Objectives

- Dynamic Programming
  - Shortest Path

Discussion

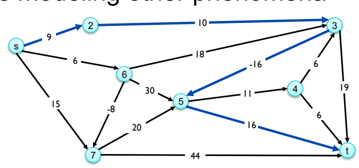
- Thoughts on Jan Cuny's talk?
- March Madness
- Dynamic programming after problem set

Shortest Paths

- **Problem**: Given a directed graph G = (V, E), with edge weights c_{vw}, find shortest path from node s to node t

allow negative weights

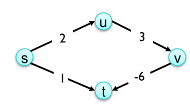
- Allows modeling other phenomena



Shortest Paths: Failed Attempts

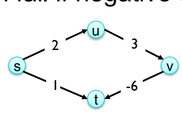
- Review: What was Dijkstra's algorithm?
  - Dijkstra can fail if negative edge costs

Shortest path from s → t?



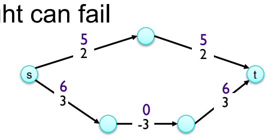
Shortest Paths: Failed Attempts

- Dijkstra. Can fail if negative edge costs



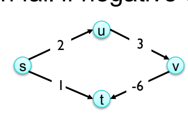
- Re-weighting. Adding a constant to every edge weight can fail

Why?



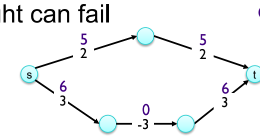
Shortest Paths: Failed Attempts

- Dijkstra. Can fail if negative edge costs



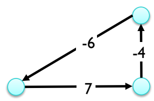
- Re-weighting. Adding a constant to every edge weight can fail

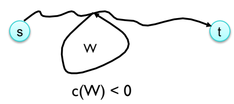
Why?



Orig:	New:
4	10
3	12

Shortest Paths: Negative Cost Cycles





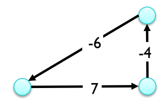
- If some path from *s* to *t* contains a negative cost cycle, there does **not** exist a shortest *s-t* path

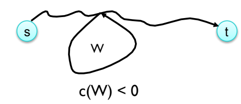
Why?

- Otherwise, there exists one that is *simple* (i.e., does not repeat nodes)

What does this mean about number of edges in path?

Shortest Paths: Negative Cost Cycles



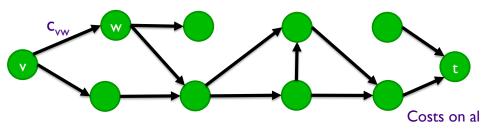


- If some path from *s* to *t* contains a negative cost cycle, there does **not** exist a shortest *s-t* path
- Otherwise, there exists one that is *simple* (i.e., does not repeat nodes)
  - Path has *at most n-1* edges, where *n* is # of nodes in graph

Towards a Recurrence

- **OPT(*i*,*v*)**: minimum cost of a ***v***-*t* path *P* using ***at most i*** edges
  - This formulation eases later discussion
- Original problem is OPT(n-1, s)

Break down into subproblems based on *i* and *v*



Costs on all edges

Shortest Paths: Dynamic Programming

- OPT(i, v) = minimum cost of a ***v***-*t* path *P* using at most ***i*** edges
  - Case 1: *P* uses at most *i-1* edges
    - OPT(i, v) = OPT(i-1, v)
  - Case 2: *P* uses exactly *i* edges
    - if (*v, w*) is first edge, then OPT uses (*v, w*), and then selects best *w-t* path using at most *i-1* edges
    - Cost: cost of chosen edge

$$OPT(i,v) = \begin{cases} 0 & \text{if } i = 0 \\ \min\left\{ OPT(i-1,\ v),\ \min\limits_{(v,w)\in E} \{ OPT(i-1,\ w) + c_{vw} \} \right\} & \text{otherwise} \end{cases}$$

Shortest Paths: Implementation

Starting node

```
Shortest-Path(G, t)
   n = number of nodes in G
   foreach node v ∈ V
      M[0, v] = ∞
   M[0, t] = 0

   for i = 1 to n-1
      foreach node v ∈ V
         M[i, v] = M[i-1, v]
         foreach edge (v, w) ∈ E
             M[i, v] = min(M[i, v], M[i-1, w] + c_vw )
```

Cost of chosen edge

- Shortest path length is M[n-1, s]

Starting node

Shortest Paths: Implementation

```
Shortest-Path(G, t)
   n = number of nodes in G
   foreach node v ∈ V
      M[0, v] = ∞    # infinite cost to reach all nodes
   M[0, t] = 0    # no cost to reach destination from dest

   for i = 1 to n-1
      foreach node v ∈ V
         M[i, v] = M[i-1, v]  # at most cost of 1 less
         foreach edge (v, w) ∈ E
             M[i, v] = min(M[i, v], M[i-1, w] + c_vw )
```

Cost of chosen edge

Analysis?

