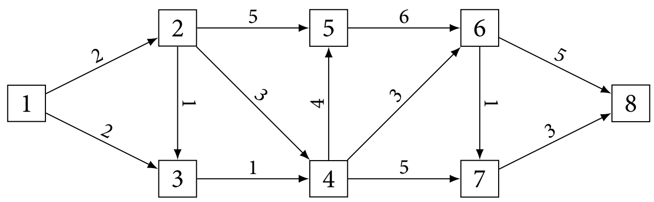
Lesson 10. The Principle of Optimality and Formulating Recursions

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: Length:

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

Path: Length:

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

Path: Length:

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
  - This contradicts the fact that
  - Therefore,

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:

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

2 Formulating recursions

- Let

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

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

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

$f(8) =$

$f(7) =$

$f(6) =$

$f(5) =$

$f(4) =$

$f(3) =$

$f(2) =$

$f(1) =$

Shortest path from node 1 to node 8:

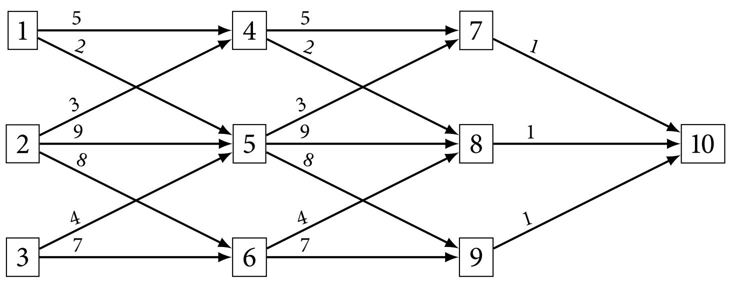
- Food for thought:
  - Does the order in which you solve the recursion matter?
  - Why did the ordering above work out for us?

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.