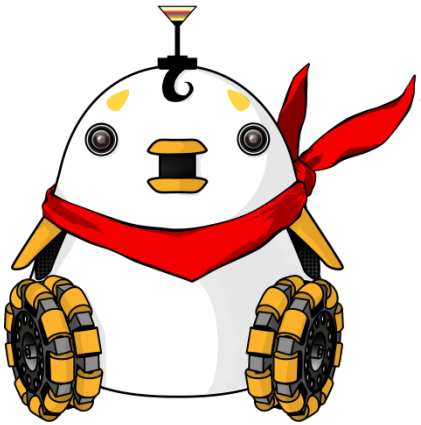
人工知能

第5回 多段決定（1）
動的計画法

立命館大学 情報理工学部
谷口彰



STORY　動的計画法

- 常に**状態や状態間のコスト(cost)が変わらず，ゴール(goal)が一つであればA*アルゴリズム(algorithm)**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**とるべき行動はそのまま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**しないといけないし，ある時刻で通りかかる**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**するかもしれない．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，**すべての行動パターン(pattern)を試すのはとても大変**だ．さてどうするべきか?

仮定　動的計画法

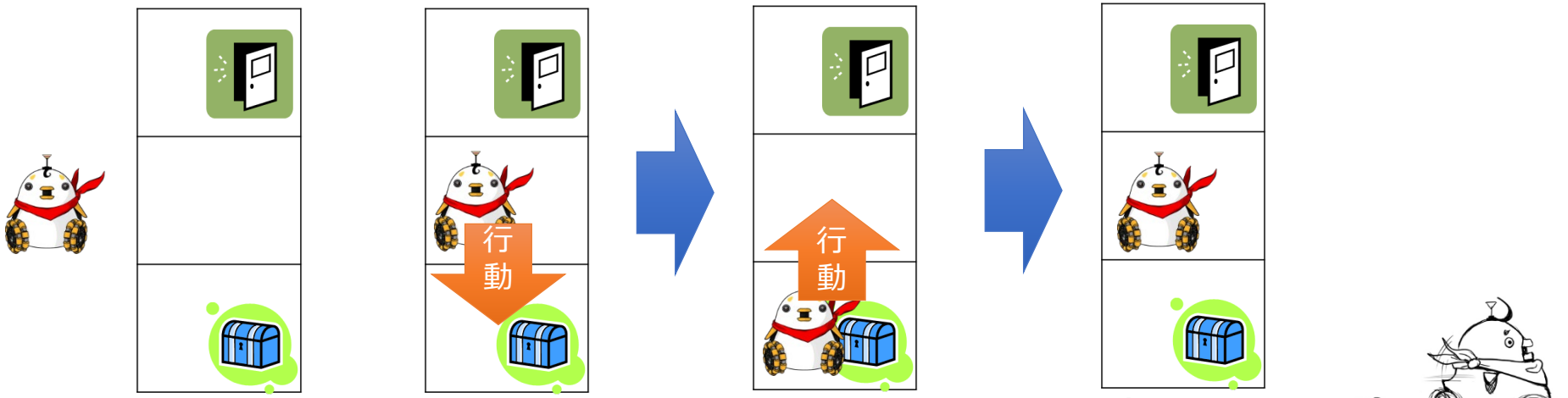
- ホイールダック２号は
  - 迷路の完全な地図を持っているものとする.
  - 迷路の中で自分がどこにいるか認識できるものとする.
  - 連続的な迷路の空間から適切な離散状態空間を構成できるものとする.
  - 各時刻で各状態間の移動にかかるコストや利得を知っているものとする.
  - 物理的につながっている場所・状態には意図すれば確定的に移動することができるものとする.

Contents

5.1.1 はじめに

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

- 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までに得られる利得の和の最大化**（もしくは，かかる費用の和を最小化），を行う計画問題を**多段決定問題**という．



