Dynamic Programming II

Inge Li Gørtz

KT section 6.6 and 6.8

Thank you to Kevin Wayne for inspiration to slides

Dynamic Programming

- Optimal substructure
- Last time
  - Weighted interval scheduling
  - Segmented least squares
- Today
  - Sequence alignment
  - Shortest paths with negative weights

Sequence Alignment

Sequence alignment

- How similar are ACAAGTC and CATGT.
- Align them such that
  - all items occurs in at most one pair.
  - no crossing pairs.
- Cost of alignment
  - gap penalty δ
  - mismatch cost for each pair of letters α(p,q).
- Goal: find minimum cost alignment.

```
A C A A G T C        A C A A - G T C
- C A T G T -        - C A - T G T -

1 mismatch, 2 gaps    0 mismatches, 4 gaps
```

Sequence Alignment

- Subproblem property.

- $SA(X_i,Y_j)$ = min cost of aligning strings X[1…i] and Y[1…j].

- Case 1. Align x_i and y_j.
  - Pay mismatch cost for x_i and y_j + min cost of aligning X_{i-1} and Y_{j-1}.
- Case 2. Leave x_i unaligned.
  - Pay gap cost + min cost of aligning X_{i-1} and Y_j.
- Case 3. Leave y_j unaligned.
  - Pay gap cost + min cost of aligning X_i and Y_{j-1}.

Sequence alignment

$$SA(X_i,Y_j) = \begin{cases} j\delta & \text{if } i=0 \\ i\delta & \text{if } j=0 \\ \min \begin{cases} \alpha(x_i,y_j)+SA(X_{i-1},Y_{j-1}), \\ \delta + SA(X_i,Y_{j-1}), \\ \delta + SA(X_{i-1},Y_j)\} \end{cases} & \text{otherwise} \end{cases}$$

		A	C	A	A	G	T	C
C								
A								
T								
G								
T								

$\delta = 1$

$SA(X_5, Y_3)$

Penalty matrix

	A	C	G	T
A	0	1	2	2
C	1	0	2	3
G	2	2	0	1
T	2	3	1	0

Sequence alignment

$$SA(X_i,Y_j) = \begin{cases} j\delta & \text{if } i=0 \\ i\delta & \text{if } j=0 \\ \min \begin{cases} \alpha(x_i,y_j)+SA(X_{i-1},Y_{j-1}), \\ \delta + SA(X_i,Y_{j-1}), \\ \delta + SA(X_{i-1},Y_j)\} \end{cases} & \text{otherwise} \end{cases}$$

		A	C	A	A	G	T	C
C								
A								
T								
G								
T								

$\delta = 1$

$SA(X_5, Y_3)$
Depends on ?

Penalty matrix

	A	C	G	T
A	0	1	2	2
C	1	0	2	3
G	2	2	0	1
T	2	3	1	0

Sequence alignment

$$SA(X_i,Y_j) = \begin{cases} j\delta & \text{if } i=0 \\ i\delta & \text{if } j=0 \\ \min \begin{cases} \alpha(x_i,y_j)+SA(X_{i-1},Y_{j-1}), \\ \delta + SA(X_i,Y_{j-1}), \\ \delta + SA(X_{i-1},Y_j)\} \end{cases} & \text{otherwise} \end{cases}$$

		A	C	A	A	G	T	C
C								
A								
T								
G								
T								

$\delta = 1$

$SA(X_5, Y_3)$
Depends on ?

Penalty matrix

	A	C	G	T
A	0	1	2	2
C	1	0	2	3
G	2	2	0	1
T	2	3	1	0

Sequence alignment

$$SA(X_i, Y_j) = \begin{cases} j\delta & \text{if } i = 0 \\ i\delta & \text{if } j = 0 \\ \min \begin{cases} \alpha(x_i, y_j) + SA(X_{i-1}, Y_{j-1}), \\ \delta + SA(X_i, Y_{j-1}), \\ \delta + SA(X_{i-1}, Y_j)\} \end{cases} & \text{otherwise} \end{cases}$$

