

동적 계획법

목차

- 다단계 결정 문제
- 동적 계획법
- 동적 계획법의 적용

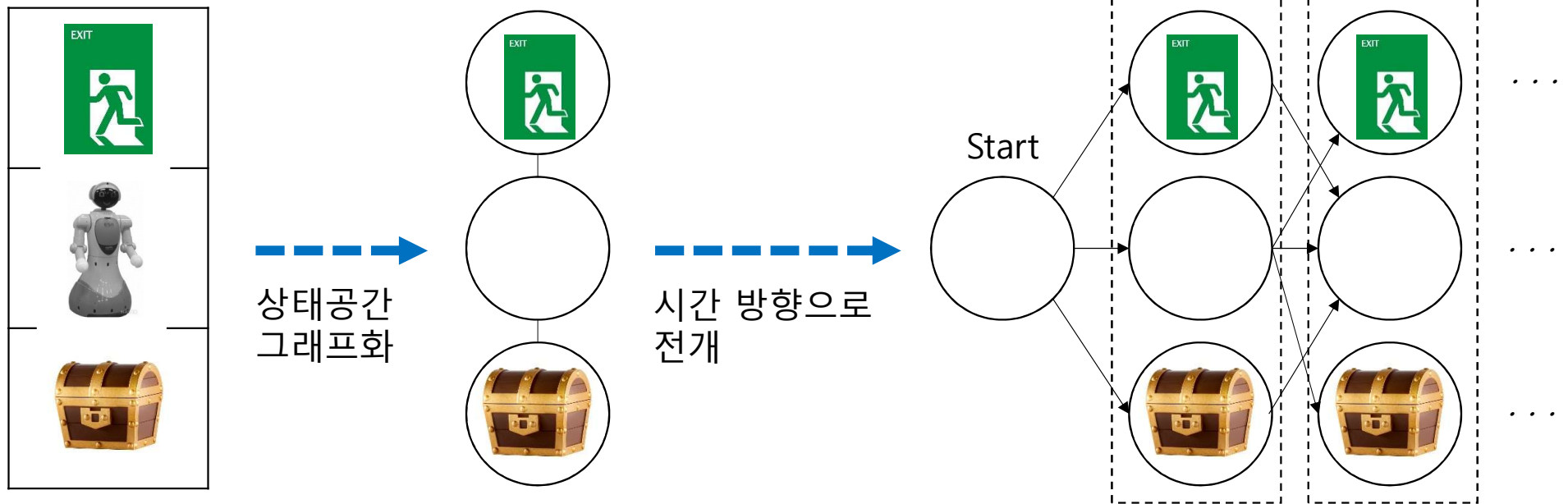
다단계 결정 문제

- 깊이우선, 너비우선 탐색, A*알고리즘 등을 사용하여 할 수 있던 것들
 - 상태를 탐색하고, 목표 상태를 도달하는 방법을 찾음
 - 비용까지 고려하여 최소비용으로 목표 상태를 도달하는 방법을 찾음
 - 경로와 목표 상태에 도달하는 것 외에 다른 것을 고려하지는 않았음
- 추가 고려 사항
 - 시간에 따른 상황변화
 - 목표 상태가 여럿인 경우

다단계 결정 문제(cont'd)

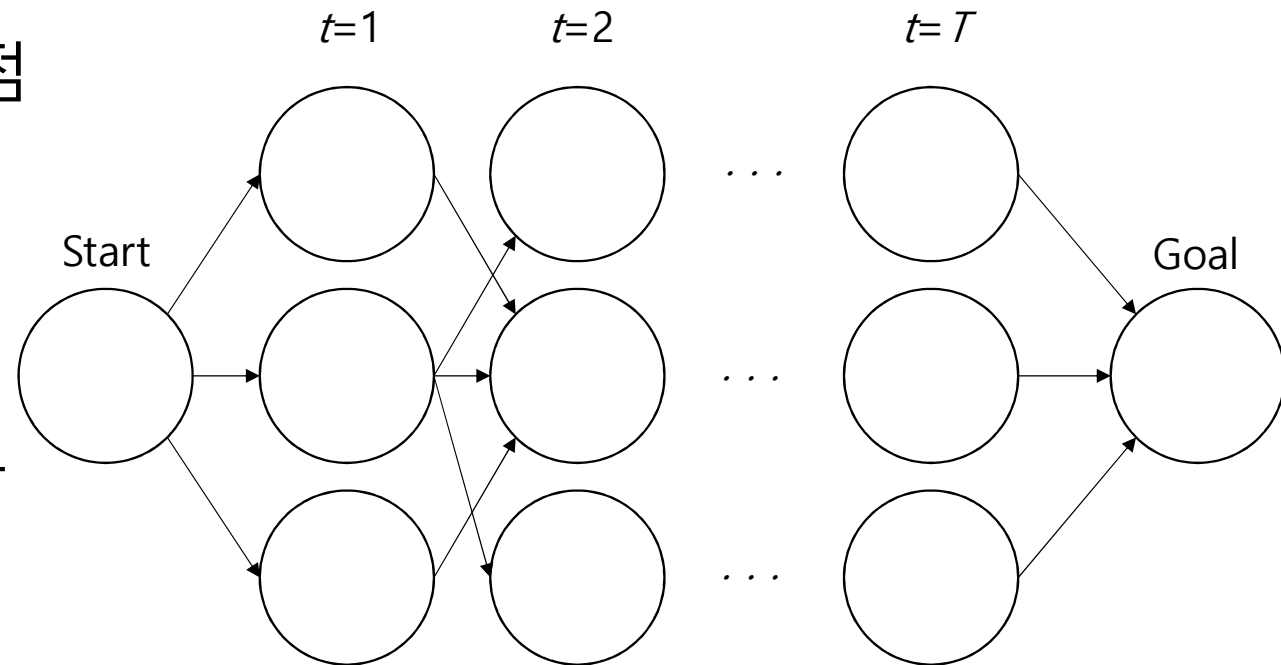
- 시간축이 존재하는 다단계 의사결정 문제
 - 어떤 시점 t 의 상태 s_t 에서 선택한 행동 a_t 가 이득 r_{t+1} 과 다음 시각의 상태 s_{t+1} 을 결정하고, 시각 $t+1$ 에서 취한 행동 a_{t+1} 이 시각 $t+2$ 의 상태 s_{t+2} 을 결정함
 - 이때 시각 T 까지 들게 될 비용의 합, 혹은 이익을 최적화하는 문제를 다단계 결정 문제(multi-stage decision problem)이라 함