状態空間（3マス迷路）　時点t　時点$t+1$　時点$t+2$

5.1.2 グラフを時間方向に展開する

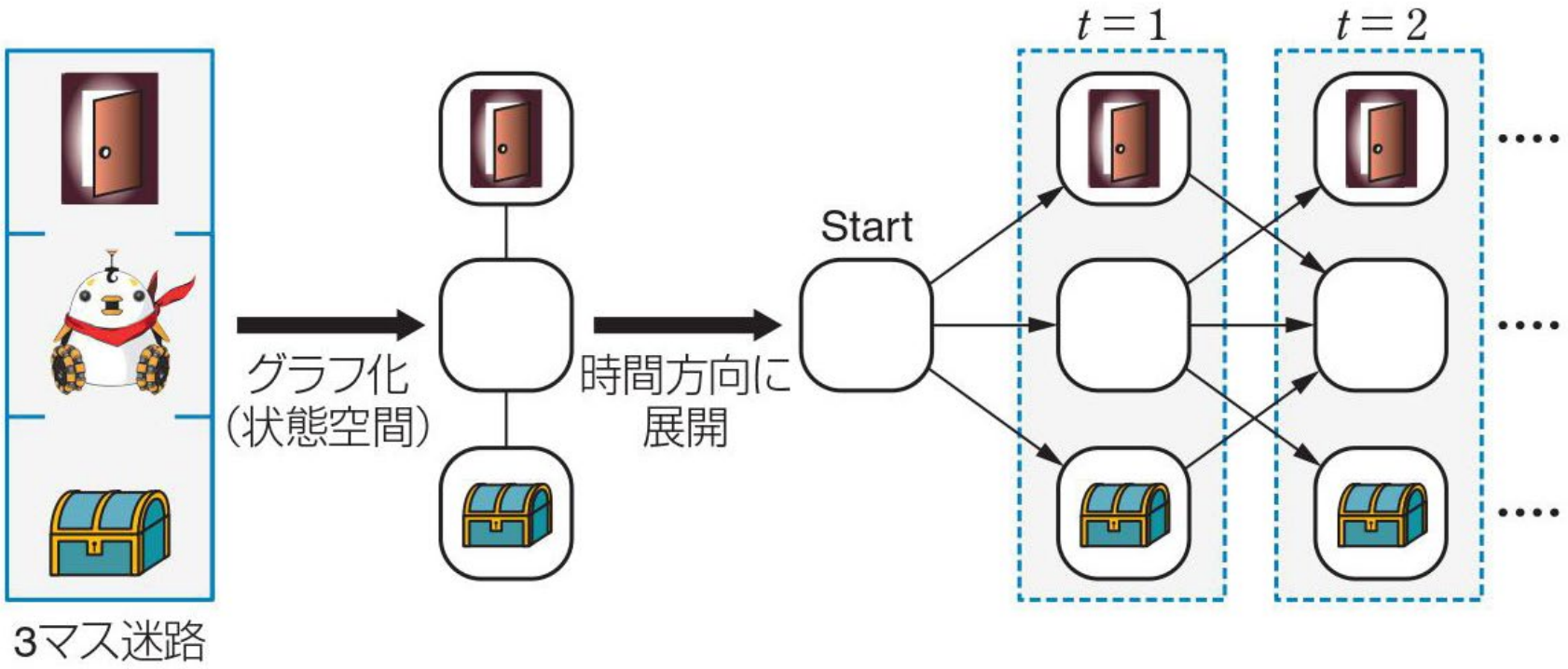
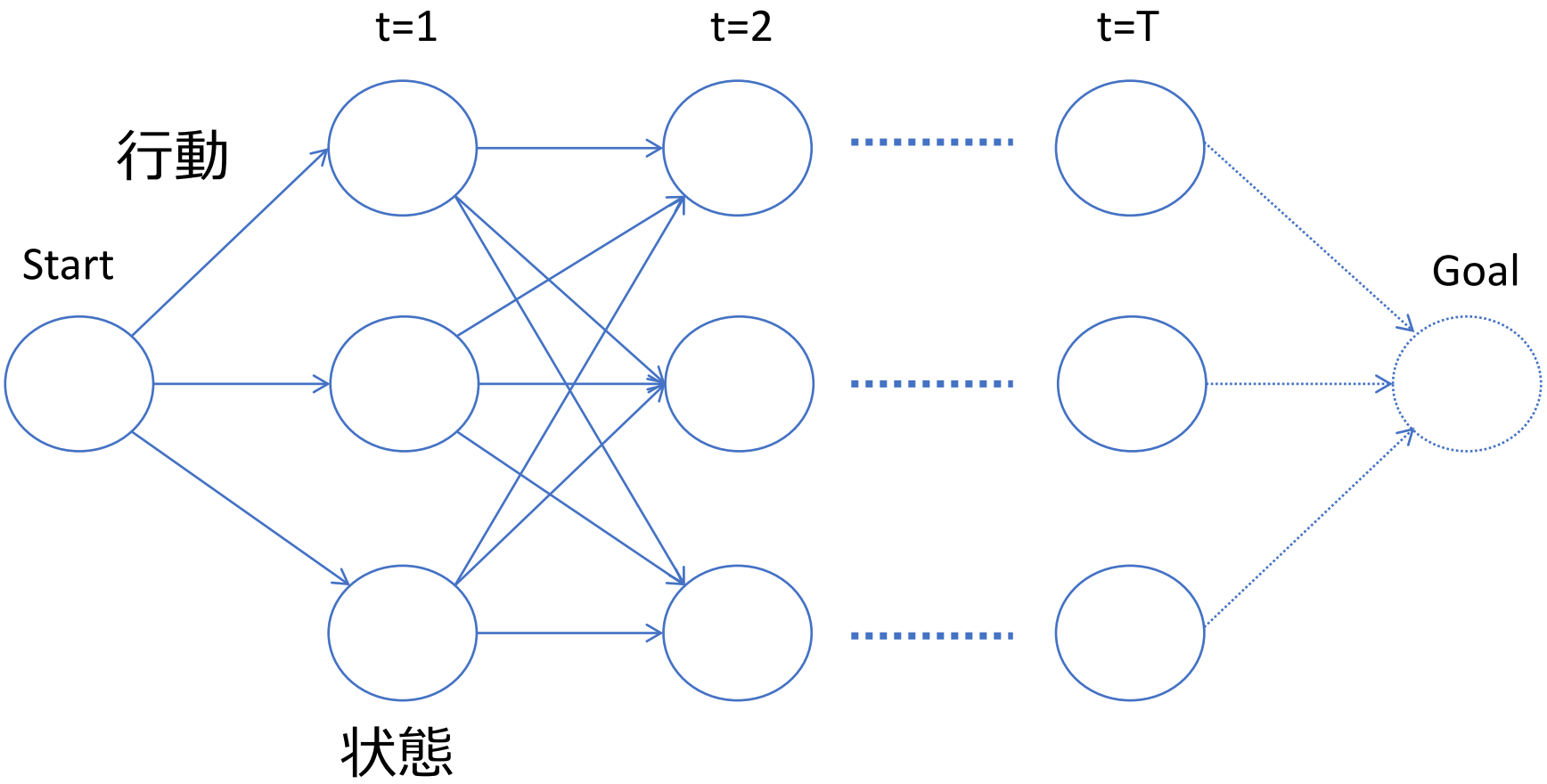
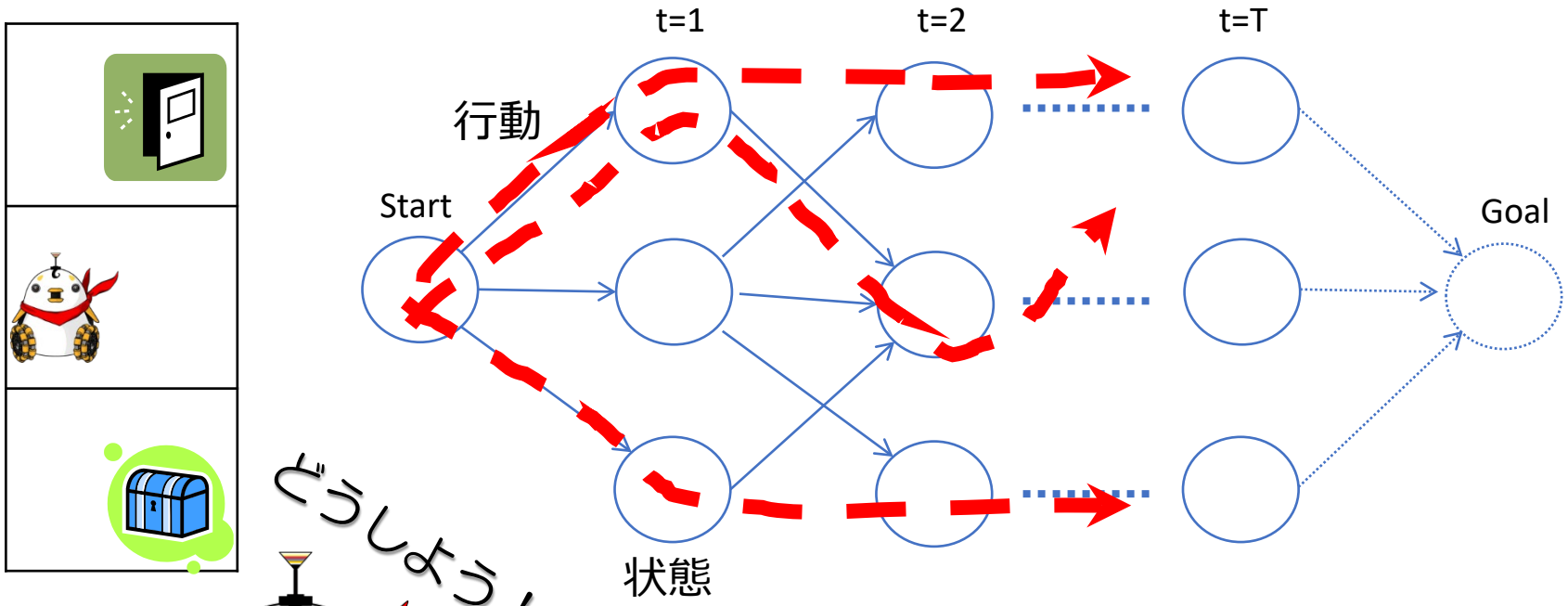


