IOE512: Dynamic Programming

MICHIGAN ENGINEERING
UNIVERSITY of MICHIGAN ■ COLLEGE of ENGINEERING

Today:

- Course summary/expectations
- Introduction to dynamic programming

Course Contact Information

Instructor: Brian Denton

Office: IOE 2893

Email: btdenton@umich.edu

Office Hours: To be determined

Course Web Site: Syllabus, course schedule, assignments, lectures will all posted to Canvas.

Note: Syllabus is subject to change. Please check the web site regularly.

Introduction

My background:

PhD in OR

5 years R&D at IBM

2 years at Mayo Clinic

5 years at NC State

Moved to UM in 2012

Research interests: optimization under uncertainty, applications to medicine

Course Prerequisites

MICHIGAN ENGINEERING
UNIVERSITY of MICHIGAN ■ COLLEGE of ENGINEERING

Goal of this class:

- The goal is for you to develop a thorough understanding of theory and mathematical methods of dynamic programming and be able to apply these concepts in different application domains.

This course assumes knowledge of:

- Linear algebra, Probability theory, and Calculus
- Linear programming (IOE 310 or IOE510), Stochastic models (IOE316 or IOE515)

Course Summary

Meet twice per week (T/TH 10:40 – Noon)

Course grading:

- Assignments approximately every 7-10 days: 15% of final grade (due at start of class, 20% deducted for late assignments + 20% each additional day)
- Midterm Exam: 30% of final grade
- Final exam: 40% of final grade
- In-Class assignments and participation: 15%

Expectations

You:

Be on-time and participate in class. No texting, Facebook, working on your laptop etc.

The work you hand in will be your own (homework assignments, tests, and exams are not collaborative; tests and exams are “closed book” unless otherwise noted)

Me:

Planned lectures

Give you timely feedback

Be available for office hours

Homework Assignments

MICHIGAN ENGINEERING
UNIVERSITY of MICHIGAN ■ COLLEGE of ENGINEERING

Include the following:

- Problem definition
- Clear explanation of mathematical model
- Assumptions made
- Conclusions in words (when appropriate)
- All work leading to the final solution
- Matlab output and code that is well commented in an appendix to your assignment

You may talk to others about basic concepts related to assignments but the work you hand in must be your own.

Grading

The final grade will be determined as follows:

- 97 ≤ A+ ≤ 100
- 93 ≤ A < 97
- 90 ≤ A- < 93
- 87 ≤ B+ < 90
- 83 ≤ B < 87
- 80 ≤ B- < 83
- 77 ≤ C+ < 80
- 73 ≤ C < 77
- 70 ≤ C- < 73
- 67 ≤ D+ < 70
- 63 ≤ D < 67
- 60 ≤ D- < 63
- F < 60

Note: I reserve the right to make “epsilon” changes based on effort and participation in class