다단계 결정 문제(cont'd)



다단계 결정 문제(cont'd)

- 시점 t 에서의 상태와 시점 $t+1$ 에서의 상태를 잇는 유향 에지가 행동을 나타냄
- 같은 장소라도 시점이 다르면 서로 다른 상태로 표현됨
- 시간은 역행하지 않으므로 유향 에지임



다단계 결정 문제(cont'd)

- 이러한 그래프에서 최대 누적 이득을 얻을 수 있는 행동 연속열은?
- 어떻게 이것을 효율적으로 계산할 수 있을까?
- 이러한 문제에 대한 해를 효율적으로 구하기 위한 방법이 동적 계획법(dynamic programming)임

동적 계획법

- 시각 $t = 1$ 에서 $t = T$ 까지에 대한 다단계 결정 문제의 해를 구하는 방법은?
 - 다단계 결정은 선택한 행동의 연속열로 볼 수 있음
 - 결정적 시스템에서는 상태의 연속열을 행동의 연속열로 생각할 수 있으므로 다단계 결정은 T 개의 상태 변수의 리스트(s_1, s_2, \dots, s_T)로 나타낼 수 있음
 - 위의 다단계 결정을 통해 얻는 경로의 평가 함수를 J 라고 할 때, 이 함수의 값이 최대가 되도록 하는 것이 경로 탐색의 목적이 됨
- $$J(s_1, s_2, \dots, s_T) \rightarrow \max$$

동적 계획법(cont'd)

- 시각 $t = 1$ 에서 $t = T$ 까지에 대한 다단계 결정 문제의 해를 구하는 방법은?(cont'd)

$$J(s_1, s_2, \dots, s_T) \rightarrow \max$$

- 이때, 상태 공간의 상태 수가 N 이라고 하면, 상태 변수의 리스트는 최대 N^T 가 존재함
- 상태가 3개에 10단계라고 하면, 3^{10} 으로 약 6만 가지이고, 20단계이면 약 35억개임
- 그러므로 모든 경로를 열거하며 평가하는 방법을 택하면 계산량은 지수적으로 증가하여 해결할 수가 없는 문제가 됨

동적 계획법(cont'd)

- 시각 $t = 1$ 에서 $t = T$ 까지에 대한 다단계 결정 문제의 해를 구하는 방법은?(cont'd)
 - 그러므로 다단계 결정 문제에서 평가 함수 J 를 상태의 쌍을 이루는 두 개의 변수에 대한 함수 h_t 의 합으로 나타내면 계산량을 많이 줄일 수 있음
 - 동적 계획법을 사용하면 실제 계산량은 지수증가에서 N 의 제곱에 비례하는 수준으로 줄어듦

$$J(s_1, s_2, \dots, s_T) = \sum_{k=2}^T h_k(s_{k-1}, s_k)$$

$$r_t = h_t(s_{t-1}, s_t)$$

동적 계획법(cont'd)

