Lesson 7.

Introduction to dynamic programming

1 The knapsack problem

Example 1. You are a thief deciding which precious metals to steal from a vault:

	Metal	Weight (kg)	Value
1	Gold	3	11
2	Silver	2	7
3	Platinum	4	12

You have a knapsack that can hold at most 8kg. If you decide to take a particular metal, you must take all of it. Which items should you take to maximize the value of your theft?

- This example problem is pretty small, so we can easily solve it by inspection
- Maximum total value:
- Items that give the maximum total value:
- We can also formulate this problem as a <u>longest</u> path problem:

- We consider filling up our knapsack in **stages**
- In stage $t = 1, 2, 3$, we decide whether to take metal t
- The last stage (stage 4) represents the end of our decision process
- Node t_n represents

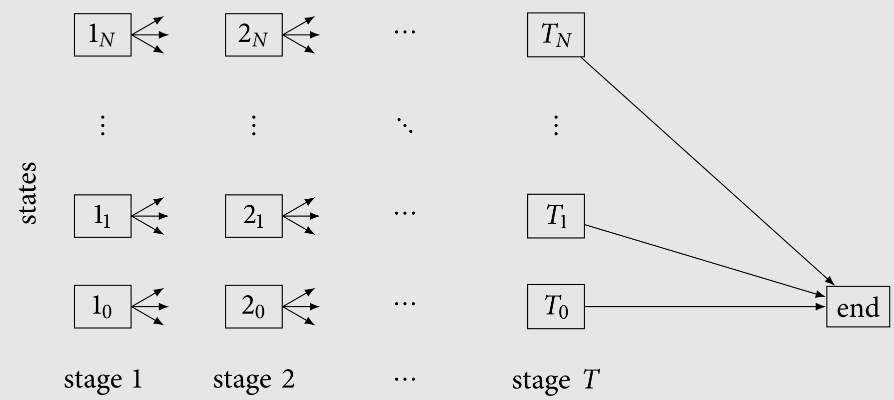
- The edges represent the **decisions** we can make
- Suppose we are deciding whether to take metal 2 (silver), and we have 5 kgs of space left in our knapsack
- Two possible decisions:
  1. Take metal 2
     - This is represented by the edge
     - This decision has a value of , so we use this as the length of this edge
  2. Don't take metal 2
     - This is represented by the edge
     - This decision has a value of , so we use this as the length of this edge
- We can complete the rest of the digraph in a similar fashion
- **Key observation.** Finding an optimal solution to the knapsack problem is equivalent to finding the longest path in this graph from node 1_8 to some stage 4 node
  - In this example, the longest path is $1_8 \to 2_5 \to 3_5 \to 4_1$ with a length of 23
  - The longest path length tells us:
  - The nodes and edges in the longest path tell us:
- To reformulate this as a shortest path problem:
  - Negate all edge lengths
  - Connect all stage 4 nodes to an "end" node with edges of length 0
  - Find the shortest path from node 1_8 to the finish node

2 Dynamic programming

- A **dynamic program** (DP) is a mathematical model that captures situations where decisions are made sequentially in order to optimize some objective
- In particular:
  - DPs divide problems into **stages** with a **decision** required at each stage
  - Each stage has a number of **states** – the possible conditions of the system at that stage
  - A decision at each stage transforms the current state into a state in the next stage with some associated **cost** or **reward**
- DPs come in several different flavors, and can be described in various ways
- For now, we will think of a DP as a specially-structured shortest/longest path problem

Dynamic program – shortest/longest path representation

- **Stages** $t = 1, 2, \ldots, T$ and **states** $n = 0, 1, 2, \ldots, N$
- Directed graph:



  - Node t_n $\longleftrightarrow$ being in state n at stage t
    - Nodes for the tth stage are put in the tth column
  - Edge $(t_n, (t+1)_m)$ $\longleftrightarrow$ the **decision** to go to state m from state n at stage t
    - Length of this edge = **cost** or **reward** of making this decision
    - An edge must connect a node in the tth column to a node in the $(t+1)$st column
  - All nodes for the last stage are connected to an "end" node with an edge of length 0
- Shortest/longest path problem:
  - Source node = one of the first stage nodes (depends on the problem)
  - Target node = end node
  - Edge lengths correspond to rewards $\Longrightarrow$ Find the longest path from source to target
  - Edge lengths correspond to costs $\Longrightarrow$ Find the shortest path from source to target

Example 2. The Simplexville Police Department wants to determine how to assign patrol cars to each precinct in Simplexville. Each precinct can be assigned 0, 1, or 2 patrol cars. The number of crimes in each precinct depends on the number of patrol cars assigned to each precinct:

	Number of patrol cars assigned to precinct		
Precinct	0	1	2
1	14	10	7
2	25	19	16
3	20	14	11

The department has 5 patrol cars. The department's goal is to minimize the total number of crimes across all 3 precincts. Formulate this problem as a dynamic program by giving its shortest/longest path representation.