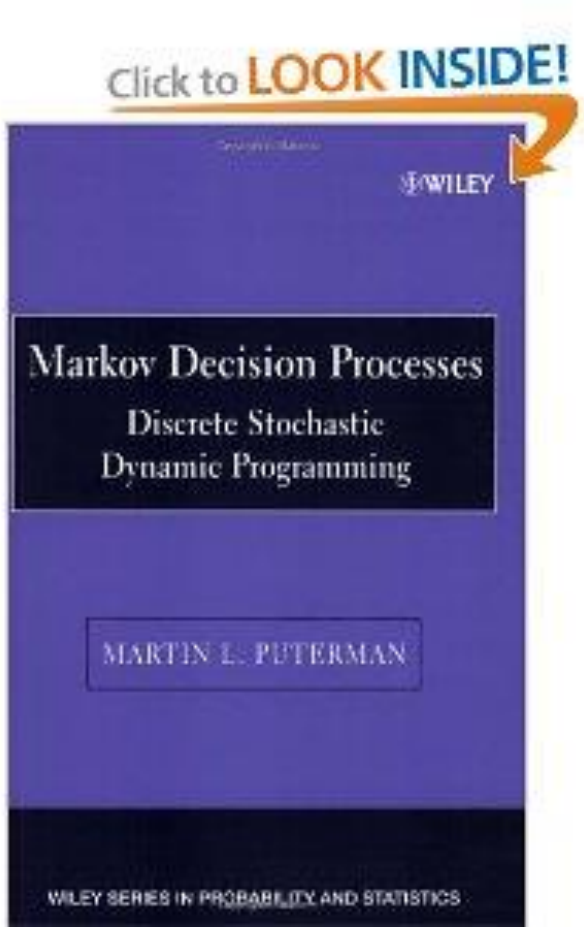
Textbook

Official Textbooks:

We will use the first 3 chapters of the following:

- The Art and Theory of Dynamic Programming, Stuart Dreyfus and Averill Law, 1977

Available on the Canvas.



Approximate Schedule

MICHIGAN ENGINEERING
UNIVERSITY of MICHIGAN ■ COLLEGE of ENGINEERING

Week 1: Introduction to DP

Weeks 2-3: Deterministic DP: Theory and Applications

Weeks 4-5: Introduction to Stochastic DP

Weeks 6-7: Finite horizon Stochastic DP: Theory and Applications

Weeks 8-11: Infinite horizon MDPs: theory, methods, and applications

Weeks 12-13: Special topics

First Class Survey

Please provide some feedback using the in-class survey about:

- Your background
- What you want to learn
- Any questions you have about this class

Introduction to DP

Dynamic programming (DP) dates back to early work of **Richard Bellman** in the 1940's

1953 Paper by Bellman describes the foundation for DP

Since its development DP has been applied to fields of mathematics, engineering, biology, chemistry, and many others

For more history on Richard Bellman see: http://www.gap-system.org/~history/Biographies/Bellman.html

Description

Dynamic programming (DP) can be used to solve many optimization problems

- Often but not always involving the element of time

In many applications DP obtains solutions by:

- Working backward from the end of the problem toward the beginning

- Breaking up a large problem into a series of smaller, easier problems

Example: Coin Game

Suppose there are 30 coins on a table. You and your opponent pick up 1, 2, or 3 coins each turn. You continue until the last coin is picked up. The player who picks up the last coin is the loser.

How can you be sure of winning the game?

Is there an optimal policy that guarantees you will win the game?

Definition

A **policy** defines the **action** to take in each possible **state** of the system

An **optimal policy** defines the **optimal action** to take for each **state** that will achieve some goal such as

- Maximize rewards
- Minimize costs
- Win a game

Optimal Policy for the Coin Game

Logical argument:

If you ensure that it will be your opponent's turn when 1 coin remains, you will win

Working backward one step, if you ensure that it will be your opponent's turn when 5 coins remain, you will win

If you force your opponent to play when 5, 9, 13, 17, 21, 25, or 29 coins remain, you will win

Optimal Policy

The optimal policy can be expressed as a **"look up" table**:

State	Action
30	1
29	*
28	3
27	2
26	1
25	*
24	3
23	2
22	1
21	*
:	:

Elements of a Dynamic Program

The above game exemplifies some of the elements of dynamic programs:

- Stages
- States
- Actions
- Rewards
- Value function
- Optimality Equations
- Optimal Policy