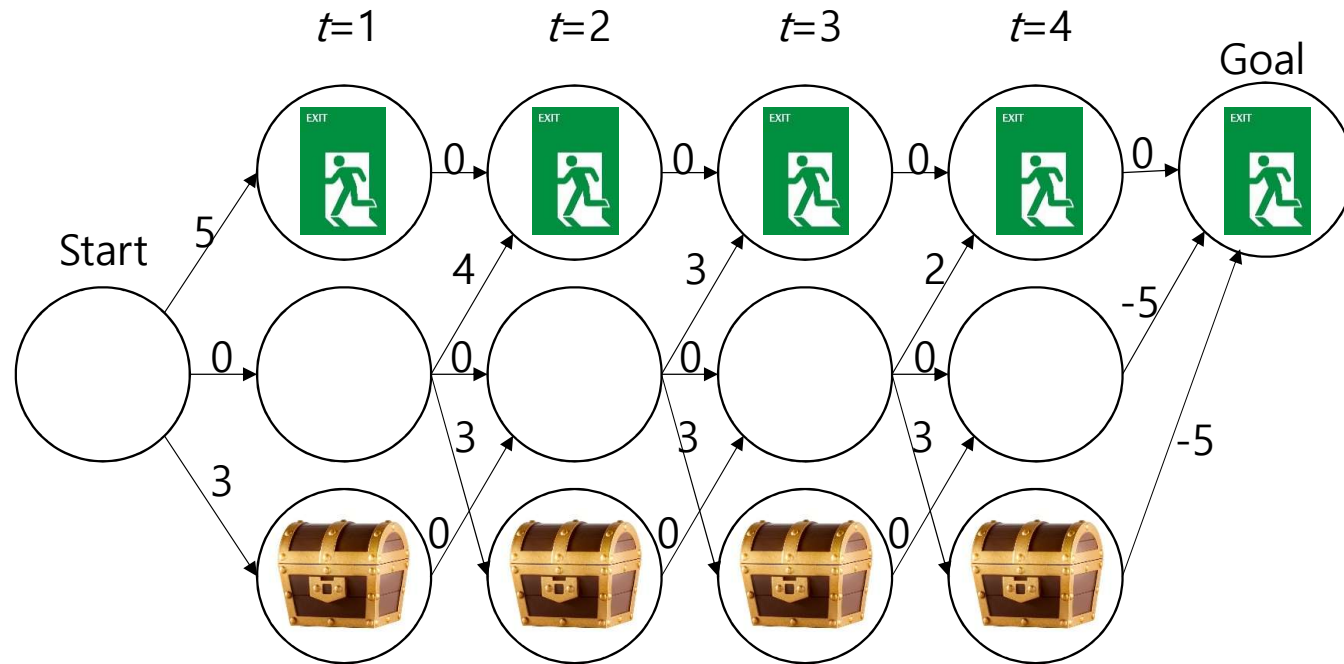
- 시각 $t = 1$ 에서 $t = T$ 까지에 대한 다단계 결정 문제의 해를 구하는 방법은?(cont'd)

$$J(s_1, s_2, \dots, s_T) = \sum_{k=2}^T h_t(s_{t-1}, s_t)$$

$$r_t = h_t(s_{t-1}, s_t)$$

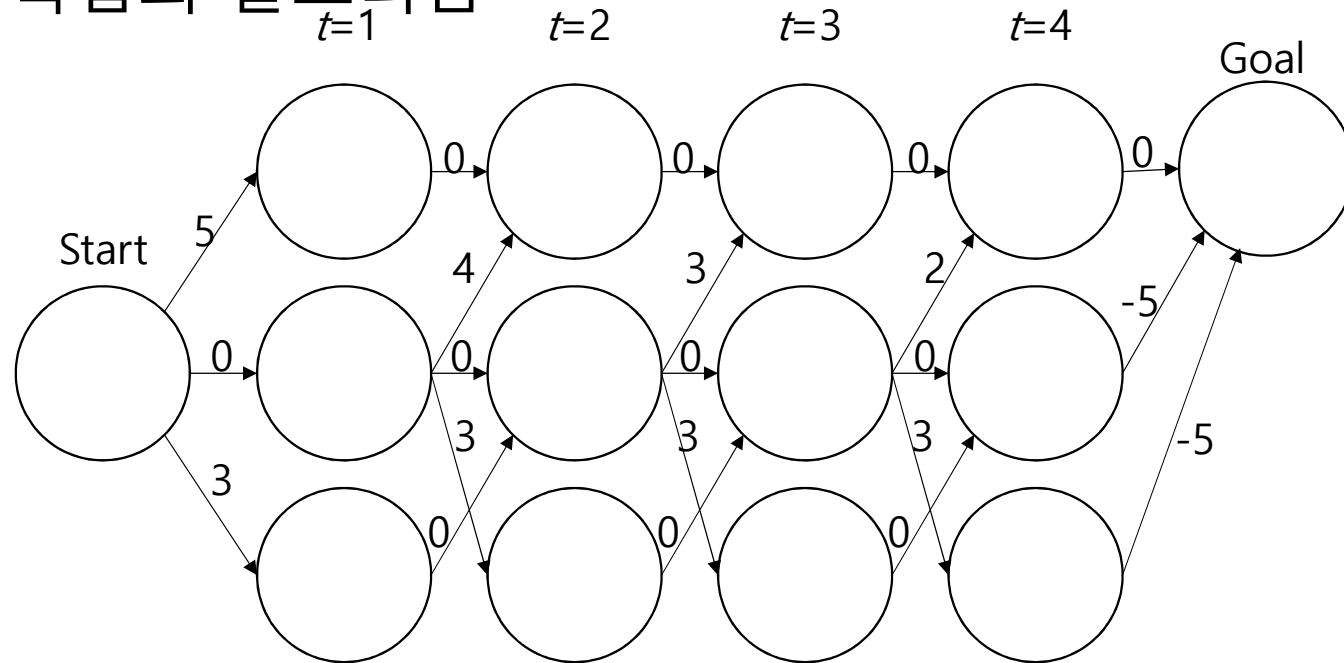
- h_t 는 시점 t 에 대한 각각의 유형 에지 (s_{t-1}, s_t) 에 대한 평가값을 부여하는 평가 함수임

