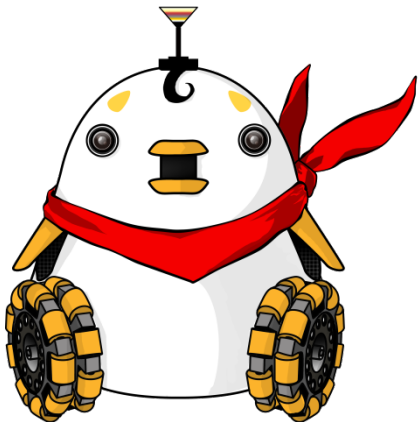


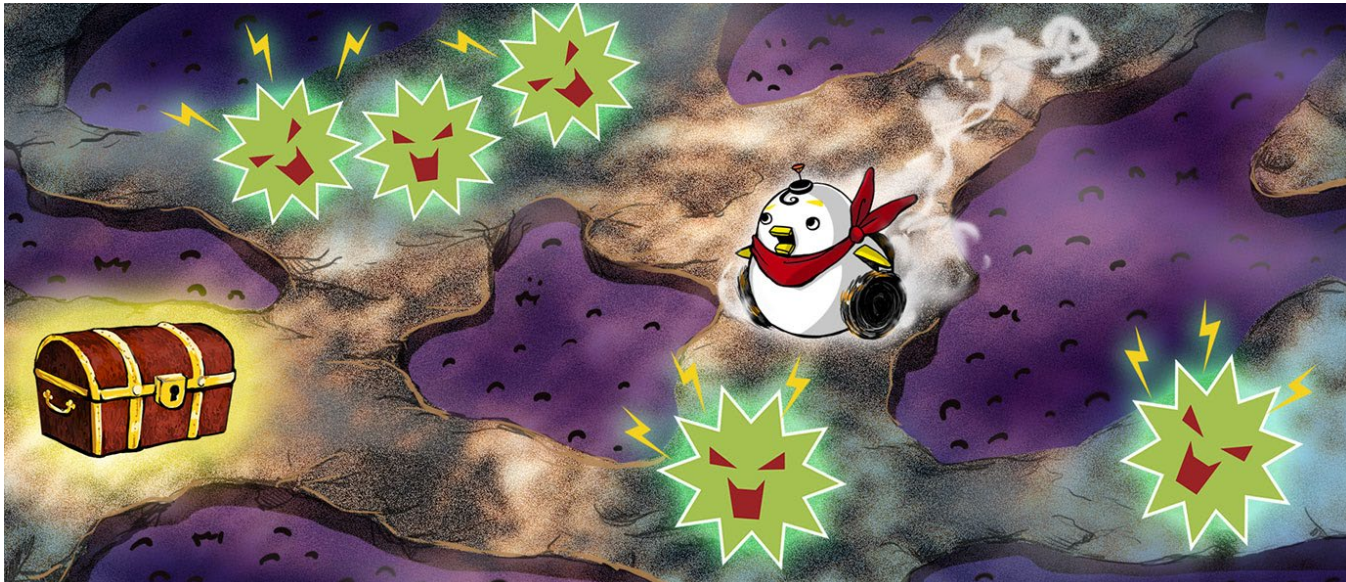
# 人工知能

## 第5章 計画と決定（1）： 動的計画法



# STORY 動的計画法

- 常に状態や状態間のコストが変わらず，ゴールが一つであればA\*アルゴリズムでゴールに向かうことができる．しかし，実際にホイールダック2号がとるべき行動は脇目も振らずにゴールに向かうことだろうか．
- ある時刻に現れるアイテムを途中で確保しないといけないかもしれないし，ある時刻に通りがかる敵を避けないといけないかもしれない．また，ゴールもいくつか存在しえるだろうし，その中でも最も「お得な」ゴールにたどり着くべきだろう．しかし，だからといってすべての行動パターンを試していたのではとてもやっていられない．さてどうすべきか．



# 仮定 動的計画法

- ホイールダック 2 号は迷路の完全な地図を持っているものとする.
- ホイールダック 2 号は迷路の中で自分がどこにいるか認識できるものとする.
- ホイールダック 2 号は連続的な迷路の空間から適切な離散状態空間を構成できるものとする.
- ホイールダック 2 号は各時刻で各状態間の移動にかかるコストや利得を知っているものとする.
- ホイールダック 2 号は物理的につながっている場所・状態には意図すれば確定的に移動することができるものとする.

# Contents

□5.1 多段決定問題

□5.2 動的計画法

□5.3 ホイールダック2号「宝箱を拾ってゴール」

□5.4 例：編集距離の計算

## 5.1.1 はじめに

- 時間軸のある意思決定問題を考える．ある時点 $t$ で選択した行動が次の時点 $t + 1$ の状態を決め，時点 $t + 1$ での行動が時点 $t + 2$ での状態を決める．
- その上で，各時点での行動選択にもとづいて利得，もしくは費用が発生する．このようなときに時刻 $T$ までにかかる費用の和を最小化，もしくは，得られる利得の和の最大化を行う計画問題を**多段決定問題**という．



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

## 5.1.2 グラフを時間方向に展開する

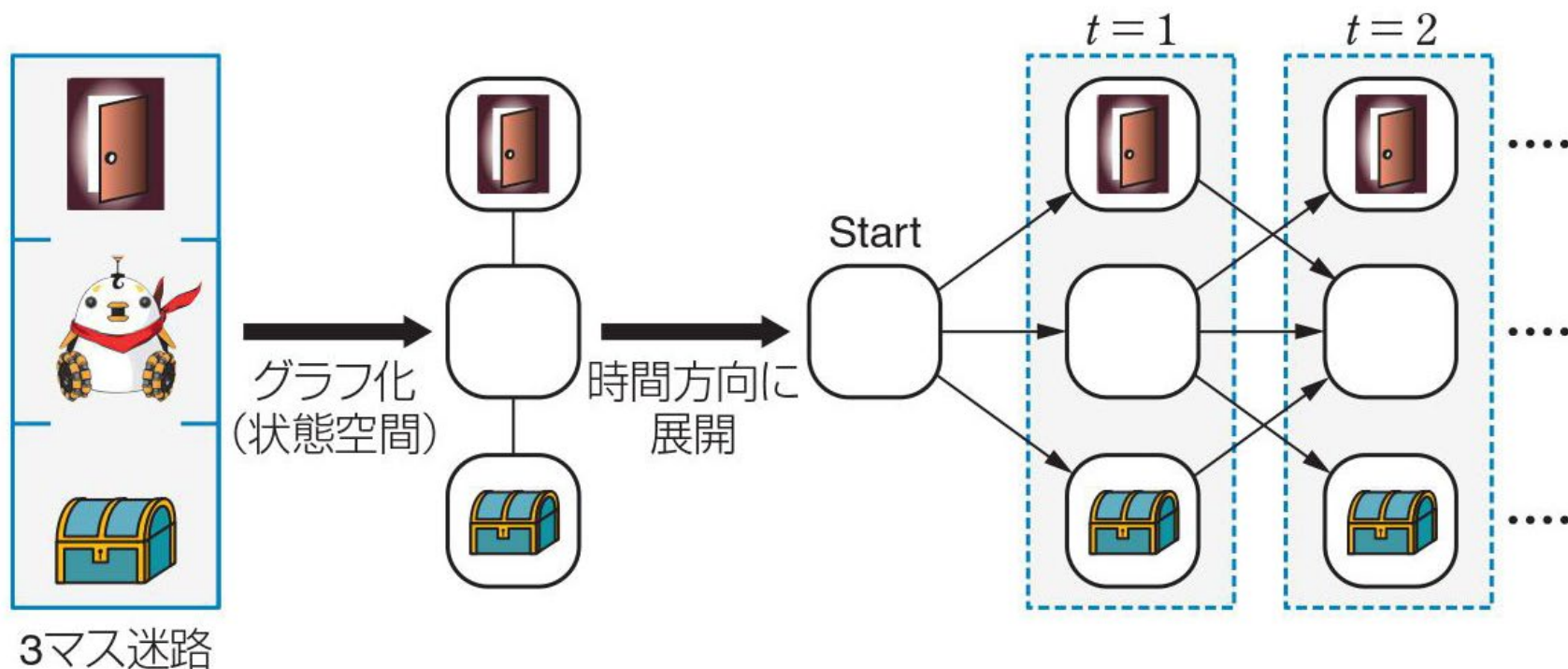
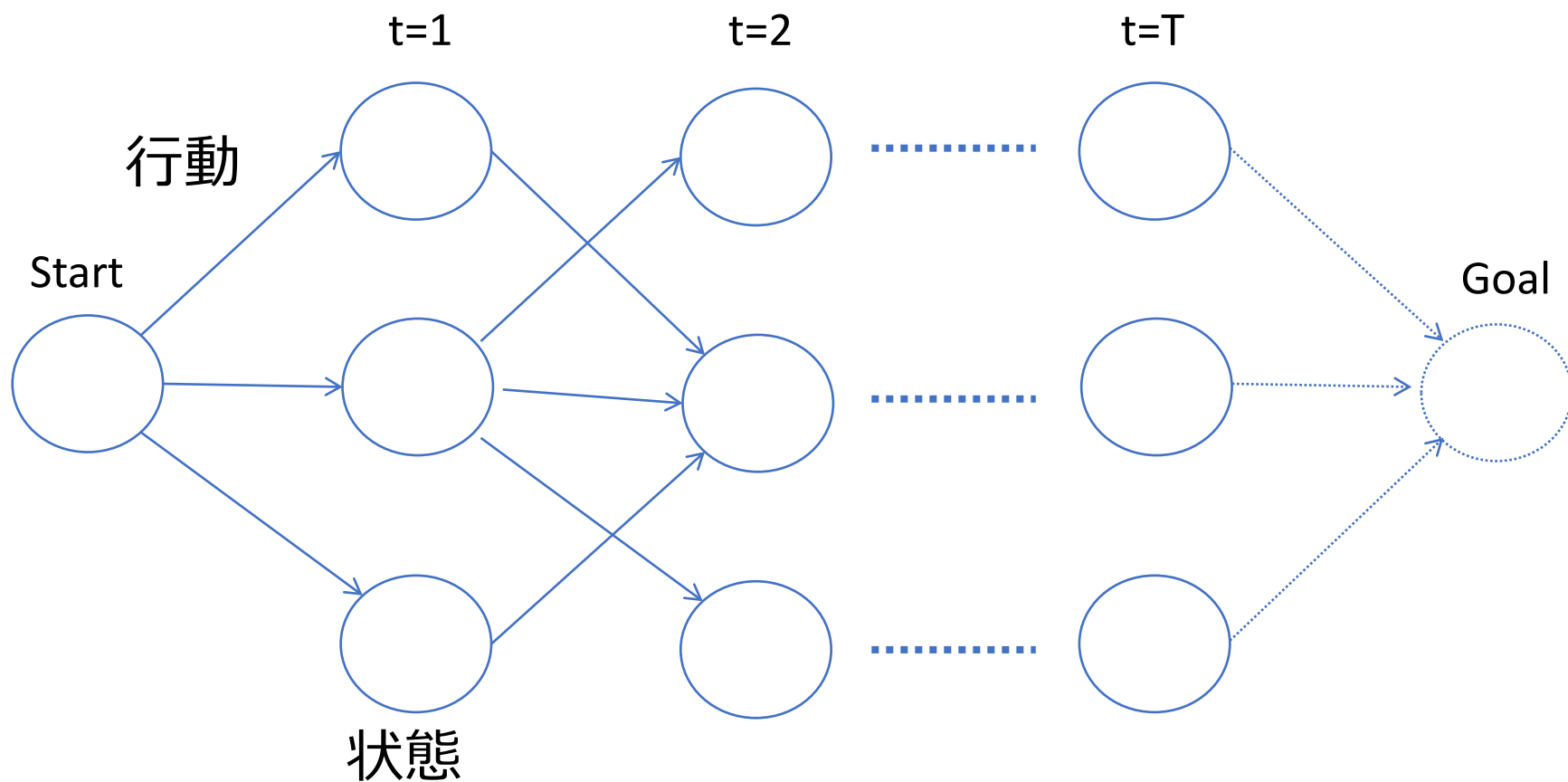


