Algorithmic Approaches for Biological Data, Lecture #19

Katherine St. John

City University of New York
American Museum of Natural History

18 April 2016

Outline

- More on Dynamic Programming

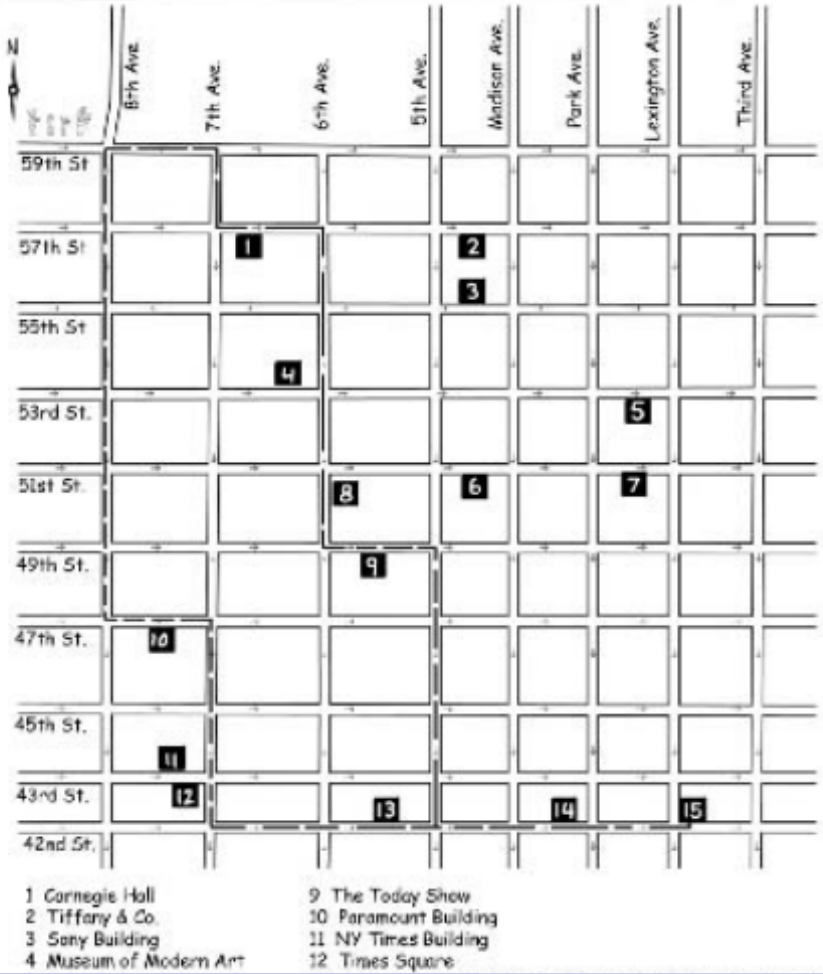
Outline

- More on Dynamic Programming
- Aligning Sequences

Outline

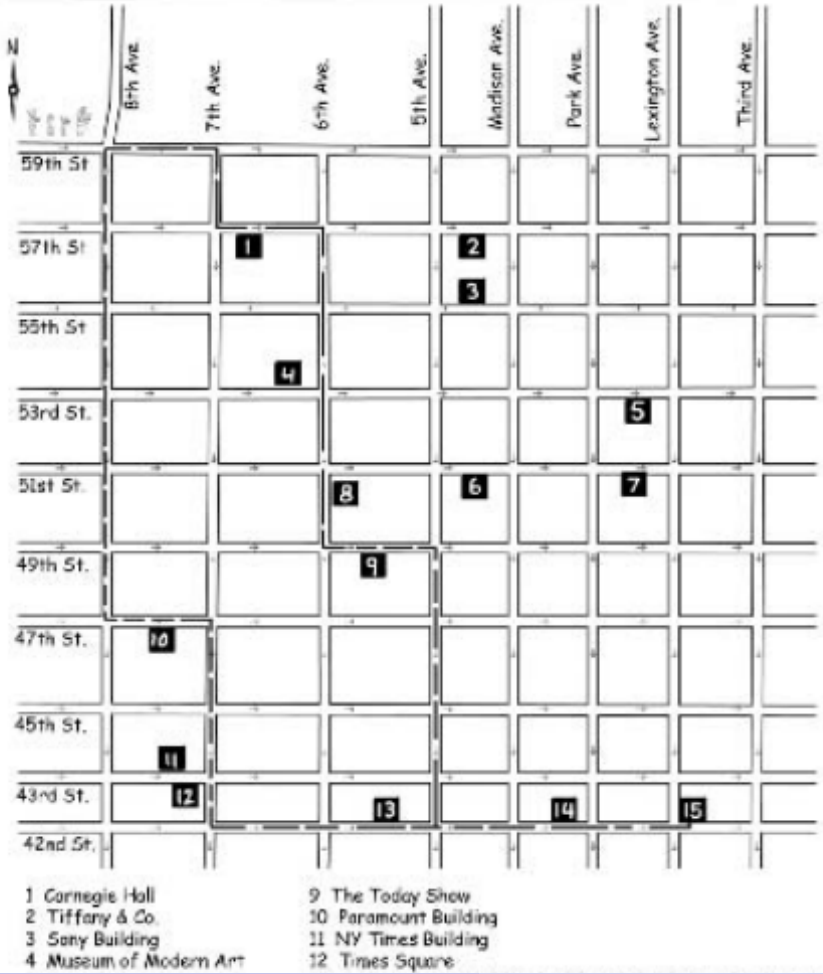
- More on Dynamic Programming
- Aligning Sequences
- Minimal Spanning Trees

Recap: Manhattan Tourist Problem



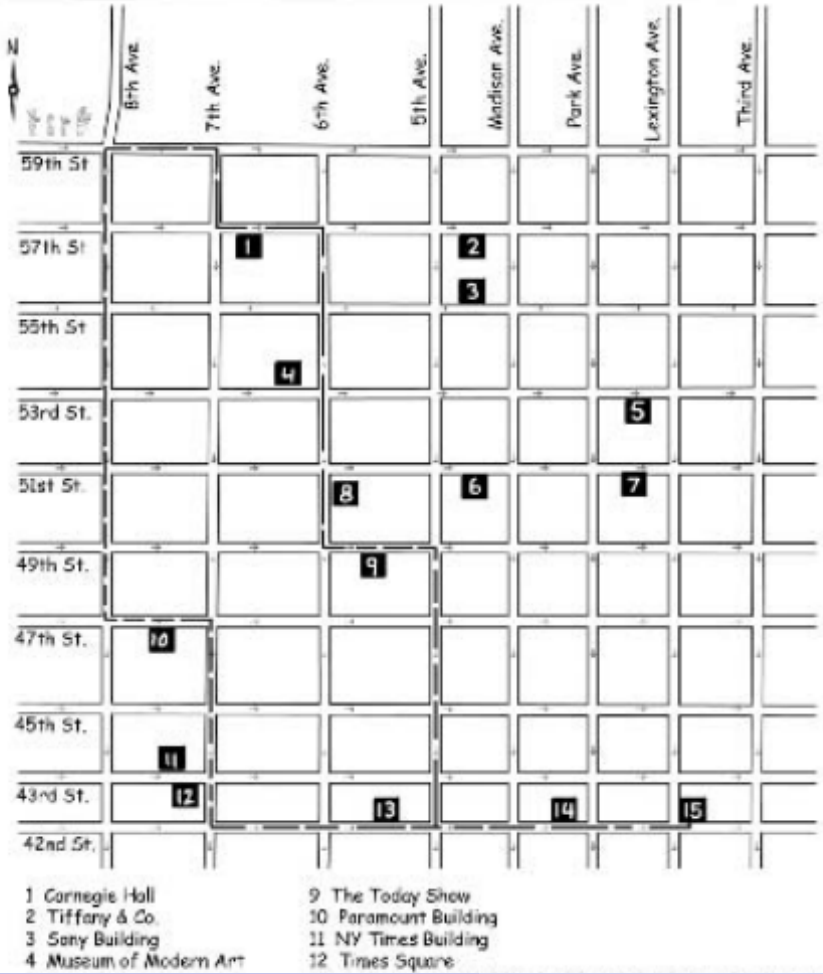
- In hurry, and want to visit as many landmarks as possible.

Recap: Manhattan Tourist Problem



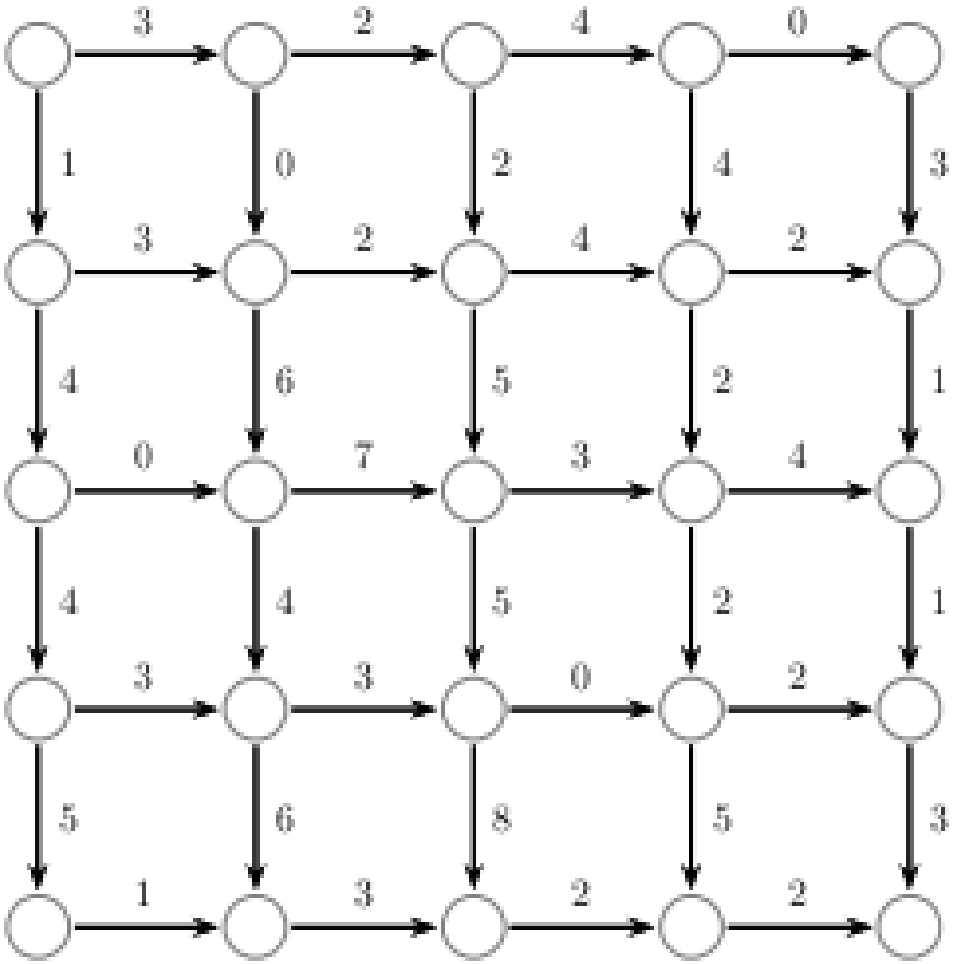
- In hurry, and want to visit as many landmarks as possible.
- Can only walk south and east.

