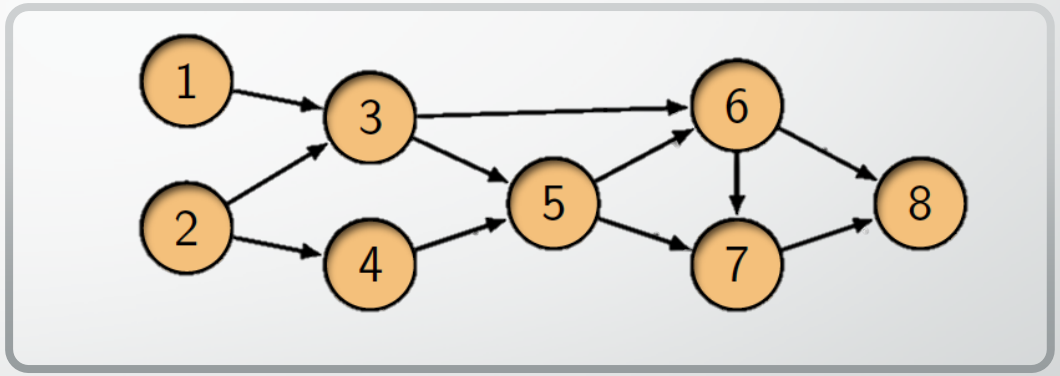
Lecture 6 – Network Flow Problems: Longest Path Problems

Module 5 – Special Cases of Linear Programs
CS/ISyE/ECE 524

Another type of transshipment problem: shortest/longest path problems

Suppose we have a directed graph with edge length. The **shortest path problem** aims to find the shortest path between two given nodes (source and sink).

- Source has supply = 1
- Sink has demand = -1
- Each edge has unlimited capacity
- Each edge has a distance (cost)
- Goal is to find the minimum "cost" path from the source to the sink



*Note: Longest path is the same, but we try to find the **maximum** cost path.*

Question: Suppose all edges have length 1 in this graph. What is the shortest distance from node 1 to every other node?

Longest Path Example: Project Planning

Like the multi-period planning problem, a project planning problem relies on the *timing* of decisions

A set of activities must be completed to complete the whole project

Activities have specified *lengths*

Activities might or might not have *predecessors* (the activity can't begin until all its predecessors are completed)

Activities may be worked on simultaneously if they don't depend on each other

The goal is to discover the minimum time needed to complete the whole project

This amounts to finding the longest path, which determines the minimum completion time

Let's look at a simple example about making breakfast that will help us understand why longest path == minimum completion time.

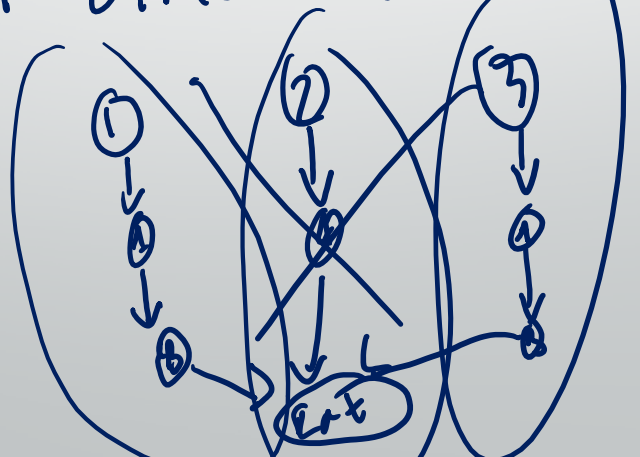
Project Planning Example: Making Breakfast