Shortest Path Problems

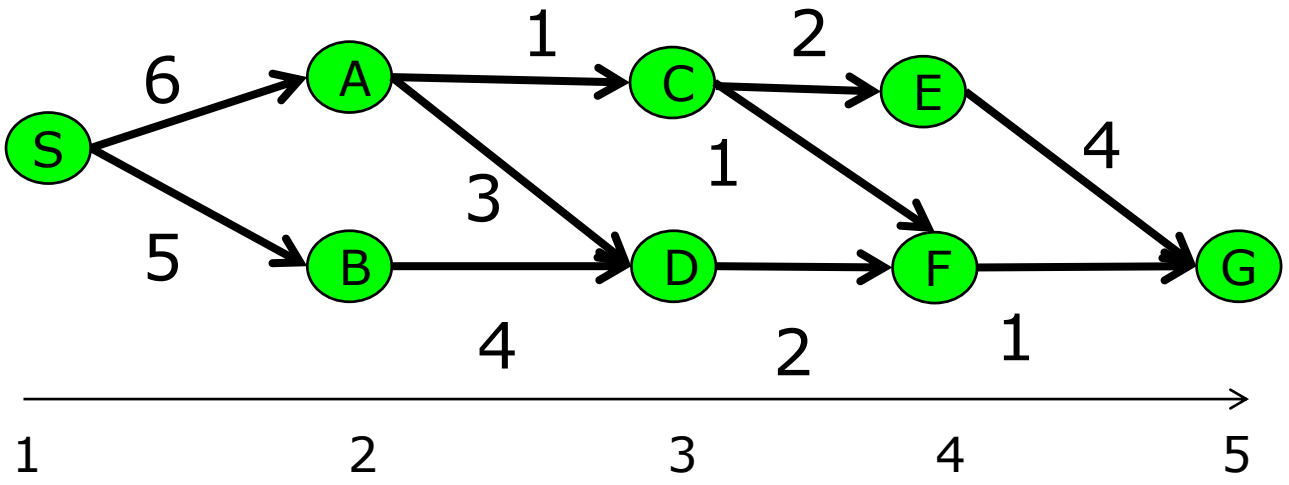
DPs can be used for finding the shortest path that joins two points in a network

- This is a fundamental problem to which DP applies (e.g. google maps)

- Many problems can be formulated as a shortest path problem

Example: DP Formulation

What is the shortest path in this directed graph?



Example: DP Formulation

Elements of the DP formulation

- States: vertices of the graph
- Actions: which vertex to move to
- Rewards: cost associated with edge (max negative costs = min costs)

Goal: Starting from vertex S, select the action at each vertex that will minimize total edge distance travelled to reach vertex G

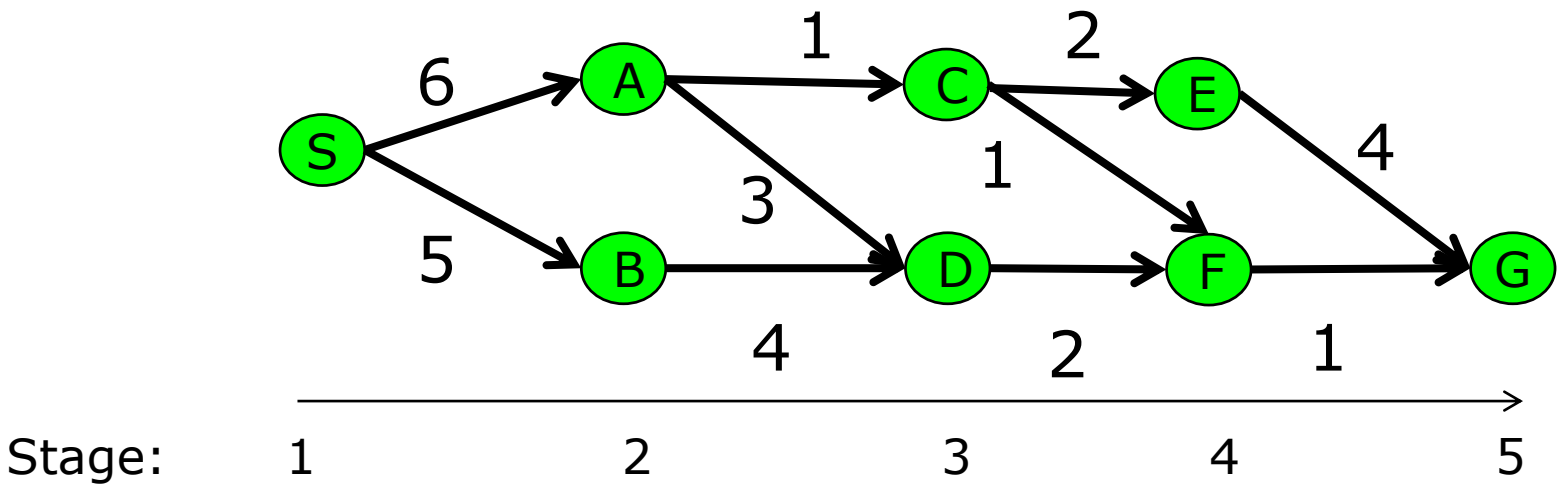
What is the optimal policy?