Recap: Manhattan Tourist Problem



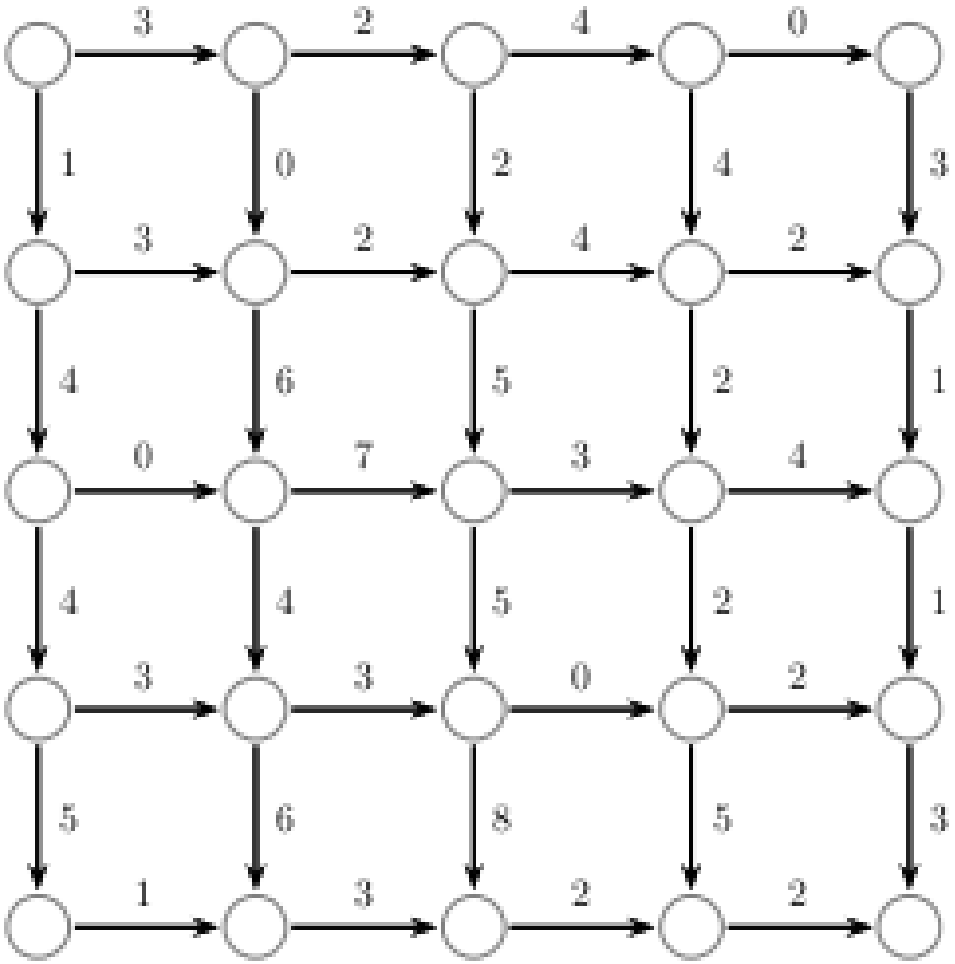
- In hurry, and want to visit as many landmarks as possible.
- Can only walk south and east.
- What's the best route?

In Pairs: Tourists Problems



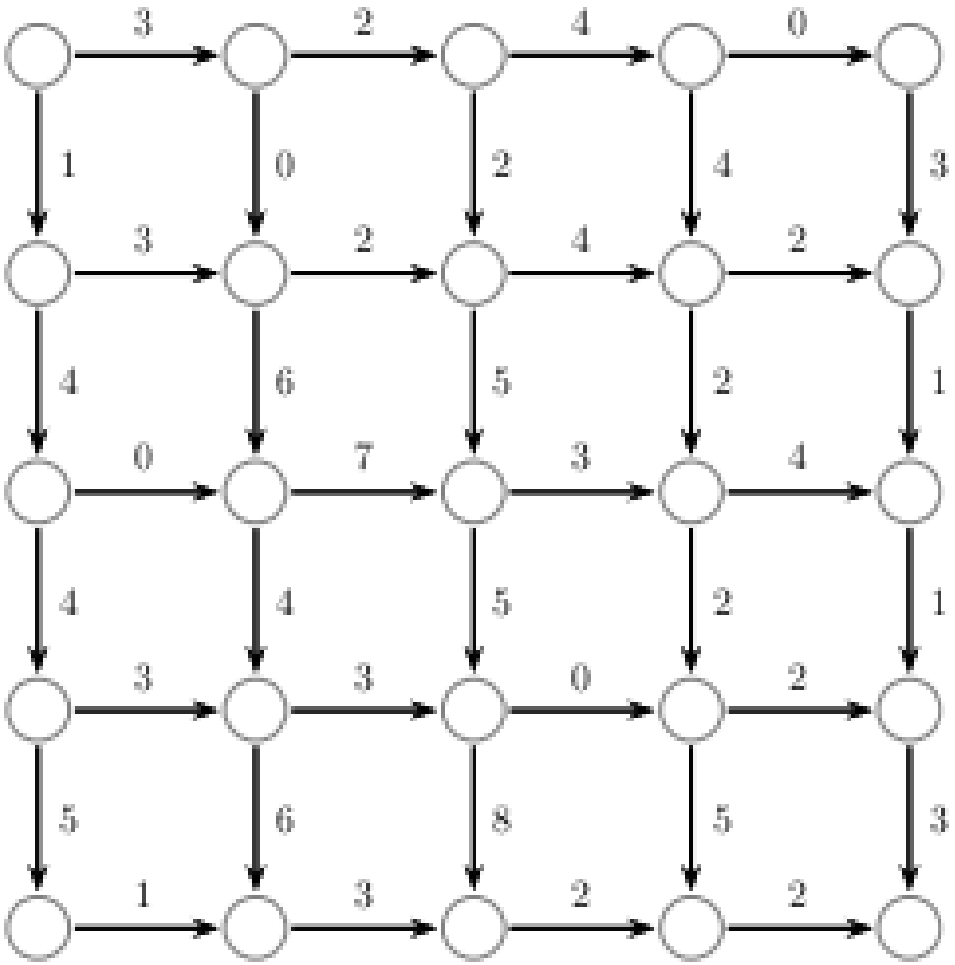
- In hurry, and want to visit as many landmarks (numbers on edges) as possible.

In Pairs: Tourists Problems



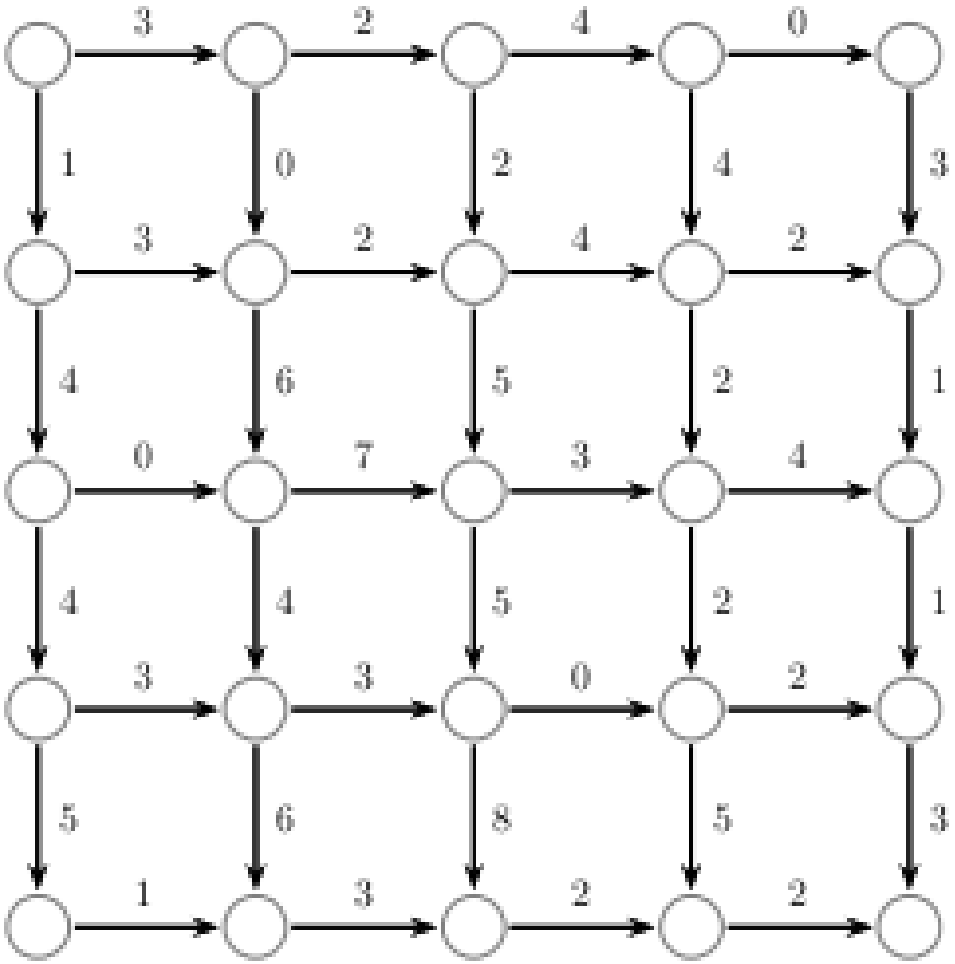
- In hurry, and want to visit as many landmarks (numbers on edges) as possible.
- Can only walk south and east.

In Pairs: Tourists Problems



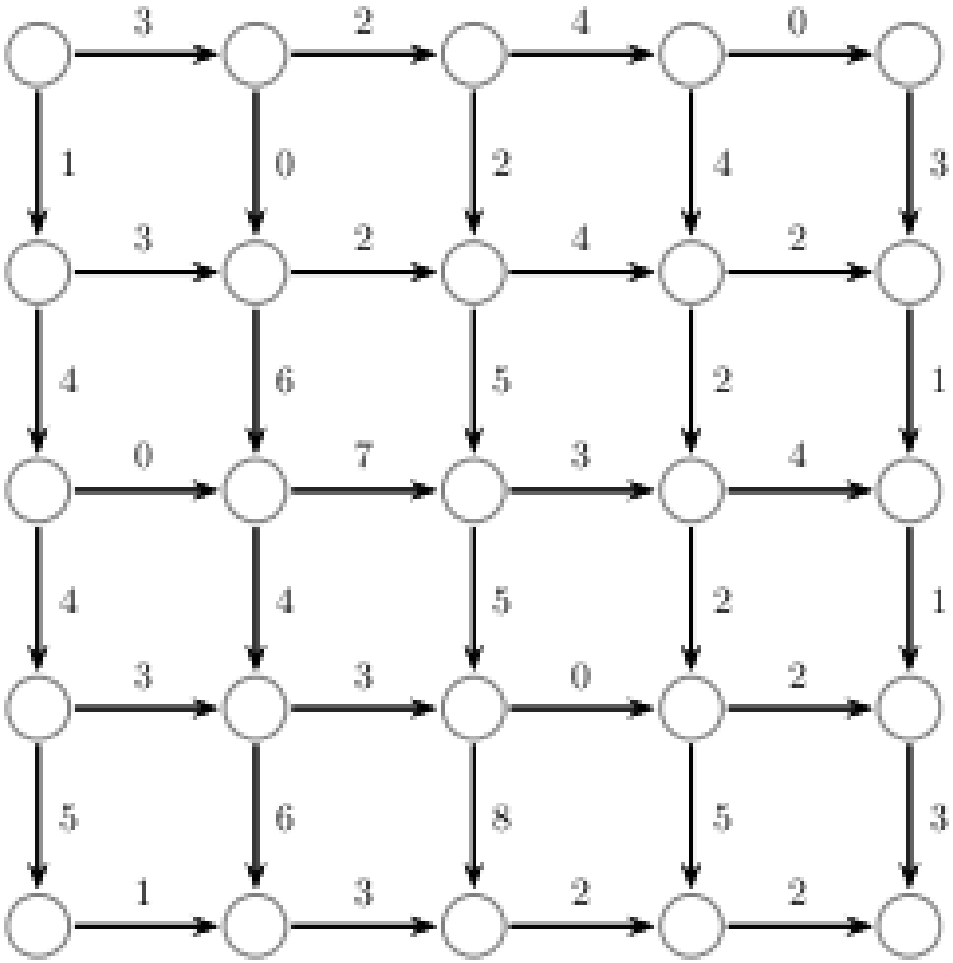
- In hurry, and want to visit as many landmarks (numbers on edges) as possible.
- Can only walk south and east.
- What's the best route found by greedy approach?

In Pairs: Tourists Problems



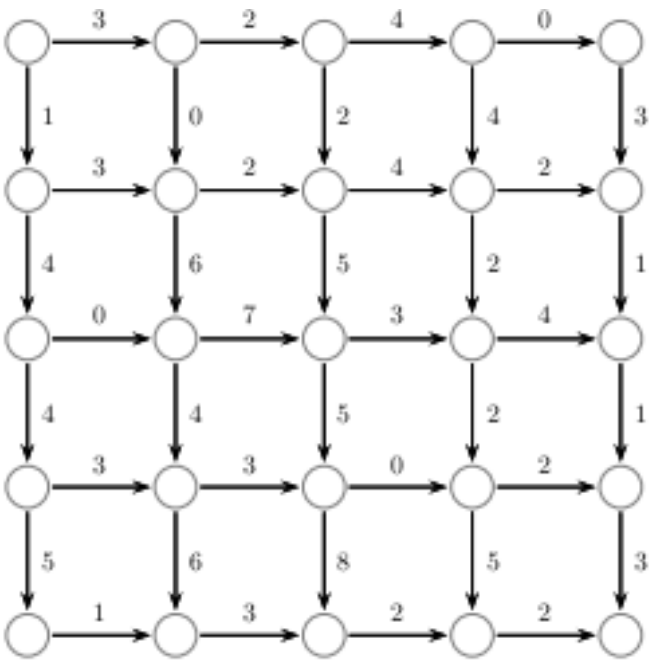
- In hurry, and want to visit as many landmarks (numbers on edges) as possible.
- Can only walk south and east.
- What's the best route found by greedy approach?
- What's the best overall?