		A	C	A	A	G	T	C
	0	1	2	3	4	5	6	7
C	1							
A	2							
T	3							
G	4							
T	5							

δ = 1

Penalty matrix

	A	C	G	T
A	0	1	2	2
C	1	0	2	3
G	2	2	0	1
T	2	3	1	0

Sequence alignment

$$SA(X_i, Y_j) = \begin{cases} j\delta & \text{if } i = 0 \\ i\delta & \text{if } j = 0 \\ \min \begin{cases} \alpha(x_i, y_j) + SA(X_{i-1}, Y_{j-1}), \\ \delta + SA(X_i, Y_{j-1}), \\ \delta + SA(X_{i-1}, Y_j)\} \end{cases} & \text{otherwise} \end{cases}$$

min(1+0, 1+1, 1+1)

		A	C	A	A	G	T	C
	0	1	2	3	4	5	6	7
C	1							
A	2							
T	3							
G	4							
T	5							

δ = 1

Penalty matrix

	A	C	G	T
A	0	1	2	2
C	1	0	2	3
G	2	2	0	1
T	2	3	1	0

Sequence alignment

$$SA(X_i, Y_j) = \begin{cases} j\delta & \text{if } i = 0 \\ i\delta & \text{if } j = 0 \\ \min \begin{cases} \alpha(x_i, y_j) + SA(X_{i-1}, Y_{j-1}), \\ \delta + SA(X_i, Y_{j-1}), \\ \delta + SA(X_{i-1}, Y_j)\} \end{cases} & \text{otherwise} \end{cases}$$

min(1+0, 1+1, 1+1)

		A	C	A	A	G	T	C
	0	1	2	3	4	5	6	7
C	1	1						
A	2							
T	3							
G	4							
T	5							

δ = 1

Penalty matrix

	A	C	G	T
A	0	1	2	2
C	1	0	2	3
G	2	2	0	1
T	2	3	1	0

Sequence alignment

$$SA(X_i, Y_j) = \begin{cases} j\delta & \text{if } i = 0 \\ i\delta & \text{if } j = 0 \\ \min \begin{cases} \alpha(x_i, y_j) + SA(X_{i-1}, Y_{j-1}), \\ \delta + SA(X_i, Y_{j-1}), \\ \delta + SA(X_{i-1}, Y_j)\} \end{cases} & \text{otherwise} \end{cases}$$

min(0+1, 1+2, 1+1)

		A	C	A	A	G	T	C
	0	1	2	3	4	5	6	7
C	1	1						
A	2							
T	3							
G	4							
T	5							

δ = 1

Penalty matrix

	A	C	G	T
A	0	1	2	2
C	1	0	2	3
G	2	2	0	1
T	2	3	1	0

Sequence alignment

$$SA(X_i, Y_j) = \begin{cases} j\delta & \text{if } i = 0 \\ i\delta & \text{if } j = 0 \\ \min \begin{cases} \alpha(x_i, y_j) + SA(X_{i-1}, Y_{j-1}), \\ \delta + SA(X_i, Y_{j-1}), \\ \delta + SA(X_{i-1}, Y_j)\} \end{cases} & \text{otherwise} \end{cases}$$

min(0+1, 1+2, 1+1)

		A	C	A	A	G	T	C
	0	1	2	3	4	5	6	7
C	1	1	1					
A	2							
T	3							
G	4							
T	5							

δ = 1

Penalty matrix

	A	C	G	T
A	0	1	2	2
C	1	0	2	3
G	2	2	0	1
T	2	3	1	0

Sequence alignment

$$SA(X_i, Y_j) = \begin{cases} j\delta & \text{if } i = 0 \\ i\delta & \text{if } j = 0 \\ \min \begin{cases} \alpha(x_i, y_j) + SA(X_{i-1}, Y_{j-1}), \\ \delta + SA(X_i, Y_{j-1}), \\ \delta + SA(X_{i-1}, Y_j)\} \end{cases} & \text{otherwise} \end{cases}$$

min(1+2, 1+3, 1+1)

		A	C	A	A	G	T	C
	0	1	2	3	4	5	6	7
C	1	1	1					
A	2							
T	3							
G	4							
T	5							

δ = 1

Penalty matrix

	A	C	G	T
A	0	1	2	2
C	1	0	2	3
G	2	2	0	1
T	2	3	1	0

Sequence alignment

$$SA(X_i, Y_j) = \begin{cases} j\delta & \text{if } i = 0 \\ i\delta & \text{if } j = 0 \\ \min \begin{cases} \alpha(x_i, y_j) + SA(X_{i-1}, Y_{j-1}), \\ \delta + SA(X_i, Y_{j-1}), \\ \delta + SA(X_{i-1}, Y_j)\} \end{cases} & \text{otherwise} \end{cases}$$

min(1+2, 1+3, 1+1)

