Session 6. Dynamic Programming

In most applications, dynamic programming obtains solutions by working *backward* from the end of a problem toward the beginning, thus breaking up a large unwieldy problem into a series of smaller, more tractable problems.

* Characteristic of DP Formulation

1. The problem can be divided into *stages* with a decision required at each stage *i*.
2. Each stage has a number of *states* s_i associated with it.
3. The *decision* d_i chosen at any stage *i* describes how the state at the current stage is transformed into the state at the next stage.
4. Given the current state, the optimal decision for each of the remaining stages must *not* depend on previously reached states or previously chosen decisions. This idea is known as the "principle of optimality".

5. When a DP problem has been classified into a series of *T* stages, there must be a *recursion* that relates the cost (or reward) incurred at stage *i* to the cost (or reward) incurred from stages *i*+1, *i*+2, …, *T*. In essence, the recursive relationship formalizes the working-backward procedure.

$$f_i(s_i) = \min_{\forall d_i}\{c_i(d_i, s_i) + f_{i+1}(s_{i+1})\} \text{ where } s_{i+1} = h(d_i, s_i)$$

* Classifications

- *Deterministic* dynamic programming
- *Probabilistic* dynamic programming

A. Recurrence Relation

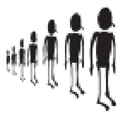
* Recursive Equation

- In mathematics, a recurrence relation is an equation that *recursively* defines a sequence; once one or more initial terms are given, each further term of the sequence is defined as a function of the preceding terms.

- A recursive equation can be thought of as analogous to a flight of stairs. In order to climb a flight of stairs, we need to be able to do the following:

  1. Get to the first step. Mathematically, this is called the *boundary* condition (or *terminal* condition in backward recursions.) For more complicated recursive equations, we sometimes need several initial conditions.

  2. Get to the next step from the previous step. Mathematically, this is the *recursive equation*.



Ex 1] Fibonacci sequence: The Fibonacci numbers are the numbers in the following integer sequence:

1, 1, 2, 3, 5, 8, 13, 21, …

The sequence F_n of Fibonacci numbers is defined by the recurrence relation:

$F_n = F_{n-1} + F_{n-2}$, for n=3, 4, …

with the initial conditions, $F_1 = 1$ and $F_2 = 1$.

# Explicit equation? $F_n = \frac{(0.5+0.5\sqrt{5})^n-(0.5-0.5\sqrt{5})^n}{\sqrt{5}}$.

Ex 2] Best-of-Seven Series: Consider a best-of-seven series that ends when one team wins *four* games (as in the NBA, MLB, or NHL playoffs). If the probability of team ***A*** defeating team ***B*** in any game is *p*, find the *average number* of games in the series. (For simplicity, let us assume that each team has a constant probability of winning a game throughout the series and games are independent of each other.)

(a) Explicit equations: *Negative binomial* distribution

$$E[n] = \sum_{i=4}^{7} i\left\{\binom{i-1}{3}p^4q^{i-4} + \binom{i-1}{3}p^{i-4}q^4\right\}.$$

(b) Recursive equations

Let $S(i, j)$ be the expected number of *remaining* games, given that Team A has won *i* games and Team B has won *j* games so far. Then, the recursive equation is

$$S(i,j) = p\ S(i+1,j) + q\ S(i,j+1) + 1$$

with the terminal (or boundary) condition,

$S(i, j) = 0$ if i=4 or j=4.

For example, if p=q=0.5, then $E[n]$=S(0, 0)=

	4	0	0	0	0	-
	3	1.875	1.75	1.5	1	0
Team B	**2**	3.5	3.125	2.5		0
(q=0.5)	**1**	4.8125	4.125		1.75	0
	0		4.8125	3.5	1.875	0
		0	**1**	**2**	**3**	**4**

Team A (p=0.5)

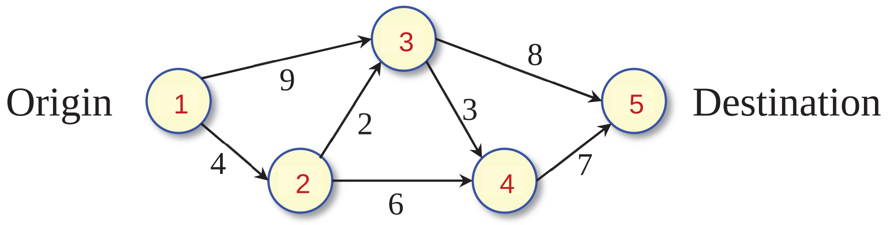
# Find the recursive equation for the probability that **Team A** will win the-best-of-seven series.