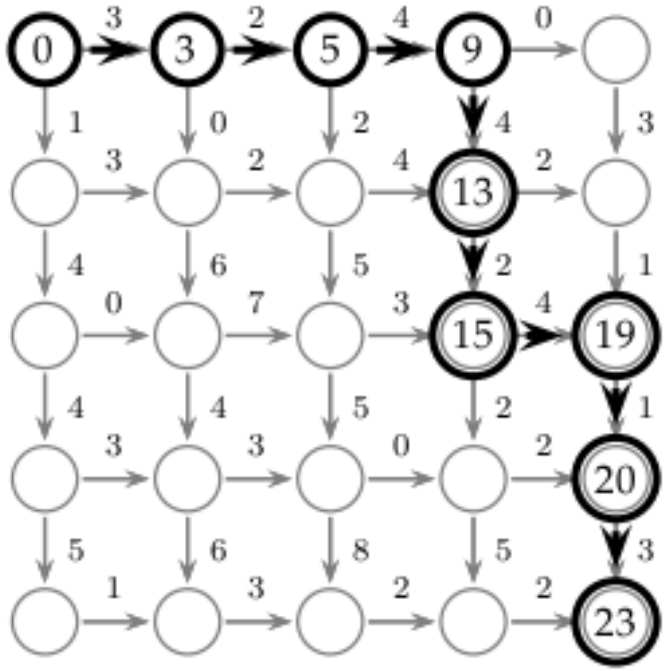
In Pairs: Tourists Problems



Greedy Algorithm: Do the best at the each step.

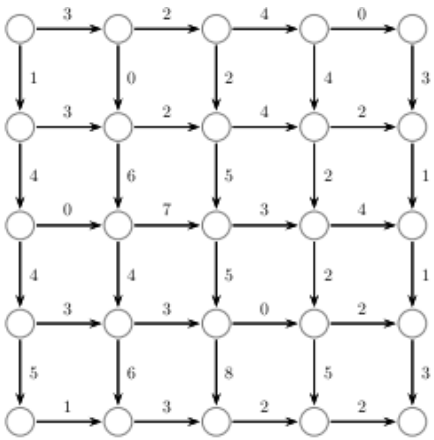
Greedy Approach: Tourists Problems





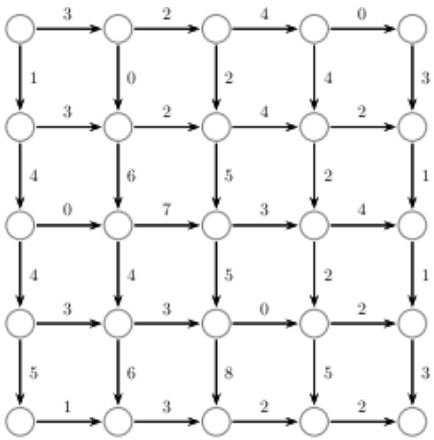
Greedy Algorithm: Do the best at the each step.

Dynamic Programming



- The best path is the larger of the best path going south and the best going east.

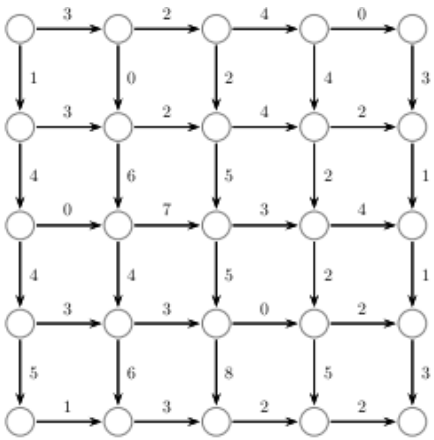
Dynamic Programming



- The best path is the larger of the best path going south and the best going east.

$$s_{i,j} = \max \begin{cases} s_{i-1,j} + \text{ weight of the edge between } (i-1,j) \text{ and } (i,j) \\ s_{i,j-1} + \text{ weight of the edge between } (i,j-1) \text{ and } (i,j) \end{cases}$$

Dynamic Programming

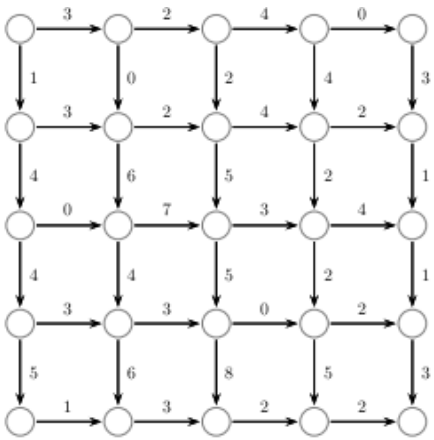


- The best path is the larger of the best path going south and the best going east.

$$s_{i,j} = \max \begin{cases} s_{i-1,j} + \text{ weight of the edge between } (i-1,j) \text{ and } (i,j) \\ s_{i,j-1} + \text{ weight of the edge between } (i,j-1) \text{ and } (i,j) \end{cases}$$

- Compute the best for each subproblem and save it for future computations.

Dynamic Programming

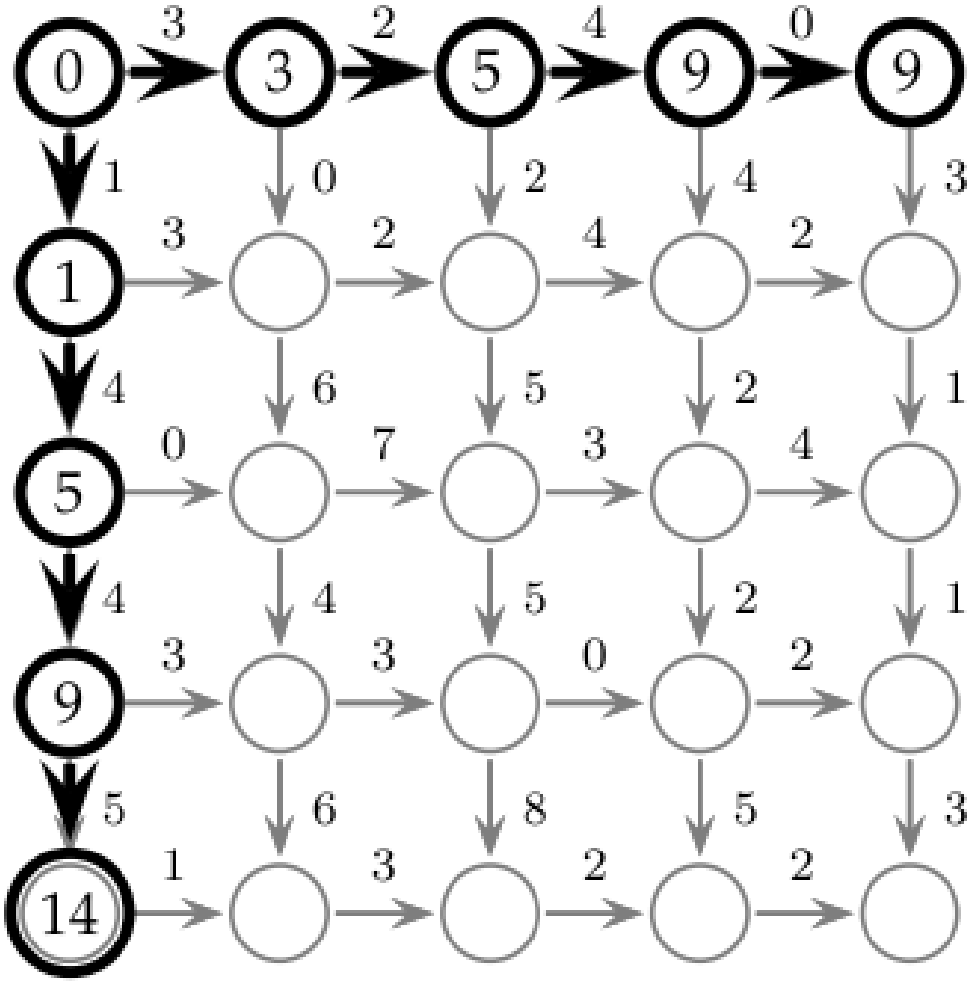


- The best path is the larger of the best path going south and the best going east.

$$s_{i,j} = \max \begin{cases} s_{i-1,j} + \text{ weight of the edge between } (i-1,j) \text{ and } (i,j) \\ s_{i,j-1} + \text{ weight of the edge between } (i,j-1) \text{ and } (i,j) \end{cases}$$

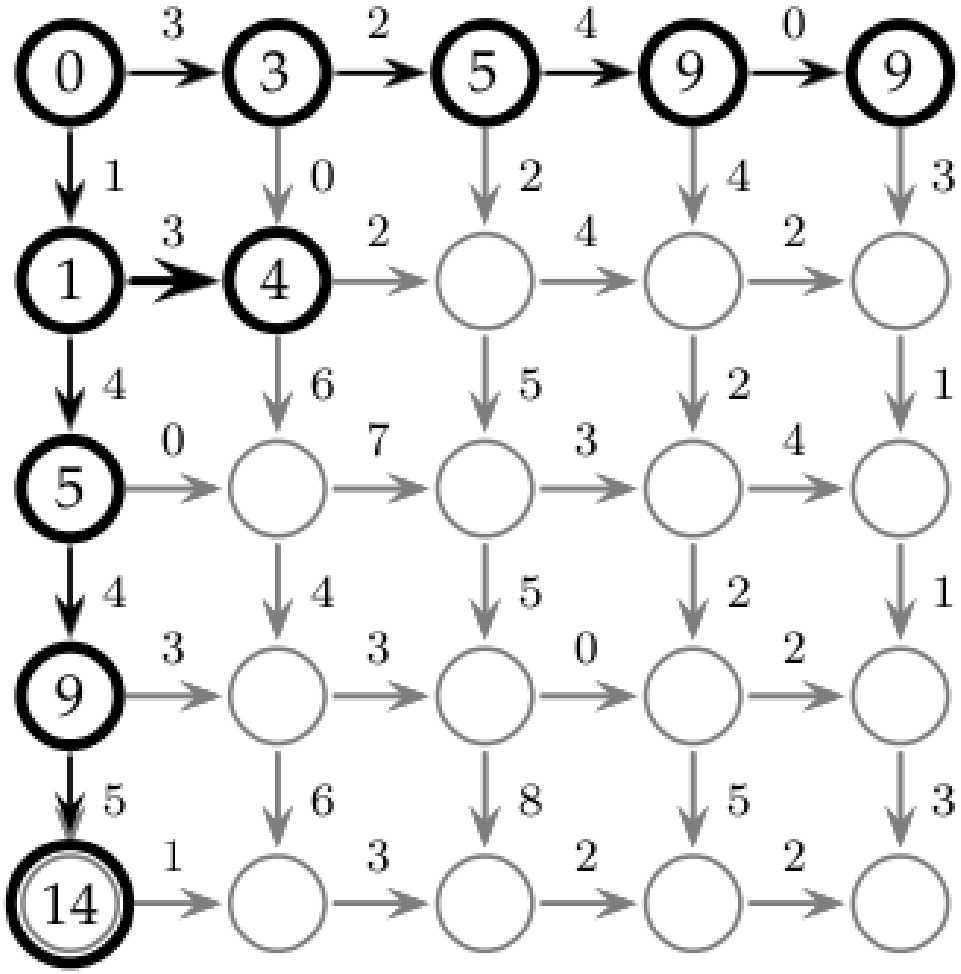
- Compute the best for each subproblem and save it for future computations.
- Store the answers in an array.

