Design and Analysis
of Algorithms I

Graph Primitives

Dijkstra's Algorithm: The Basics

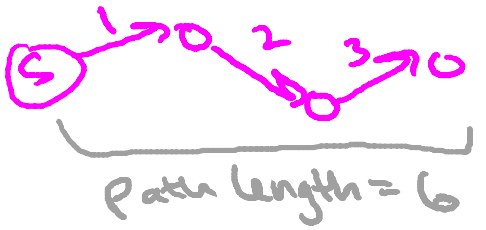
Single-Source Shortest Paths

Input: directed graph $G = (V, E)$. ($m = |E|$, $n = |V|$)

- each edge has nonnegative length ℓ_e
- source vertex s

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

(length of path = sum of edge lengths)



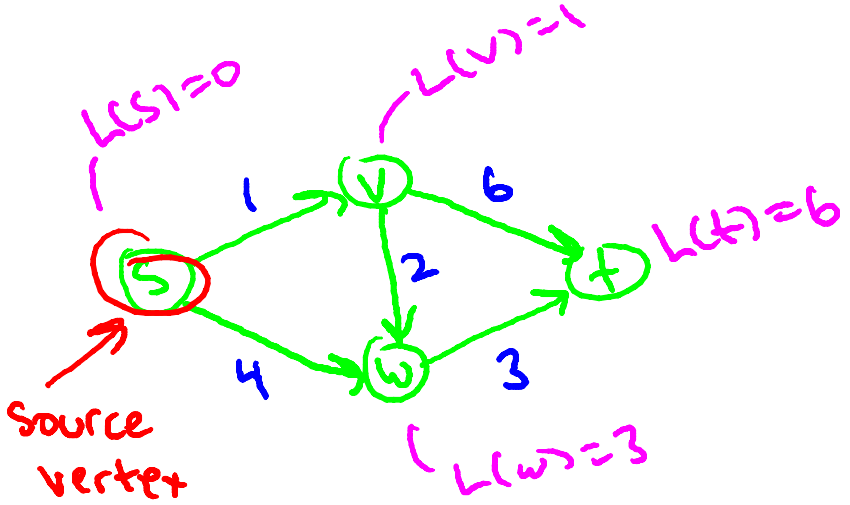
Assumptions:

① [for convenience] $\forall v \in V$, $\exists$ an $s \rightsquigarrow v$ path

② [important] $\ell_e \geq 0$ $\forall e \in E$

One of the following is the list of shortest-path distances for the nodes s, v, w, t, respectively. Which is it?

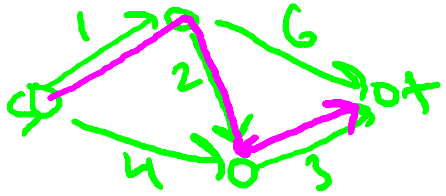
- 0,1,2,3
- 0,1,4,7
- 0,1,4,6
- 0,1,3,6



Why Another Shortest-Path Algorithm?

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

Answer: yes, IF $\ell_e = 1$ 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

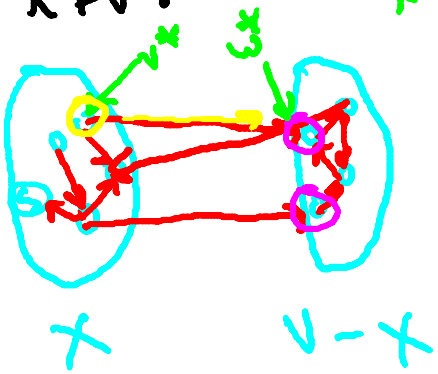
Initialize:

- $X = \{s\}$ [vertices processed so far]
- $A[s] = 0$ [computed shortest path distances]
- $B[s] =$ empty path [computed shortest paths]

this array only to help explanation!