B. Various Recursions

The recursive equation could be expressed as an additive, multiplicative, or mini-max recursion in dynamic programming.

I. Additive Recursion

I. Q. Smart needs to drive from city 1 to city 5. The length c_{ij} of the arc connecting city i and city j represents the distance from city i to city j. Find the shortest path.



▪ Optimality equation

Let $f_i(s_i)$ be the minimum distance at stage i when the state is s_i. Then, the optimality equation is

$$f_i(s_i) = min \{ c(s_i, d_i) + f_{i+1}(d_i) \} \text{ for all } d_i.$$

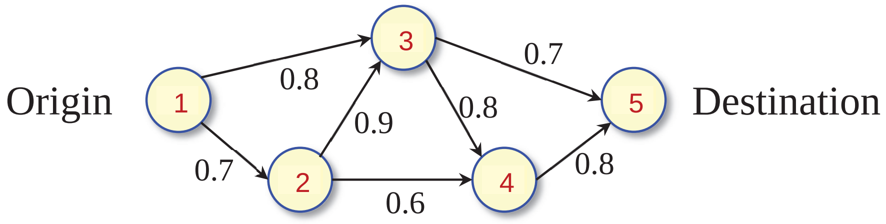
▪ Optimality table:

Stage	Decision, d_i	State, s_i					
		1	2	3	3'	4	5
3	5				8	7	
	min				8	7	
2	3'			8			
	4			10			
	min			8			
1	2	14					
	3	17					
	min	14					

▪ Optimal decision:

II. Multiplicative Recursion

Unfortunately, some routes are heavily patrolled by police. Smart thus decided to choose a route that maximizes the probability of not being stopped by police. Observing all the feasible road segments, the associated probabilities were compiled as given below:



Find the route which gives the maximum probability of not being stopped by police.

▪ Optimality table:

Stage	1	2	3	4	5	Max
4	-	-	-	-	0.8	0.8
3	-	-	-	.64	[0.7]	0.7
2	-	-			-	
1	-	0.441	[0.56]	-	-	**0.56**

▪ Optimality equation

Let f_i = probability of driving from city i to the destination without being stopped by police.

$f_5 = 1.0$ <= Terminal condition!

$f_4 = 0.8\, f_5 = 0.8$

$f_3 = \max\{\ 0.8\, f_4,\ 0.7\, f_5\ \} = \max\{\ 0.64,\ 0.70\ \} = 0.70$

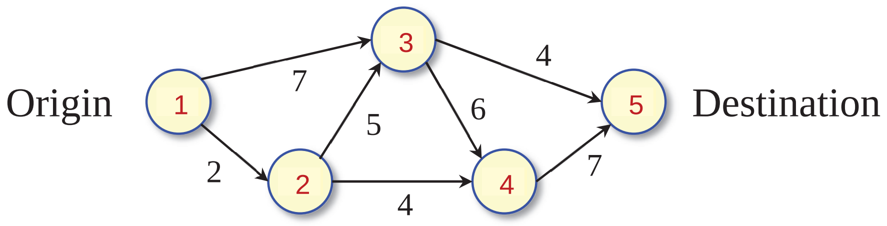
$f_2 = \max\{\ 0.9\, f_3,\ 0.6\, f_4\ \} = \max\{\ 0.63,\ 0.48\ \} = 0.63$

$f_1 = \max\{\ 0.7\, f_2,\ 0.8\, f_3\ \} = \max\{\ 0.441,\ 0.56\ \} = 0.56$

▪ Optimal decision:

III. Min-Max Recursion

I. Q. Smart is interested in *minimizing* the maximum altitude above sea level that he will encounter during his drive. Now a_{ij} of the arc connecting city i and city j represents the maximum altitude (in thousands of feet above sea level) encountered when driving from city i to city j.



Determine how Smart should proceed from city 1 to city 5.