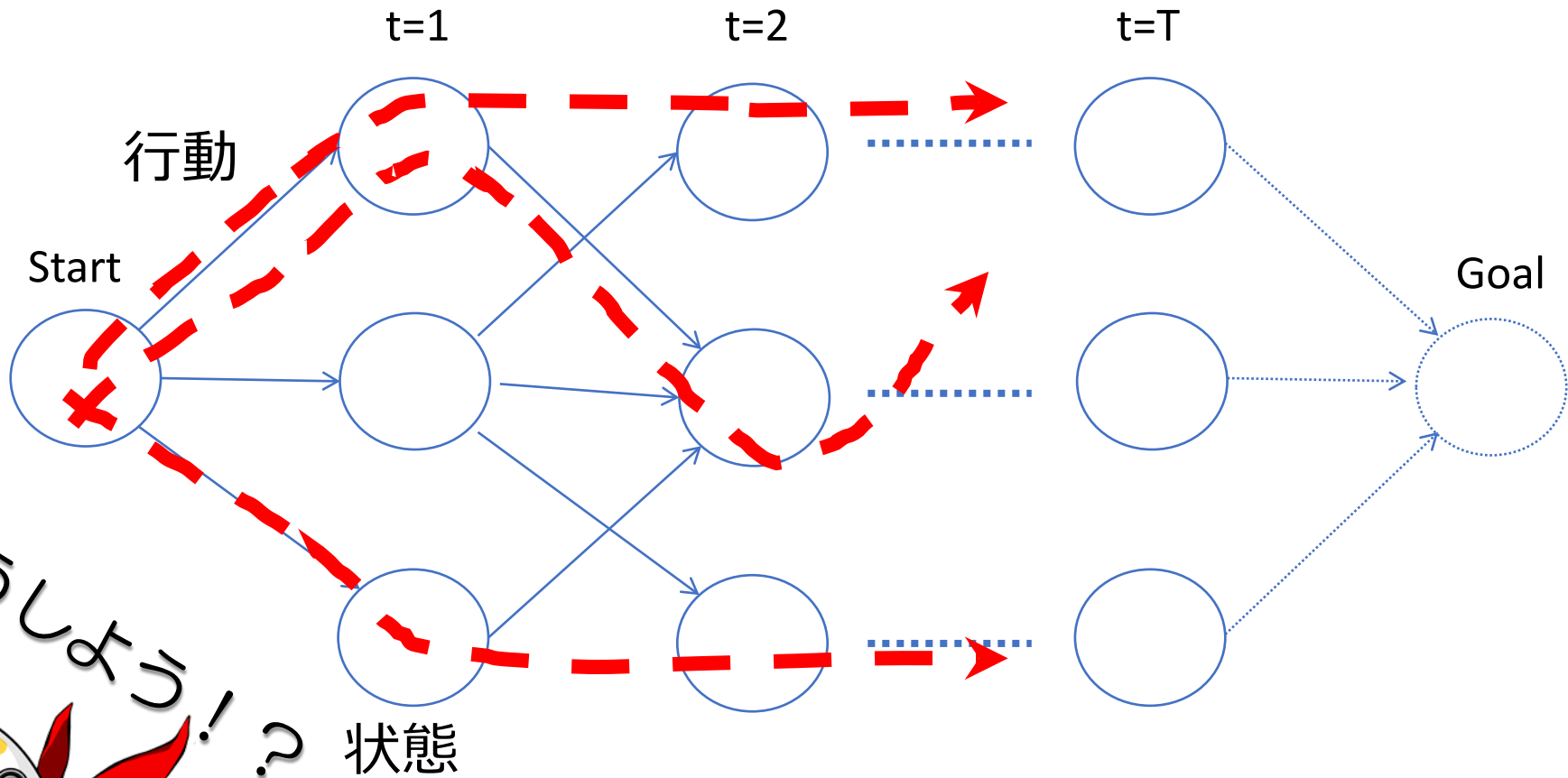
図 5.2

状態空間の時間方向への展開とグラフの作成

## 5.1.2 多段決定問題のグラフ表現



# あらゆる経路を列挙的に探索する



N個の選択肢がT回ある！ 計算量は  $O(N^T)$



# Contents

□5.1 多段決定問題

□5.2 動的計画法

□5.3 ホイールダック2号「宝箱を拾ってゴール」

□5.4 例：編集距離の計算

## 5.2.1 経路と計算量

□この経路の評価関数を $J$ とすると、これを最大化することが経路探索の目的となる。

計算量爆発！

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

計算量は  $O(N^T)$

□動的計画法は多段決定問題において、各評価値が状態の対ごとの二変数関数の和で書けることを利用してこれを効率化するアルゴリズムである。

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

計算量を  $O(N^2 T)$  まで縮減

# 指数オーダー⇒2次オーダー のインパクト

- $N=100$ 状態,  $T=34$ ステップの場合

- $O(N^T)$

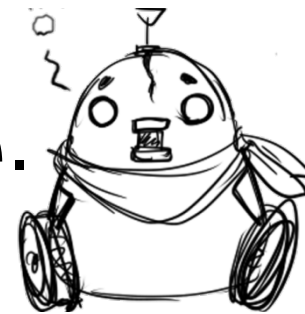
- 1無量大数回

- ⇒現実的には終わらない.

- $O(N^2T)$

- 34万回

- ⇒数GHz=数十億Hz の  
CPUならあっという間



## 5.2.2 動的計画法のアルゴリズム

### Algorithm 5.1 動的計画法

① for  $t = 1$  to  $T$  do

②

$$F_t(s_t) = \max_{s_{t-1}} [F_{t-1}(s_{t-1}) + h_t(s_{t-1}, s_t)] \quad (5.4)$$

および、その最大値を与える  $s_{t-1}$  を  $\hat{s}_{t-1}(s_t)$  としてメモリに保持する。

メモ化

③ end for

④  $F_T(s_T)$  を最大にする  $s_T$  の値  $s_T^*$  を探索し、その最大値を  $J^* \leftarrow F_T(s_T^*)$  とする。

⑤ for  $t = T - 1$  to  $1$  do

⑥  $s_t^* = \hat{s}_t(s_{t+1}^*)$  を計算する。

⑦ end for

⑧ return 経路  $(s_1^*, s_2^*, \dots, s_T^*)$  および  $J^*$  を返す。

# Contents

□5.1 多段決定問題

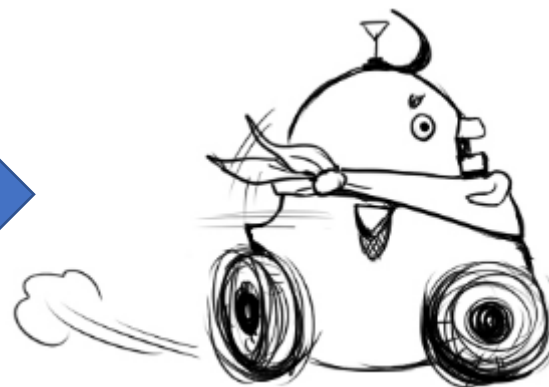
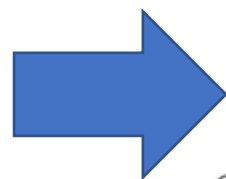
□5.2 動的計画法

