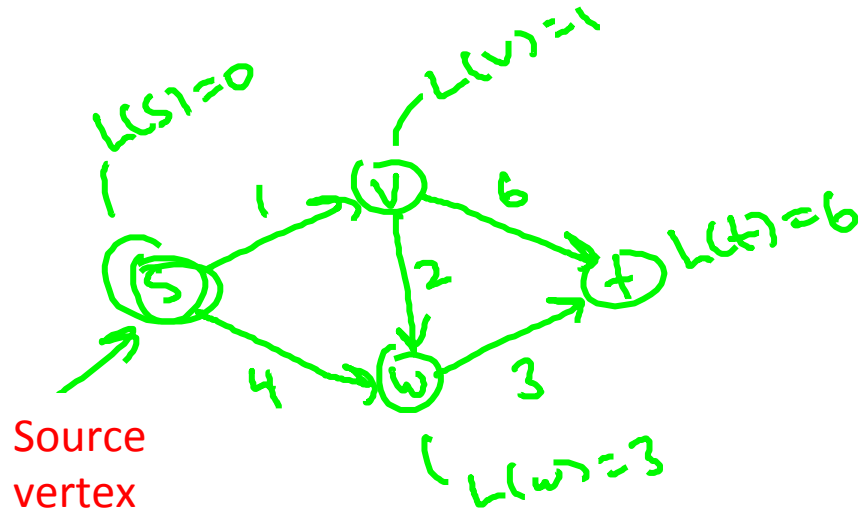2. [important] $le \geq 0 \ \forall e \in E$
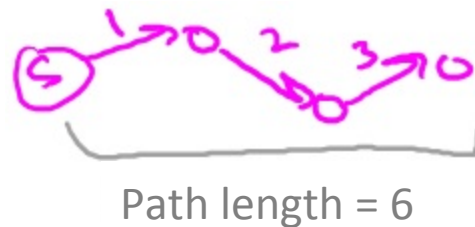
One of the following is the list of shortest-path distances for the nodes s,v,w,t, respectively. Which is it?

○ 0,1,2,3

○ 0,1,4,7

○ 0,1,4,6

○ 0,1,3,6

Source vertex

$L(s)=0$

$L(v)=1$

$L(t)=6$

$L(w)=3$

1

6

2

5
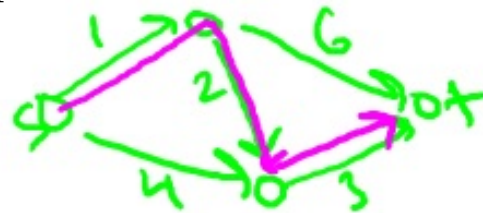
4

3

Why Another Shortest-Path Algorithm?

Question: doesn't BFS already compute shortest paths in linear time?

Answer: yes, IF $l_e = 1$ for every edge e.

Question: why not just replace each edge e by directed path of l_e unit length edges:

Answer: blows up graph too much

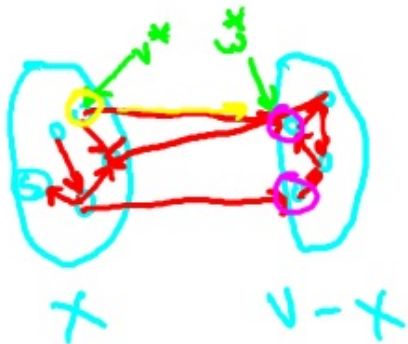Solution: Dijkstra's shortest path algorithm.

Dijkstra's Algorithm

<u>Initialize:</u>
- X = [s]   [vertices processed so far]
- A[s] = 0 [computed shortest path distances]
- B[s] = empty path [computed shortest paths]

<u>Main Loop</u>
- while X≠V:

-need to grow x by one node

<u>Main Loop cont'd:</u>
- among all edges $(v, w) \in E$
  with $v \in X, w \notin X$,
  pick the one that minimizes
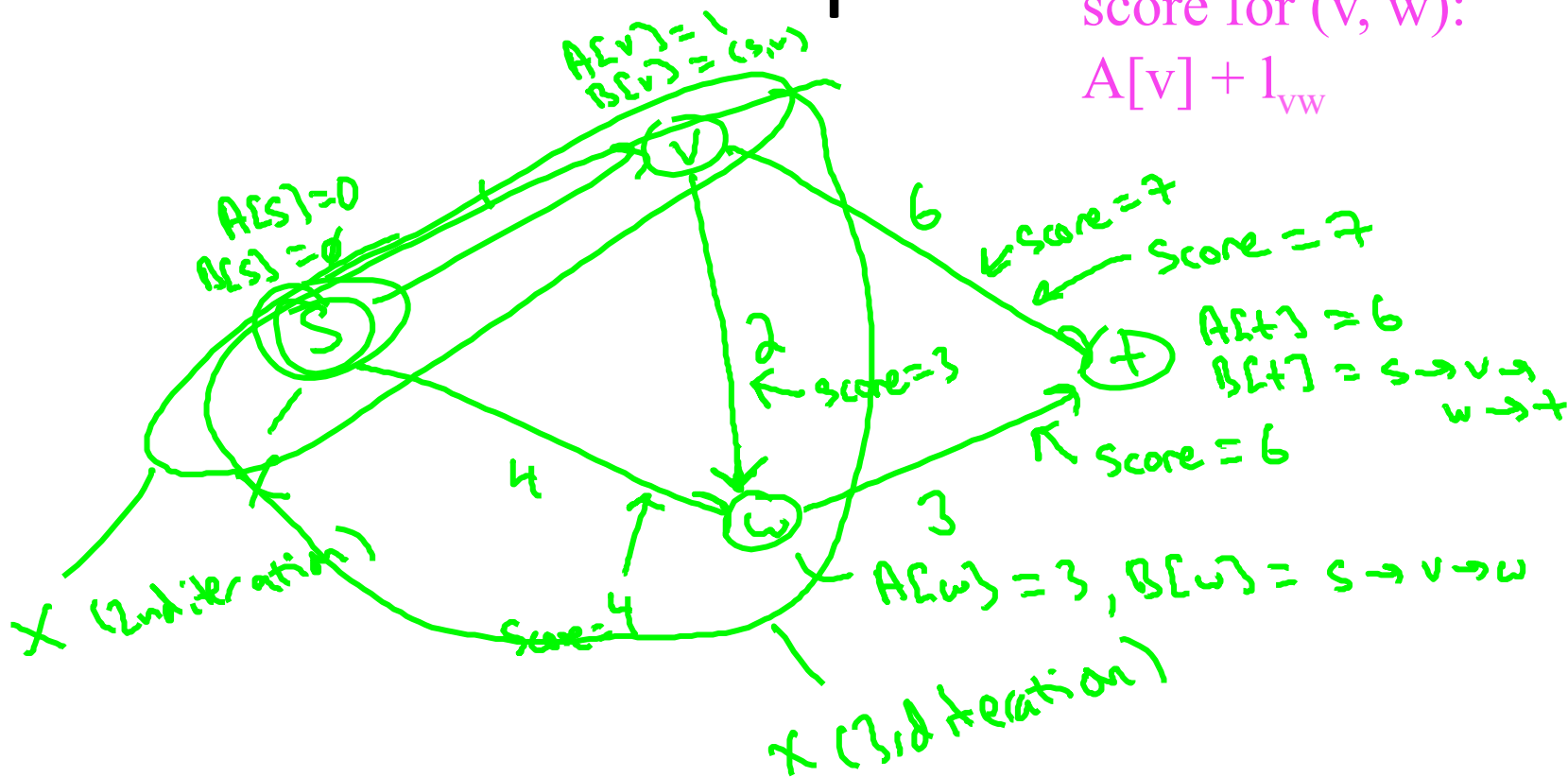 $A[v] + l_{vw}$

[call it (v*, w*)]

Already computed in earlier iteration

- add w* to X
- set $A[w^*] := A[v^*] + l_{v^*w^*}$
- set $B[w^*] := B[v^*]u(v^*, w^*)$

Tim Roughgarden

Example

Dijkstra's greedy score for (v, w):
$$A[v] + l_{vw}$$

A[v] = 1
B[v] = (s,v)

A[s] = 0
B[s] = ∅

6

score = 7
score = 7

A[t] = 6
B[t] = s → v → w → t

2

score = 3

score = 6

4

A[w] = 3, B[w] = s → v → w

X (2nd iteration)

score = 4

3

X (3rd iteration)

Non-Example

Question: why not reduce computing shortest paths with negative edge lengths to the same problem with non negative lengths? (by adding large constant to edge lengths)

Problem: doesn't preserve shortest paths !

Also: Dijkstra's algorithm incorrect on this graph !
(computes shortest s-t distance to be -2 rather than -4)