▪ Optimality table:

Stage	1	2	3	4	5	Min
4	-	-	-	-	7	7
3	-	-	-	7	[4]	4
2	-	-			-	
1	-	[5]	7	-	-	**5**

▪ Optimality equation

Let f_i = maximum altitude encountered when driving from city i to the destination.

$f_5 = 0.0$ <= Terminal condition!

$f_4 = \max \{ 7, f_5 \} = 7$

$f_3 = \min \{ \max[6, f_4], \max[4, f_5] \} = \min\{ 7, 4 \} = 4$

$f_2 = \min \{ \max[5, f_3], \max[4, f_4] \} = \min\{ 5, 7 \} = 5$

$f_1 = \min \{ \max[2, f_2], \max[7, f_3] \} = \min\{ 5, 7 \} = 5$

▪ Optimal decision:

C. Deterministic DP Models

Ex 1] Optimal Assignment Problem

The number of crimes in each of a city's three police precincts depends on the number of patrol cars assigned to each precinct.

c_{ij}	Number of patrol cars assigned to a precinct			
	0	1	2	3
Precinct 1	14	10	7	4
Precinct 2	25	19	16	14
Precinct 3	20	14	11	8

A total of 3 patrol cars are available. Determine how many patrol cars should be assigned to each precinct.

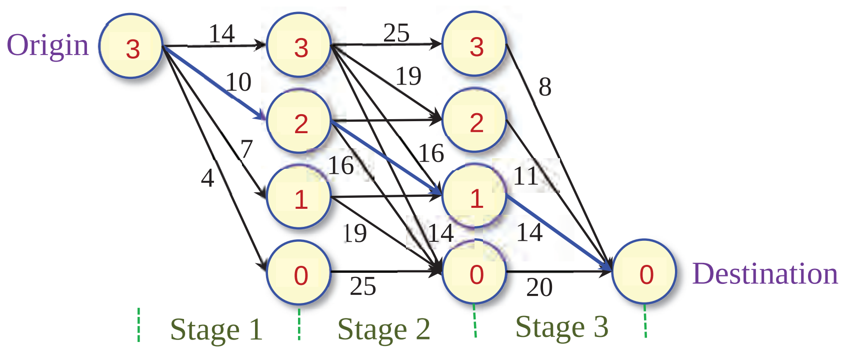
(a) IP formulation

- Variables:

- Objective function:

- Constraints:

(b) Network representation: Shortest path problem!



(c) DP formulation

- Stage i: Precinct i, where i = 1, 2, and 3.
- State s_i at stage i: Total number of patrol cars available.
- Decision d_i: Number of cars assigned to precinct i.
- Optimality equation:

  Let $f_i(d_i, s_i)$ be the minimum number of crimes at stage i when the state is s_i and the action is d_i. Then, we have

  $$f_i(d_i, s_i) = c_i(d_i) + f^*_{i+1}(s_i - d_i) \quad \text{for } d_i \le s_i,$$

  $$\text{where } f_i^*(s_i) = \min_{d_i \le s_i} f_i(d_i,\ s_i), \quad \text{for } i=1, 2, 3.$$

  The boundary condition is $f_4^*(s_4)=0$ for any s_4.

- Optimality table:

Stage i	d_i	State s_i			
		0	1	2	3
3	0	20	20	20	20
	1	-	[14]	14	14
	2	-	-	11	11
	3	-	-	-	8
	min	20	14	11	8
2	0	45	39	36	33
	1	-			
	2	-	-		
	3	-	-	-	
	min				
1	0	59	53	47	44
	1	-	55	49	[43]
	2	-	-	52	46
	3	-	-	-	49
	min	59	53	47	**43**

- Optimal decision:

Ex 2] Cargo-Loading Problem

Consider loading a vessel with stocks of 3 items. Maximum cargo weight is 6 tons. The weight and the value of each unit of items are listed as follows:

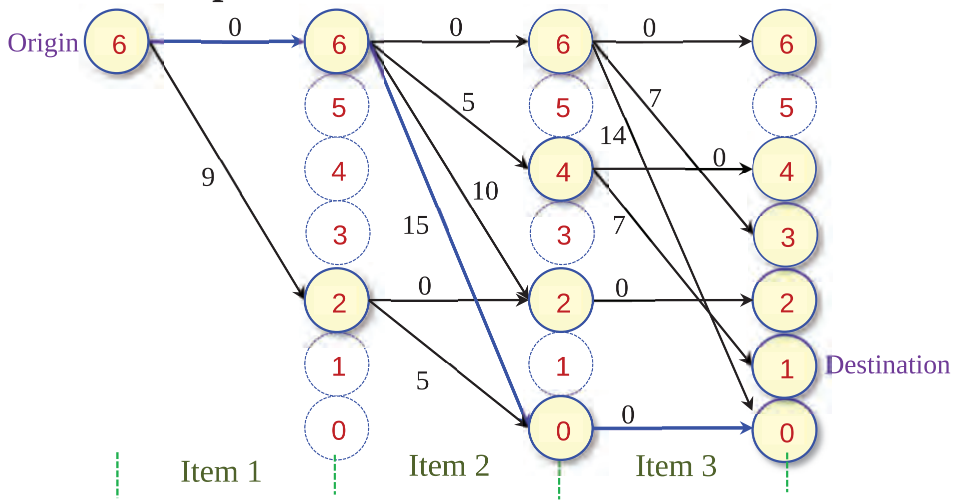
Item i	Weight (ton) w_i	Value v_i
1	4	\$9
2	2	\$5
3	3	\$7

How should the vessel be loaded to maximize the total value?

(a) IP formulation

- Variables:
- Objective function:
- Constraints:

(b) Network representation:



Find the *longest* path from the origin to the destination!

(c) DP formulation

- Stage i: The ith item, where i=1, 2, and 3.
- State s_i at stage i: Total remaining weight.
- Decision d_i: Number of units of item i to be loaded.

- Optimality equation:

Let $f_i(d_i, s_i)$ be the *maximum* value obtained when the remaining weight is s_i and we load d_i units of item i:

$$f_i(d_i, s_i) = v_i d_i + f_{i+1}^*(s_i - w_i d_i) \quad \text{for } w_i d_i \le s_i,$$

where $f_i^*(s_i) = \max_{w_i d_i \le s_i} f_i(d_i, s_i)$, for i=1, 2, 3.

The terminal condition is $f_4^*(s_4) = 0$.

- Optimality table:

Stage i		State s_i 0	1	2	3	4	5	6	w_i	v_i
3	0	[0]	0	0	0	0	0	0	3	7
	1	-	-	-	7	7	7	7		
	2	-	-	-	-	-	-	14		
	max	0	0	0	7	7	7	14		
2	0								2	5
	1	-	-							
	2	-	-	-	-					
	3	-	-	-	-	-	-			
	max									
1	0	0	0	5	7	10	12	[15]	4	9
	1	-	-	-	-	9	9	14		
	max	-	-	-	-	10	12	**15**		

- Optimal decision:

Ex 3] Resource Allocation Problem

ET is about to fly home. For the trip to be successful, the ship's solar relay, wrap drive, and candy maker must all function properly. ET has found 3 mechanics who are willing to help get the ship ready for take-off. The probability that each component will function properly during the trip home is given as a function of the number of mechanics assigned to repair each component:

p_{ij}	Number of mechanics assigned to component i			
	0	1	2	3
1. Wrap drive	0.30	0.55	0.65	0.95
2. Solar relay	0.40	0.50	0.70	0.90
3. Candy maker	0.45	0.55	0.80	0.98

(a) IP formulation

- Variables:
- Objective function:
- Constraints:

(b) DP formulation

▪ Stage i:	The ith task, where i=1, 2, 3.
▪ State s_i at stage i:	Number of mechanics still available.
▪ Decision d_i:	Number of mechanics assigned to task i

▪ Optimality equation:

- Let $p_i(d_i)$ denote the *probability* of success for task i if it is assigned d_i mechanics.

- Let $f_i(d_i, s_i)$ be the *maximum* probability obtained at stage i when, among s_i mechanics still available, we assign d_i mechanics to task i. Then, the recursive equation is

$$f_i(d_i, s_i) = p_i(d_i) f_{i+1}^*(s_i - d_i) \quad \text{for } d_i \le s_i,$$

where $f_i^*(s_i) = \max_{d_i \le s_i} f_i(d_i, s_i)$, for i=1, 2, 3.