		A	C	A	A	G	T	C
	0	1	2	3	4	5	6	7
C	1	1	1	2				
A	2							
T	3							
G	4							
T	5							

δ = 1

Penalty matrix

	A	C	G	T
A	0	1	2	2
C	1	0	2	3
G	2	2	0	1
T	2	3	1	0

Sequence alignment

$$SA(X_i, Y_j) = \begin{cases} j\delta & \text{if } i = 0 \\ i\delta & \text{if } j = 0 \\ \min \begin{cases} \alpha(x_i, y_j) + SA(X_{i-1}, Y_{j-1}), \\ \delta + SA(X_i, Y_{j-1}), \\ \delta + SA(X_{i-1}, Y_j)\} \end{cases} & \text{otherwise} \end{cases}$$

min(1+3, 1+4, 1+2)

		A	C	A	A	G	T	C
	0	1	2	3	4	5	6	7
C	1	1	1	2				
A	2							
T	3							
G	4							
T	5							

δ = 1

Penalty matrix

	A	C	G	T
A	0	1	2	2
C	1	0	2	3
G	2	2	0	1
T	2	3	1	0

Sequence alignment

$$SA(X_i, Y_j) = \begin{cases} j\delta & \text{if } i = 0 \\ i\delta & \text{if } j = 0 \\ \min \begin{cases} \alpha(x_i, y_j) + SA(X_{i-1}, Y_{j-1}), \\ \delta + SA(X_i, Y_{j-1}), \\ \delta + SA(X_{i-1}, Y_j) \} \end{cases} & \text{otherwise} \end{cases}$$

min(1+3, 1+4, 1+2)

		A	C	A	A	G	T	C
	0	1	2	3	4	5	6	7
C	1	1	1	2	3			
A	2							
T	3							
G	4							
T	5							

δ = 1

Penalty matrix

	A	C	G	T
A	0	1	2	2
C	1	0	2	3
G	2	2	0	1
T	2	3	1	0

Sequence alignment

$$SA(X_i, Y_j) = \begin{cases} j\delta & \text{if } i = 0 \\ i\delta & \text{if } j = 0 \\ \min \begin{cases} \alpha(x_i, y_j) + SA(X_{i-1}, Y_{j-1}), \\ \delta + SA(X_i, Y_{j-1}), \\ \delta + SA(X_{i-1}, Y_j) \} \end{cases} & \text{otherwise} \end{cases}$$

min(2+4, 1+5, 1+3)

		A	C	A	A	G	T	C
	0	1	2	3	4	5	6	7
C	1	1	1	2	3	4		
A	2							
T	3							
G	4							
T	5							

δ = 1

Penalty matrix

	A	C	G	T
A	0	1	2	2
C	1	0	2	3
G	2	2	0	1
T	2	3	1	0

Sequence alignment

$$SA(X_i, Y_j) = \begin{cases} j\delta & \text{if } i = 0 \\ i\delta & \text{if } j = 0 \\ \min \begin{cases} \alpha(x_i, y_j) + SA(X_{i-1}, Y_{j-1}), \\ \delta + SA(X_i, Y_{j-1}), \\ \delta + SA(X_{i-1}, Y_j) \} \end{cases} & \text{otherwise} \end{cases}$$

		A	C	A	A	G	T	C
	0	1	2	3	4	5	6	7
C	1	1	1	2	3	4	5	6
A	2	1	2	1	2	3	4	5
T	3	2	3	2	3	3	3	4
G	4	3	4	3	4	3	4	5
T	5	4	5	4	5	4	3	4

δ = 1

Penalty matrix

	A	C	G	T
A	0	1	2	2
C	1	0	2	3
G	2	2	0	1
T	2	3	1	0

Sequence alignment

```
SA(X[1…m],Y[1…n],δ,A){
 for i=0 to m
   M[i,0] := iδ

 for j=0 to n
   M[0,j] := jδ

 for i=1 to m
   for j = 1 to n
     M[i,j] := min{ A[i,j] + M[i-1,j-1],
                    δ + M[i-1,j],
                    δ + M[i,j-1]}

 Return M[m,n]
}
```

- Time: Θ(mn)
- Space: Θ(mn)