Dynamic Programming



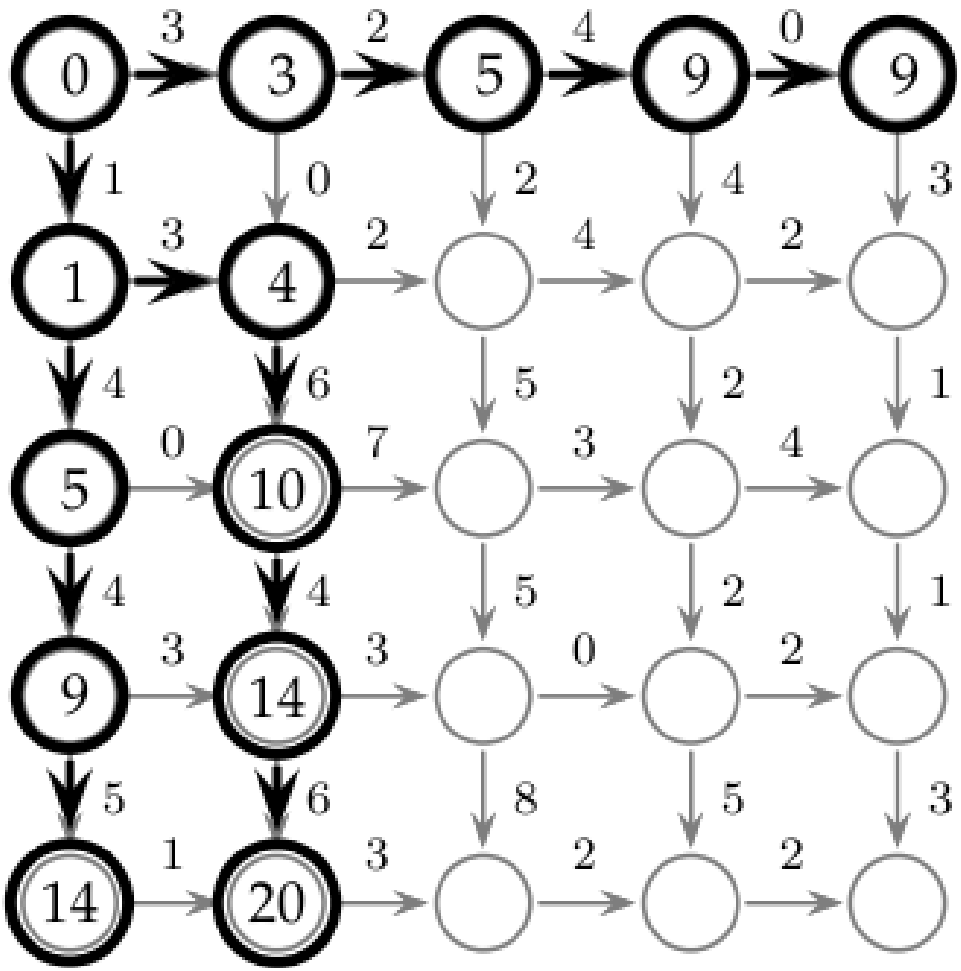
$$s_{i,j} = \max \begin{cases} s_{i-1,j} + \text{ weight of the edge between } (i-1,j) \text{ and } (i,j) \\ s_{i,j-1} + \text{ weight of the edge between } (i,j-1) \text{ and } (i,j) \end{cases}$$

Dynamic Programming



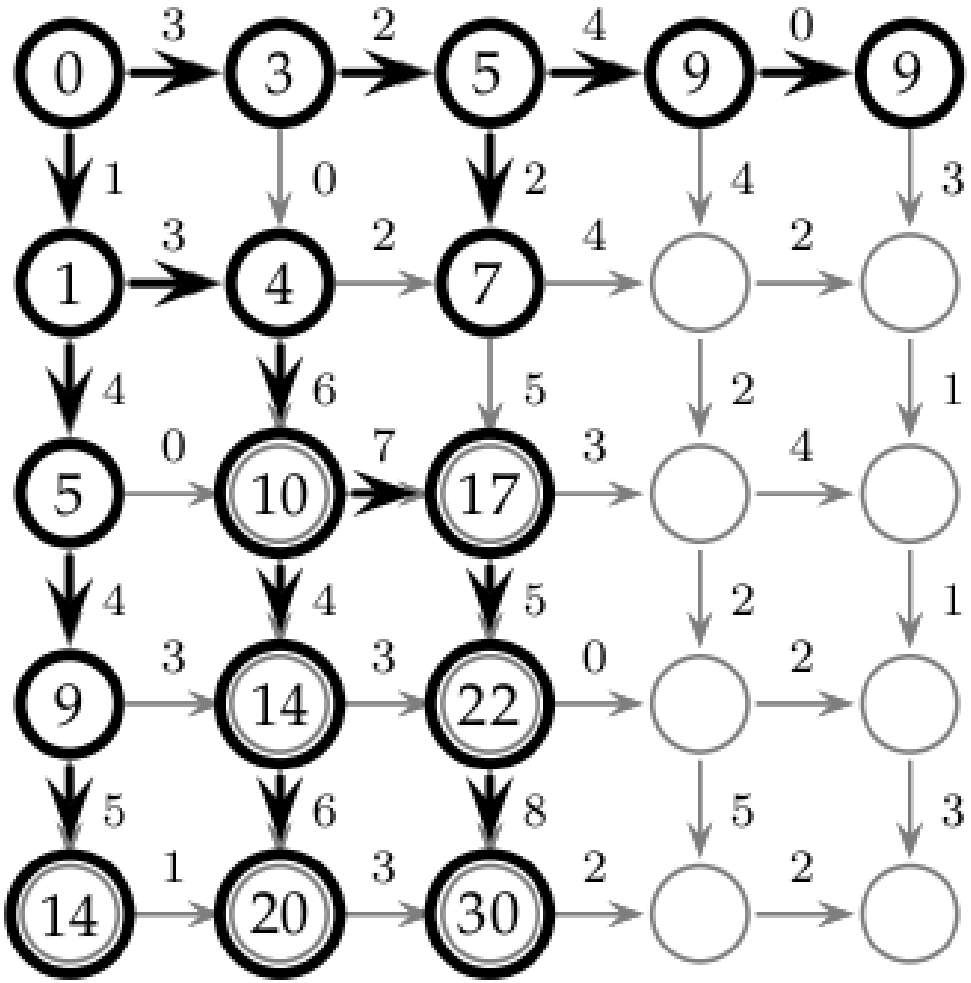
$$s_{i,j} = \max \begin{cases} s_{i-1,j} + \text{ weight of the edge between } (i-1,j) \text{ and } (i,j) \\ s_{i,j-1} + \text{ weight of the edge between } (i,j-1) \text{ and } (i,j) \end{cases}$$

Dynamic Programming



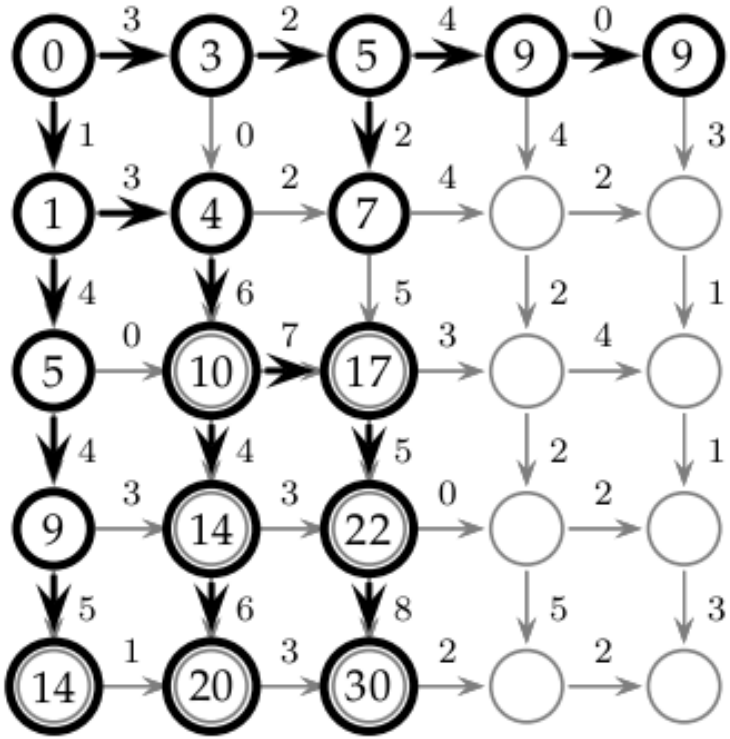
$$s_{i,j} = \max \begin{cases} s_{i-1,j} + \text{ weight of the edge between } (i-1,j) \text{ and } (i,j) \\ s_{i,j-1} + \text{ weight of the edge between } (i,j-1) \text{ and } (i,j) \end{cases}$$

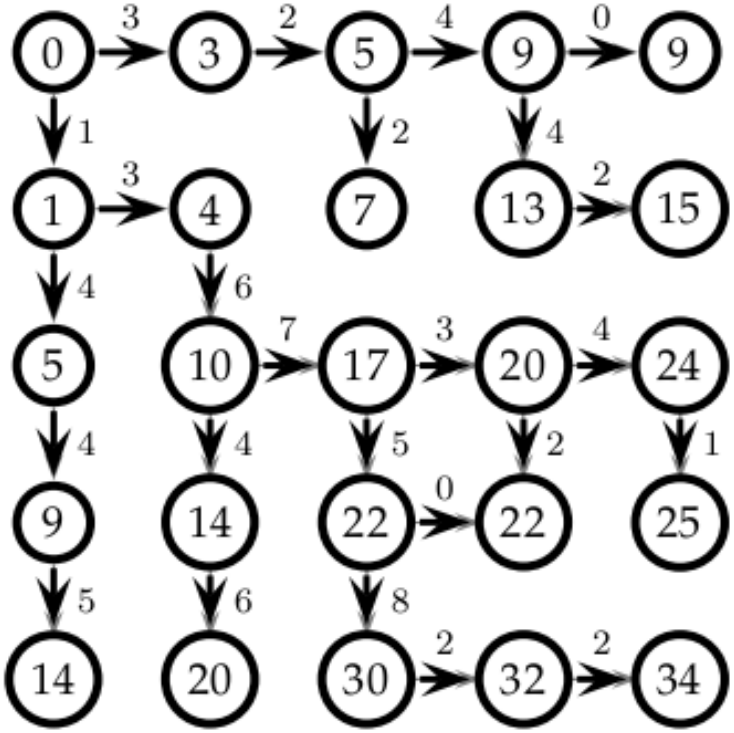
Dynamic Programming



$$s_{i,j} = \max \begin{cases} s_{i-1,j} + \text{ weight of the edge between } (i-1,j) \text{ and } (i,j) \\ s_{i,j-1} + \text{ weight of the edge between } (i,j-1) \text{ and } (i,j) \end{cases}$$

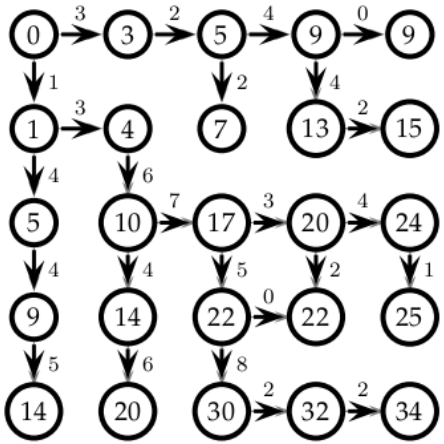
Dynamic Programming





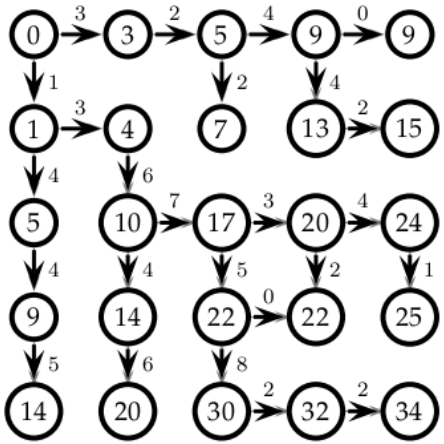
$$s_{i,j} = \max \begin{cases} s_{i-1,j} + \text{ weight of the edge between } (i-1,j) \text{ and } (i,j) \\ s_{i,j-1} + \text{ weight of the edge between } (i,j-1) \text{ and } (i,j) \end{cases}$$

Dynamic Programming



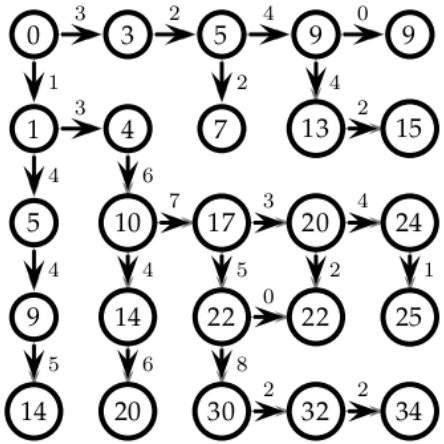
- How do you code this?