□5.3 ホイールダック2号「宝箱を拾ってゴール」

□5.4 例：編集距離の計算

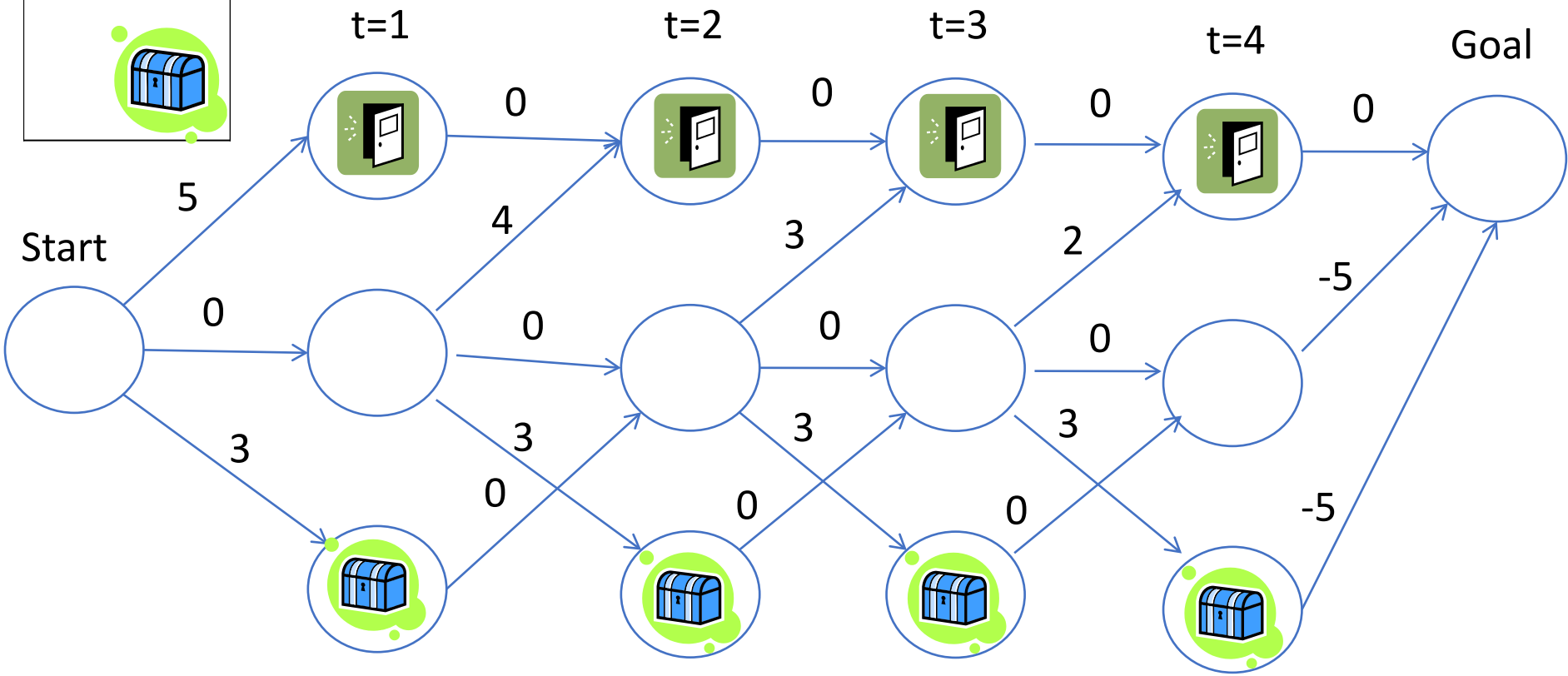
# 例: 「宝箱を拾ってゴール」

$t = 4$  までにゴールできなかった場合、ペナルティとして-5の利得を没収される。宝箱をとることは何度でもでき、この時には3の利得を得る。また、早くゴールしたほうが利得は高く、ゴールが一時刻遅れるたびにゴール時の利得は減っていく。宝箱の場所にはとどまることはできない。また、一度ゴールすると、ゴールから再度出てくることはできない。



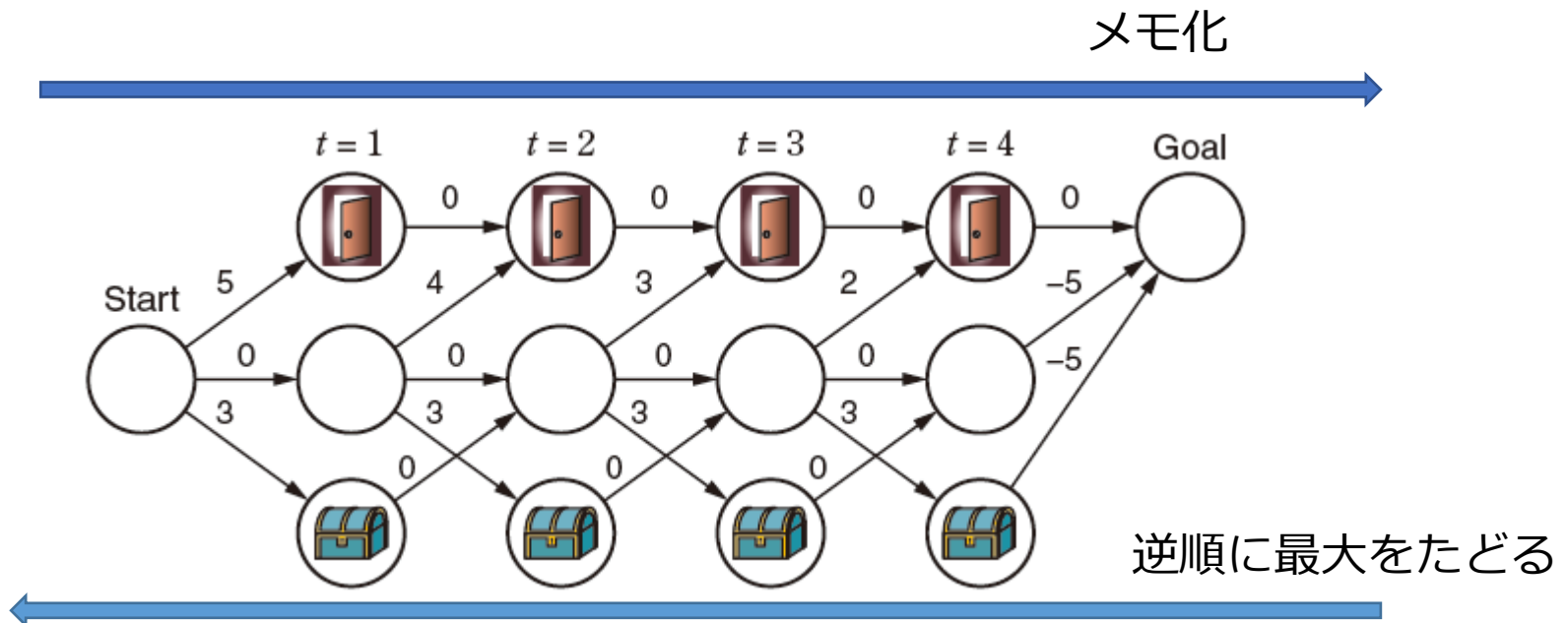


# 例: 「宝箱を拾ってゴール」



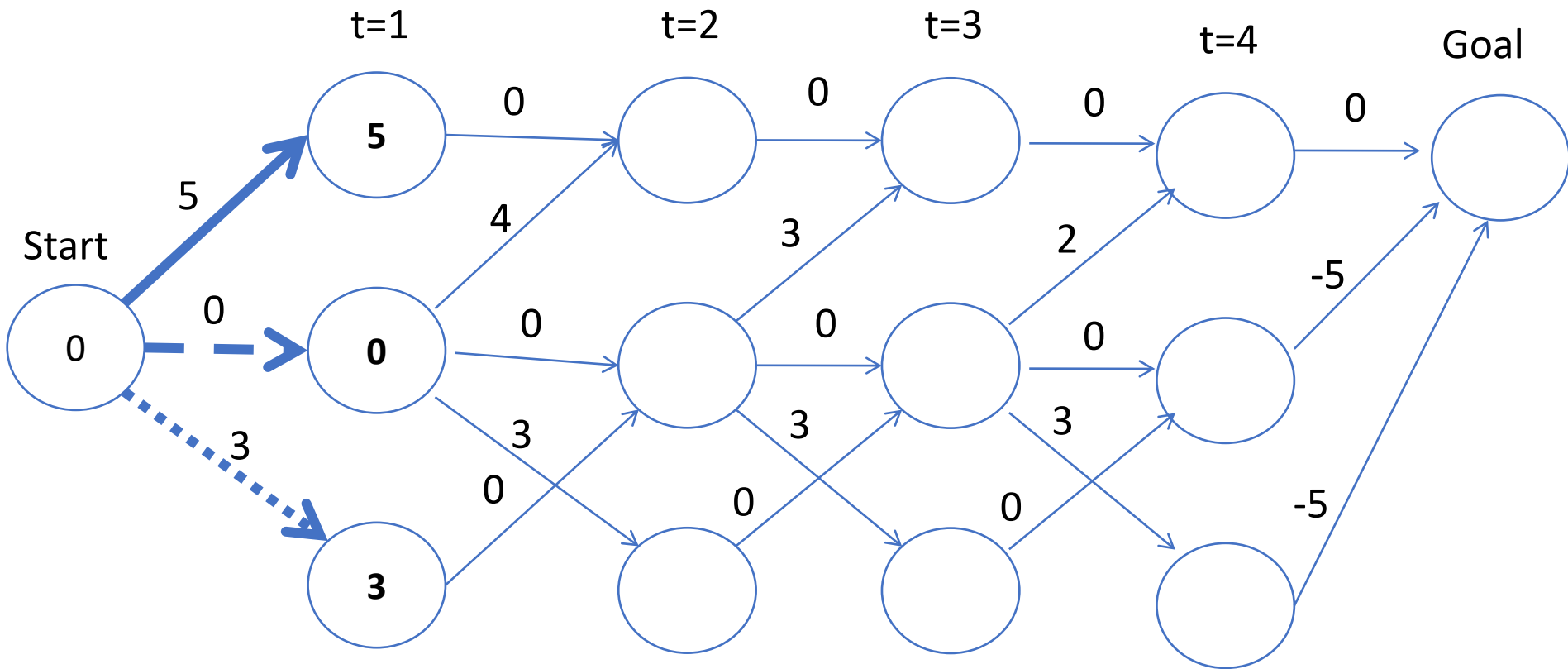
# (ポイント) 動的計画法のアルゴリズム

- まず、左から順に各状態までの最適パスを計算し、その時の評価値を状態に記述していく。これを**メモ化(Memoization)**という。これを繰り返していくことで、最終時刻に至った段階で、これを逆順にたどることで最適なパスがひと通りに決まる。