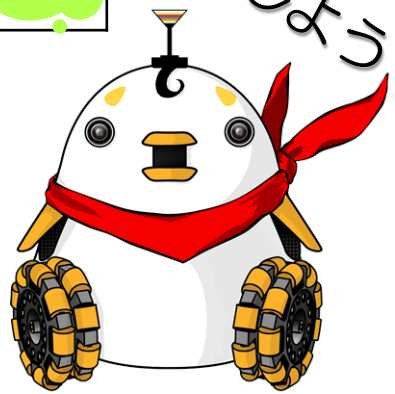
図 5.2 状態空間の時間方向への展開とグラフの作成

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

5.2.1 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！

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

計算量は $O(N^T)$

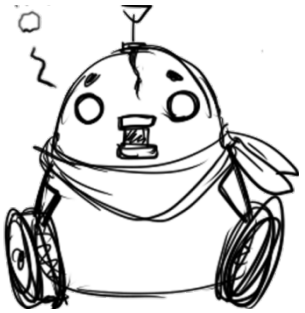
- **動的計画法**は多段決定問題において，各評価値が状態の対ごとの二変数関数の和で書けることを利用して**計算量を縮減**するアルゴリズムである．

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

計算量を $O(N^2 T)$ まで縮減

指数オーダー(Exponential order)⇒ 2次オーダー(Secondary order)の影響

- $N = 100$状態，$T = 34$ステップの場合
  - $O(N^T)$
    - 1無量大数回＝10^{68}
    - ⇒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. $$F_t(s_t) = \max_{s_{t-1}}[F_{t-1}(s_{t-1}) + h_t(s_{t-1}, s_t)] \qquad (5.4)$$

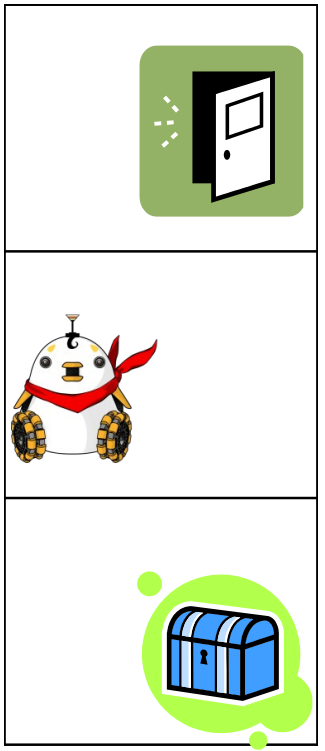
   および，その最大値を与える s_{t-1} を $\hat{s}_{t-1}(s_t)$ としてメモリに保持する．
3. end for
4. $F_T(s_T)$ を最大にする s_T の値 s_T^* を探索し，その最大値を $J^* \leftarrow F_T(s_T^*)$ とする．
5. for $t = T - 1$ to 1 do
6. $s_t^* = \hat{s}_t(s_{t+1}^*)$ を計算する．
7. end for
8. return 経路 $(s_1^*, s_2^*, \ldots, s_T^*)$ および J^* を返す．

