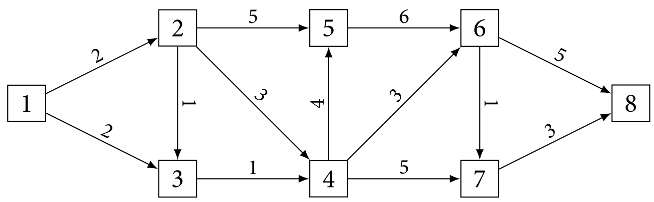
Lesson 10. The Principle of Optimality and Formulating Recursions

0 Warm up

Example 1. Consider the following directed graph. The labels on the edges are edge lengths.



In this order:

a. Find a shortest path from node 1 to node 8. What is its length?

Path:	$1 \to 3 \to 4 \to 6 \to 7 \to 8$	Length:	10

b. Find a shortest path from node 3 to node 8. What is its length?

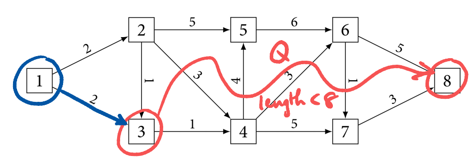
Path:	$3 \to 4 \to 6 \to 7 \to 8$	Length:	8

c. Find a shortest path from node 4 to node 8. What is its length?

Path:	$4 \to 6 \to 7 \to 8$	Length:	7

1 The principle of optimality

- Let P be the path $1 \to 3 \to 4 \to 6 \to 7 \to 8$ in the graph for Example 1
  - P is a shortest path from node 1 to node 8, and has length 10
- Let P' be the path $3 \to 4 \to 6 \to 7 \to 8$
  - P' is a **subpath** of P with length 8



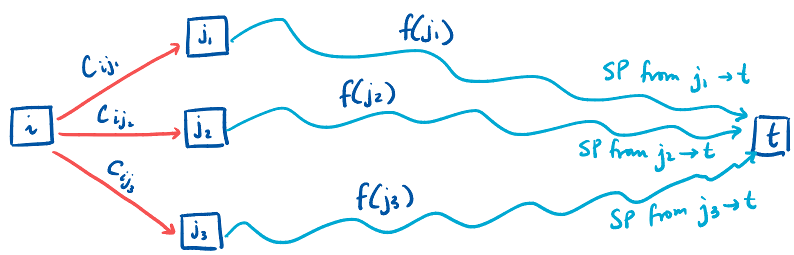
- Is P' a shortest path from node 3 to node 8?
  - Suppose we had a path Q from node 3 to node 8 with length < 8
  - Let R be the path consisting of edge $(1, 3) + Q$
  - Then, R is a path from node 1 to node 8 with length $< 2 + 8 = 10$
  - This contradicts the fact that P is a shortest path from node 1 to node 8
  - Therefore, there cannot be a path from node 3 to node 8 with length < 8 (shorter than the subpath P')

The principle of optimality (for shortest path problems)

In a directed graph with no negative cycles, optimal paths must have optimal subpaths.

- How can we exploit this?
- Suppose we want to find a shortest path
  - from source node s to sink node t
  - in a directed graph (N, E)
  - with edge lengths c_{ij} for $(i, j) \in E$
- We consider the **subproblems** of finding a shortest path from node i to node t, for every node $i \in N$
- By the principle of optimality, the shortest path from node i to node t must be:

$$\text{edge } (i, j) + \text{shortest path from } j \text{ to } t \qquad \text{for some } j \in N \text{ such that } (i, j) \in E$$



$$\text{SP length} \quad f(i) = \min\{c_{ij_1} + f(j_1),\ c_{ij_2} + f(j_2),\ c_{ij_3} + f(j_3)\}$$

2 Formulating recursions

- Let

$$f(i) = \text{length of a shortest path from node } i \text{ to node } t \qquad \text{for every node } i \in N$$