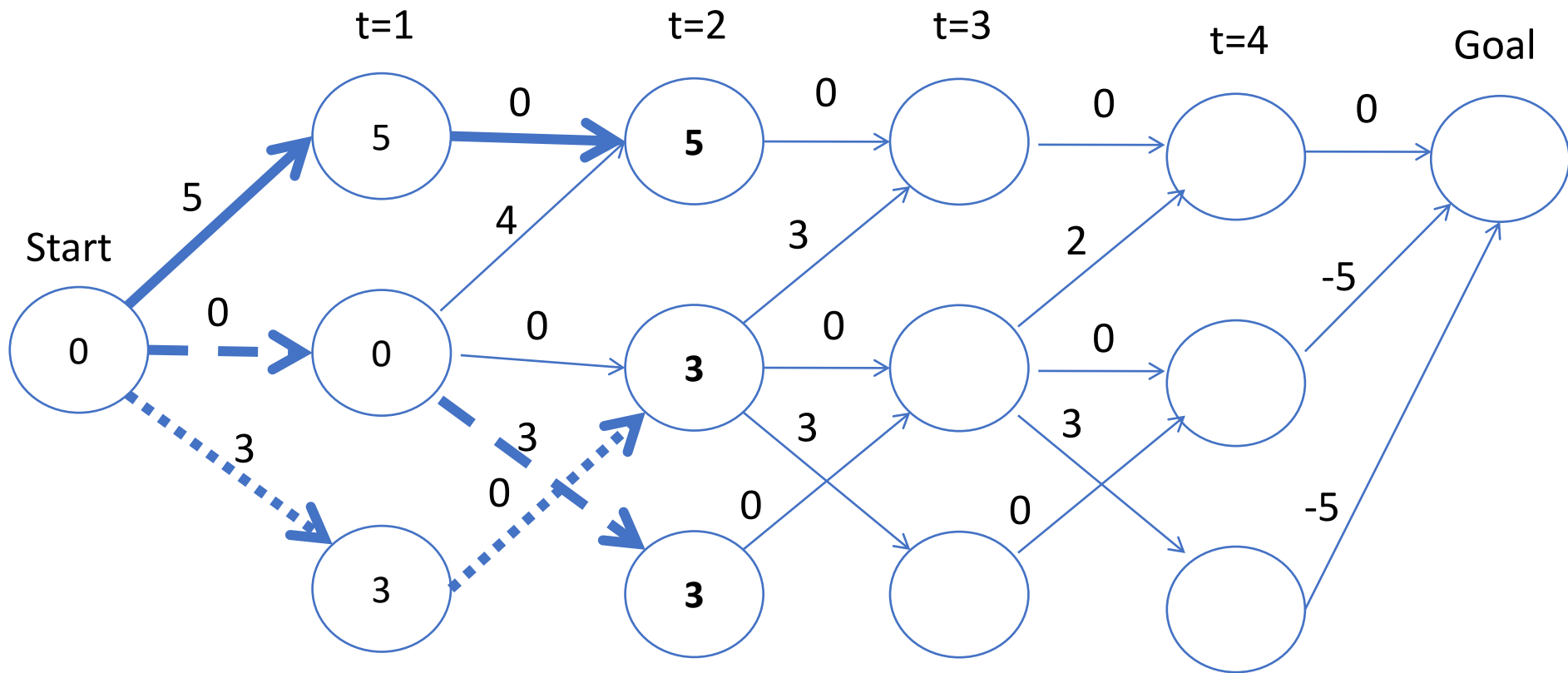




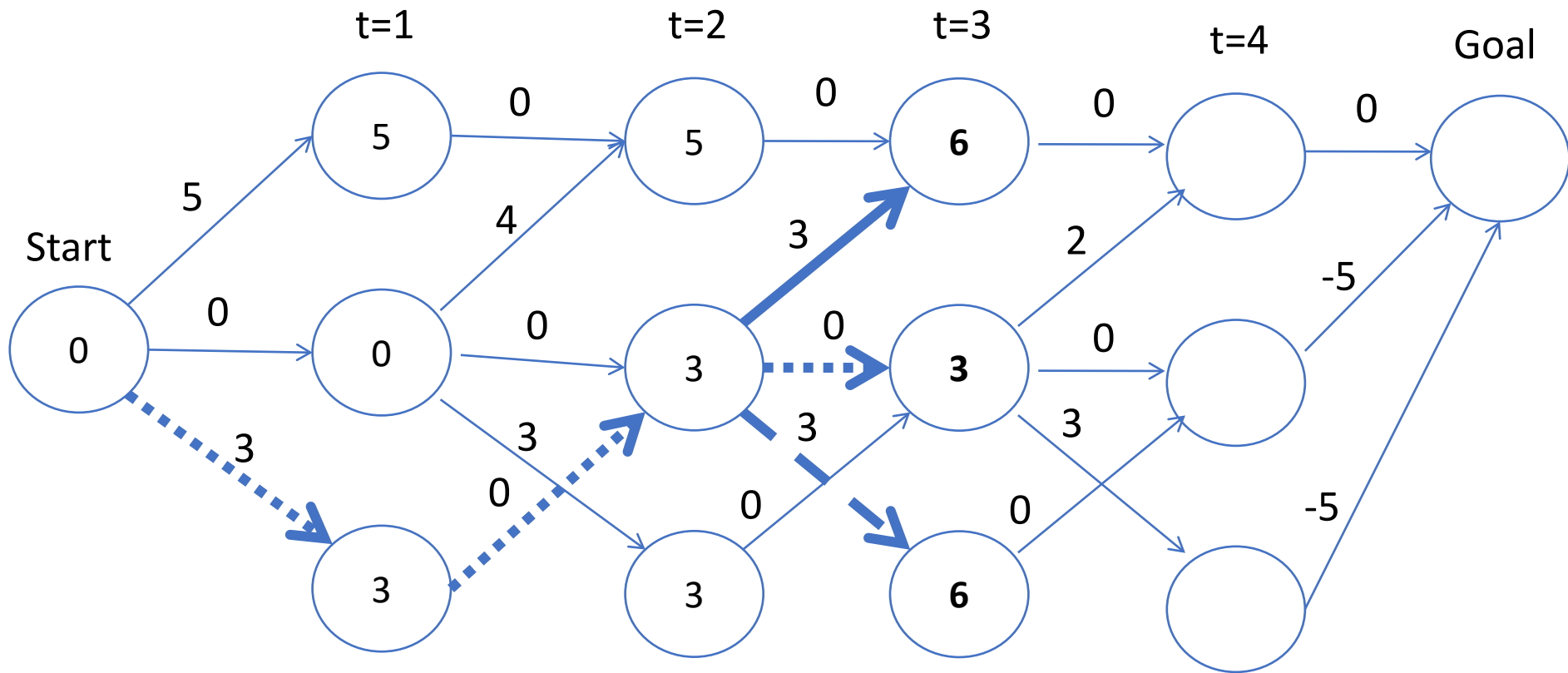
# Step 1



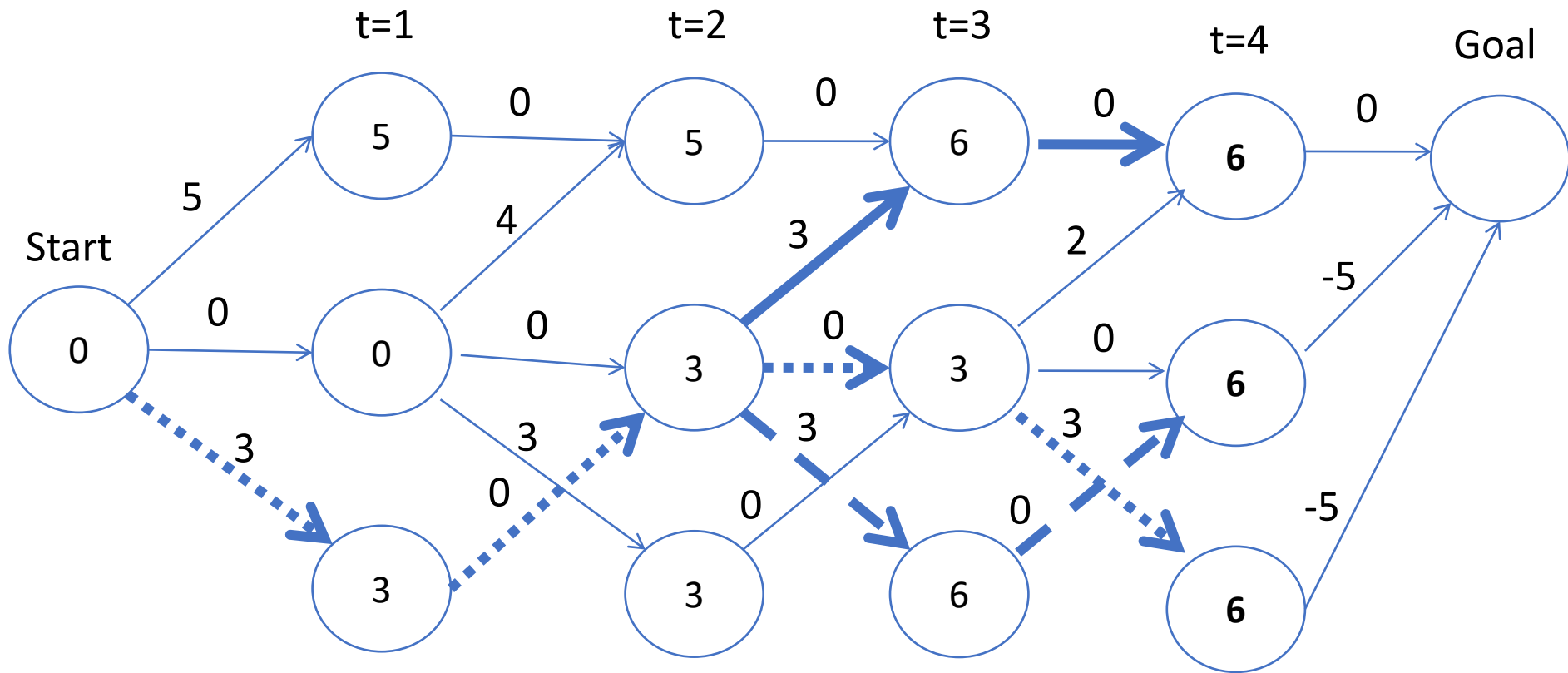