- The terminal condition is $f_4^*(s_4) = 1$.

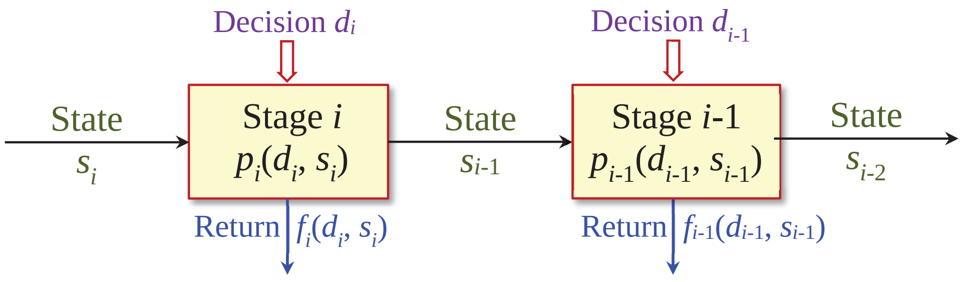
▪ Optimality table:

		0	1	2	3
3. Candy maker	0	0.45	0.45	0.45	0.45
	1	.	0.55	0.55	0.55
	2	.	.	[0.80]	0.80
	3	.	.	.	0.98
	max	0.45	0.55	0.80	0.98
2. Solar relay	0	0.18	0.22	[0.32]	0.392
	1	.			
	2	.	.		
	3	.	.	.	
	max				
1. Wrap drive	0	0.054	0.0675	0.096	0.1215
	1	.	0.099	0.12375	[0.176]
	2	.	.	0.117	0.14625
	3	.	.	.	0.171
	max	0.054	0.099	0.12375	**0.176**

▪ Optimal decision:

D. Probabilistic DP Models*

In deterministic DP problems, a specification of the current state and current decision is enough to tell us *with certainty* the new state and the costs during the current stage. In probabilistic (or stochastic) DP problems, the current period's costs and/or the next period's state may not be known with certainty, even if the current state and decision are known.



* Formulation

- Stage i, where i=0, 1, …, n. (*finite* time-horizon!)
- State s_i: At the beginning of each stage i, the system is in some state x_i.
- Decision (or action) d_i: For every state s_i, the allowable decision d_i is available.
- Transition probabilities from the current state to the next state: $p_i(d_i, s_i)$
- Expected reward obtained at stage i: $f_i(d_i, s_i)$

* Classification

- *Finite* stage model
- *Infinite* stage model => Markov decision process

*** Case I: Uncertain Payoffs**. A supermarket chain has purchased 6 gallons of milk from a local dairy. The chain must allocate the 6 gallons to its 3 stores. If a store sells a gallon of milk, then the chain receives revenue of $2. Any unsold milk is worth just $.50. Unfortunately, the demand for milk is uncertain, and is given in the following table:

	Daily Demand		
	1	2	3
Store 1	0.6	0.0	0.4
Store 2	0.5	0.1	0.4
Store 3	0.4	0.3	0.3

Use dynamic programming to determine how the supermarket should allocate the 6 gallons of milk among the three stores to maximize the expected revenue.

▪ Expected revenue $r_i(d_i)$ for each allocation of milk to a store:

$r_i(d_i)$	Allocation			
	0	1	2	3
Store 1	$0.00	$2.00	$3.10	$4.20
Store 2	$0.00			$4.35
Store 3	$0.00	$2.00	$3.40	$4.35

▪ **DP formulation**

- Stage i: The ith store, where i=1, 2, 3.
- State s_i: Amount of milk available for allocation.
- Decision d_i: Amount of milk allocated for store i.
- Payoff $r_i(d_i)$ Expected revenue earned at store i with d_i.

▪ Optimality equation

Let $f_i(d_i, s_i)$ be the expected revenue earned when s_i gallons of milk is available and d_i gallons of them are allocated for store i. Then, the recursive equation is

$$f_i(d_i, s_i) = r_i(d_i) + \ f_{i+1}^*(s_i - d_i) \text{ for } d_i \leq s_i,$$

$$\text{where } f_i^*(s_i) = \max_{d_i \leq s_i} f_i(d_i, s_i)\}, \text{ for } i=1, 2, 3.$$

The boundary condition is $f_4^*(s_4) = 0.5s_4$ for all s_4.