1. Coffee → Brewing (2 min) → Pouring (30 sec)

2. Toast → Toast Bread (2 min)

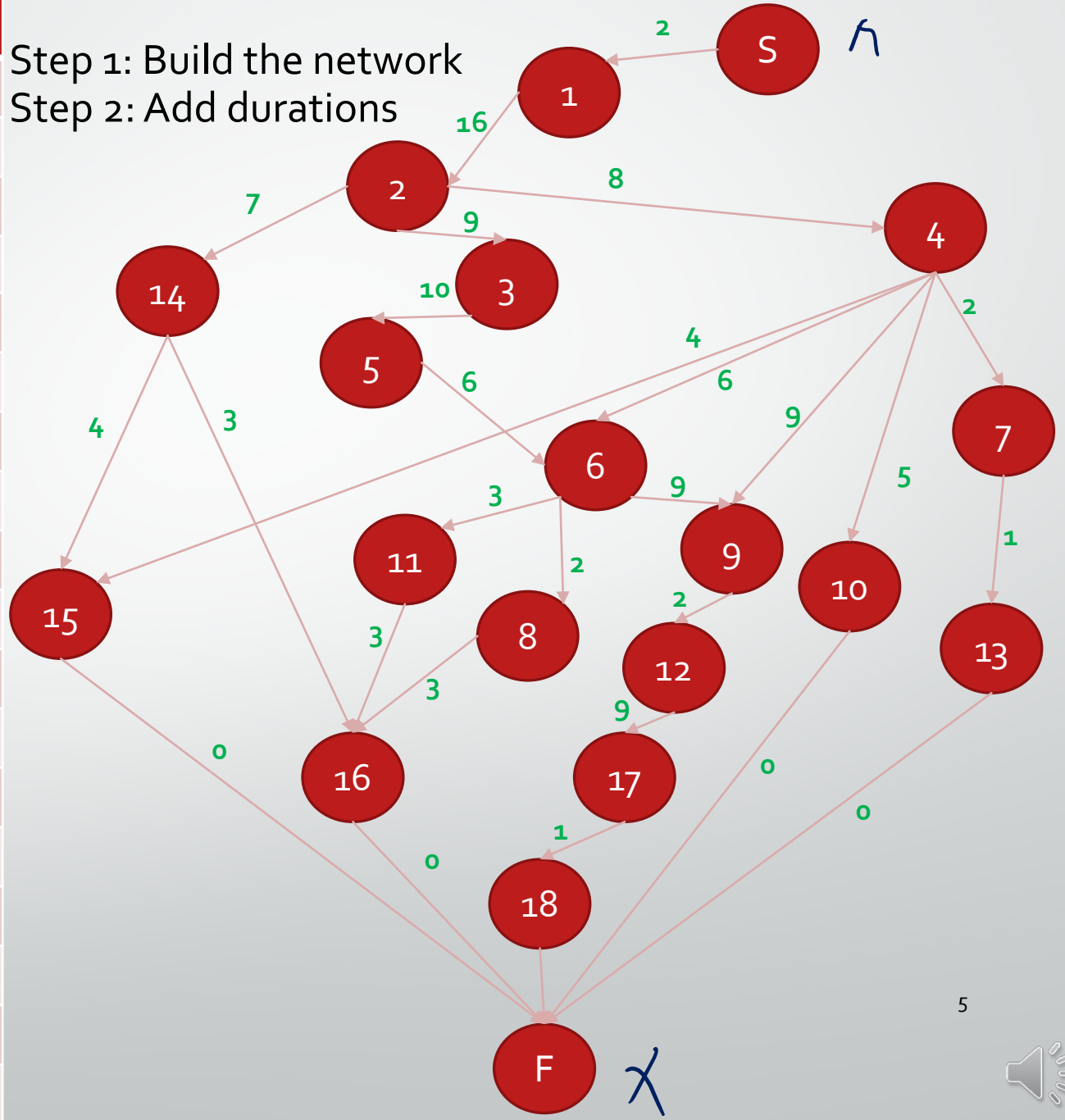
3. Eggs → Cook eggs (2 min) → Put on plate (30 sec)