동적 계획법(cont'd)



동적 계획법(cont'd)

- 동적 계획법의 알고리즘

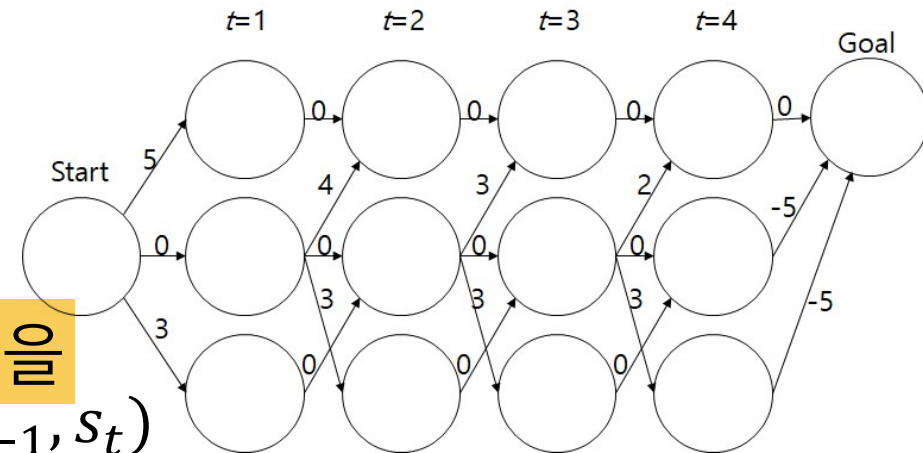


동적 계획법(cont'd)

- 동적 계획법의 알고리즘(cont'd)

```
for  $t=1$  to  $T$  do  
   $F_t(s_T) =$ 
```

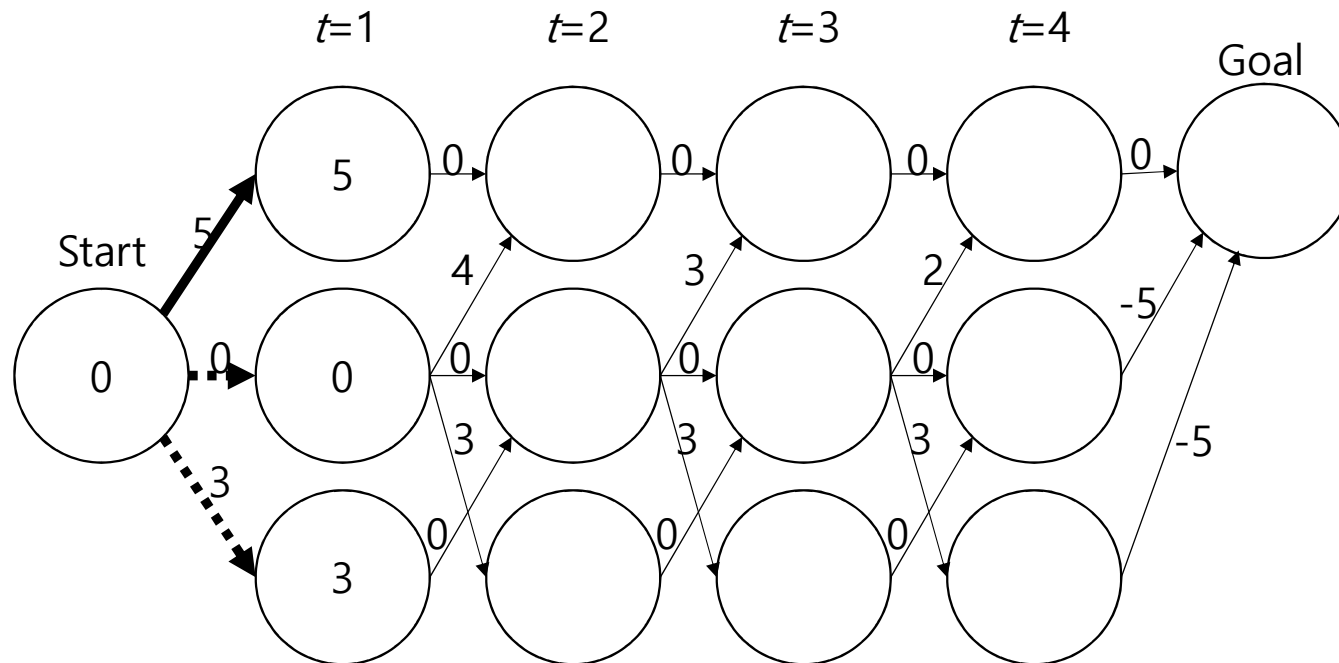
동적 계획법의 적용



- 각각의 유형 에지위의 숫자는 **그 행동을 취할 때 얻을 수 있는 이득**, $r_t = h_t(s_{t-1}, s_t)$
- 출구에 이르기까지 이득의 합이 최대가 되도록 하는 것이 목표임, $t=4$ 까지 출구에 도달하도록 함
- 만약 $t=4$ 가 지나도록 출구에 도달하지 못하면 -5의 이득
- 보물 상자를 얻을 때 이득은 3이며 횟수에 제한이 없음
- 출구에 일찍 도달할 수록 이득이 높고 도착이 시각이 1씩 늦을 수록 이득이 줄어듦
- 보물 상자에는 머물러 있을 수가 없음
- 출구에 오면 떠날 수가 없음