Sequence alignment: Finding the solution

$$SA(X_i, Y_j) = \begin{cases} j\delta & \text{if } i = 0 \\ i\delta & \text{if } j = 0 \\ \min \begin{cases} \alpha(x_i, y_j) + SA(X_{i-1}, Y_{j-1}), \\ \delta + SA(X_i, Y_{j-1}), \\ \delta + SA(X_{i-1}, Y_j)\} \end{cases} & \text{otherwise} \end{cases}$$

Penalty matrix

	A	C	G	T
A	0	1	2	2
C	1	0	2	3
G	2	2	0	1
T	2	3	1	0

δ = 1

		A	C	A	A	G	T	C
	0	1	2	3	4	5	6	7
C	1	1	1	2	3	4	5	6
A	2	1	2	1	2	3	4	5
T	3	2	3	2	3	3	3	4
G	4	3	4	3	4	3	4	5
T	5	4	5	4	5	4	3	4

		A	C	A	A	G	T	C
		←	←	←	←	←	←	←
C	↑	↖	↖	←	←	←	←	↖
A	↑	↖	↖	↖	↖	←	←	←
T	↑	↑	↑	↑	↖	↖	↖	←
G	↑	↑	↖	↑	↖	↖	↖	↖
T	↑	↑	↑	↑	↖	↑	↖	←

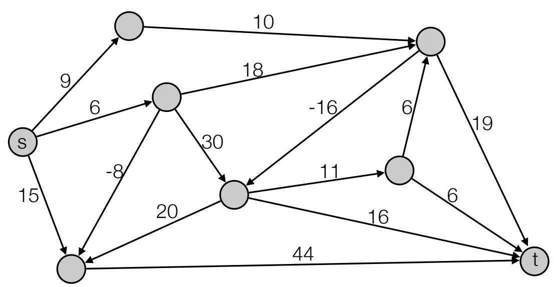
Sequence alignment

- Use dynamic programming to compute an optimal alignment.
  - Time: Θ(mn)
  - Space: Θ(mn)
- Find actual alignment by backtracking (or saving information in another matrix).
- Linear space?
  - Easy to compute value (save last and current row)
  - How to compute alignment? Hirschberg. (not part of the curriculum).

Shortest paths

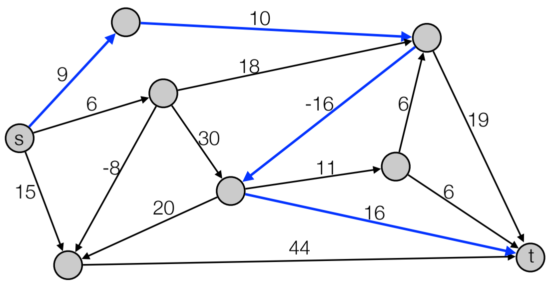
Shortest Paths

- All-Pairs Shortest Path Problem (APSP)
  - Given directed weighted graph G=(V,E).
  - Weights of edges c_{ij} are real numbers (might be negative).
  - Let n = |V| and m= |E|.
  - Weight of a path is the sum of the weights on its edges.
  - Goal: Compute the shortest path for from node s to node t.



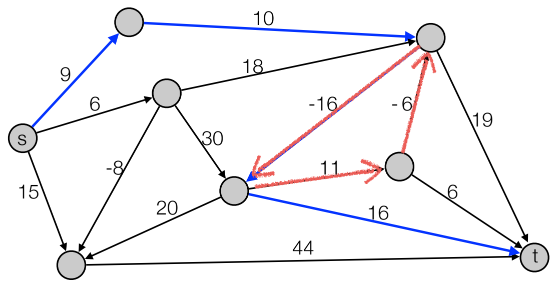
Shortest Paths

- All-Pairs Shortest Path Problem (APSP)
  - Given directed weighted graph G=(V,E).
  - Weights of edges c_{ij} are real numbers (might be negative).
  - Let n = |V| and m= |E|.
  - Weight of a path is the sum of the weights on its edges.
  - Goal: Compute the shortest path for from node s to node t.