## Step 2



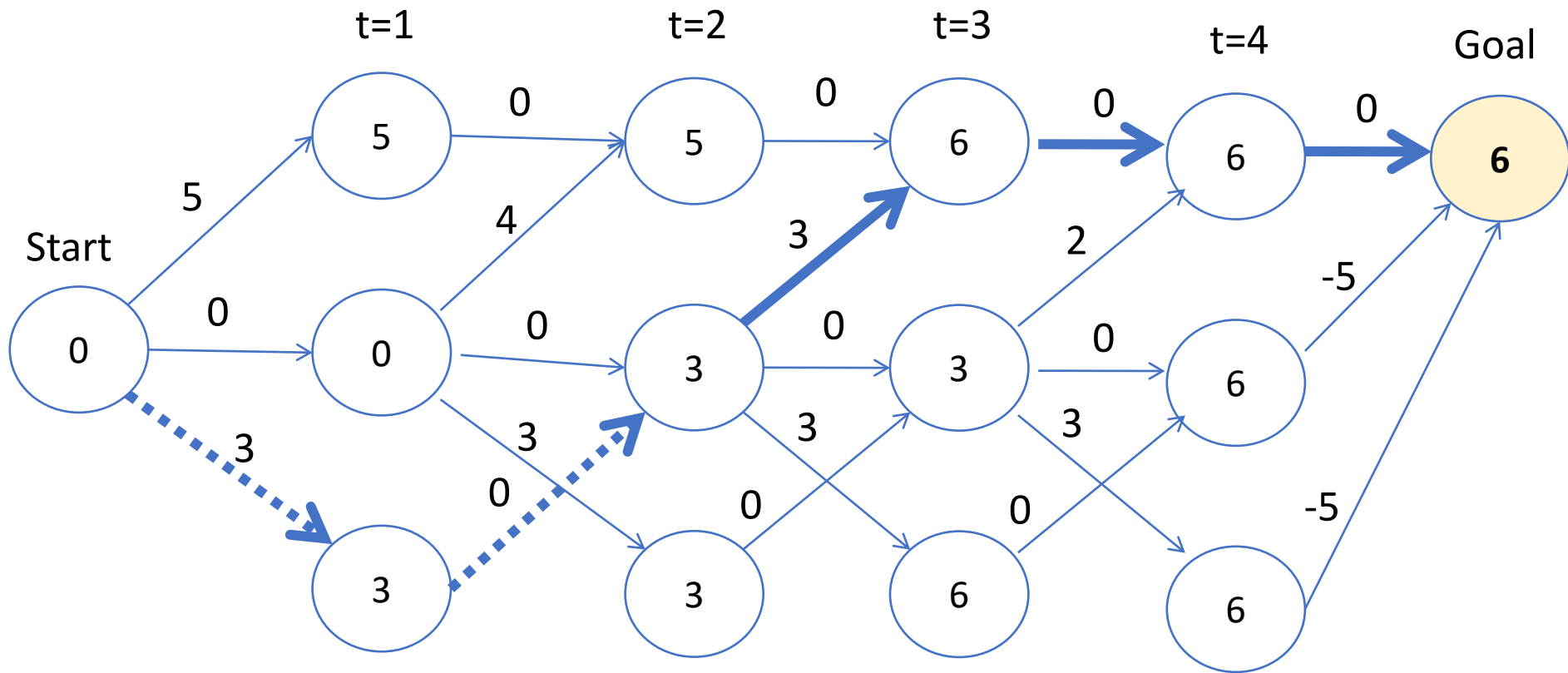
# Step 3



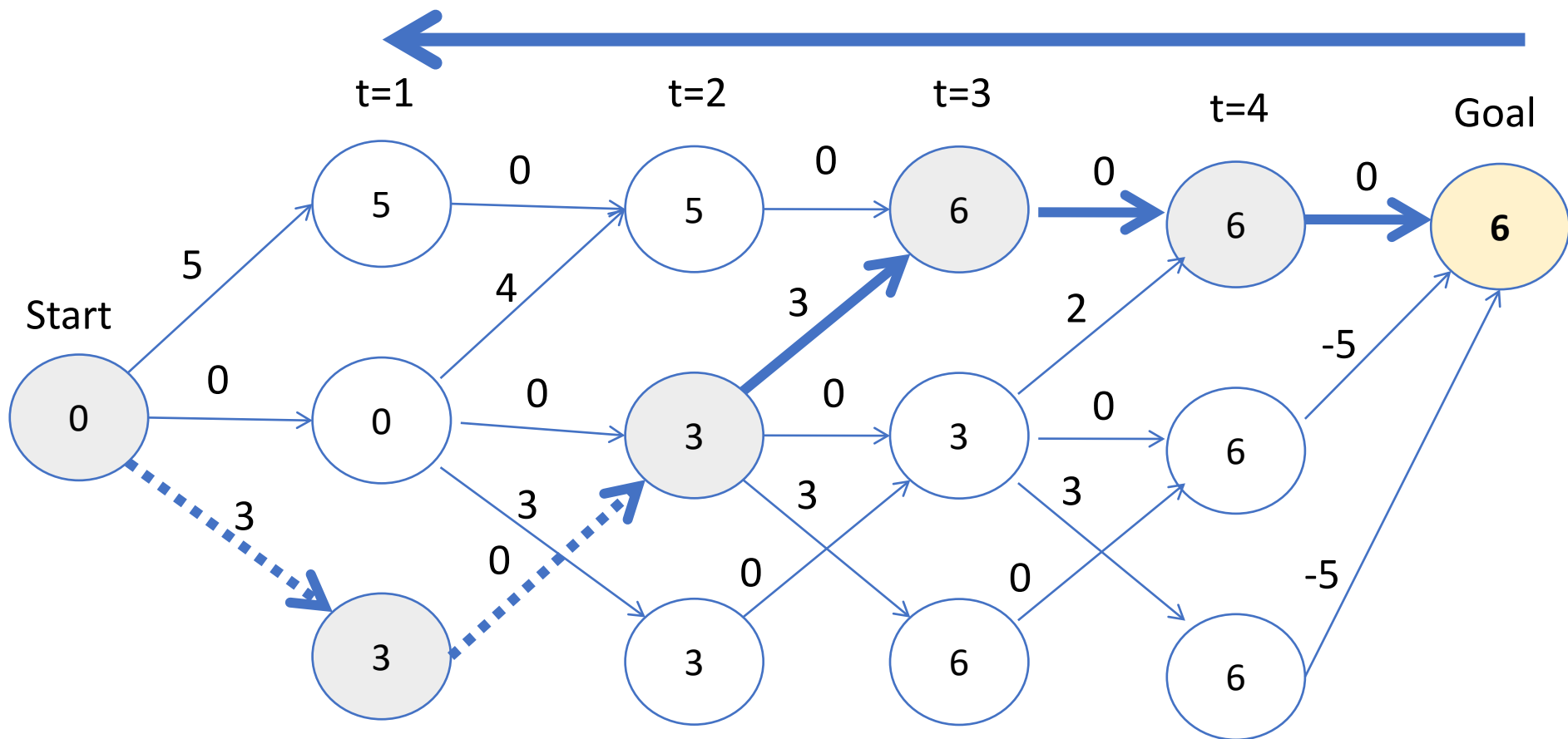
# Step 4



# Step 5

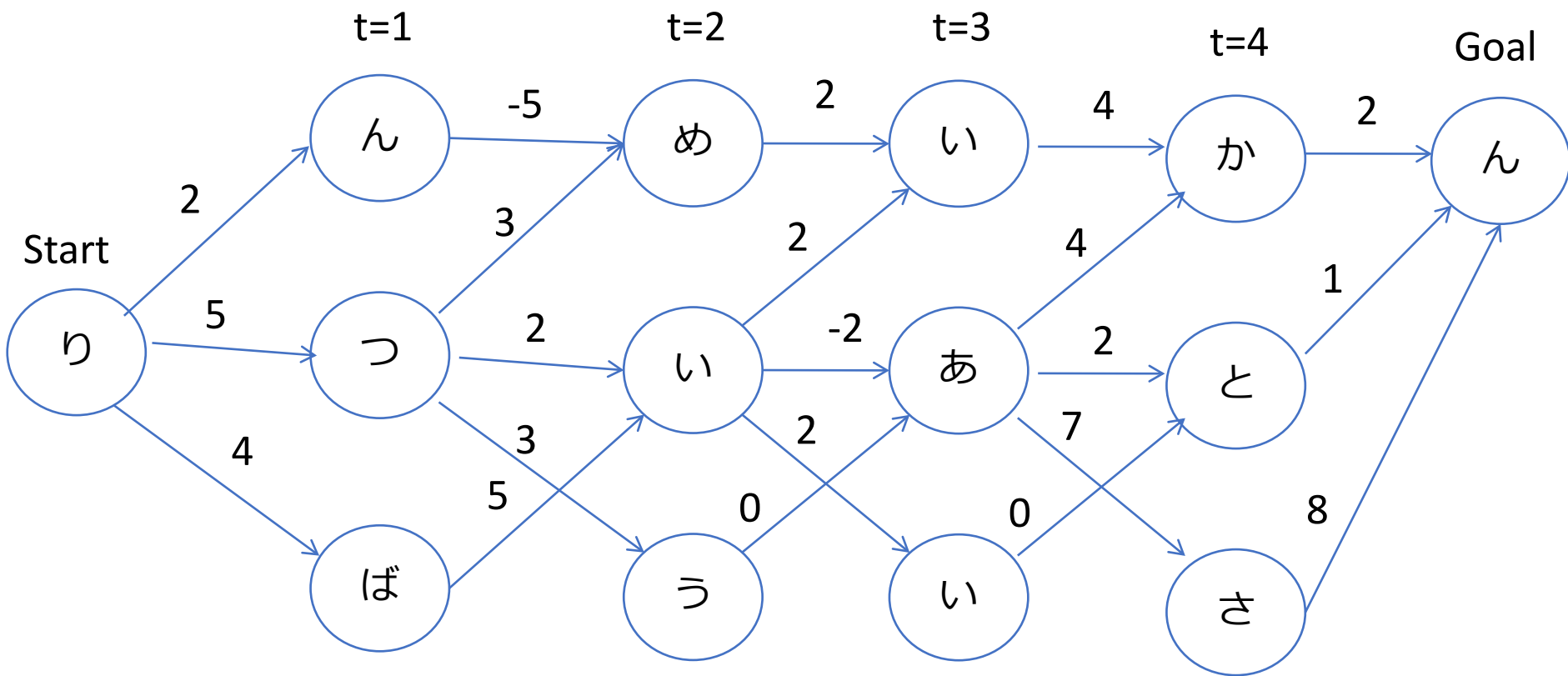