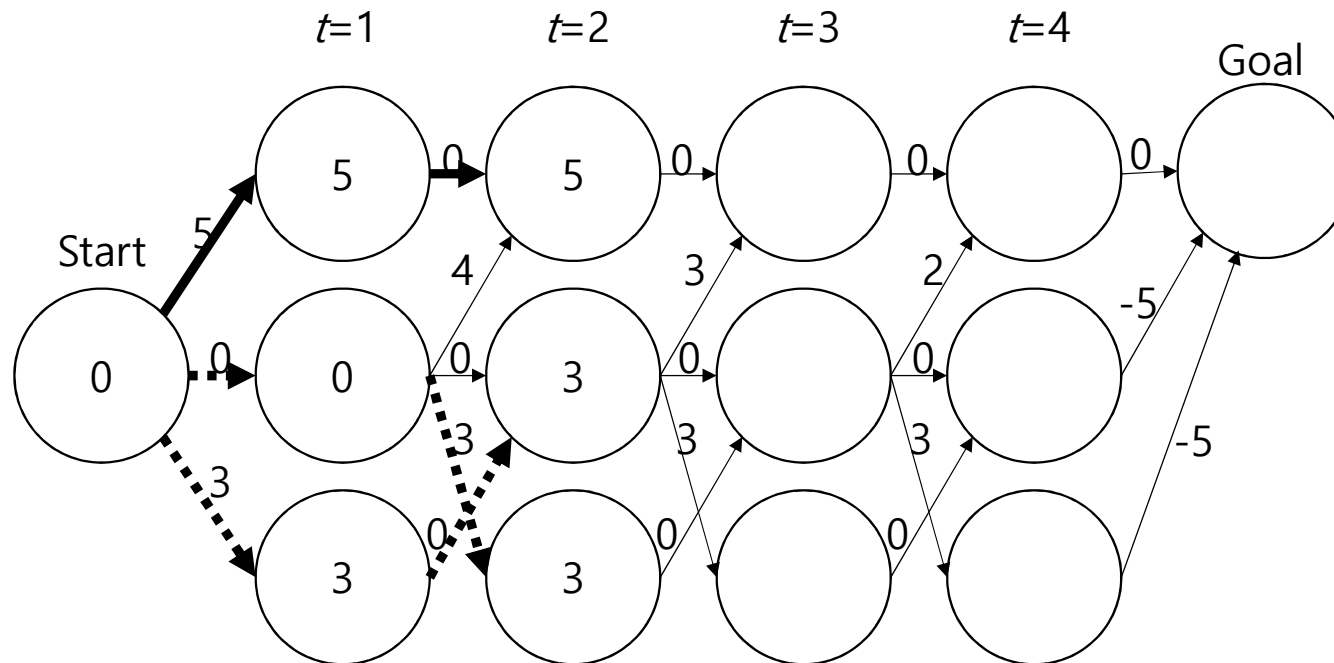
동적 계획법의 적용(cont'd)

- 1단계



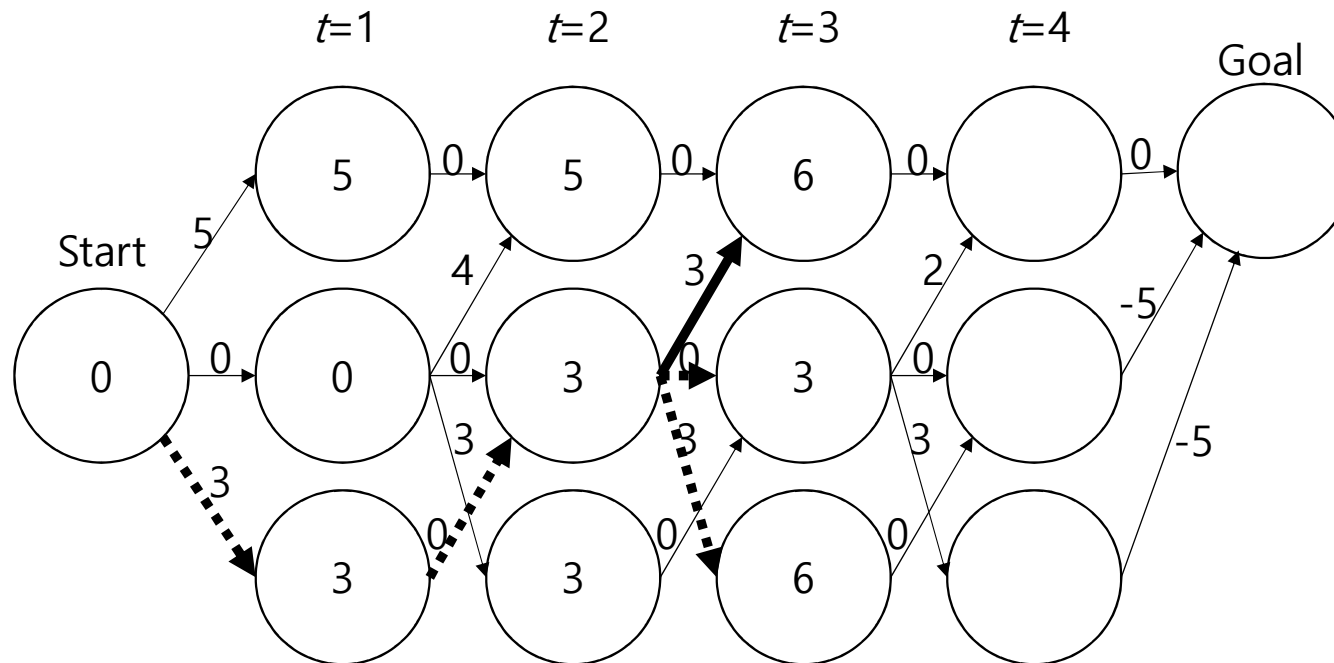
동적 계획법의 적용(cont'd)