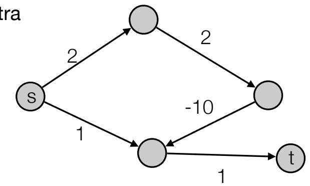
Shortest Paths

- All-Pairs Shortest Path Problem (APSP)
  - Given directed weighted graph G=(V,E).
  - Weights of edges c_{ij} are real numbers (might be negative).
  - Let n = |V| and m= |E|.
  - Weight of a path is the sum of the weights on its edges.
  - Goal: Compute the shortest path for from node s to node t.

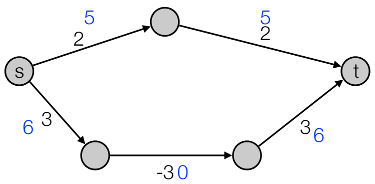


Failed attempts

- Dijkstra

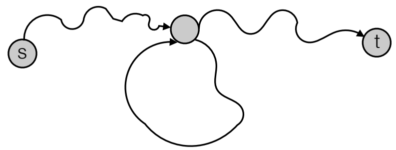


- Re-weighting



Observations

- Negative cycle. If some path from s to t contains a negative cost cycle, then there does not exist a shortest s-t path. Otherwise, there exists one that is simple.



- Optimal substructure. Subpaths of shortest paths are shortest paths

Recurrence

- OPT(i,v) = length of shortest v-t path P using at most i edges.
  - Case 1: P uses at most i-1 edges.

shortest v → t path using at most i-1 edges

OPT(i,v) = OPT(i-1,v)

  - Case 2: P uses exactly i edges.

shortest w → t path using at most i-1 edges

OPT(i,v) = OPT(i-1,w) + c_{vw}

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

- If no negative cycles then OPT(n-1,v) = length of shortest path

Bellman-Ford

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

```
Bellmann-Ford(G,s,t)

 for each node v ∈ V
   M[0,v] = ∞

 M[0,t] = 0.
 for i=1 to n-1
   for each node v ∈ V
     M[i,v] = M[i-1,v]
     for each edge (v,w) ∈ E
       M[i,v] = min(M[i,v], M[i-1,w] + cvw)
```

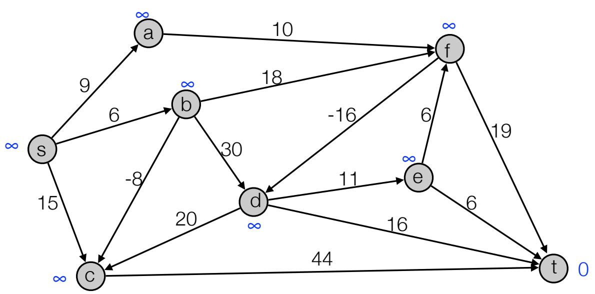

Example

```
Bellmann-Ford(G,s,t)

 for each node v ∈ V
   M[0,v] = ∞

 M[0,t] = 0.
 for i=1 to n-1
   for each node v ∈ V
     M[i,v] = M[i-1,v]
     for each edge (v,w) ∈ E
       M[i,v] = min(M[i,v], M[i-1,w] + cvw)
```

	0	1	2	3	4	5	6	7
s	∞							
a	∞							
b	∞							
c	∞							
d	∞							
e	∞							
f	∞							
t	0							



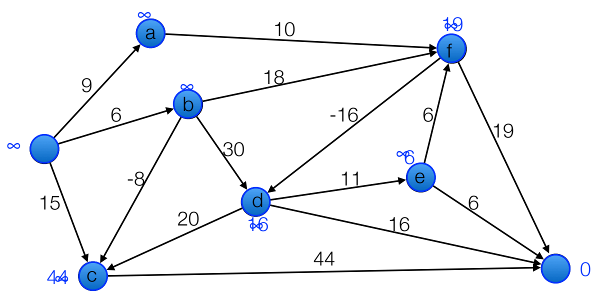
Example

```
Bellmann-Ford(G,s,t)

 for each node v ∈ V
   M[0,v] = ∞

 M[0,t] = 0.
 for i=1 to n-1
   for each node v ∈ V
     M[i,v] = M[i-1,v]
     for each edge (v,w) ∈ E
       M[i,v] = min(M[i,v], M[i-1,w] + cvw)
```

	0	1	2	3	4	5	6	7
s	∞	∞						
a	∞	∞						
b	∞	∞						
c	∞	44						
d	∞	16						
e	∞	6						
f	∞	19						
t	0	0						