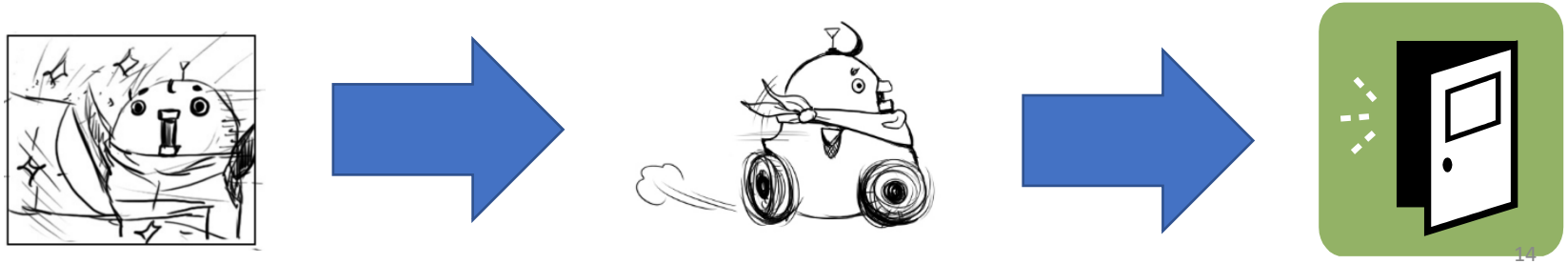
メモ化

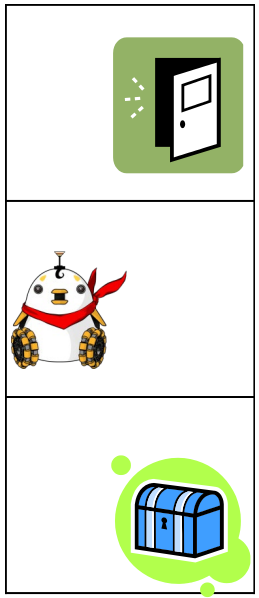
Contents

例:「宝箱を拾ってゴール」

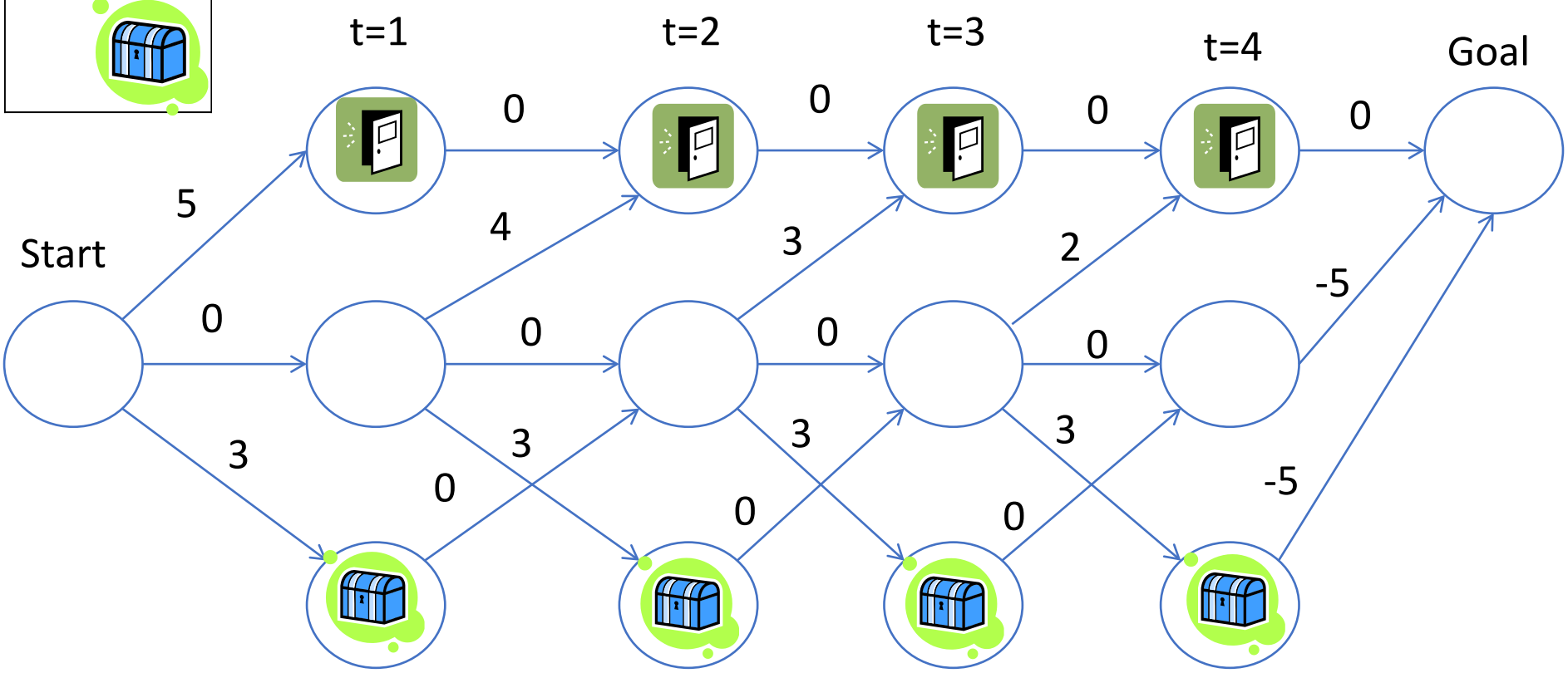
- $t = 4$までにゴールできなかった場合，ペナルティとして-5の利得（5のコスト）を与えられる.
- 宝箱をとることは何度でもでき，この時には3の利得を得る.
- また，早くゴールしたほうが利得は高く，ゴールが一時刻遅れるたびにゴール時の利得は減っていく.
- 宝箱の場所にはとどまることはできない．また，一度ゴールすると，ゴールから再度出てくることはできない.