▪ Optimality table

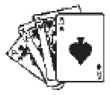
		State s_i							$r_i(d_i)$
	d_i	0	1	2	3	4	5	6	
3	0	0	0.50	1.00	1.50	2.00	2.50	3.00	0.00
	1		2.00	2.50	3.00	3.50	4.00	4.50	2.00
	2		-	[3.40]	3.90	4.40	4.90	5.40	3.40
	3		-	-	4.35	4.85	5.35	5.85	4.35
	max	0	2.00	3.40	4.35	4.85	5.35	5.85	
2	0	0	2.00	3.40		4.85	5.35	5.85	0.00
	1		2.00	4.00	5.40	6.35		7.35	2.00
	2		-	3.25		6.65	7.60	8.10	3.25
	3		-	-	4.35	6.35		8.70	4.35
	max	0		4.00		6.65		8.70	
1	0	0	2.00	4.00	5.40	6.65	7.75	8.70	0.00
	1		2.00	4.00	6.00	7.40	8.65	[9.75]	2.00
	2		-	3.10	5.10	7.10	8.50	9.75	3.10
	3		-	-	4.20	6.20	8.20	9.60	4.20
	max	0	2.00	4.00	6.00	7.40	8.65	**9.75**	

▪ Multiple optimal solutions

Assign (1, 3, 2) or (2, 2, 2) with $z =$

*** Case II: Uncertain States**: A gambler has $2. He is allowed to play a game of chance 4 times, and his goal is to maximize his probability of finishing with at least $6. If the gambler bets *d* dollars on a play of the game, then with probability 0.40, he wins the game and increases his capital position by *d* dollars; There is a probability of 0.6 of losing the game, in which case he loses *d* dollars. On any play of the game, the gambler may not bet more money than he has. We assume that bets of zero dollars (i.e., not betting) are permissible.

Using dynamic programming, determine the betting strategy that will maximize his chances of finishing with at least $6 by the end of the fourth game.



▪ **DP formulation**

- Stage i:	He has i more games to play.
- State s_i:	Number of $1 chips in hand at stage i.
- Decision d_i:	Number of $1 chips to bet at stage i.
- Payoff $r_i(d_i)$	\$$d$ if he wins or -\$$d$ if he loses a game.

▪ **Optimality equation**

- Let $f_i(d_i, s_i)$ be the probability that he will finish with at least $6, given that he bets d_i among s_i dollars available at stage i. Then, the recursive equation is

$$f_i(d_i, s_i) = 0.6 f_{i-1}^*(s_i - d_i) + 0.4 f_{i-1}^*(s_i + d_i) \text{ for } d_i \le s_i$$

where $f_i^*(s_i) = \max_{d_i \le s_i} f_i(d_i, s_i)$, for i=1, 2, 3, 4.

- The boundary condition is

$$f_0(s_0) = \begin{cases} 1 & \text{if } s_0 \ge \$6 \\ 0 & \text{if } s_0 < \$6 \end{cases}$$

▪ Optimality table

		State						
Stage	Decision	0	1	2	3	4	5	6
0	max	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	1
	1		0	0	0	0	0.4	0.4
	2			0	0	0.4	0.4	0.4
	3				0.4	0.4	0.4	0.4
	4					0.4	0.4	0.4
	5						0.4	0.4
	max	0	0	0	0.4	0.4	0.4	1
2	0	0	0	0	0.4	0.4	0.4	1
	1		0	0.16	0.16	0.4	0.64	0.64
	2			0.16	0.16	0.4	0.64	0.64
	3				0.4	0.4	0.4	0.64
	4					0.4	0.4	0.4
	5						0.4	0.4
	max	0	0	0.16	0.4	0.4	0.64	1
3	0	0	0	0.16	0.4	0.4	*0.64*	*1*
	1		0.064	0.16	0.256	0.496	*0.64*	*0.784*
	2			0.16	0.256	0.496	*0.64*	*0.64*
	3				0.4	0.4	*0.496*	*0.64*
	4					0.4	*0.4*	*0.496*
	5						*0.4*	*0.4*
	max	0	0.064	0.16	0.4	0.496	*0.64*	*1*
4	0	0	0.064	0.16	*0.4*	*0.496*	*0.64*	*1*
	1		0.064	0.1984	*0.2944*	*0.496*	*0.6976*	*0.784*
	2			0.1984	*0.2944*	*0.496*	*0.64*	*0.6976*
	3				*0.4*	*0.4384*	*0.496*	*0.64*
	4					*0.4*	*0.4384*	*0.496*
	5						*0.4*	*0.4384*
	max	0	0.064	**0.1984**	*0.4*	*0.496*	*0.6976*	*1*