- 2단계



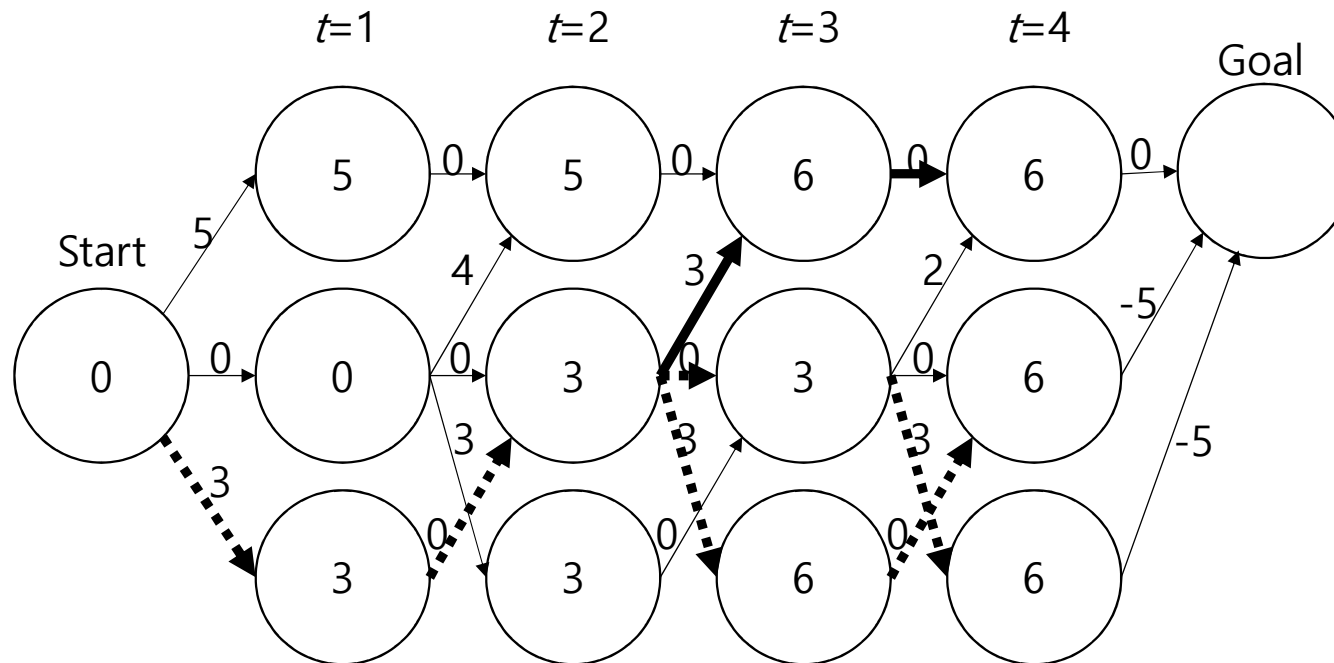
동적 계획법의 적용(cont'd)