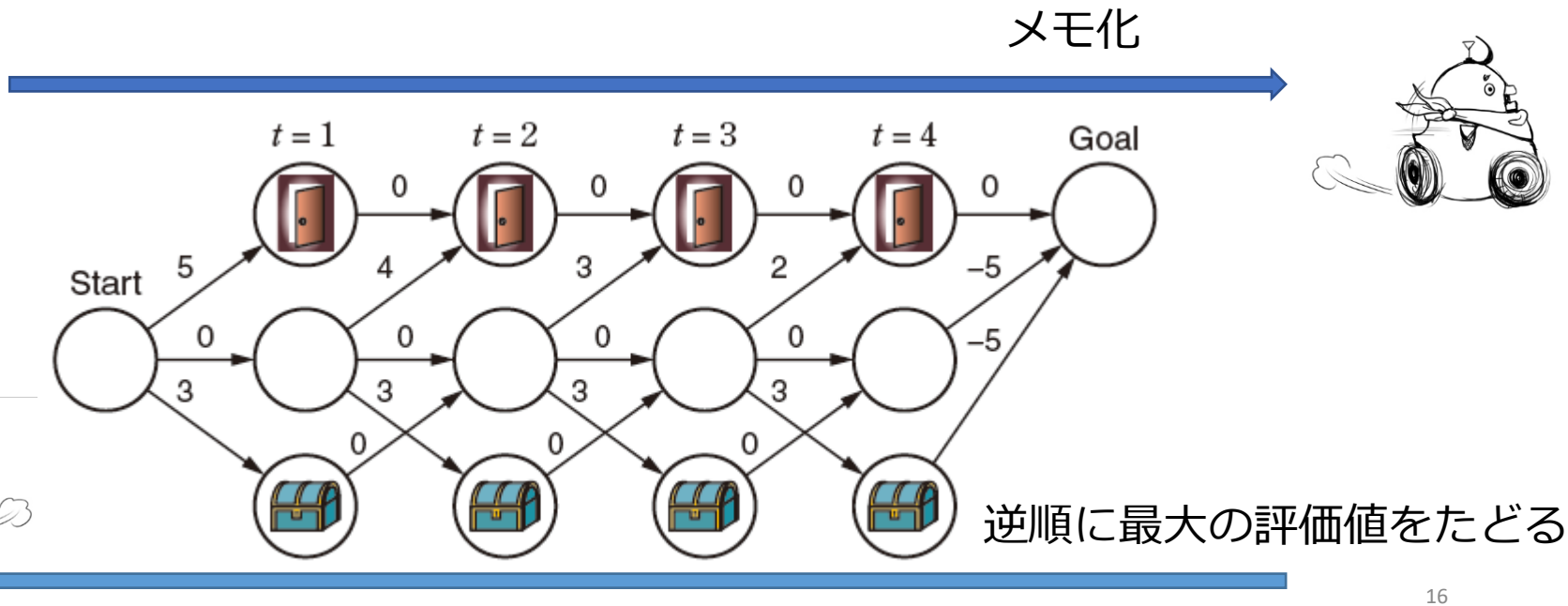


例:「宝箱を拾ってゴール」

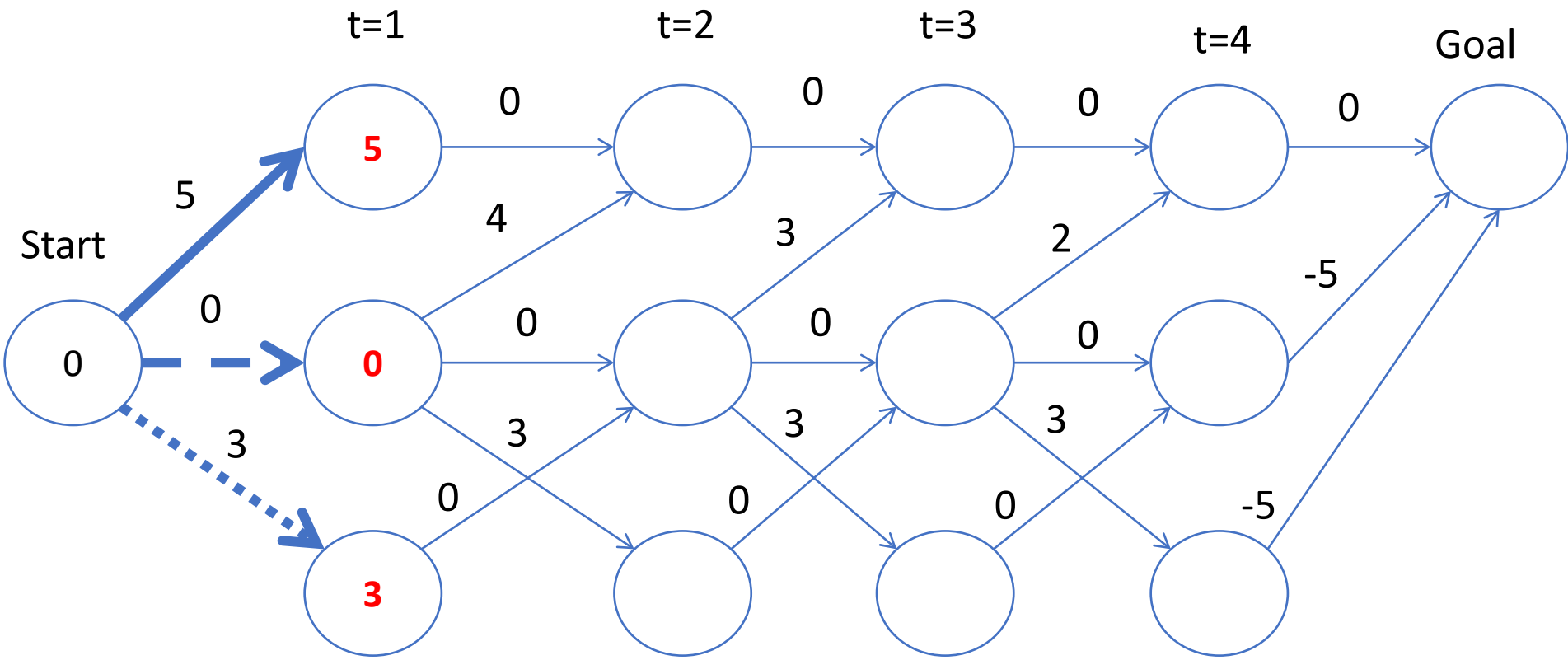


（ポイント） 動的計画法のアルゴリズム

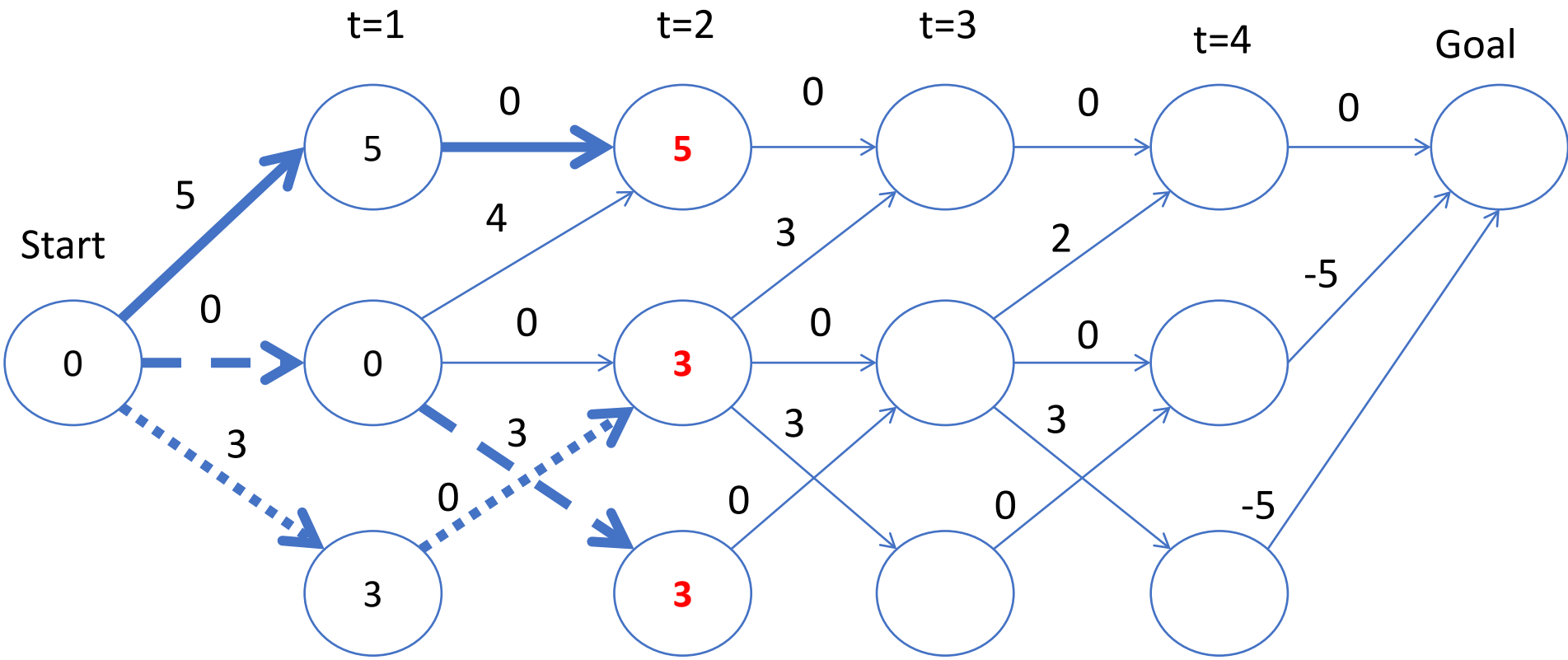
- まず，左から**順に各状態までの最適パスを計算**し，そのときの**評価値を状態に記述**していく．これを**メモ化(Memoization)** という．
- メモ化を繰り返して最終時刻に至った段階で，**最大の評価値を逆順にたどる**ことで最適な経路が一通りに決まる．