Main Loop

- while $X \neq V$:
  - need to grow X by one node



Main Loop cont'd

- among all edges $(v,w) \in E$ with $v \in X$, $w \notin X$, pick the one that minimizes

$$A[v] + \ell_{vw}$$

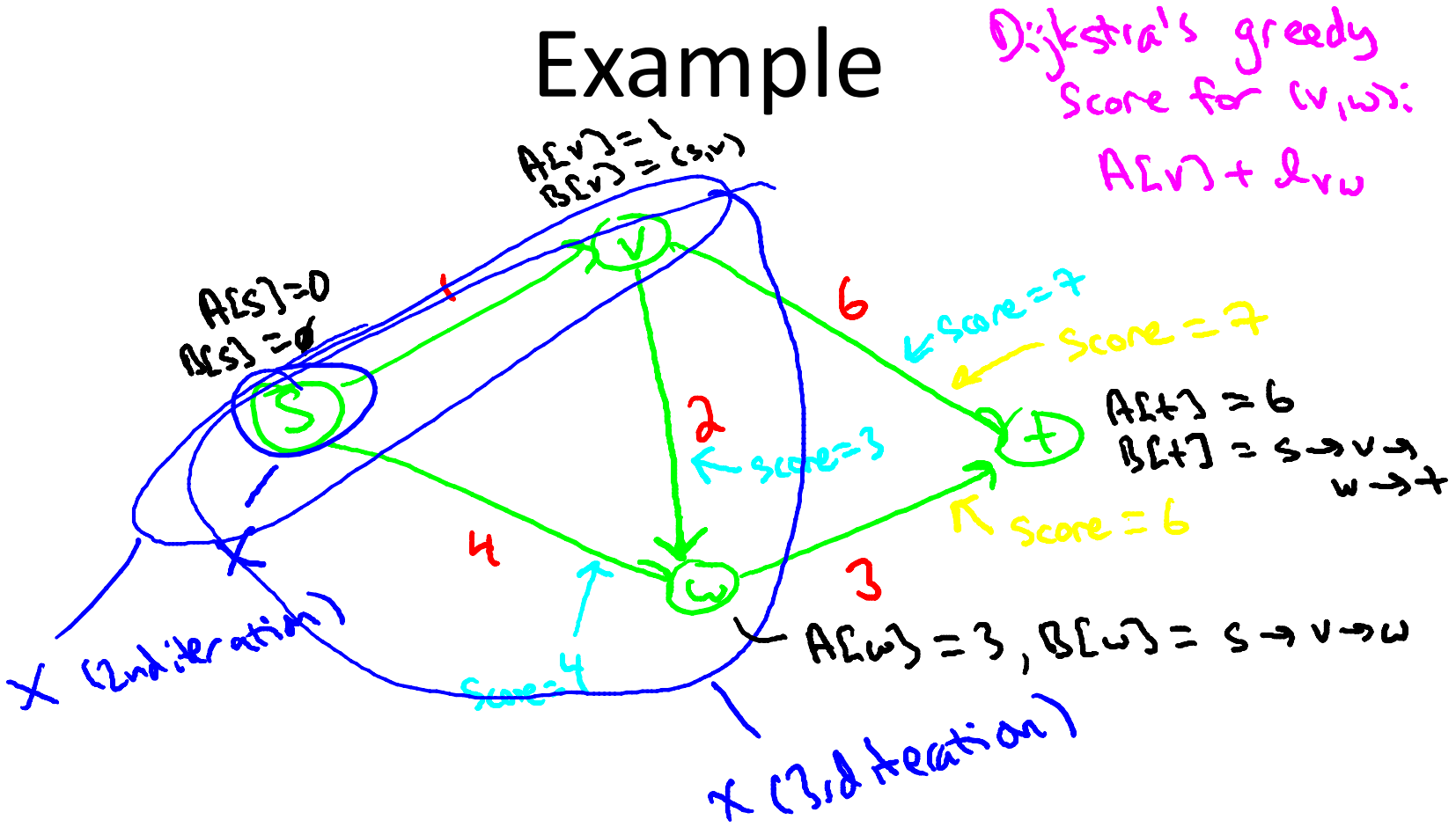
(Dijkstra's greedy criterion)

already computed in earlier iteration

[call it (v^*, w^*)]

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

Example


Dijkstra's greedy score for (v,w): A[v] + l_vw
A[v] = 1
B[v] = (s,v)
A[s] = 0
B[s] = ∅
s
v
w
t
1
6
2
4
3
score = 7
score = 7
score = 3
score = 6
score = 4
A[t] = 6
B[t] = s → v → w → t
A[w] = 3, B[w] = s → v → w
X (2nd iteration)
X (3rd iteration)

Non-Example

Question: why not reduce computing shortest paths with negative edge lengths to the same problem with nonnegative edge 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 -47)