# 最適経路



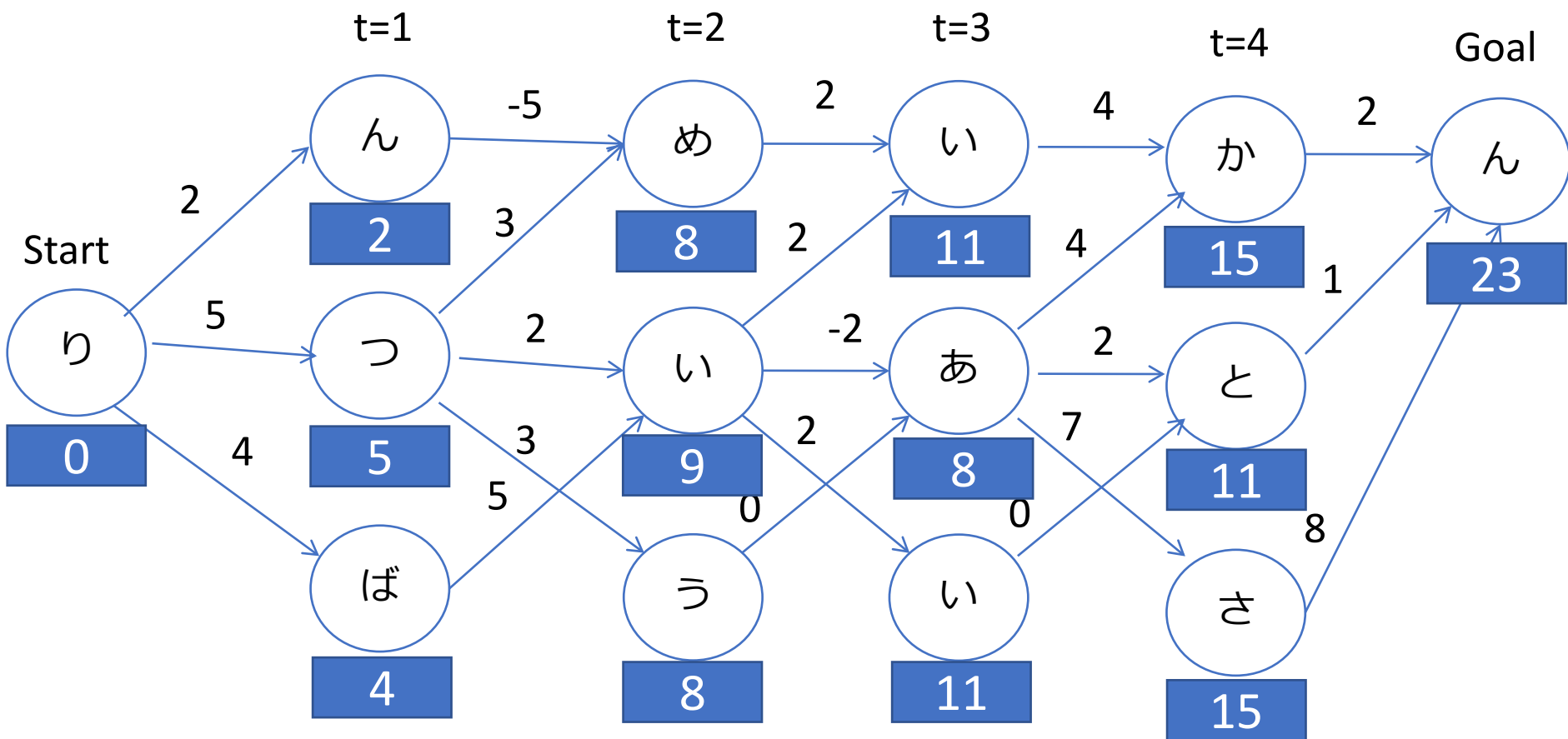
# 演習問題5-1

## 文字の接続コストによる単語生成



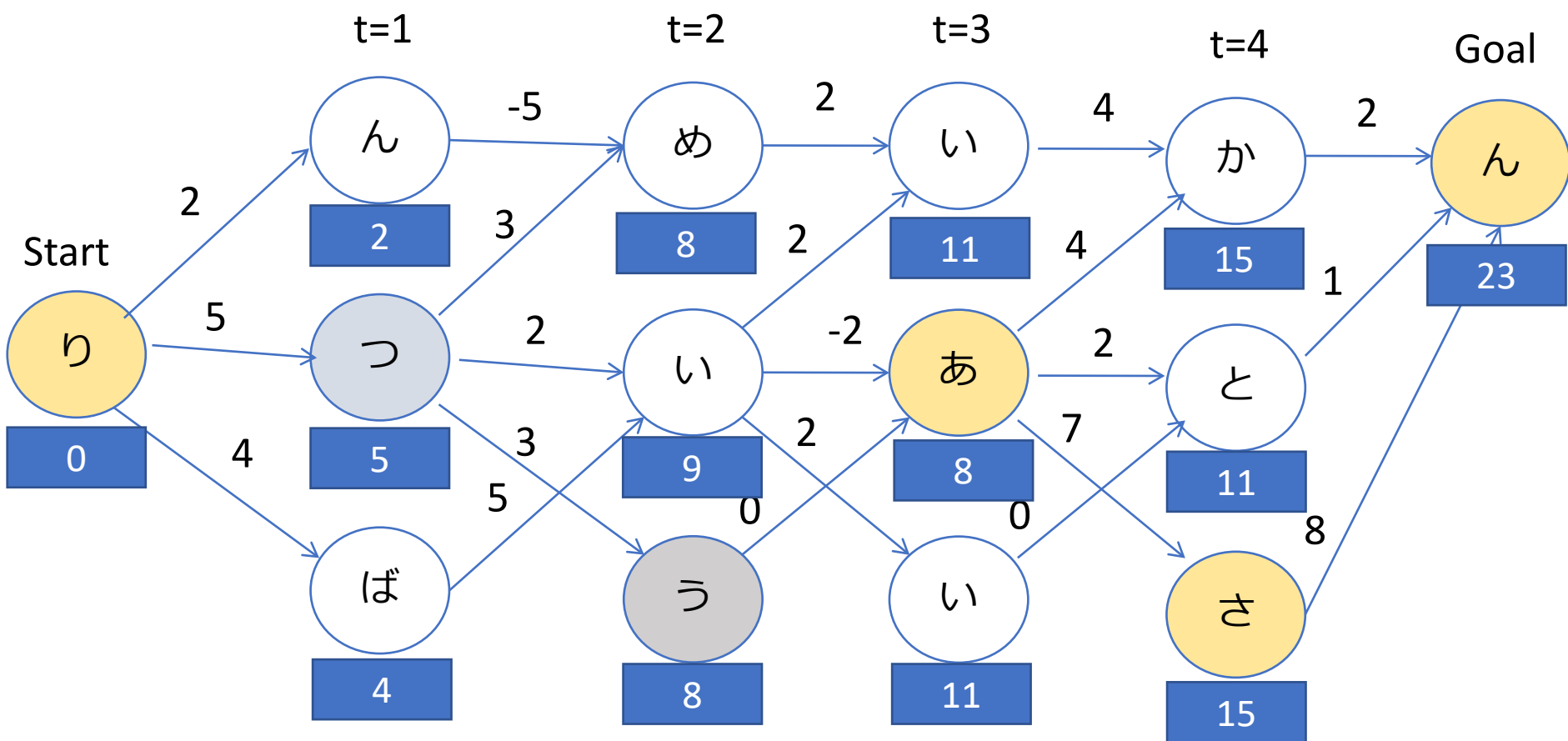
文字のつながりの利得がリンク上に表示してある。  
最も得点の高くなる経路を見つけよ。