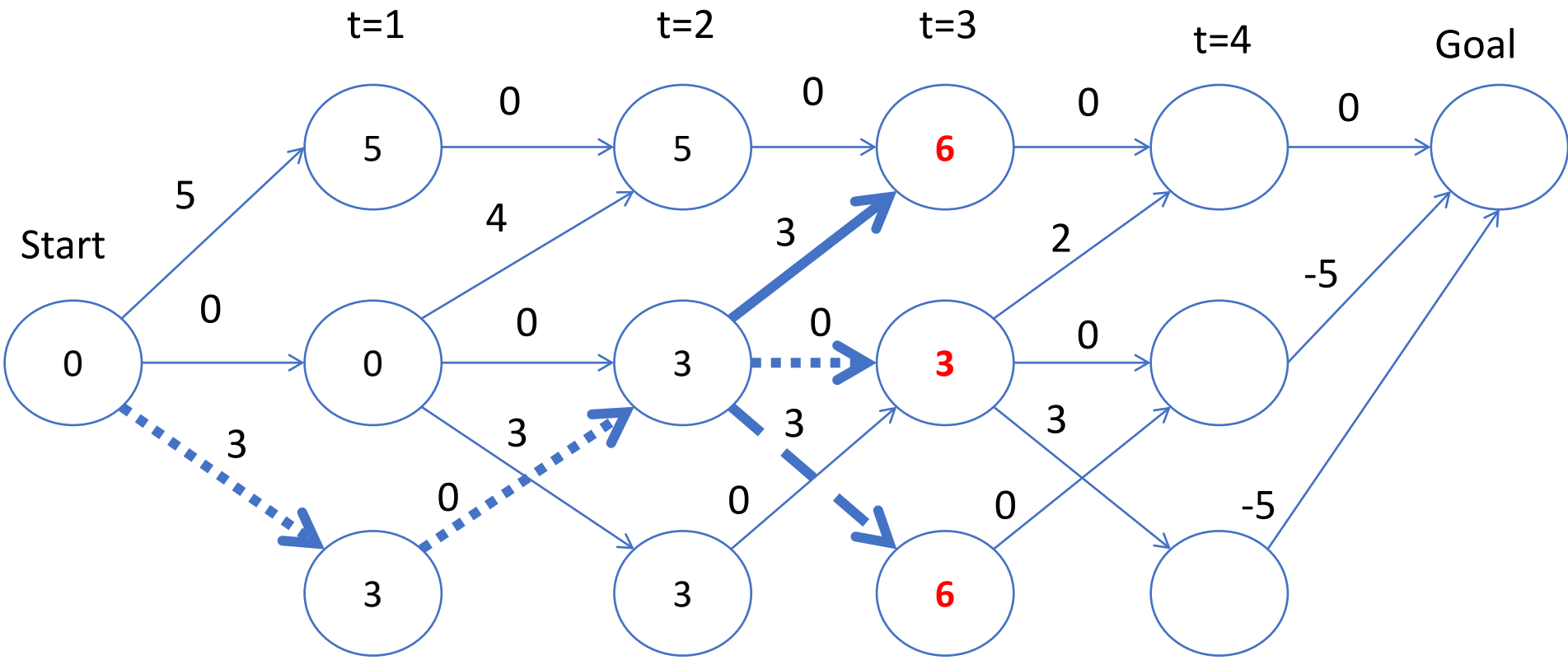
Step 1



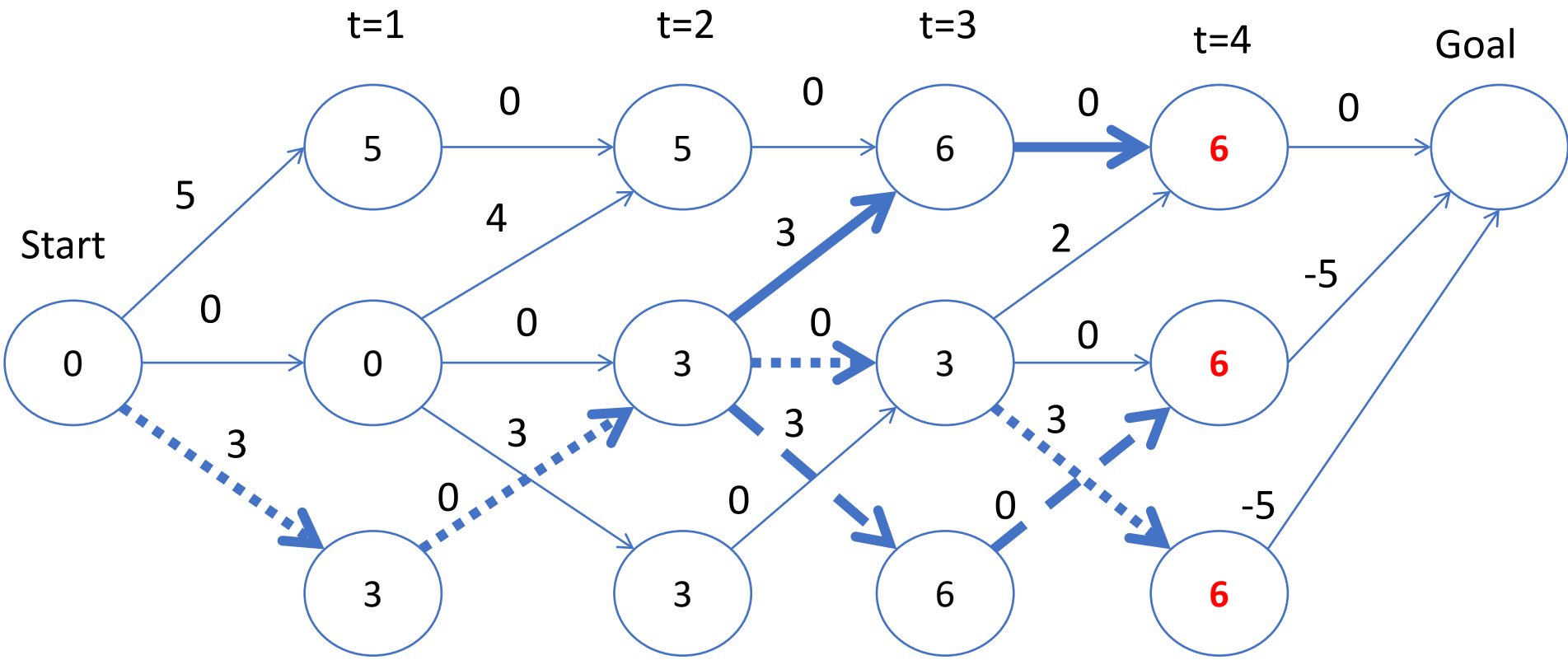
Step 2



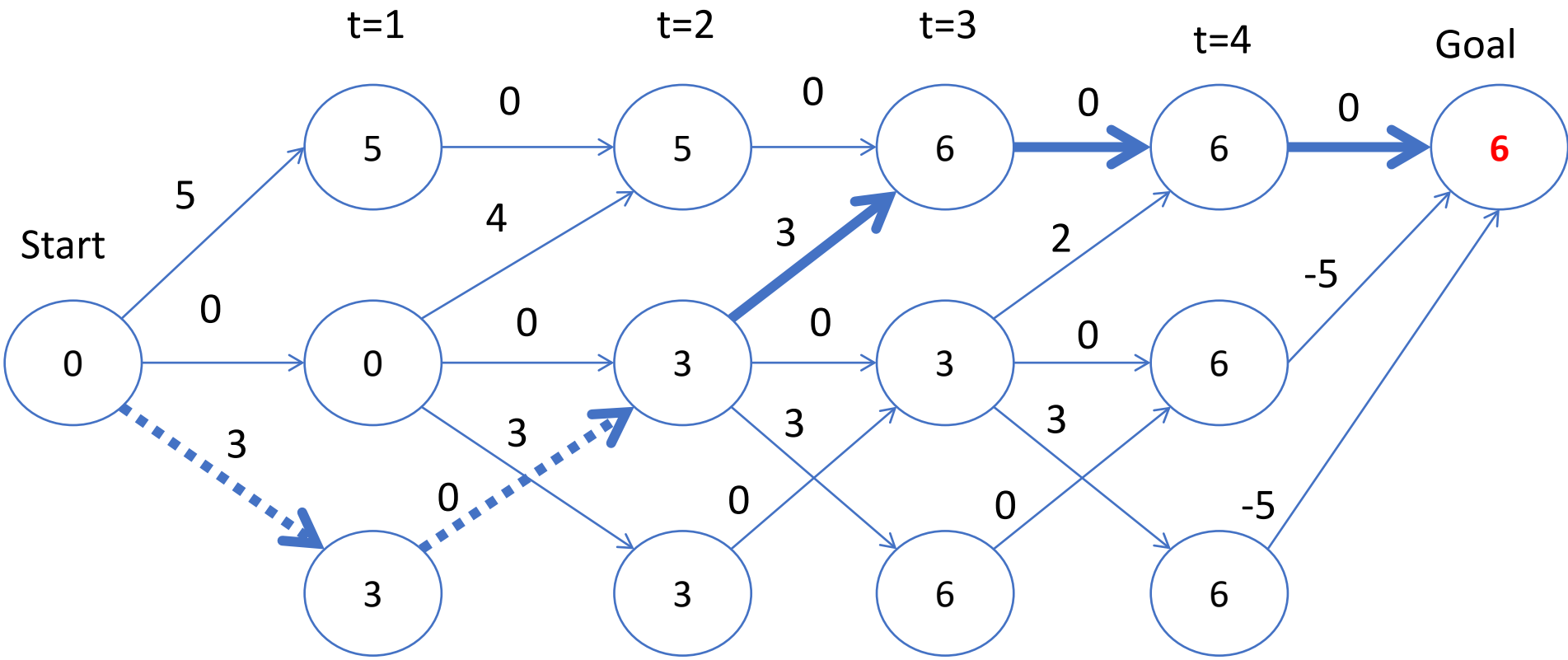
Step 3



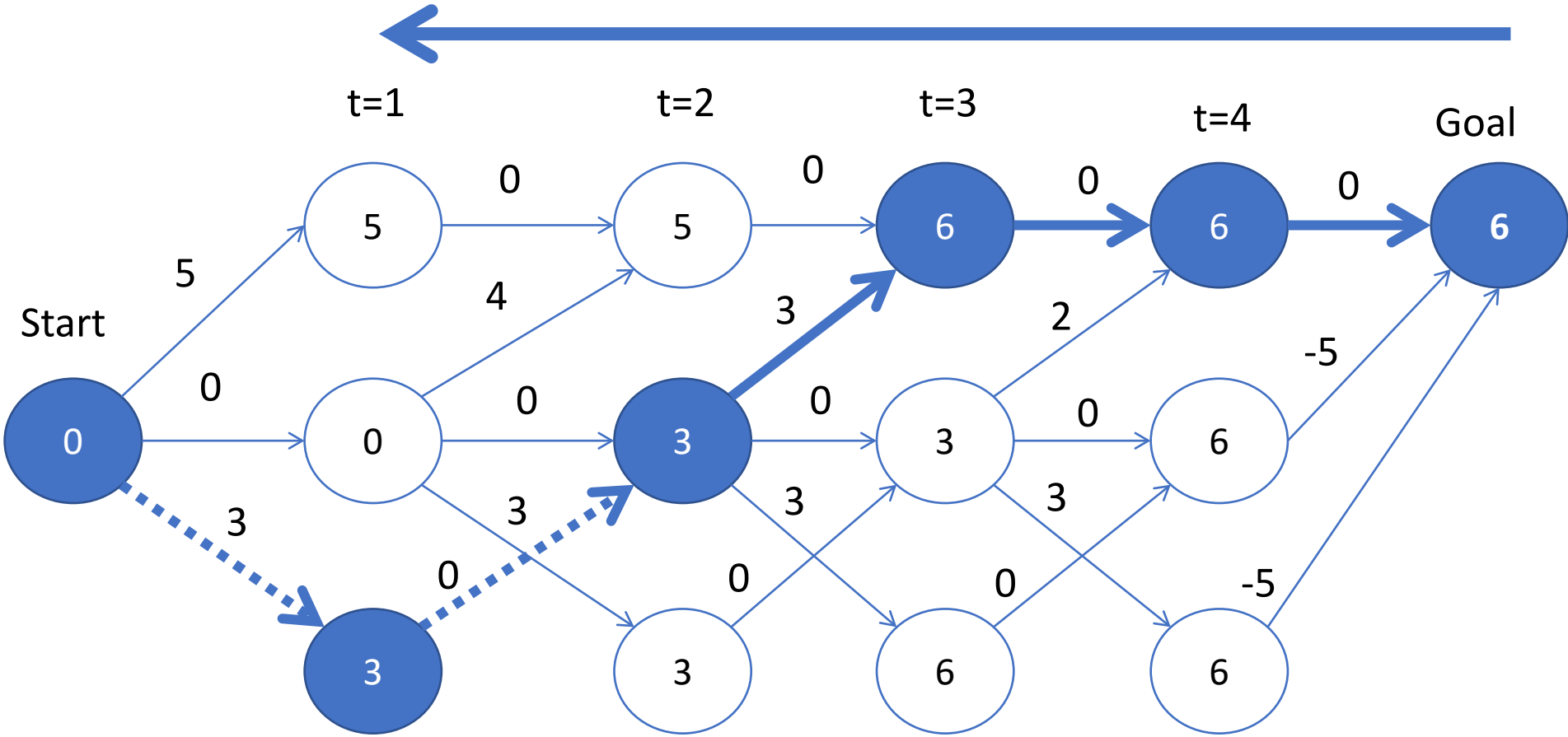
Step 4


t=1
t=2
t=3
t=4
Goal
Start
0
5
0
5
0
6
0
6
0
5
4
0
0
3
3
0
2
0
6
-5
0
3
3
0
3
0
3
0
6
-5
3
3
6
6

Step 5


t=1
t=2
t=3
t=4
Goal
Start
0
5
0
5
0
6
0
6
0
6
5
4
3
2
-5
0
0
0
0
0
3
0
3
0
6
3
3
3
3
0
0
0
-5
3
3
6
6

最適経路


t=1
t=2
t=3
t=4
Goal
Start
0
5
0
5
0
6
0
6
0
6
5
4
3
2
-5
0
0
0
0
0
3
0
3
0
6
3
3
3
3
0
0
0
-5
3
3
6
6