▪ Optimal betting strategy

If he plays the game with \$2, P[$s_0 \geq \$6$ | 4 plays] = 0.1984.

# From the Markov decision process,
it can be shown that P[$s_0 \geq \$6$ | ∞ plays] = 0.2105.

E. Optimal Stopping Problems*

The theory of optimal stopping is concerned with the problem of choosing a time to take a particular action, based on *sequentially* observed random variables, in order to maximize an expected reward or minimize an expected cost. Problems of this type are found in the area of statistics, economics, and mathematical finance.

Optimal stopping problems can often be written in the form of an optimality equation, and are therefore often solved using dynamic programming:

- Stage i:	A sequence of random variables.
- State s_i:	Observed value of the random variable.
- Decision d_i:	Stop (d_i=1) or go (d_i=0).
- Reward $r_i(d_i, s_i)$	

However, they often have a special structure that simplifies the determination of optimal policies.

Ex 1] Asset Selling Problem

The market value of a used car is estimated at \$3,000. The owner can get more than this amount, but is willing to entertain offers from the first three prospective buyers who respond to the advertisement (which means that a decision must be made no later than the time the third offer is received). The offers are expected to be \$3,000, \$3,300, \$3,700, and \$4,200, with equal probabilities. Naturally, once an offer is accepted, all later offers are discarded. The objective is to set an acceptance limit that can be used to evaluate each offer.

Develop an optimal stopping rule for the owner.

▪ DP formulation

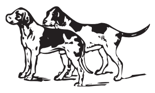
- Stage *i*:	The *i*th buyer, where *i*=1, 2, 3.
- State s_i:	Buyer's offer at stage *i*.
- Decision d_i:	Accept the offer (d_i=1) or reject it (d_i=0).

▪ Optimality equation

Let $f_i(d_i, s_i)$ be the expected profit on day *i* given that the offer is s_i and the decision is d_i. Then, the recursive equation is

$$f_i(d_i, s_i) = \begin{cases} s_i & \text{if } d_i = 1 \\ \sum_{\forall s_{i+1}} p(s_{i+1}) f_{i+1}^*(s_{i+1}) & \text{if } d_i = 0 \end{cases}$$

where $f_i^*(s_i) = \max_{d_i=\{0,1\}} f_i(d_i, s_i)$ for *i*=1, 2,…,3.



The terminal condition is $f_4(s_4) = 0$.

▪ Optimality table

Stage	Decision	State			
		3000	3300	3700	4200
3	Stop	3000	3300	3700	4200
	Go	0	0	0	0
	max	3000	3300	3700	4200
2	Stop	3000		3700	4200
	Go	3550		3550	3550
	max	3550		3700	4200
1	Stop	3000	3300		4200
	Go	3750	3750		3750
	max	3750	3750		4200

▪ Expected revenue: f_0 =

Ex 2] Random Walk Model

An urn initially has 6 red and 4 blue balls. At each stage, you may *randomly* choose a ball from the urn; if the ball is blue, then \$1 is earned, and if it is red, then \$1 is lost. The chosen ball is discarded. At any time, you can decide to stop playing or choose another ball. To maximize the player's total *expected net return*, let's analyze this as a *probabilistic* DP problem.

▪ **DP formulation**

- State (i, j): Currently i red and j blue balls in the urn.
- Decision when the state is (i, j): (*Stop*, *Continue*)
- Transition probabilities from (i, j):

  P[Pick a *red* ball] = P[from (i, j) to $(i\text{-}1, j)$] = $i/(i+j)$

  P[Pick a *blue* ball]= P[from (i, j) to $(i, j\text{-}1)$] = $j/(i+j)$
- *Maximum* expected additional return at state (i, j): $v(i, j)$

▪ **Optimality equation:**

$$v(i, j) = max\left\{0,\ \frac{i}{i+j}[v(i-1,\ j)-1]+\frac{j}{i+j}[v(i,\ j-1)+1]\right\}$$

with the *boundary* conditions, $v(i, 0) = 0$ and $v(0, j) = j$.