# 演習問題5-1 解答

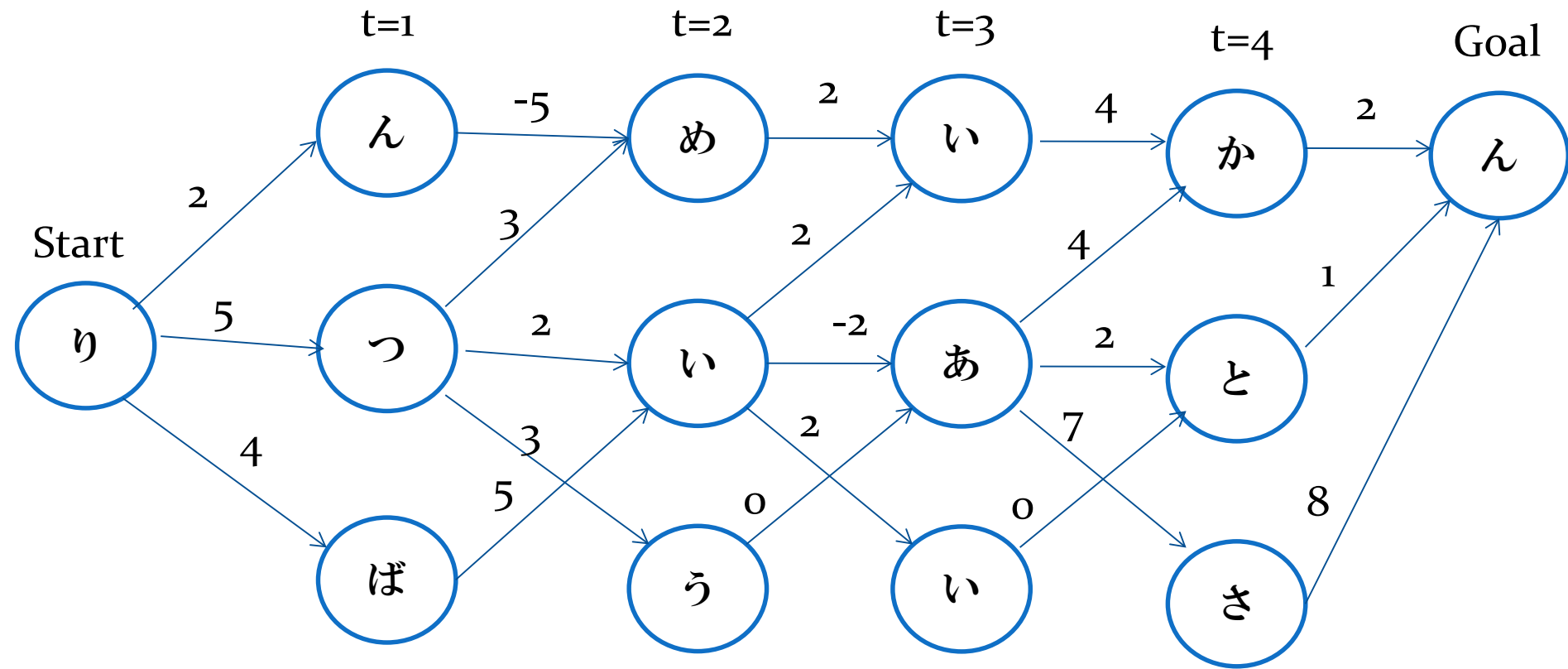




# 演習問題5-1 解答

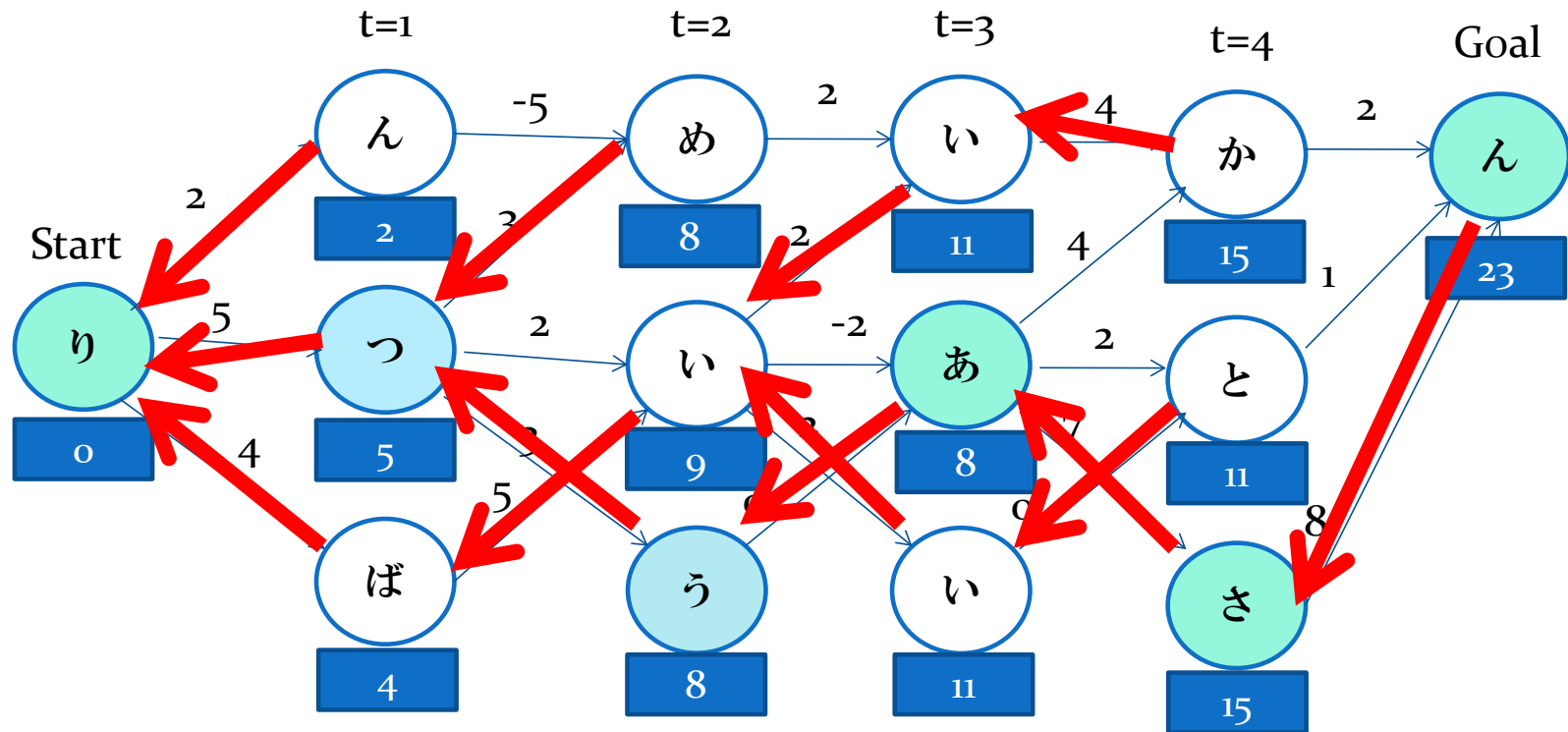


# 演習問題5-2 アルゴリズムの確認



□動的計画法のアルゴリズムと演習問題5-1の結果を比較し、最終的なメモリに格納された各 $s_t$ に対する $F_t(s_t)$ と $\hat{s}_{t-1}(s_t)$  はどのようなになっているか、示せ。

# 演習問題5-2 アルゴリズムの確認 解答



- 動的計画法のアルゴリズムと演習問題5-1の結果を比較し，最終的なメモリに格納された各 $s_t$ に対する $F_t(s_t)$ と $\hat{s}_{t-1}(s_t)$  はどのようにになっているか，示せ．
- $F_t(s_t)$  . . . 各状態にメモ化した値そのもの．
- $\hat{s}_{t-1}(s_t)$  . . . 各状態から示した赤矢印の指し示す先

# Contents

□5.1 多段決定問題

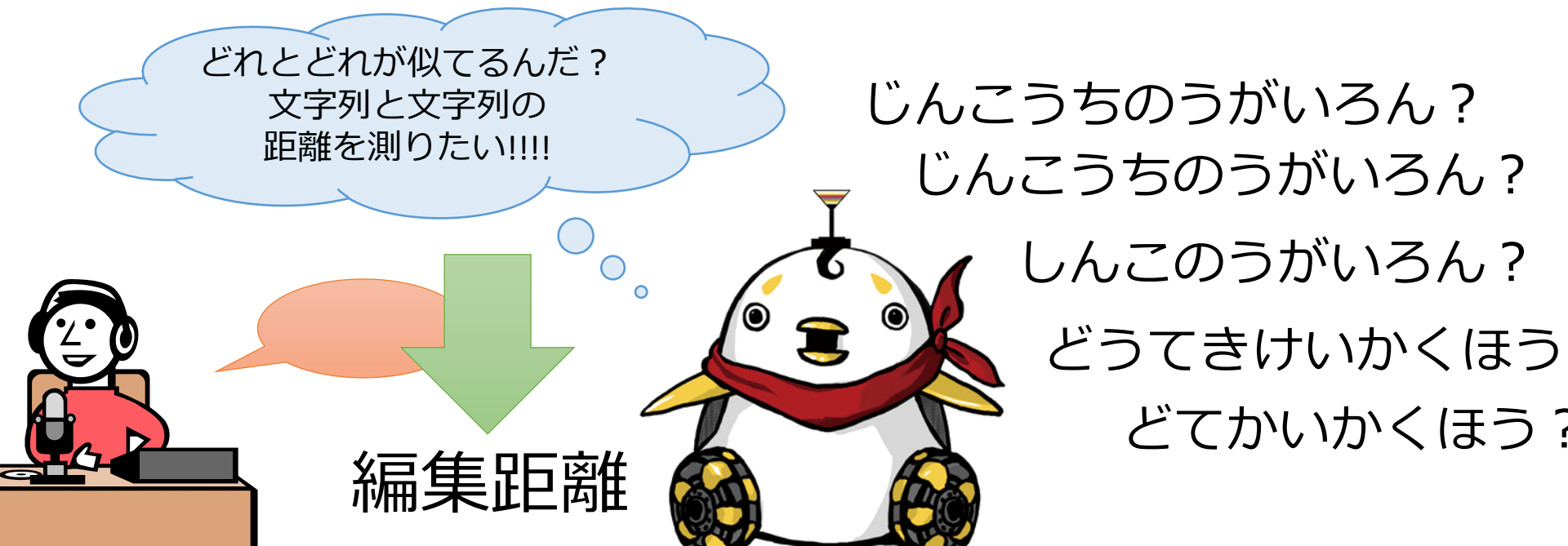
□5.2 動的計画法

□5.3 ホイールダック2号「宝箱を拾ってゴール」

□5.4 例：編集距離の計算

## 5.4.1 編集距離の計算

- 動的計画法は確定的システムにおける多段階決定の一般的な解法である.
- ロボットの移動のみならず, 様々な多段階決定問題に帰着されうる問題に対して利用することが出来る.



# 例：編集距離の計算

- 編集距離(edit distance) は文字列と文字列の尺度
- ハミング距離では文字の置き換えには対応できるが、文字の挿入や削除に対応できない.
- 編集距離の計算にはストリングマッチングが必要.