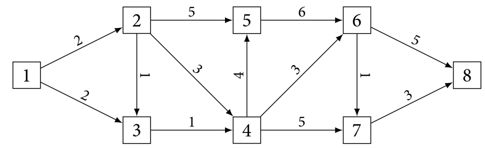
  - In other words, the function f defines the optimal values of the subproblems
- A **recursion** defines the value of a function in terms of other values of the function
- Using the principle of optimality, we can define f recursively by specifying
  - (i) the **boundary conditions** and
  - (ii) the **recursion**
- The boundary conditions provide a "base case" for the values of f:

$$f(t) = \text{length of a SP from node } t \text{ to node } t = 0$$

- The recursion specifies how the values of f are connected:

$$f(i) = \min_{\underbrace{j \text{ s.t. } (i,j) \in E}_{\substack{j \text{ can be reached} \\ \text{by an outgoing} \\ \text{edge from } i}}} \{c_{ij} + f(j)\} \qquad \text{for } i \in N,\ i \neq t$$

Example 2. Use the recursion defined above to find the length of a shortest path from nodes 1, …, 8 to $\overset{t}{\text{node 8}}$ in the graph for Example 1. Use your computations to find a shortest path from node 1 to node 8.



$f(8) =$ 0

$f(7) =$ $\min\{c_{78} + f(8)\} = \min\{\underbrace{3+0}_{(7,8)}\} = 3$

$f(6) =$ $\min\{c_{67} + f(7),\ c_{68} + f(8)\} = \min\{\underbrace{1+3}_{(6,7)},\ 5+0\} = 4$

$f(5) =$ $\min\{c_{56} + f(6)\} = \min\{6+4\} = 10$

$f(4) =$ $\min\{c_{45} + f(5),\ c_{46} + f(6),\ c_{47} + f(7)\} = \min\{4+10,\ \underbrace{3+4}_{(4,6)},\ 5+3\} = 7$

$f(3) =$ $\min\{c_{34} + f(4)\} = \min\{\underbrace{1+7}_{(3,4)}\} = 8$

$f(2) =$ $\min\{c_{23} + f(3),\ c_{24} + f(4),\ c_{25} + f(5)\} = \min\{1+8,\ 3+7,\ 5+10\} = 9$

$f(1) =$ $\min\{c_{12} + f(2),\ c_{13} + f(3)\} = \min\{2+9,\ \underbrace{2+8}_{(1,3)}\} = 10$

Shortest path from node 1 to node 8:

(1,3), (3,4), (4,6), (6,7), (7,8) ~ $1 \to 3 \to 4 \to 6 \to 7 \to 8$

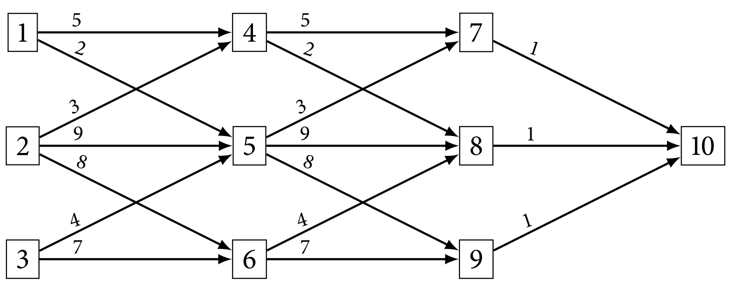
- Food for thought:
  - Does the order in which you solve the recursion matter?
  - Why did the ordering above work out for us?

3 Next lesson...

- Dynamic programs are not usually given as shortest/longest path problems as we have done over the past few lessons
- Instead, dynamic programs are usually given as recursions
- We'll get some practice using this "standard language" to describe dynamic programs

A Problems

Problem 1 (Shortest path recursions)**.** Consider the following directed graph. The edge labels correspond to edge lengths.



Use the recursion for the shortest path problem defined in Lesson 10 to

(i) Find the length of a shortest path from nodes 1, …, 10 to node 10.
(ii) Find a shortest path from node 1 to node 10.