Min time: 2.5 min



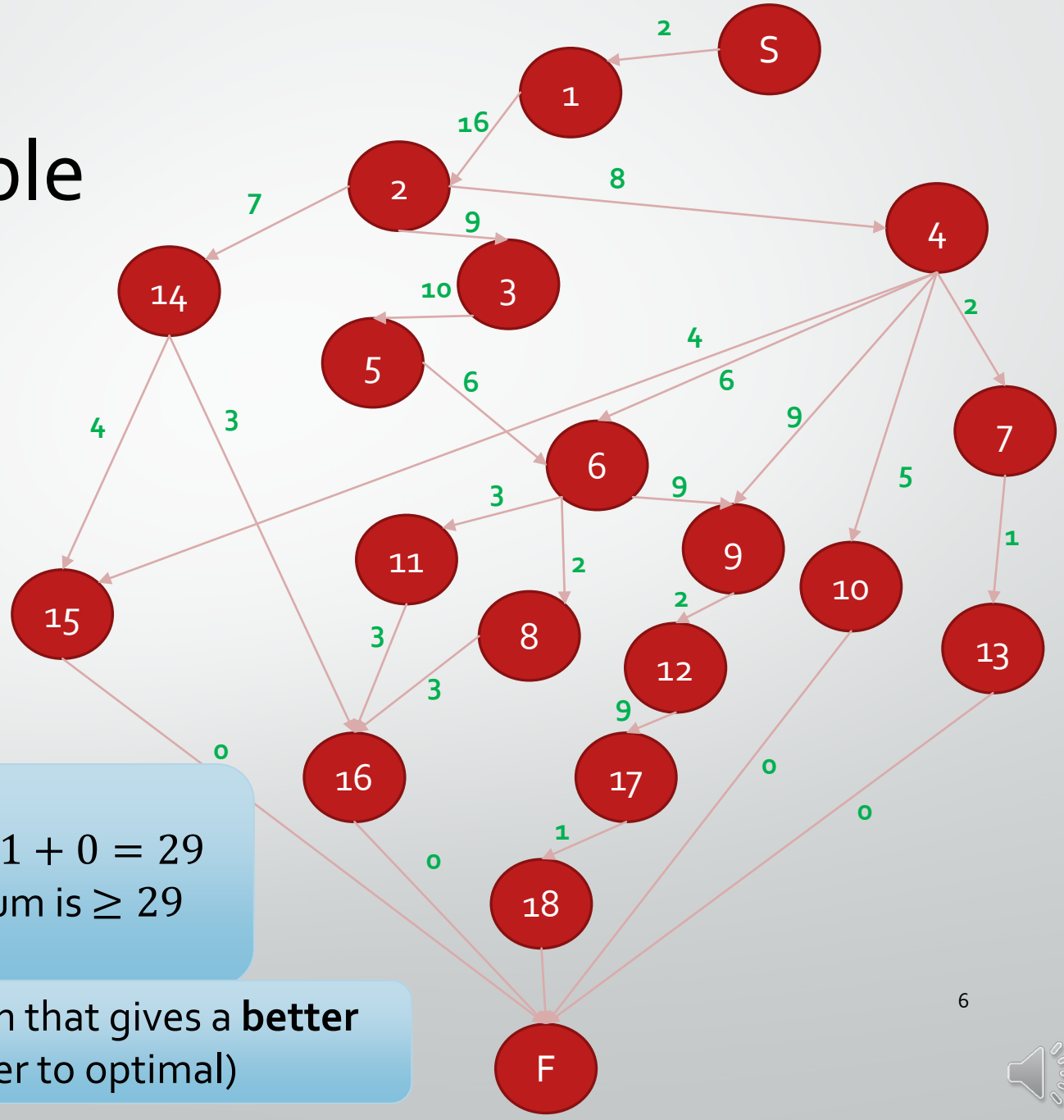
Task	Description	Duration (weeks)	Predecessors
1	Installing construction site	2	None
2	Terracing	16	1
3	Constructing foundations	9	2
4	Access roads and other networks	8	2
5	Erecting the basement	10	3
6	Main floor	6	4,5
7	Dividing up changing rooms	2	4
8	Electrifying terraces	2	6
9	Constructing roof	9	4,6
10	Lighting	5	4
11	Installing terraces	3	6
12	Sealing roof	2	9
13	Finishing changing rooms	1	7
14	Constructing ticket office	7	2
15	Secondary access roads	4	4,14
16	Means of signaling	3	8,11,14
17	Lawn and sport accessories	9	12
18	Handing over the building	1	17

Step 1: Build the network
Step 2: Add durations



Stadium building (longest path) example

- Any sequence of nodes from S to F (start to finish) is called a **path**
- Every path has an associated length (sum of times connecting sequence of nodes)
- Every path provides a **lower bound** on the time required to complete the whole stadium



Example path: S-1-2-4-7-13-F

- This path has length $2 + 16 + 8 + 2 + 1 + 0 = 29$
- The earliest we can complete the stadium is ≥ 29 weeks

Question: Can we find a path that gives a **better lower bound**? (Better = closer to optimal)

The general longest path model

Data

- $T = \{S, 1, 2, \dots, 18, F\}$ – the set of tasks
- c_{ij} – Time it takes to travel from the end of task i to the end of task j (time to complete task j)
- b_i – Supply/demand at node i (1 at node S, -1 at node F, 0 everywhere else)

Variables

- Select which edges are on the chosen path:

$$x_{ij} = \begin{cases} 1 \text{ if edge } (i,j) \text{ is on the chosen path} \\ 0 \quad \text{otherwise} \end{cases}$$

Note that x_{ij} is a *binary variable.*

This is a nice way to relate this problem to other MCNF problems, but we never implement these problems this way. Think about the incidence matrix….

Objective

- Maximize time:

$$\max_{x} \sum_{i \in T} \sum_{j \in T} c_{ij} x_{ij}$$

Constraints

- Balance flow:

$$\sum_{j \in T} x_{kj} - \sum_{i \in T} x_{ik} = b_k \quad \forall k \in T$$

- *Note why this works: only one arc can be chosen leaving S. At the node where the chosen arc terminates, exactly one arc must be chosen leaving that node. Repeat until F is reached. Flow balance enforces the selection of a single complete path from S to F in the longest path model.*

An equivalent, much easier formulation of the project planning model

Data

- $T = \{S, 1, 2, \dots, 18, F\}$ – the set of tasks
- c_i – Time it takes to complete task i

Objective

- Minimize time we finish project:

$$\min_{x} \; x_F$$

Variables

- The time we start working on task $i \in T$:

$$x_i \quad \forall i \in T \text{ (nonnegative)}$$

Constraints

- Task j must start after task i if i is a predecessor for j:

$$x_j \geq x_i + c_i \quad \forall i \in T, j \in T \text{ s.t. } i \text{ precedes } j$$

This is (probably) a much more intuitive model.

In Julia: Project Planning Example

To think about: Are there always two (or more) equivalent ways to model the same problem?