表 5.1 ハミング距離と編集距離の比較

文字列 1	文字列 2	ハミング距離	編集距離	説明
abcd	abcd	0	0	文字が変化していないときは距離 0 になる.
abcd	abed	1	1	1 文字の置換は距離 1 になる. “c” から “e” へ.
abcd	acd	3	1	ハミング距離では 3 文字の置換とされるが、編集距離では “b” 1 文字の削除と捉えられる.
abcd	axbcd	4	1	ハミング距離では 4 文字の置換とされるが、編集距離では “x” 1 文字の挿入と捉えられる.

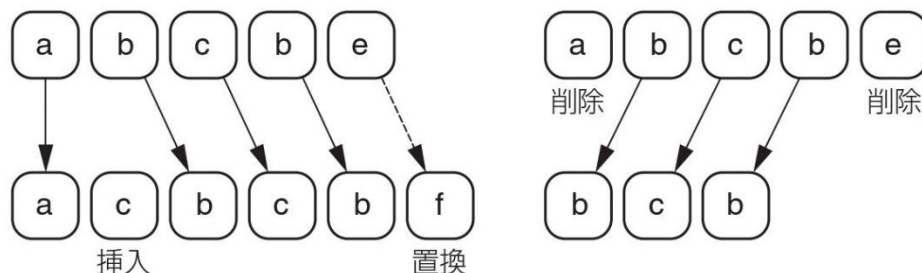


図 5.11 ストリングマッチング

# 編集距離を計算時の各移動コスト

- ① 元の文字列が“ab”であって編集後の文字列が“a”であった後に、“e”が**挿入**された場合.
- ② 元の文字列が“a”であって編集後の文字列が“ae”であった後に、元の文字列に“b”が追加されたが、これが**削除**された場合.
- ③ 元の文字列が“a”であって編集後の文字列が“a”であった後に、元の文字列に“b”が追加されたが、これが“e”に**置換**されて出力された場合.

編集後の文字列

	\$	a	e
\$	0	1	2
a	1	0	1
b	2	1	

元の文字列

③ 置換：1  
④ 一致：0

② 削除：1

① 挿入：1

元の文字列が入力されて、編集されて編集後の文字列が出力されるプロセスが、左のような表の移動経路として表現される。

図 5.13 編集距離計算時の各移動コスト

# 編集距離計算のための グラフとメモ化のための表

元の文字列

	編集後の文字列				
	\$	a	e	b	c
\$	0	1	2	3	4
a	1				
b	2				
c	3				
d	4				Goal

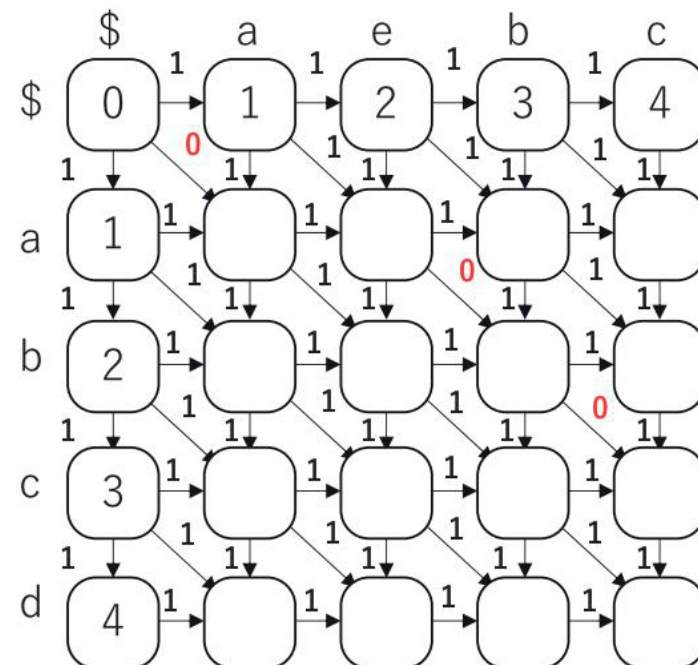


図 5.12

編集距離を計算するための表



# 編集距離の計算結果

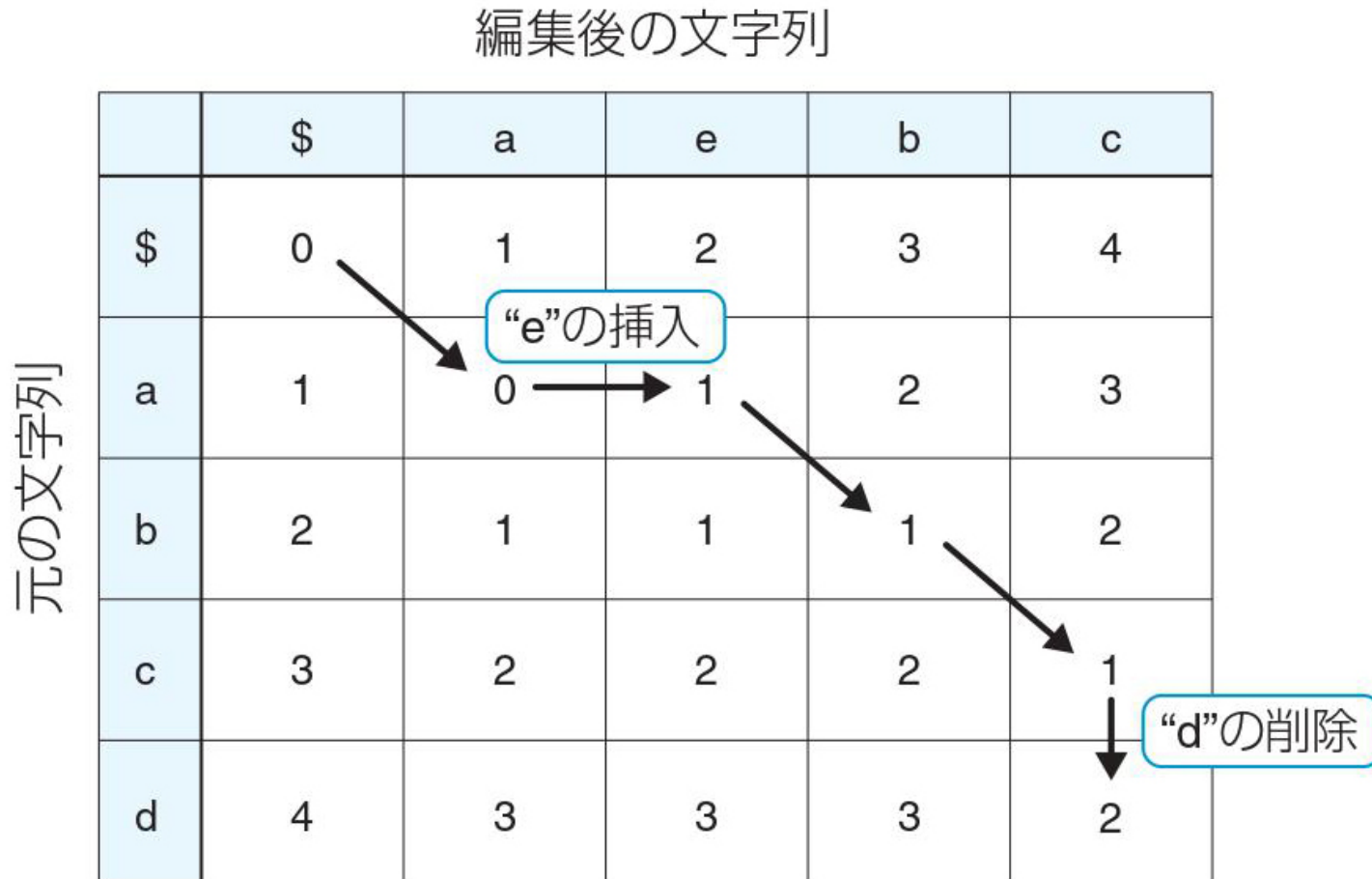


図 5.14

編集距離の計算結果

# 演習問題 5-3

- 「りつめいかん」と「はつめいか」の編集距離を動的計画法を用いて求めよ.

	\$	は	つ	め	い	か
\$	0	1	2	3	4	5
り	1					
つ	2					
め	3					
い	4					
か	5					
ん	6					Goal

# 演習問題 5-3 解答

## • 答え 2

- 意味的な違いは全く考慮されないことに注意しよう

	\$	は	つ	め	い	か
\$	0	1	2	3	4	5
り	1	1	2	3	4	5
つ	2	2	1	2	3	4
め	3	3	2	1	2	3
い	4	4	3	2	1	2
か	5	5	4	3	2	1
ん	6	6	5	4	3	2

## 演習5-4 編集距離と文書検索

1. 長さ $n$ の文字列と長さ $m$ の文字列の編集距離を計算するのに必要な計算量を見積もれ.
2.  $L$ 文字で書かれた文書がある. この文書には「ホイールダック」を「ホイルダック」とするなど, 文字の書き間違いが含まれているとする. この場合, 部分文字列の検索では正しい検索結果を得ることが出来ない. そこで与えられたクエリ文字列について編集距離の小さな文字列を文章中から探してくるプログラムをつくりたい. 前から順番に長さ $M$ の部分文字列をとってきては長さ $K$ のクエリ (検索) 文字列との編集距離を計算していく. このアルゴリズムの計算量はどれほどになるか. (ただし $L \gg M$ とする)

※ともに $O$ (オーダー)表記で答えよ

## 演習5-4 編集距離と文書検索 解答

1. 長さ $n$ の文字列と長さ $m$ の文字列の編集距離を計算するのに必要な計算量.

□  $O(mn)$  :  $m \times n$  個のマスを埋めないといけないから.

2.  $L$ 文字で書かれた文書がある. 単純に, 前から順番に長さ $M$ の部分文字列をとってきては長さ $K$ のクエリ (検索) 文字列との編集距離を計算していく. このアルゴリズムの計算量を見積もれ.

□  $O(LMK)$  :  $O(MK)$ の計算を $L-M+1$ 回やるから

# 第5章のまとめ

- 確定システムにおける多段決定問題の定式化を行った.
- 状態空間の時間方向へのグラフ展開について学んだ.
- 動的計画法のアルゴリズムについて学んだ.
- 動的計画法の応用として, スtringマッチングと編集距離の計算方法について学んだ.