Dynamic Programming



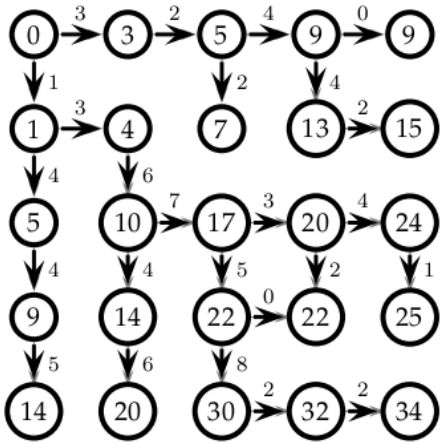
- How do you code this?
  - What are the inputs?

Dynamic Programming



- How do you code this?
  - What are the inputs?
  - What are the outputs?

Dynamic Programming



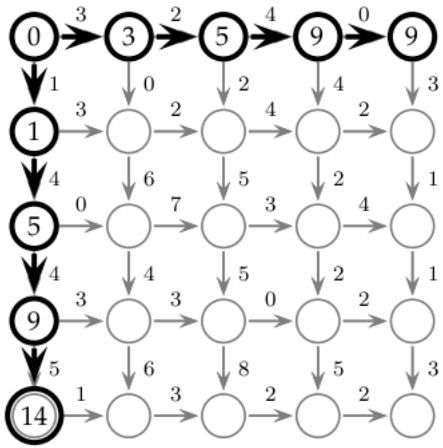
- How do you code this?
  - What are the inputs?

    Graph with weighted edges
  - What are the outputs?

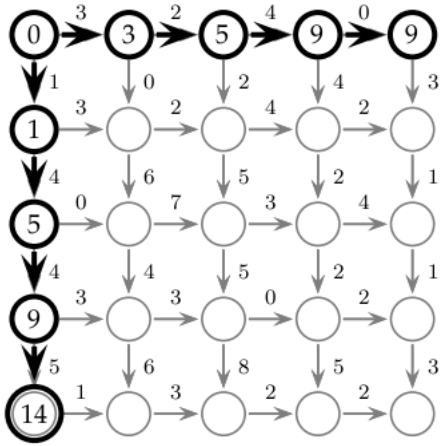
    Final score (could also give path)

Dynamic Programming

- How do you code this?

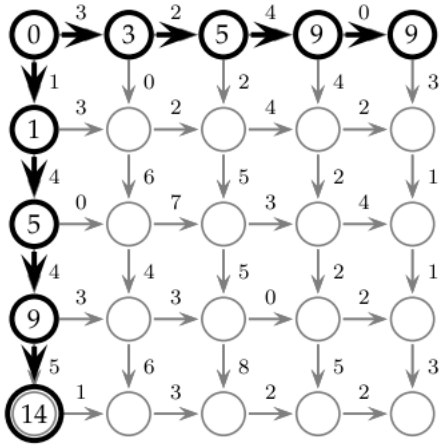


Dynamic Programming



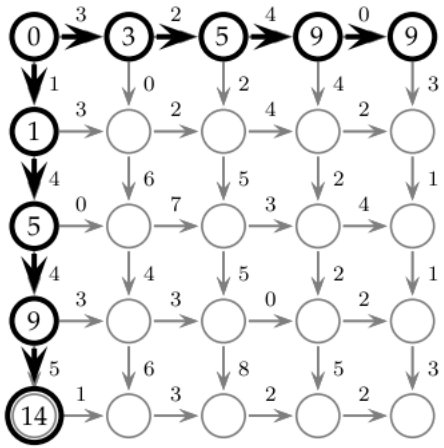
- How do you code this?
  1. Fill in the edge values.

Dynamic Programming



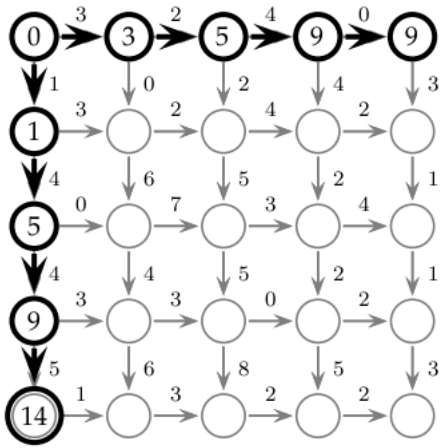
- How do you code this?
  1. Fill in the edge values.
  2. Fill in the second column, using the first.

Dynamic Programming



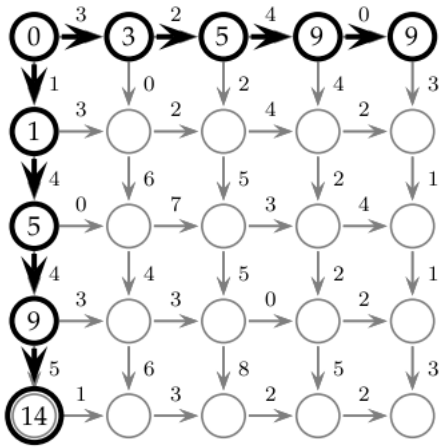
- How do you code this?
  1. Fill in the edge values.
  2. Fill in the second column, using the first.
  3. Fill in the third column, using the second.

Dynamic Programming



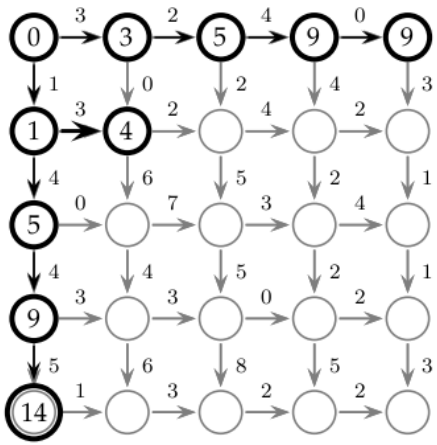
- How do you code this?
  1. Fill in the edge values.
  2. Fill in the second column, using the first.
  3. Fill in the third column, using the second.
  4. Keep repeating until entire array is filled.

Dynamic Programming



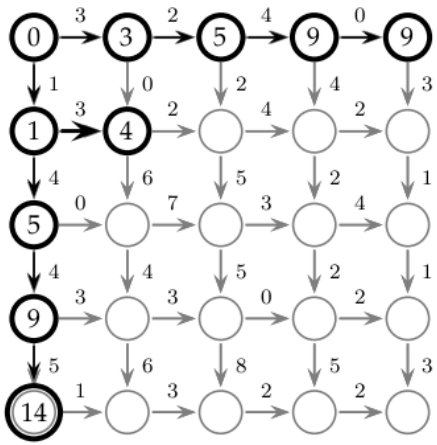
- How do you code this?
  1. Fill in the edge values.
  2. Fill in the second column, using the first.
  3. Fill in the third column, using the second.
  4. Keep repeating until entire array is filled. Nested `for`-loops: for each column and for each row.
- What data structures do you need?

Dynamic Programming



- How do you code this?
  1. Fill in the edge values.
  2. Fill in the second column, using the first.
  3. Fill in the third column, using the second.
  4. Keep repeating until entire array is filled. Nested `for`-loops: for each column and for each row.
- What data structures do you need?
  - *Grid (array) to store the best subproblems answers.*

Dynamic Programming



- How do you code this?
  1. Fill in the edge values.
  2. Fill in the second column, using the first.
  3. Fill in the third column, using the second.
  4. Keep repeating until entire array is filled. Nested `for`-loops: for each column and for each row.
- What data structures do you need?
  - *Grid (array) to store the best subproblems answers.*
  - *The path, or could 'read' if off grid.*

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

Dynamic Programming & Sequence Alignment

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

- How do you (globally) align two sequences?

Dynamic Programming & Sequence Alignment

A C G T C C T C

A C G C C T A C

- How do you (globally) align two sequences?
- Needleman-Wunsch Algorithm: use dynamic programming.

Pairwise Sequence Alignment

- Look at smaller example: `AGG` and `AGAG`.

		A	G	A	G
	0				
A					
G					
G					

Pairwise Sequence Alignment

		A	G	A	G
	0				
A					
G					
G					

- Look at smaller example: `AGG` and `AGAG`.
- Same basic set up as a 'grid'.

Pairwise Sequence Alignment

		A	G	A	G
	0				
A					
G					
G					

- Look at smaller example: `AGG` and `AGAG`.
- Same basic set up as a 'grid'.
- Add an extra row and column to allow for a gap at the beginning of sequence.

Pairwise Sequence Alignment

		A	G	A	G
	0				
A					
G					
G					

- Look at smaller example: `AGG` and `AGAG`.
- Same basic set up as a 'grid'.
- Add an extra row and column to allow for a gap at the beginning of sequence.
- Start the grid with `0`.

Pairwise Sequence Alignment

		A	G	A	G
	0				
A					
G					
G					

- Look at smaller example: `AGG` and `AGAG`.
- Same basic set up as a 'grid'.
- Add an extra row and column to allow for a gap at the beginning of sequence.
- Start the grid with `0`.
- Scoring:

Pairwise Sequence Alignment

		A	G	A	G
	0				
A					
G					
G					

- Look at smaller example: `AGG` and `AGAG`.
- Same basic set up as a 'grid'.
- Add an extra row and column to allow for a gap at the beginning of sequence.
- Start the grid with `0`.
- Scoring:
  - +1 for a match

Pairwise Sequence Alignment

		A	G	A	G
	0				
A					
G					
G					

- Look at smaller example: `AGG` and `AGAG`.
- Same basic set up as a 'grid'.
- Add an extra row and column to allow for a gap at the beginning of sequence.
- Start the grid with `0`.
- Scoring:
  - +1 for a match
  - −1 for a mismatch

Pairwise Sequence Alignment

		A	G	A	G
	0				
A					
G					
G					

- Look at smaller example: `AGG` and `AGAG`.
- Same basic set up as a 'grid'.
- Add an extra row and column to allow for a gap at the beginning of sequence.
- Start the grid with `0`.
- Scoring:
  - +1 for a match
  - −1 for a mismatch
- Now allowed to go 'diagonally' (match the elements)

Pairwise Sequence Alignment

		A	G	A	G
	0				
A	-1				
G					
G					

- Look at smaller example of `AGG` vs `AGAG`.
- Same basic set up as a 'grid'.
- Add an extra row and column to allow for a gap at the beginning of sequence.
- Start the grid with `0`.
- Scoring:
  - +1 for a match.
  - −1 for a mismatch.
- Now allowed to go 'diagonally' (match the elements).

Pairwise Sequence Alignment

		A	G	A	G
	0				
A	-1				
G	-2				
G					

- Look at smaller example of `AGG` vs `AGAG`.
- Same basic set up as a 'grid'.
- Add an extra row and column to allow for a gap at the beginning of sequence.
- Start the grid with `0`.
- Scoring:
  - +1 for a match.
  - −1 for a mismatch.
- Now allowed to go 'diagonally' (match the elements).

Pairwise Sequence Alignment

		A	G	A	G
	0				
A	-1				
G	-2				
G	-3				

- Look at smaller example of `AGG` vs `AGAG`.
- Same basic set up as a 'grid'.
- Add an extra row and column to allow for a gap at the beginning of sequence.
- Start the grid with `0`.
- Scoring:
  - +1 for a match.
  - −1 for a mismatch.
- Now allowed to go 'diagonally' (match the elements).

Pairwise Sequence Alignment

		A	G	A	G
	0	-1			
A	-1				
G	-2				
G	-3				

- Look at smaller example of `AGG` vs `AGAG`.
- Same basic set up as a 'grid'.
- Add an extra row and column to allow for a gap at the beginning of sequence.
- Start the grid with `0`.
- Scoring:
  - +1 for a match.
  - −1 for a mismatch.
- Now allowed to go 'diagonally' (match the elements).

Pairwise Sequence Alignment

		A	G	A	G
	0	-1	-2	-3	-4
A	-1				
G	-2				
G	-3				

- Look at smaller example of `AGG` vs `AGAG`.
- Same basic set up as a 'grid'.
- Add an extra row and column to allow for a gap at the beginning of sequence.
- Start the grid with `0`.
- Scoring:
  - +1 for a match.
  - −1 for a mismatch.
- Now allowed to go 'diagonally' (match the elements).

Pairwise Sequence Alignment

- For each, we take the best of:

		A	G	A	G
	0	-1	-2	-3	-4
A	-1	1			
G	-2				
G	-3				

Pairwise Sequence Alignment

		A	G	A	G
	0	-1	-2	-3	-4
A	-1	1			
G	-2				
G	-3				

- For each, we take the best of:
  - matching elements $(+1)$.