Example: DP Formulation

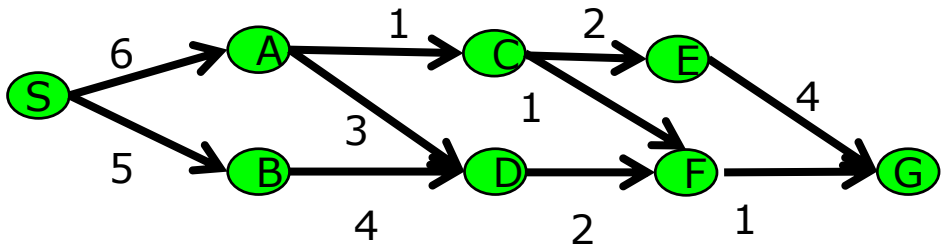
Principle of Optimality: Any "subpath" in the optimal path must be the shortest path between the two vertices

To find the best action from node S we need to compare

- Distance from S to A + Shortest Path from A to G
- Distance from S to B + Shortest Path from B to G



Example: DP Formulation



- Start by finding the stage 5 distance:

  $v_5(G)=0$

- Stage 4: compute shortest path from E and F:

  from E→ G and F→ G: $v_4(E) = 4+0$, $v_4(F) = 1+0$

- Stage 3: compute the shortest path from C and D:

  $v_3(C) = \min\{2+V_4(E), 1+V_4(F)\}=2$, $v_3(D) = 2+ v_4(F) = 3$

- Stage 2: compute the shortest path from A and B:

  $v_2(A) = \min\{1+v_3(C), 3+v_3(D)\} = 3$, $v_2(B) = 4+v_3(D) = 7$

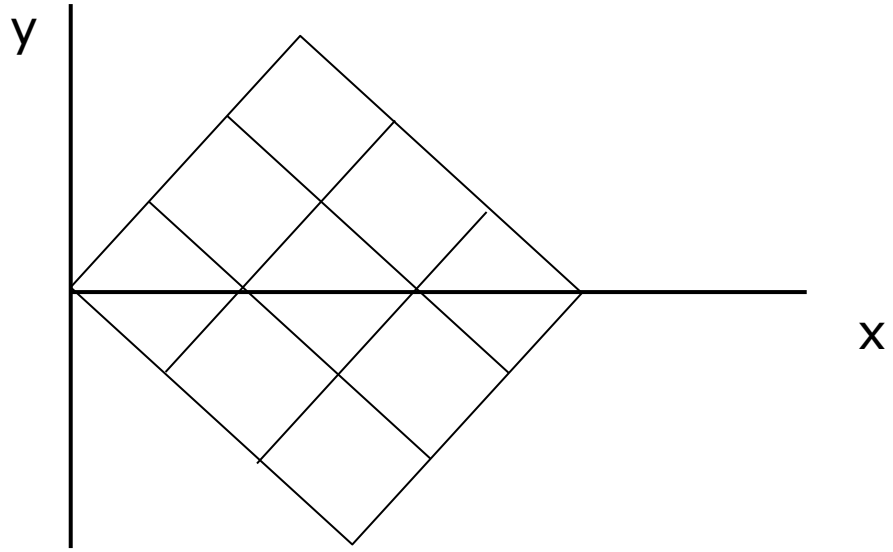
- Stage 1: compute the shortest path from S:

  $v_1(S) = \min\{6+v_2(A), 5+v_2(B)\} = 9$

Answer: Shortest path is S→A→C→F→G

Shortest Path DP Formulation

- Following is a general formulation for shortest path problems on directed graphs
- Let the state be denoted by coordinates (x,y)
- Let cost of a path be denoted by c(x,y)
- Assume the following coordinate system in which all admissible paths move from the left to the right:



Shortest Path DP Formulation

- Let $c^+(x,y)$ be the cost of moving from state (x,y) to (x+1, y+1) and let $c^-(x,y)$ be the cost of moving from state (x,y) to (x+1, y-1)

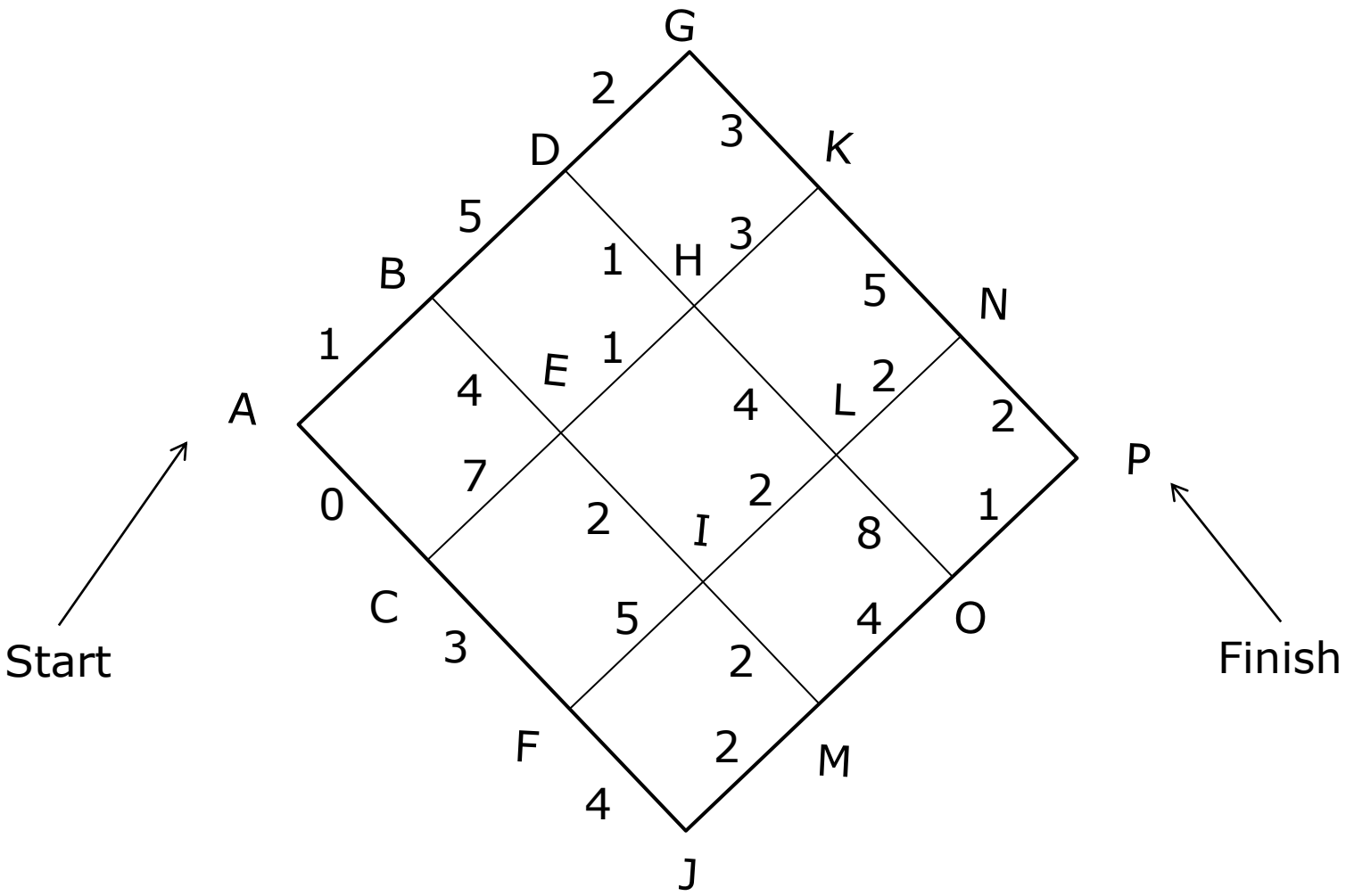
- Let (x^F,y^F) denote the terminal (destination) state

- The optimality equations are:

$$v(x,y) = min \begin{cases} c^+(x,y) + v(x+1,y+1) \\ c^-(x,y) + v(x+1,y-1) \end{cases}$$

$$v(x^F,y^F) = 0$$

Exercise: What is the shortest path?