- 3단계



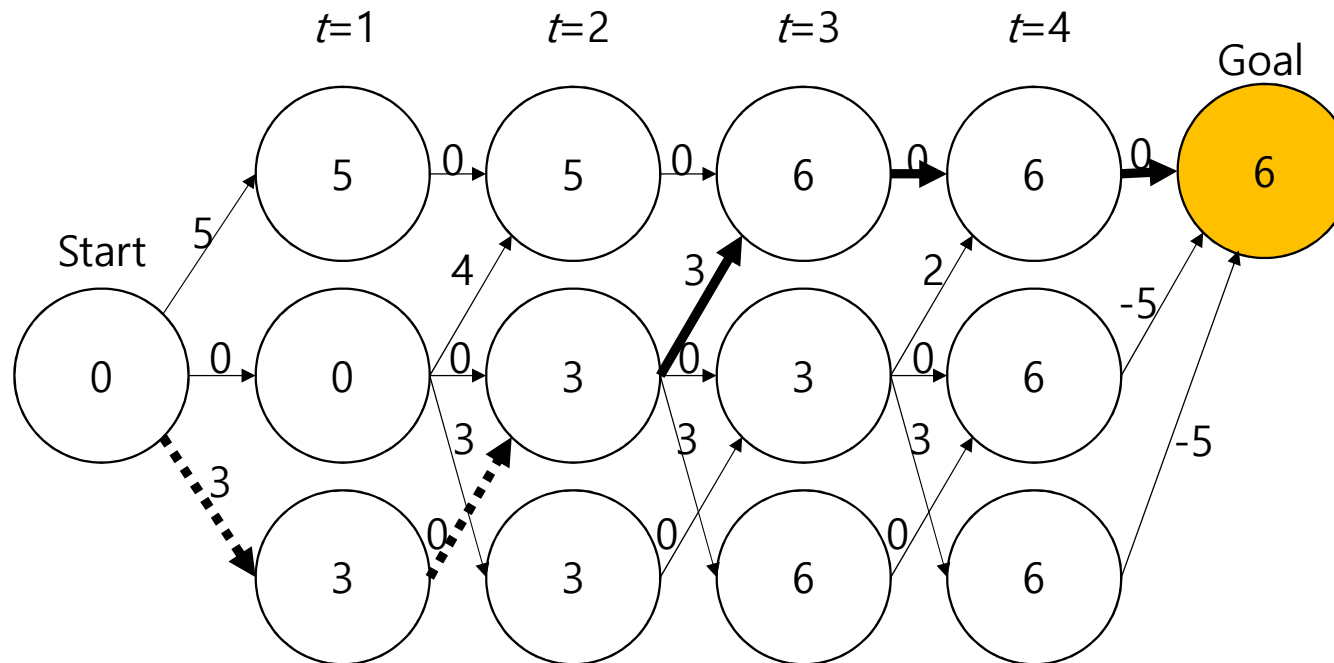
동적 계획법의 적용(cont'd)

- 4단계



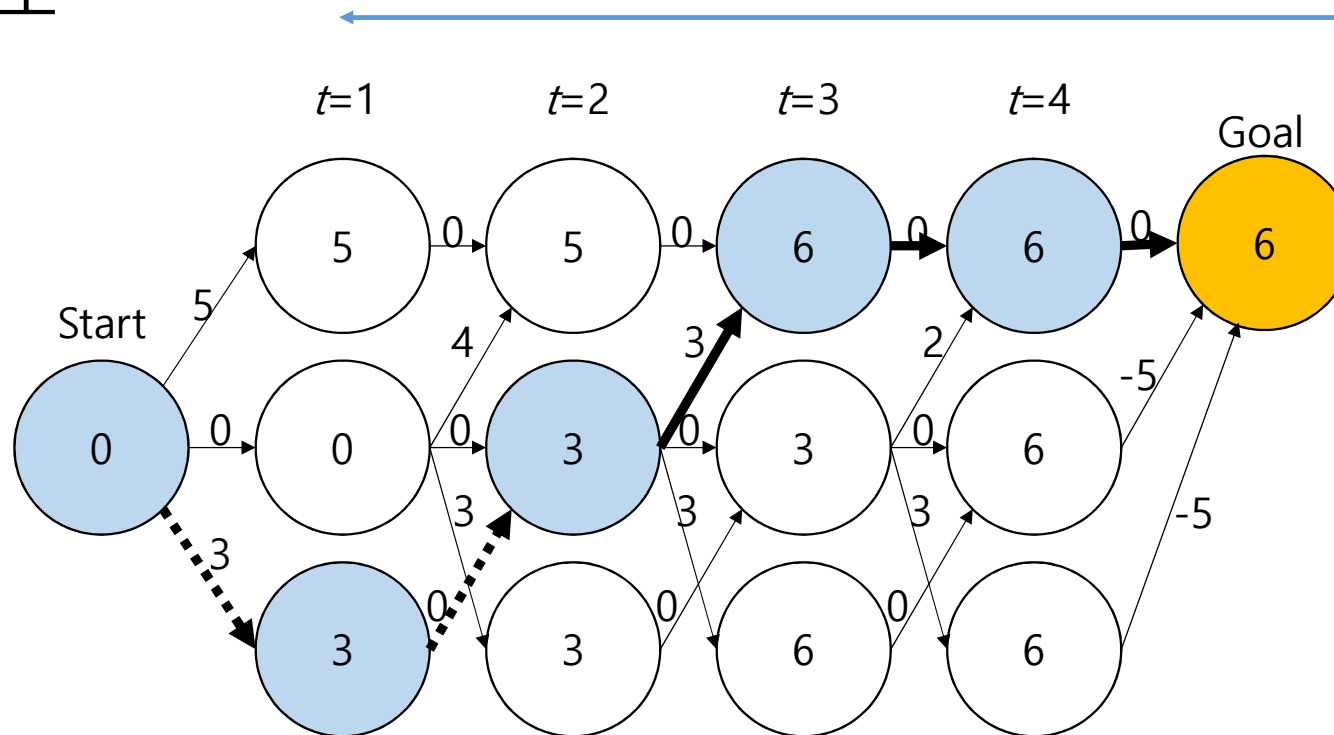
동적 계획법의 적용(cont'd)