Pairwise Sequence Alignment

		A	G	A	G
	0	-1	-2	-3	-4
A	-1	1			
G	-2				
G	-3				

- For each, we take the best of:
  - matching elements $(+1)$.
  - adding a gap to first sequence (-1).

Pairwise Sequence Alignment

		A	G	A	G
	0	-1	-2	-3	-4
A	-1	1			
G	-2				
G	-3				

- For each, we take the best of:
  - matching elements $(+1)$.
  - adding a gap to first sequence (-1).
  - adding a gap to second sequence (-1).

Pairwise Sequence Alignment

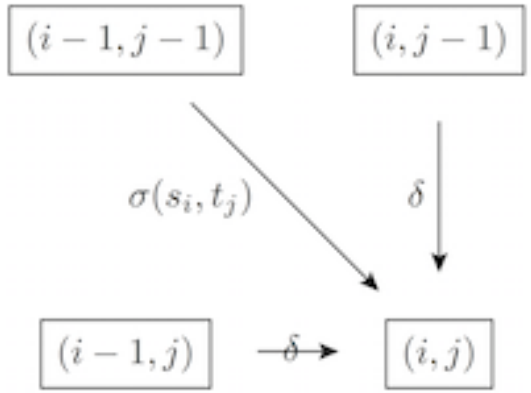
		A	G	A	G
	0	-1	-2	-3	-4
A	-1	1			
G	-2				
G	-3				

- For each, we take the best of:
  - matching elements $(+1)$.
  - adding a gap to first sequence (-1).
  - adding a gap to second sequence (-1).
- Pictorially:

Pairwise Sequence Alignment

		A	G	A	G
	0	-1	-2	-3	-4
A	-1	1			
G	-2				
G	-3				

- For each, we take the best of:
  - matching elements $(+1)$.
  - adding a gap to first sequence (-1).
  - adding a gap to second sequence (-1).
- Pictorially:



Pairwise Sequence Alignment

		A	G	A	G
	0	-1	-2	-3	-4
A	-1	1			
G	-2				
G	-3				

- For each, we take the best of:
  - matching elements (+1).
  - adding a gap to first sequence (-1).
  - adding a gap to second sequence (-1).
- As equations:

$$S(s_{0..i}, t_{0..j}) = \max \begin{cases} \sigma(s_i, t_j) + S(s_{0..i-1}, t_{0..j-1}) \\ -\delta + S(s_{0..i-1}, t_{0..j}) \\ -\delta + S(s_{0..i}, t_{0..j-1}) \end{cases}$$

where:

$$\delta = 1 \text{ and } \sigma(s,t) = \begin{cases} 1 \text{ if } s = t \\ -1 \text{ otherwise} \end{cases}.$$

Pairwise Sequence Alignment

		A	G	A	G
	0	-1	-2	-3	-4
A	-1	1			
G	-2	0			
G	-3				

- For each, we take the best of:
  - matching elements (+1).
  - adding a gap to first sequence (-1).
  - adding a gap to second sequence (-1).
- As equations:

$$S(s_{0..i}, t_{0..j}) = \max \begin{cases} \sigma(s_i, t_j) + S(s_{0..i-1}, t_{0..j-1}) \\ -\delta + S(s_{0..i-1}, t_{0..j}) \\ -\delta + S(s_{0..i}, t_{0..j-1}) \end{cases}$$

where:

$$\delta = 1 \text{ and } \sigma(s,t) = \begin{cases} 1 \text{ if } s = t \\ -1 \text{ otherwise} \end{cases}.$$

Pairwise Sequence Alignment

		A	G	A	G
	0	-1	-2	-3	-4
A	-1	1			
G	-2	0			
G	-3	-1			

- For each, we take the best of:
  - matching elements (+1).
  - adding a gap to first sequence (-1).
  - adding a gap to second sequence (-1).
- As equations:

$$S(s_{0..i}, t_{0..j}) = \max \begin{cases} \sigma(s_i, t_j) + S(s_{0..i-1}, t_{0..j-1}) \\ -\delta + S(s_{0..i-1}, t_{0..j}) \\ -\delta + S(s_{0..i}, t_{0..j-1}) \end{cases}$$

where:

$$\delta = 1 \text{ and } \sigma(s,t) = \begin{cases} 1 \text{ if } s = t \\ -1 \text{ otherwise} \end{cases}.$$

In Pairs

		A	G	A	G
	0	-1	-2	-3	-4
A	-1	1			
G	-2	0			
G	-3	-1			

$$S(s_{0..i}, t_{0..j}) = \max \begin{cases} \sigma(s_i, t_j) + S(s_{0..i-1}, t_{0..j-1}) \\ -\delta + S(s_{0..i-1}, t_{0..j}) \\ -\delta + S(s_{0..i}, t_{0..j-1}) \end{cases}$$

where

$$\delta = 1 \text{ and } \sigma(s,t) = \begin{cases} 1 \text{ if } s = t \\ -1 \text{ otherwise} \end{cases}.$$

1. Finish this alignment.
   - What is the score?
   - What is the actual alignment?

In Pairs

		A	G	A	G
	0	-1	-2	-3	-4
A	-1	1			
G	-2	0			
G	-3	-1			

$$S(s_{0..i}, t_{0..j}) = \max \begin{cases} \sigma(s_i, t_j) + S(s_{0..i-1}, t_{0..j-1}) \\ -\delta + S(s_{0..i-1}, t_{0..j}) \\ -\delta + S(s_{0..i}, t_{0..j-1}) \end{cases}$$

where

$$\delta = 1 \text{ and } \sigma(s,t) = \begin{cases} 1 \text{ if } s = t \\ -1 \text{ otherwise} \end{cases}.$$

1. Finish this alignment.
   - What is the score?
   - What is the actual alignment?
2. Align the sequences:

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

In Pairs

		A	G	A	G
	0	-1	-2	-3	-4
A	-1	1			
G	-2	0			
G	-3	-1			

$$S(s_{0..i}, t_{0..j}) = \max \begin{cases} \sigma(s_i, t_j) + S(s_{0..i-1}, t_{0..j-1}) \\ -\delta + S(s_{0..i-1}, t_{0..j}) \\ -\delta + S(s_{0..i}, t_{0..j-1}) \end{cases}$$

where

$$\delta = 1 \text{ and } \sigma(s,t) = \begin{cases} 1 \text{ if } s = t \\ -1 \text{ otherwise} \end{cases}.$$

1. Finish this alignment.
   - What is the score?
   - What is the actual alignment?
2. Align the sequences:

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

3. What happens if we score +5 for match and -4 for mismatch (NCBI's `blastn`)?

In Pairs

		A	G	A	G
	0	-1	-2	-3	-4
A	-1	1			
G	-2	0			
G	-3	-1			

$$S(s_{0..i}, t_{0..j}) = \max \begin{cases} \sigma(s_i, t_j) + S(s_{0..i-1}, t_{0..j-1}) \\ -\delta + S(s_{0..i-1}, t_{0..j}) \\ -\delta + S(s_{0..i}, t_{0..j-1}) \end{cases}$$

where

$$\delta = 1 \text{ and } \sigma(s,t) = \begin{cases} 1 \text{ if } s = t \\ -1 \text{ otherwise} \end{cases}.$$

1. Finish this alignment.
   - What is the score?
   - What is the actual alignment?
2. Align the sequences:

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

3. What happens if we score +5 for match and -4 for mismatch (NCBI's `blastn`)?
4. What happens if σ scores pairs differently?

$$\sigma(s,t) = \begin{cases} 2 & \text{if } s = t \\ -1 & \text{if } s \neq t, s, t \in \{A, G\} \\ -1 & \text{if } s \neq t, s, t \in \{C, T\} \\ -2 & \text{otherwise} \end{cases}$$

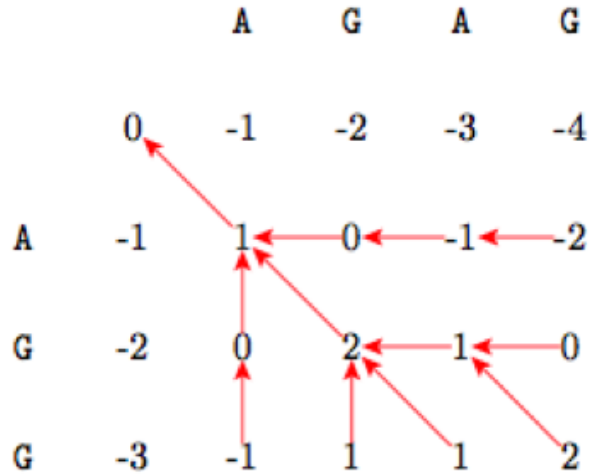
(Weighing transversions and transitions differently.)

First Example

		A	G	A	G
	0	-1	-2	-3	-4
A	-1	1	0	-1	-2
G	-2	-1	2	1	0
G	-3	-1	1	1	2

First Example

		A	G	A	G
	0	-1	-2	-3	-4
A	-1	1	0	-1	-2
G	-2	-1	2	1	0
G	-3	-1	1	1	2



Recap: In Pairs

		A	G	A	G
	0				
A					
G					
G					

$$S(s_{0..i}, t_{0..j}) = \max \begin{cases} \sigma(s_i, t_j) + S(s_{0..i-1}, t_{0..j-1}) \\ -\delta + S(s_{0..i-1}, t_{0..j}) \\ -\delta + S(s_{0..i}, t_{0..j-1}) \end{cases}$$

where

$$\delta = 1 \text{ and } \sigma(s,t) = \begin{cases} 1 \text{ if } s = t \\ -1 \text{ otherwise} \end{cases}.$$

1. Finish this alignment.
   - What is the score?
   - What is the actual alignment?
2. Align the sequences:

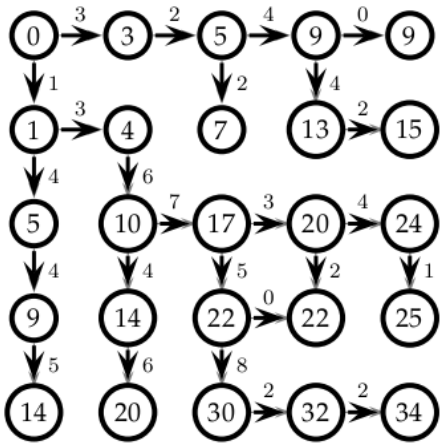
   ```
   A C G T C C T C
   A C G C C T A C
   ```

3. What happens if we score +5 for match and -4 for mismatch (NCBI's `blastn`)?
4. What happens if σ scores pairs differently?

$$\sigma(s,t) = \begin{cases} 1 & \text{if } s = t \\ -1 & \text{if } s \neq t, s, t \in \{A, G\} \\ -1 & \text{if } s \neq t, s, t \in \{C, T\} \\ -2 & \text{otherwise} \end{cases}$$

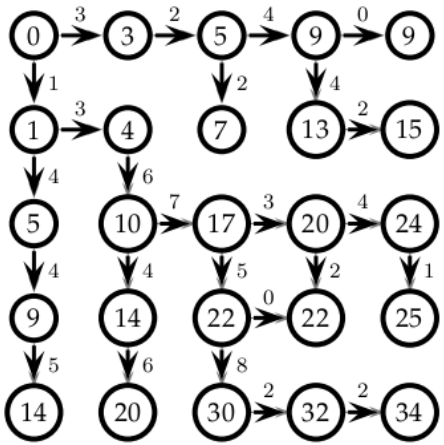
(Transversions with twice the weight as transitions.)

Variations on the Theme



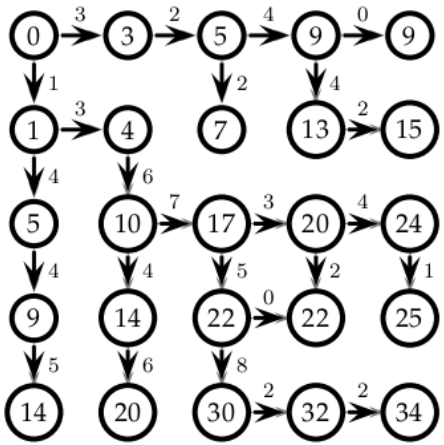
- Algorithm above is due to Needleman & Wunsch.
- On Wednesday, we will look at variations on the theme:
  - Locally aligning sections of the sequences (Smith-Waterman).