Solution

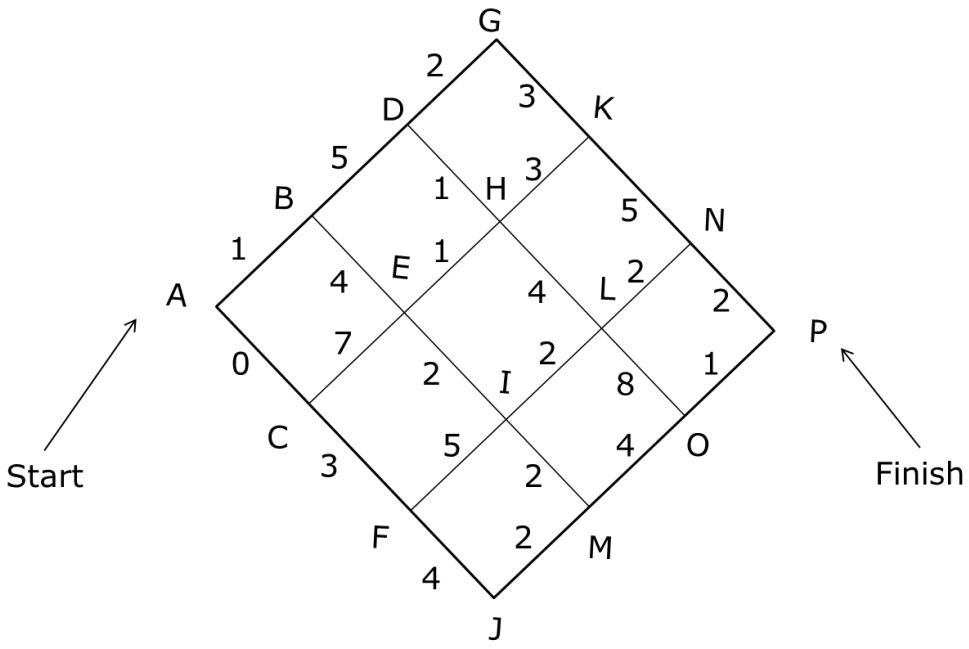
Stage 7: $v_7(P)=0$

Stage 6: $v_6(N)= 2+v_7(P)=2$
$v_6(O)= 1+v_7(P)=1$

Stage 5: $v_5(K)=5+v_6(N)=7$
$v_5(L)=\min\{2+v_6(N)$
$8+v_6(O\}=4$
$v_5(M)= 4 + v_6(O)=5$

Stage 4: $v_4(G)=3+v_5(K)=10$
$v_4(H)= \min\{3+v_5(K),$
$4+v_5(L)\}=8$
$v_4(I)= \min\{2+ v_5(L),$
$2+ v_5(M)\}=6$
$v_4(J)= 2+ v_5(M)=7$

Stage 3: $v_3(D)= \min\{2+v_4(G),$
$1+ v_4(H)\}=9$
$v_3(E)= \min\{1+v_4(H),$
$2+v_4(I)\}=8$
$v_3(F)= \min\{5+ v_4(I),$
$4+v_4(J)\} = 11$



Stage 2: $v_2(B)= \min\{5+v_3(D),$
$4+v_3(E)\}=12$
$v_2(C)= \min\{7+v_3(E),$
$3+v_3(F)\}=14$