▪ **Optimality table:**

Number	4	***4***	3.2000	2.4000	1.6571	1.0000	0.4444	0.0667
of	3	***3***	2.2500	1.5000	0.8500	0.3429	0.0000	0.0000
blue	2	***2***	1.3333	0.6667	0.2000	0.0000	0.0000	0.0000
balls	1	***1***		0.0000	0.0000	0.0000	0.0000	0.0000
j	0	***0***	***0***	***0***	***0***	***0***	***0***	***0***
		0	1	2	3	4	5	6

Number of red balls, *i*

- Optimal strategy?
- Total expected net return, $v(6, 4)$ =

Ex 3] Secretary Problem

An employer would like to choose the best candidate among 4 competing applicants that arrive in a random order. After each interview, the position of the interviewee in the total order is revealed vis-á-vis already interviewed applicants. The interviewer has to decide, irrevocably, whether to accept the candidate for the position or to reject the candidate. The objective is to accept the *best* candidate with high probability.

▪ DP formulation

- *Stage* *i* = The *i*th applicant, where *i*=1, 2, …, *n*.
- *State* at the *i*th stage=$\begin{cases}1 & \text{if the } i\text{th applicant is a candidate} \\ 0 & \text{Otherwise}\end{cases}$
- *Decision* at stage *i* = {Stop, Go}
- *Reward*: Probability of selecting the *absolutely* best choice.

ϕ_i	if we *stop* at stage *i* with a “candidate”.
π_i	if we *go* at stage *i*.

		State	
		Candidate	Not a candidate
Decision	Stop	ϕ_i	0
	Go	π_i	π_i

▪ Optimality equation

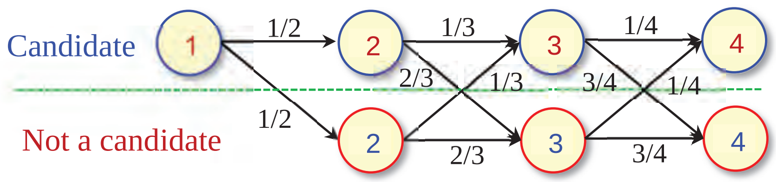
- Recursive equation:

If stop at stage *i* $\qquad \phi_i = i/n.$

If go at stage *i* $\qquad \pi_i = \frac{1}{i+1} max(\phi_{i+1}, \ \pi_{i+1}) + \frac{i}{i+1}\pi_{i+1}.$

The boundary condition is $\pi_n = 0$.

▪ Consider the case in which $n = 4$.



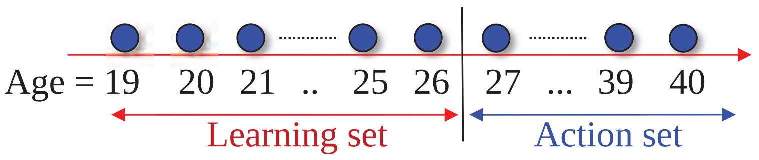
- Optimality table

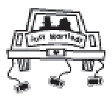
Stage	Decision	State: Candidate	State: Not candidate	Transition to candidate
4	stop	[1]	0	1/4
	go	0	0	
	max	1	0	
3	stop	[3/4]	0	1/3
	go	1/4	[1/4]	
	max	3/4	1/4	
2	stop	[2/4]	0	1/2
	go			
	max	2/4	5/12	
1	stop	1/4	*0*	1/1
	go	[11/24]	*11/24*	
	max	11/24	*11/24*	

▪ Stopping region: {2, 3, 4}

▪ Success rate: $P[\text{Win} \mid n=4] = \pi_0 =$

# Lindley (1961) said, “If, in real life, this process works between 18 and 40 (i.e., for 22 years), one should never propose until age 26.”





He explained, “Either many people do not pursue an optimal strategy or else they have a different utility function.”