Variations on the Theme



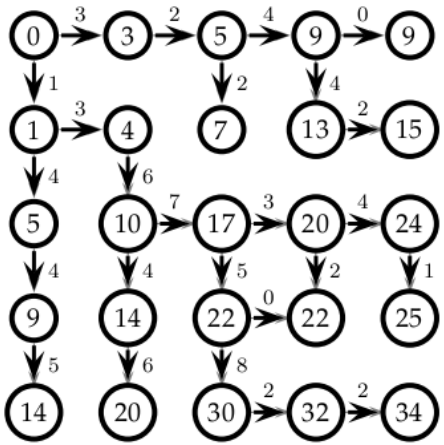
- Algorithm above is due to Needleman & Wunsch.
- On Wednesday, we will look at variations on the theme:
  - Locally aligning sections of the sequences (Smith-Waterman).
  - Different models for scoring gaps.

Variations on the Theme



- Algorithm above is due to Needleman & Wunsch.
- On Wednesday, we will look at variations on the theme:
  - Locally aligning sections of the sequences (Smith-Waterman).
  - Different models for scoring gaps.
  - Aligning protein sequences using scores that differ (PAM & BLOSUM scoring matrices).

Variations on the Theme



- Algorithm above is due to Needleman & Wunsch.
- On Wednesday, we will look at variations on the theme:
  - Locally aligning sections of the sequences (Smith-Waterman).
  - Different models for scoring gaps.
  - Aligning protein sequences using scores that differ (PAM & BLOSUM scoring matrices).

Pairwise vs. Multiple Sequence Alignment

- While pairwise alignment can be done efficiently, multiple sequence alignment is hard.

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

Pairwise vs. Multiple Sequence Alignment

- While pairwise alignment can be done efficiently, multiple sequence alignment is hard.
- Which sequences do you align first?

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

Pairwise vs. Multiple Sequence Alignment

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

- While pairwise alignment can be done efficiently, multiple sequence alignment is hard.
- Which sequences do you align first?
- Often use a guess at which species are most closely related ('guide tree'):

Pairwise vs. Multiple Sequence Alignment

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

- While pairwise alignment can be done efficiently, multiple sequence alignment is hard.
- Which sequences do you align first?
- Often use a guess at which species are most closely related ('guide tree'):
  - Align pairwise (closely related sequences first and work towards the root),

Pairwise vs. Multiple Sequence Alignment

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

- While pairwise alignment can be done efficiently, multiple sequence alignment is hard.
- Which sequences do you align first?
- Often use a guess at which species are most closely related ('guide tree'):
  - Align pairwise (closely related sequences first and work towards the root),
  - Re-compute the guide tree based on the new alignment, and

Pairwise vs. Multiple Sequence Alignment

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

- While pairwise alignment can be done efficiently, multiple sequence alignment is hard.
- Which sequences do you align first?
- Often use a guess at which species are most closely related ('guide tree'):
  - Align pairwise (closely related sequences first and work towards the root),
  - Re-compute the guide tree based on the new alignment, and
  - Repeat.

Pairwise vs. Multiple Sequence Alignment

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

- While pairwise alignment can be done efficiently, multiple sequence alignment is hard.
- Which sequences do you align first?
- Often use a guess at which species are most closely related ('guide tree'):
  - Align pairwise (closely related sequences first and work towards the root),
  - Re-compute the guide tree based on the new alignment, and
  - Repeat.
- Other approaches build the tree and alignment simultaneously.

Finding Trees in Graphs

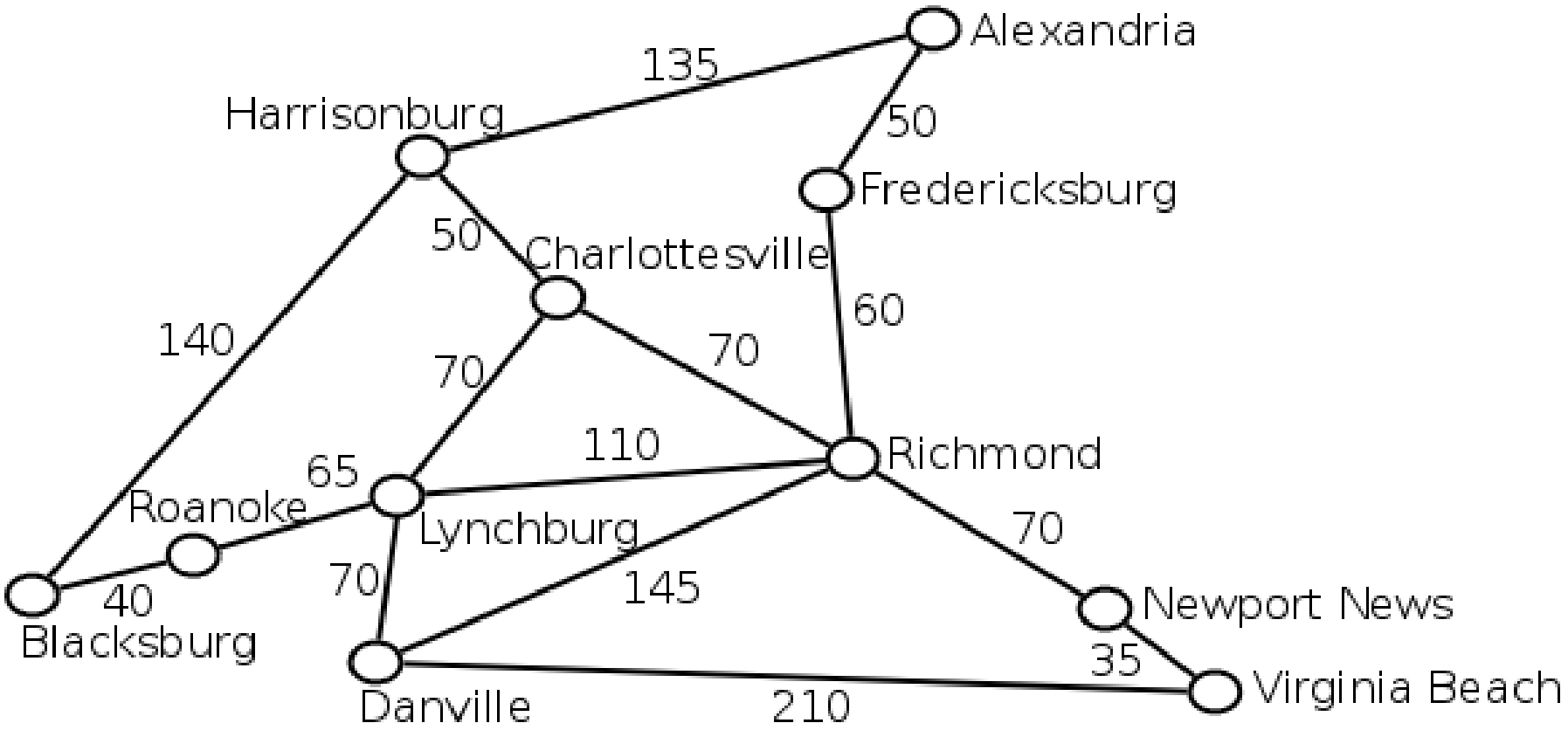


Image from Iam Finlayson (U. Mary Washington).

Finding Trees in Graphs

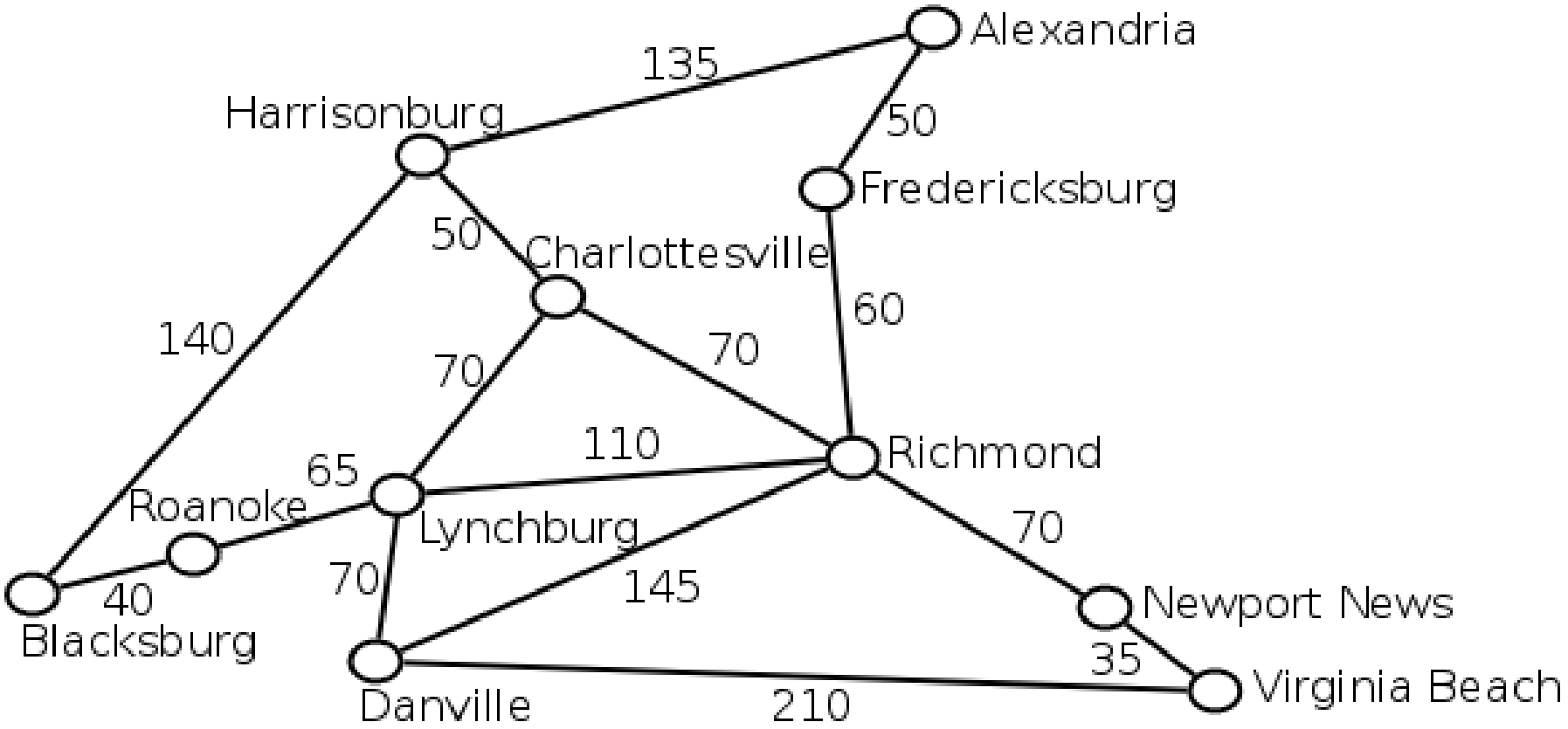
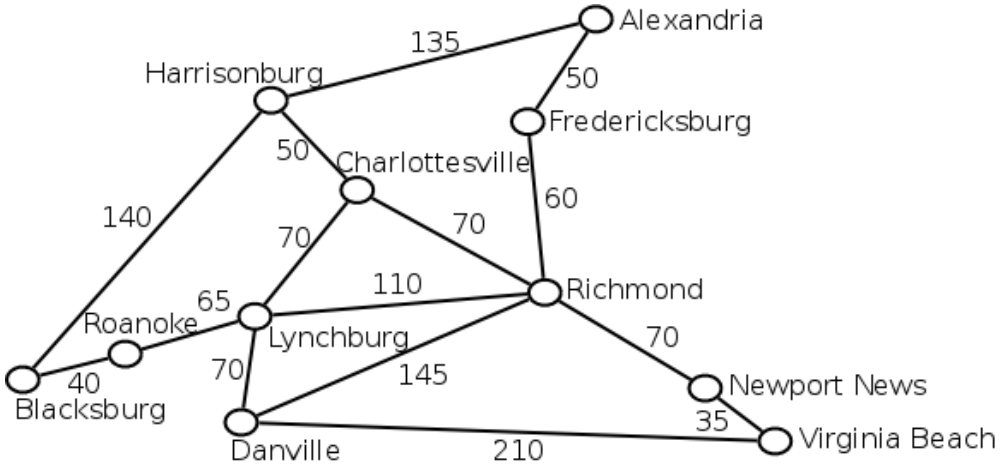


Image from Iam Finlayson (U. Mary Washington).

What is the shortest way to connect all the cities?