- 5단계



동적 계획법의 적용(cont'd)

- 최적 경로



동적 계획법의 적용(cont'd)

- 편집 거리(edit distance)
 - 문자열과 문자열의 거리를 재는 척도
 - 두 문자열이 얼마나 다른가?(ex. DNA 염기 서열이 얼마나 다른가?)
- 문자열 사이의 거리를 정의하는 가장 심플한 방법으로 해밍 거리(Hamming distance)도 있음
 - 문자열에 포함된 문자를 앞에서부터 하나씩 비교하여 몇 개나 다른가를 출력하는 거리 함수임
 - 'ILOVEYOU'와 'YLOVEIOU'는 첫 번째 I, Y 그리고 여섯 번째, Y와 I가 달라서 해밍 거리는 2임

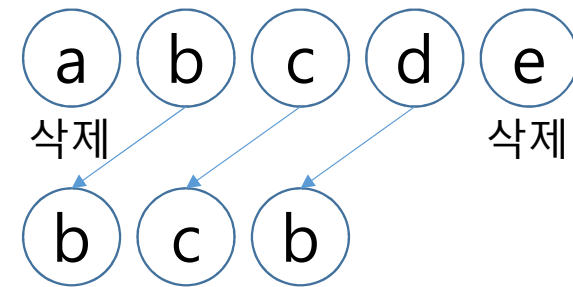
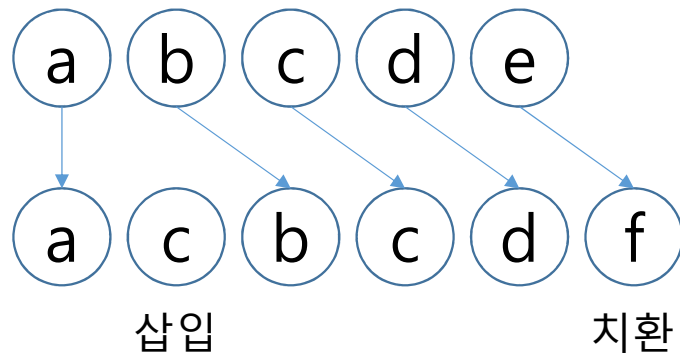
동적 계획법의 적용(cont'd)

- 그런데 서로 어긋나는 문자열을 해밍 거리로 계산하면 우리가 직관적으로 인식하는 것보다 거리 차이가 많이 남
 - ILOVEYOU와 LOVEYOU는 l가 하나 빠진 것임에도 불구하고 해밍 거리가 8임
 - 그러므로 문자의 추가나 생략에 대한 변화도 한 글자의 (상태)변화로 생각하여 거리를 계산하는 방법이 편집 거리임

동적 계획법의 적용(cont'd)

문자열1	문자열2	해밍 거리	편집 거리	설명
abcd	abcd	0	0	문자가 바뀌지 않았고 거리는 0임
abcd	abed	1	1	한 문자가 바뀌어 거리 1이 됨
abcd	acd	3	1	해밍 거리로는 3문자가 바뀌었고, 편집 거리로는 'b' 한 글자만 지워진 것임
abcd	axbcd	4	1	해밍 거리로는 4문자가 바뀌었고, 편집 거리로는 'x' 한 글자만 추가된 것임

동적 계획법의 적용(cont'd)



동적 계획법의 적용(cont'd)

	\$	a	e	b	c
\$	0	1	2	3	4
a	1				
b	2				
c	3				
d	4				Goal

동적 계획법의 적용(cont'd)

- 두 문자열 'ab'와 'ae'를 비교하는 경우

	\$	a	e
	0	1	
a	1		
b			

Diagram illustrating the dynamic programming table for comparing the strings 'ab' and 'ae'. The table shows the minimum edit distance (0 for match, 1 for mismatch or insertion/deletion) between the prefixes of the two strings.

Annotations (Red boxes):

- 치환:1, 일치: 0 (Mismatch/Match cost)
- 삽입: 1 (Insertion cost)
- 삭제: 1 (Deletion cost)

Arrows indicate the path from the top-left cell (0) to the bottom-right cell (1) via the middle-right cell (1).

동적 계획법의 적용(cont'd)

	\$	a	e	b	c
\$	0	1	2	3	4
a	1	0	1	2	3
b	2	1	1	1	2
c	3	2	2	2	1
d	4	3	3	3	2

A가 E로 치환됨.

0 -> 1

1 -> 2

2 -> 3

이중 최소를 선택!

이동하면서 얻게되는 값중 최소 값을 선택!! 최소값을 유지해 가기.

Summary

- 결정 시스템에 대한 다단계 결정 문제를 형식화 하였음
- 상태 공간을 시간 방향의 그래프로 전개하는 방법을 살폈음
- 동적 계획법의 알고리즘을 배웠음
- 동적 계획법을 이용하여 문자열의 편집 거리를 계산하는 방법을 살폈음