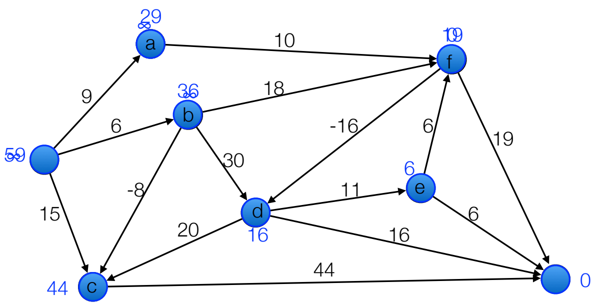
Example

```
Bellmann-Ford(G,s,t)

  for each node v ∈ V
     M[0,v] = ∞

  M[0,t] = 0.
  for i=1 to n-1
     for each node v ∈ V
        M[i,v] = M[i-1,v]
        for each edge (v,w) ∈ E
           M[i,v] = min(M[i,v], M[i-1,w] + c_vw)
```

	0	1	2	3	4	5	6	7
s	∞	∞	59					
a	∞	∞	29					
b	∞	∞	36					
c	∞	44	44					
d	∞	16	16					
e	∞	6	6					
f	∞	19	0					
t	0	0	0					



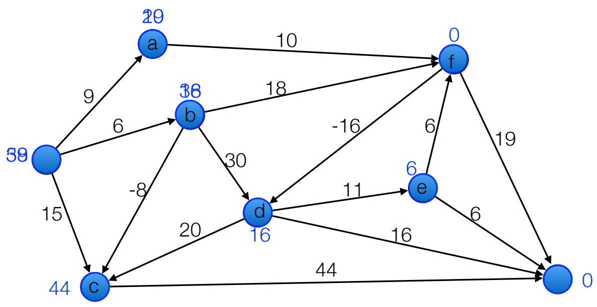
Example

```
Bellmann-Ford(G,s,t)

  for each node v ∈ V
     M[0,v] = ∞

  M[0,t] = 0.
  for i=1 to n-1
     for each node v ∈ V
        M[i,v] = M[i-1,v]
        for each edge (v,w) ∈ E
           M[i,v] = min(M[i,v], M[i-1,w] + c_vw)
```

	0	1	2	3	4	5	6	7
s	∞	∞	59	38				
a	∞	∞	29	10				
b	∞	∞	36	18				
c	∞	44	44	44				
d	∞	16	16	16				
e	∞	6	6	6				
f	∞	19	0	0				
t	0	0	0	0				



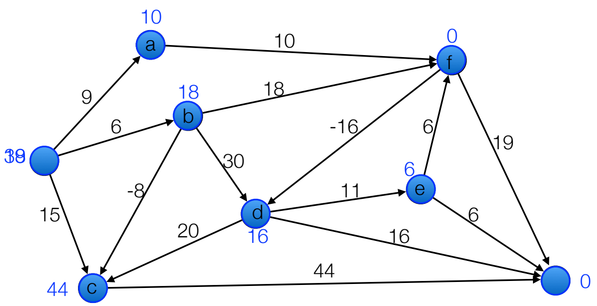
Example

```
Bellmann-Ford(G,s,t)

  for each node v ∈ V
     M[0,v] = ∞

  M[0,t] = 0.
  for i=1 to n-1
     for each node v ∈ V
        M[i,v] = M[i-1,v]
        for each edge (v,w) ∈ E
           M[i,v] = min(M[i,v], M[i-1,w] + c_vw)
```

	0	1	2	3	4	5	6	7
s	∞	∞	59	38	19			
a	∞	∞	29	10	10			
b	∞	∞	36	18	18			
c	∞	44	44	44	44			
d	∞	16	16	16	16			
e	∞	6	6	6	6			
f	∞	19	0	0	0			
t	0	0	0	0	0			



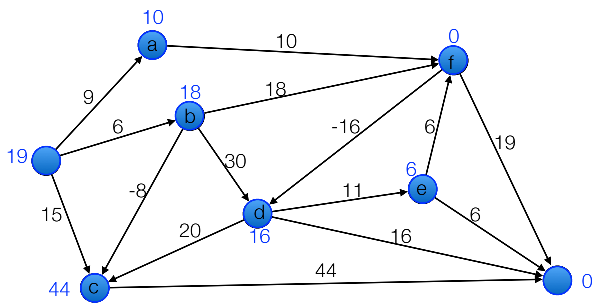
Example

```
Bellmann-Ford(G,s,t)

  for each node v ∈ V
     M[0,v] = ∞

  M[0,t] = 0.
  for i=1 to n-1
     for each node v ∈ V
        M[i,v] = M[i-1,v]
        for each edge (v,w) ∈ E
           M[i,v] = min(M[i,v], M[i-1,w] + c_vw)
```