Finding Trees in Graphs



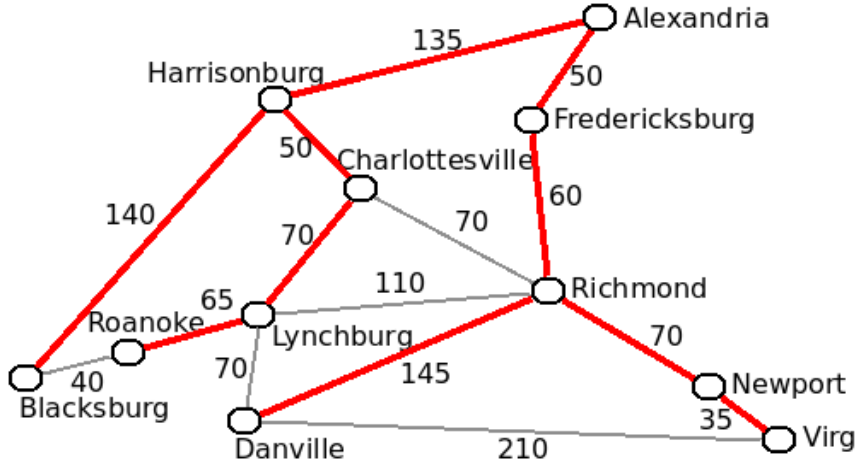
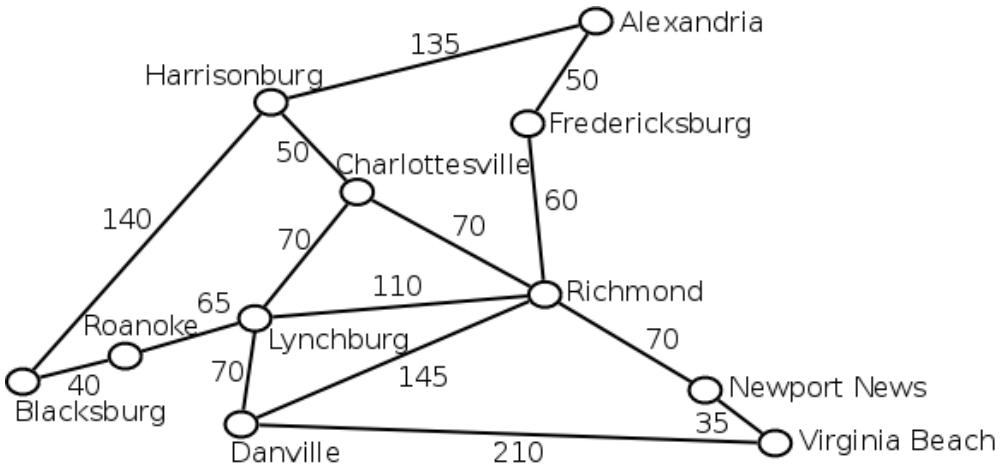


Image from Iam Finlayson (U. Mary Washington).

Finding Trees in Graphs



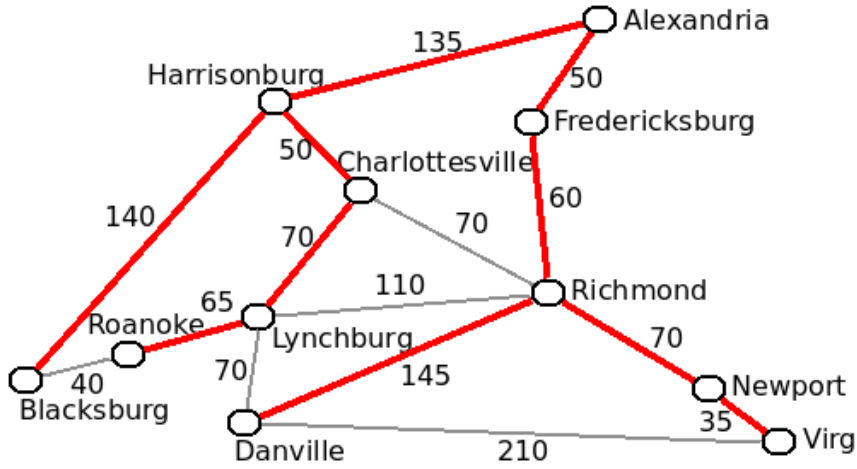
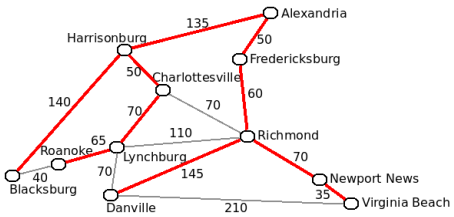


Image from Iam Finlayson (U. Mary Washington).

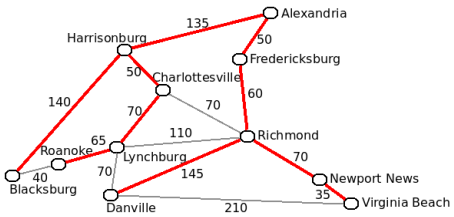
What is the shortest way to connect all the cities?

Finding Trees in Graphs



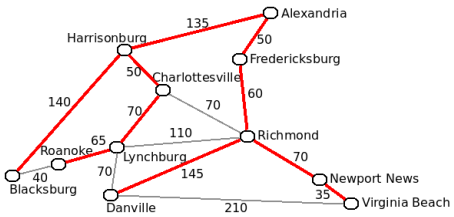
- Given a weighted graph, a minimum spanning tree (MST) is a tree on a subset of the edges that has minimum weight.

Finding Trees in Graphs



- Given a weighted graph, a minimum spanning tree (MST) is a tree on a subset of the edges that has minimum weight.
- Different than phylogenetic trees: MSTs can only use nodes given; does not add in hypothetical ancestors.

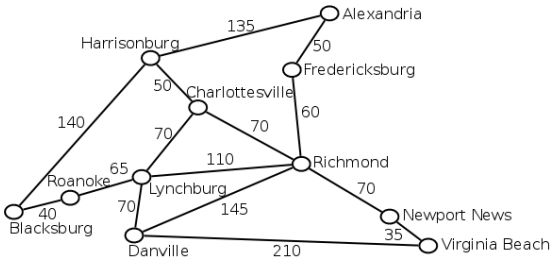
Finding Trees in Graphs



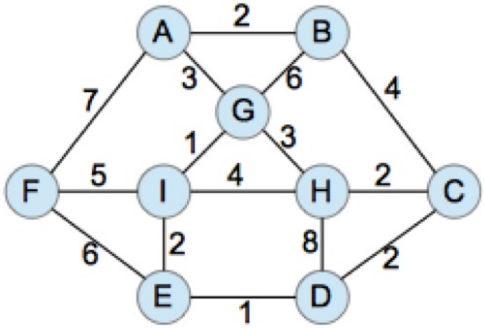
- Given a weighted graph, a minimum spanning tree (MST) is a tree on a subset of the edges that has minimum weight.
- Different than phylogenetic trees: MSTs can only use nodes given; does not add in hypothetical ancestors.
- MST's can be computed quickly and very common.

In Pairs

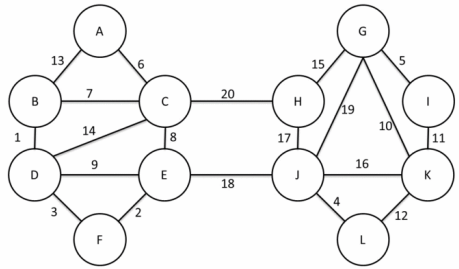
Find a minimum spanning tree (MST) for each:



(Ian Finlayson)

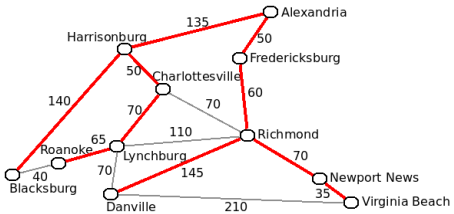


Chegg



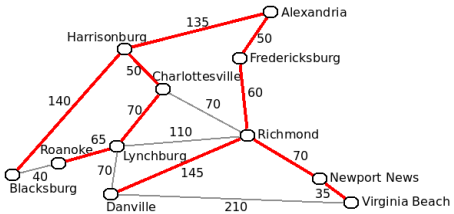
Aliesaraj (wiki)

Kruskal's & Prim's Algorithms



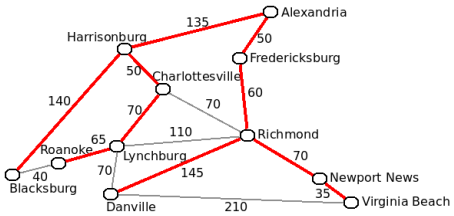
- Kruskal: Sort edges, add in lowest weight remaining edge that does not make a cycle.

Kruskal's & Prim's Algorithms



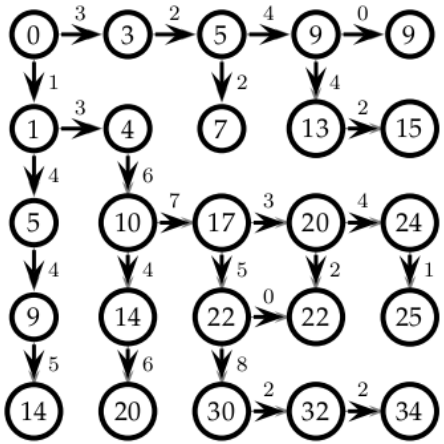
- Kruskal: Sort edges, add in lowest weight remaining edge that does not make a cycle.
- Prim: Build tree by adding next connected edge that does not make a cycle.

Kruskal's & Prim's Algorithms



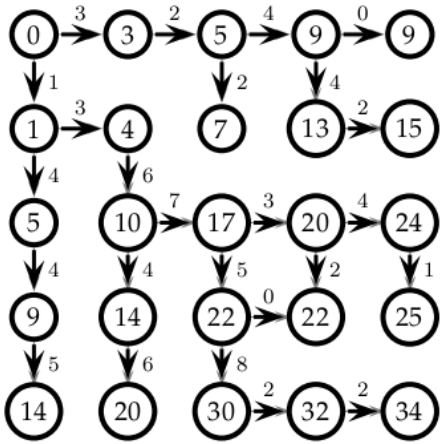
- Kruskal: Sort edges, add in lowest weight remaining edge that does not make a cycle.
- Prim: Build tree by adding next connected edge that does not make a cycle.
- Both are fast: if n is the number of vertices:
  - Kruskal: $O(n^2 \log n)$.
  - Prim: $O(n^2)$ with adjacency matrix.
  - Both can be made faster with fancier ways to store possible edges.

Recap



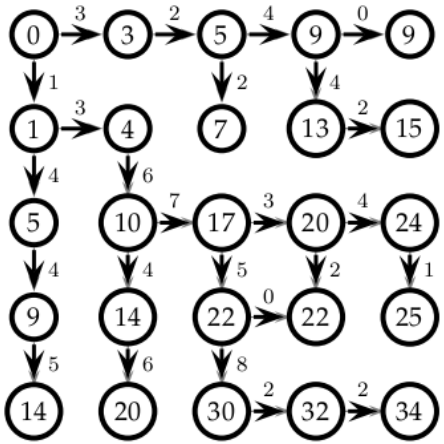
- Dynamic Programming: very useful tool when problems can be broken down into well structured parts.

Recap



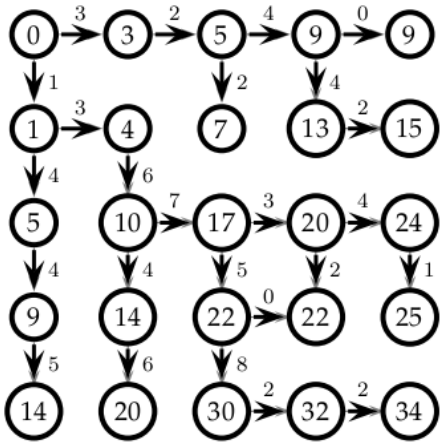
- Dynamic Programming: very useful tool when problems can be broken down into well structured parts.
- Minimum Spanning Trees: fast way to extract a tree from a graph (different from phylo trees since no new ancestor nodes).

Recap



- Dynamic Programming: very useful tool when problems can be broken down into well structured parts.
- Minimum Spanning Trees: fast way to extract a tree from a graph (different from phylo trees since no new ancestor nodes).
- Email lab reports to `kstjohn@amnh.org`.

Recap



- Dynamic Programming: very useful tool when problems can be broken down into well structured parts.
- Minimum Spanning Trees: fast way to extract a tree from a graph (different from phylo trees since no new ancestor nodes).
- Email lab reports to `kstjohn@amnh.org`.
- Challenges available at `rosalind.info`.