Stage 1: $v_1(A)=\min\{1+v_2(B),$
$0+ v_2(C)\}=13$

Solution

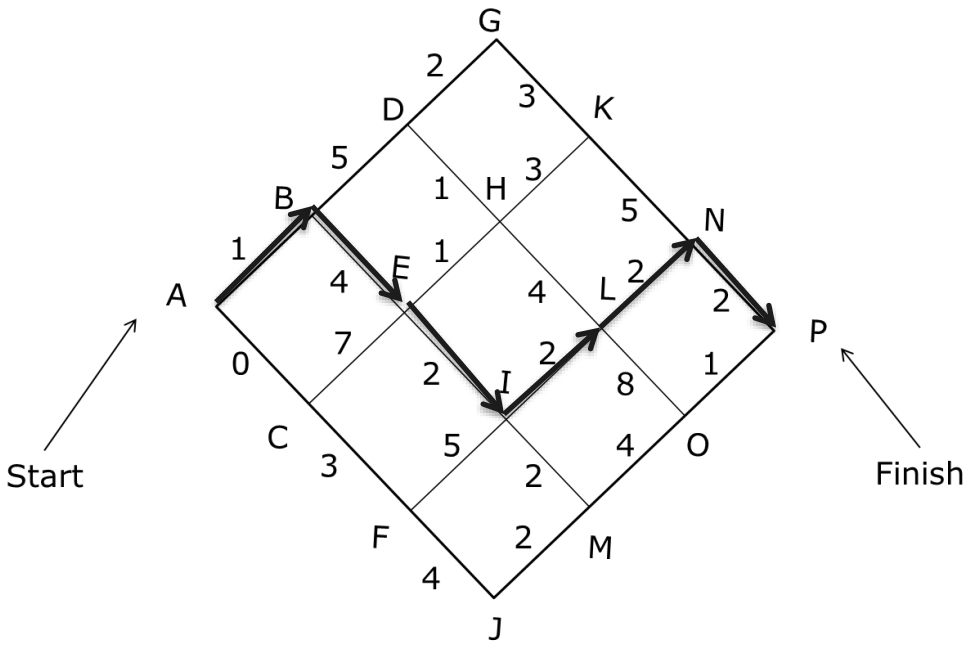
Stage 7: $v_7(P)=0$

Stage 6: $v_6(N)= \underline{2+v_7(P)}=2$
$v_6(O)= 1+v_7(P)=1$

Stage 5: $v_5(K)=5+v_6(N)=7$
$v_5(L)=\min\{\underline{2+v_6(N)},$
$8+v_6(O)\}=4$
$v_5(M)= 4 + v_6(O)=5$

Stage 4: $v_4(G)=3+v_5(K)=10$
$v_4(H)= \min\{3+v_5(K),$
$4+v_5(L)\}=8$
$v_4(I)= \min\{\underline{2+ v_5(L)},$
$2+ v_5(M)\}=6$
$v_4(J)= 2+ v_5(M)=7$

Stage 3: $v_3(D)= \min\{2+v_4(G),$
$1+ v_4(H)\}=9$
$v_3(E)= \min\{1+v_4(H),$
$\underline{2+v_4(I)}\}=8$
$v_3(F)= \min\{5+ v_4(I),$
$4+v_4(J)\} = 11$



Stage 2: $v_2(B)= \min\{5+v_3(D),$
$\underline{4+v_3(E)}\}=12$
$v_2(C)= \min\{7+v_3(E),$
$3+v_3(F)\}=14$

Stage 1: $v_1(A)=\min\{\underline{1+v_2(B)},$
$0+ v_2(C)\}=13$

DP Characteristics

Characteristic 1:

- The problem can be divided into **stages** with an action (decision) required at each stage

Characteristic 2:

- Each stage has **states** which define the information needed to make an optimal decision

Characteristic 3:

- The decision chosen at any stage describes how the state changes in next stage

DP Characteristics

Characteristic 4

- The action at a given state does not depend on previously reached states or actions
  - This is known as the **principle of optimality**.

Characteristic 5

- Given T stages, there is a **recursion** that relates the cost or reward earned during stage *t* with that of t+1, t+2, …. *T*.

Next Time

Read Chapter 1 of Dreyfus and Law