- Shortest path length is M[n-1, s]

Starting node

Shortest Paths: Analysis

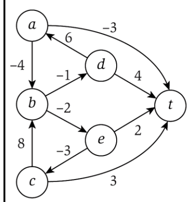
```
Shortest-Path(G, t)
  n = number of nodes in G
  foreach node v ∈ V
     M[0, v] = ∞    # infinite cost to reach all nodes
  M[0, t] = 0    # no cost to reach destination from dest

  for i = 1 to n-1     O(n)
     foreach node v ∈ V
        M[i, v] = M[i-1, v]  # at most cost of 1 less     O(m)
        foreach edge (v, w) ∈ E
           M[i, v] = min(M[i, v], M[i-1, w] + c_vw )
```

Time: $O(n^3)$, $\Theta(mn)$
Space: $\Theta(n^2)$

Mar 18, 2011 CSCI211 - Sprenkle 13

Example

Trying to get to t

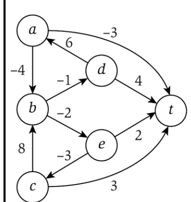


Number of edges in path

	0	1	2	3	4	5
t	0	0	0	0	0	0
a	∞					
b	∞					
c	∞					
d	∞					
e	∞					

Mar 18, 2011 CSCI211 - Sprenkle 14

Example



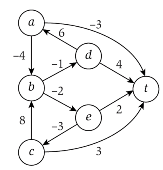
Number of edges in path

	0	1	2	3	4	5
t	0	0	0	0	0	0
a	∞					
b	∞					
c	∞					
d	∞					
e	∞					

What edges do we need to look at for each node?

Mar 18, 2011 CSCI211 - Sprenkle 15

Example



Edges

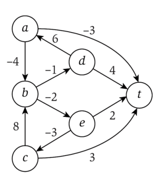
b, t
d, e
b, t
a, t
c, t

Number of edges in path

	0	1	2	3	4	5
t	0	0	0	0	0	0
a	∞	-3				
b	∞	∞				
c	∞	3				
d	∞	4				
e	∞	2				

Mar 18, 2011 CSCI211 - Sprenkle 16

Example



Edges

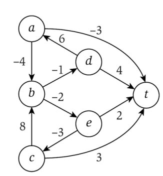
b, t
d, e
b, t
a, t
c, t

Number of edges in path

	0	1	2	3	4	5
t	0	0	0	0	0	0
a	∞	-3				
b	∞	∞				
c	∞	3				
d	∞	4				
e	∞	2				

Mar 18, 2011 CSCI211 - Sprenkle 17

Example



Edges

b, t
d, e
b, t
a, t
c, t

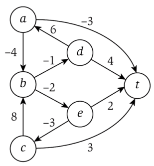
Number of edges in path

	0	1	2	3	4	5
t	0	0	0	0	0	0
a	∞	-3	-3			
b	∞	∞	0			
c	∞	3	3			
d	∞	4	3			
e	∞	2	0			

Mar 18, 2011 CSCI211 - Sprenkle 18

Example

Number of edges in path

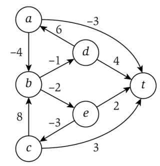


Edges
b, t
d, e
b, t
a, t
c, t

	0	1	2	3	4	5
t	0	0	0	0	0	0
a	∞	-3	-3	-4		
b	∞	∞	0	-2		
c	∞	3	3	3		
d	∞	4	3	3		
e	∞	2	0	0		

Example

Number of edges in path

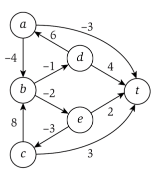


Edges
b, t
d, e
b, t
a, t
c, t

	0	1	2	3	4	5
t	0	0	0	0	0	0
a	∞	-3	-3	-4	-6	
b	∞	∞	0	-2	-2	
c	∞	3	3	3	3	
d	∞	4	3	2	0	
e	∞	2	0	0	0	

Example

Number of edges in path



Edges
b, t
d, e
b, t
a, t
c, t

	0	1	2	3	4	5
t	0	0	0	0	0	0
a	∞	-3	-3	-4	-6	-6
b	∞	∞	0	-2	-2	-2
c	∞	3	3	3	3	3
d	∞	4	3	2	0	0
e	∞	2	0	0	0	0

Shortest Paths: Implementation

```
Shortest-Path(G, t)
   n = number of nodes in G
   foreach node v ∈ V
      M[0, v] = ∞    # infinite cost to reach all nodes
   M[0, t] = 0    # no cost to reach destination from dest

   for i = 1 to n-1
      foreach node v ∈ V
         M[i, v] = M[i-1, v]  # at most cost of 1 less
         foreach edge (v, w) ∈ E
             M[i, v] = min(M[i, v], M[i-1, w] + c_vw )
```

- Shortest path length is M[n-1, s]

Review: Dynamic Programming Process

1. Determine the optimal substructure of the problem → define the recurrence relation
2. Define the algorithm to find the ***value*** of the optimal solution
3. Optionally, change the algorithm to an iterative rather than recursive solution
4. Define algorithm to find the ***optimal solution***
5. Analyze running time of algorithms

Should see all parts in exam answers

This Week

- Exam 2 due **Friday @ 4:30 p.m.**
  - To Jacque
  - No class next week: all work days
    - Conference in Germany to present research paper
  - No “outside resources”
  - OK: Your notes, my slides, book
- No Wiki due until following Wednesday
  - Keep reading Chapter 6
- Problem Set 8 due following Friday
  - Starter code on course web site