	0	1	2	3	4	5	6	7
s	∞	∞	59	38	19	19		
a	∞	∞	29	10	10	10		
b	∞	∞	36	18	18	18		
c	∞	44	44	44	44	44		
d	∞	16	16	16	16	16		
e	∞	6	6	6	6	6		
f	∞	19	0	0	0	0		
t	0	0	0	0	0	0		



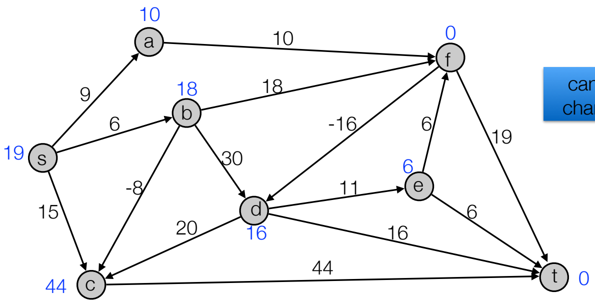
Example

```
Bellmann-Ford(G,s,t)

  for each node v ∈ V
     M[0,v] = ∞

  M[0,t] = 0.
  for i=1 to n-1
     for each node v ∈ V
        M[i,v] = M[i-1,v]
        for each edge (v,w) ∈ E
           M[i,v] = min(M[i,v], M[i-1,w] + c_vw
```

	0	1	2	3	4	5	6	7
s	∞	∞	59	38	19	19	19	19
a	∞	∞	29	10	10	10	10	10
b	∞	∞	36	18	18	18	18	18
c	∞	44	44	44	44	44	44	44
d	∞	16	16	16	16	16	16	16
e	∞	6	6	6	6	6	6	6
f	∞	19	0	0	0	0	0	0
t	0	0	0	0	0	0	0	0



can stop when no changes in a round

Bellman-Ford

```
Bellmann-Ford(G,s,t)

  for each node v ∈ V
     M[v] = ∞

  M[t] = 0.
  for i=1 to n-1
     for each node v ∈ V
        M[i,v] = M[i-1,v]
        for each edge (v,w) ∈ E
           M[i,v] = min(M[i,v], M[i-1,w] + c_vw
```

- Running time. O(mn)
- Space. $O(n^2)$

Bellman-Ford

- Improvements to basic implementation
  - Maintain only one array
  - No need to check edges of form (v,w) if M[w] didn't change in previous iteration.
- Space: O(m+n)
- Running time: O(mn) worst case, but substantially faster in practice.

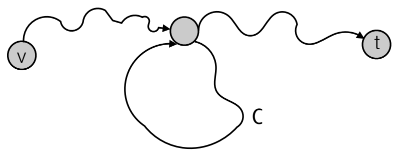
```
Bellmann-Ford-push-based(G,s,t)
  for each node v ∈ V
     M[v] = ∞
     succ[v] = nil

  M[t] = 0.
  for i=1 to n-1
     for each node w ∈ V
        if M[w] was updated in previous iteration do
           for each node v such that (v,w) ∈ E
              if M[v] > M[w] + c_vw do
                 M[v] = M[w] + c_vw
                 succ[v] = w
     if no M[w] changed in iteration i, stop.
```


Detecting negative cycles

- Lemma. If OPT(n,v) < OPT(n-1,v) for some node, then (any) shortest path from v to t contains a cycle C with negative cost.

- Proof. By contradiction.
  - OPT(n,v) < OPT(n-1,v) => P has exactly n edges
  - => P contains a cycle C.
  - Deleting C gives a v-t path with < n edges => C makes v-t path shorter => C has negative cost.



- Lemma. If OPT(n,v) = OPT(n-1,v) for all v, then no negative cycles.

Detecting negative cycles

- Detect negative cost cycles in O(mn) time.
  - Add new node t and connect all nodes to t with 0-cost edge.
  - Check if OPT(n,v) = OPT(n-1,v) for all nodes v.
    - Yes: No negative cycles.
    - No: Can find negative cycle from shortest path from v to t.