5.2.2 動的計画法のアルゴリズム

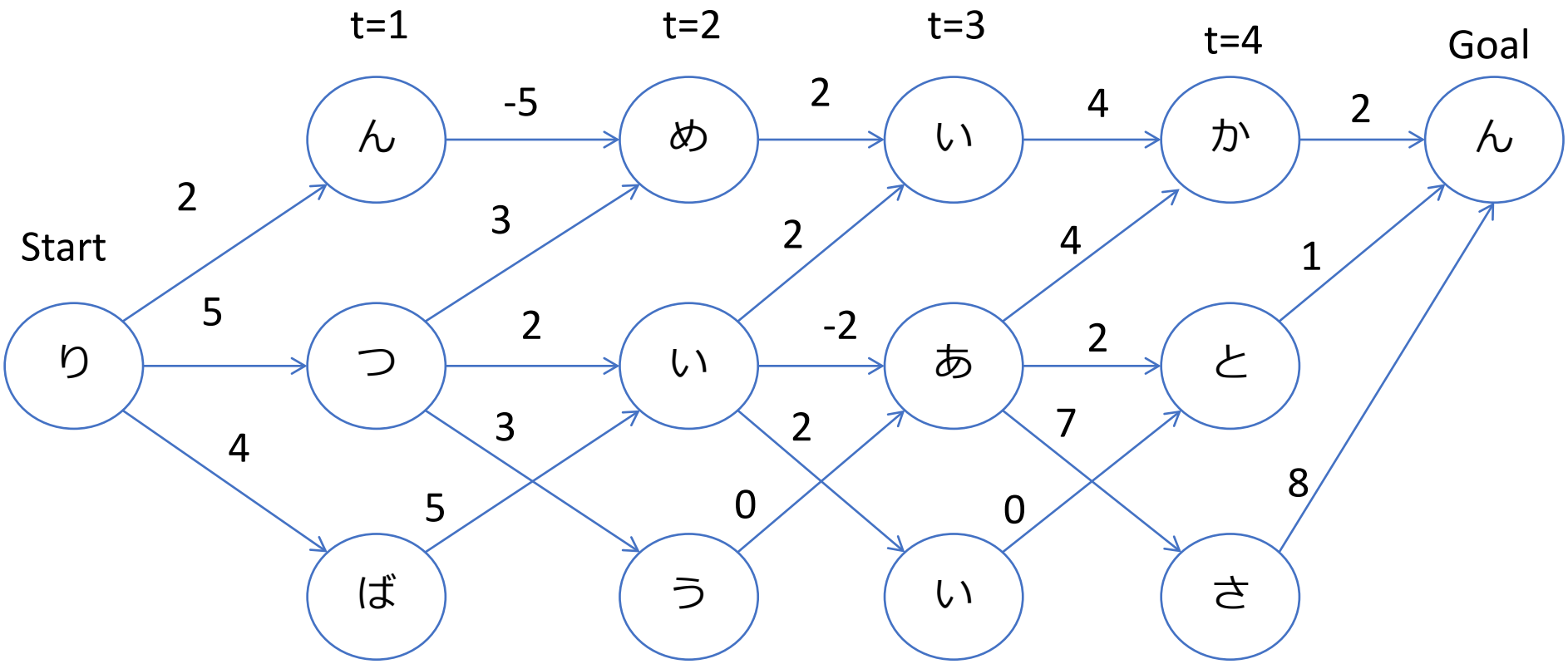
Algorithm 5.1 動的計画法

1. for $t = 1$ to T do
2. $$F_t(s_t) = \max_{s_{t-1}}[F_{t-1}(s_{t-1}) + h_t(s_{t-1}, s_t)] \qquad (5.4)$$

   および，その最大値を与える s_t を $\hat{s}_{t-1}(s_t)$ としてメモリに保持する．
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 $(s_1^*, s_2^*, \ldots, s_T^*)$ および J^* を返す．

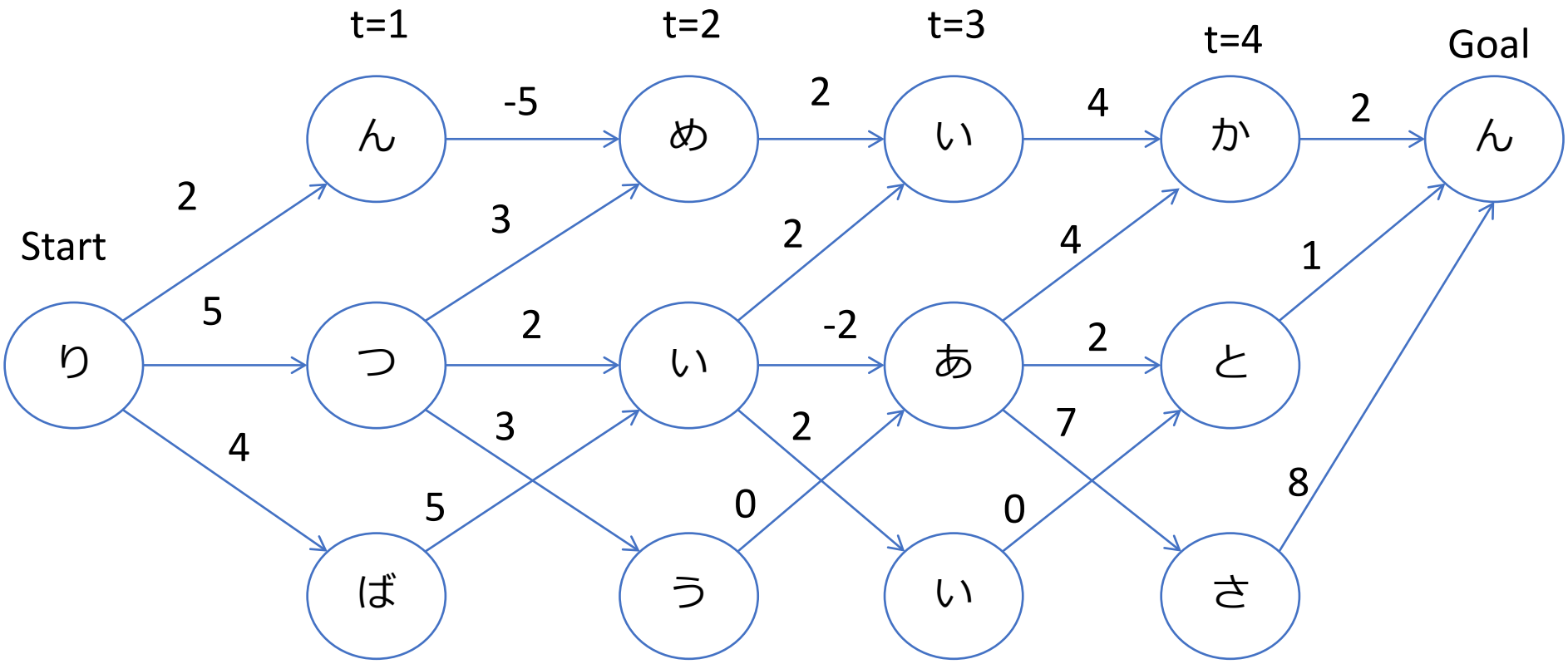
メモ化

演習問題5-1
文字の接続コスト（bi-gram）による単語生成



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

演習問題5-2 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$F_t(s_t)$と$\hat{s}_{t-1}(s_t)$のリストはどのようになっているか，示せ.

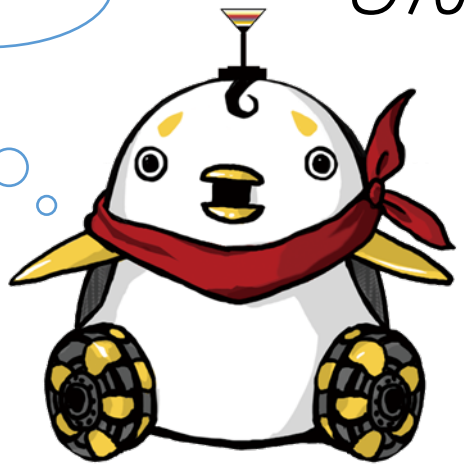
Contents

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.
- ロボットの移動だけではなく，様々な多段階決定問題に対して利用することが出来る.

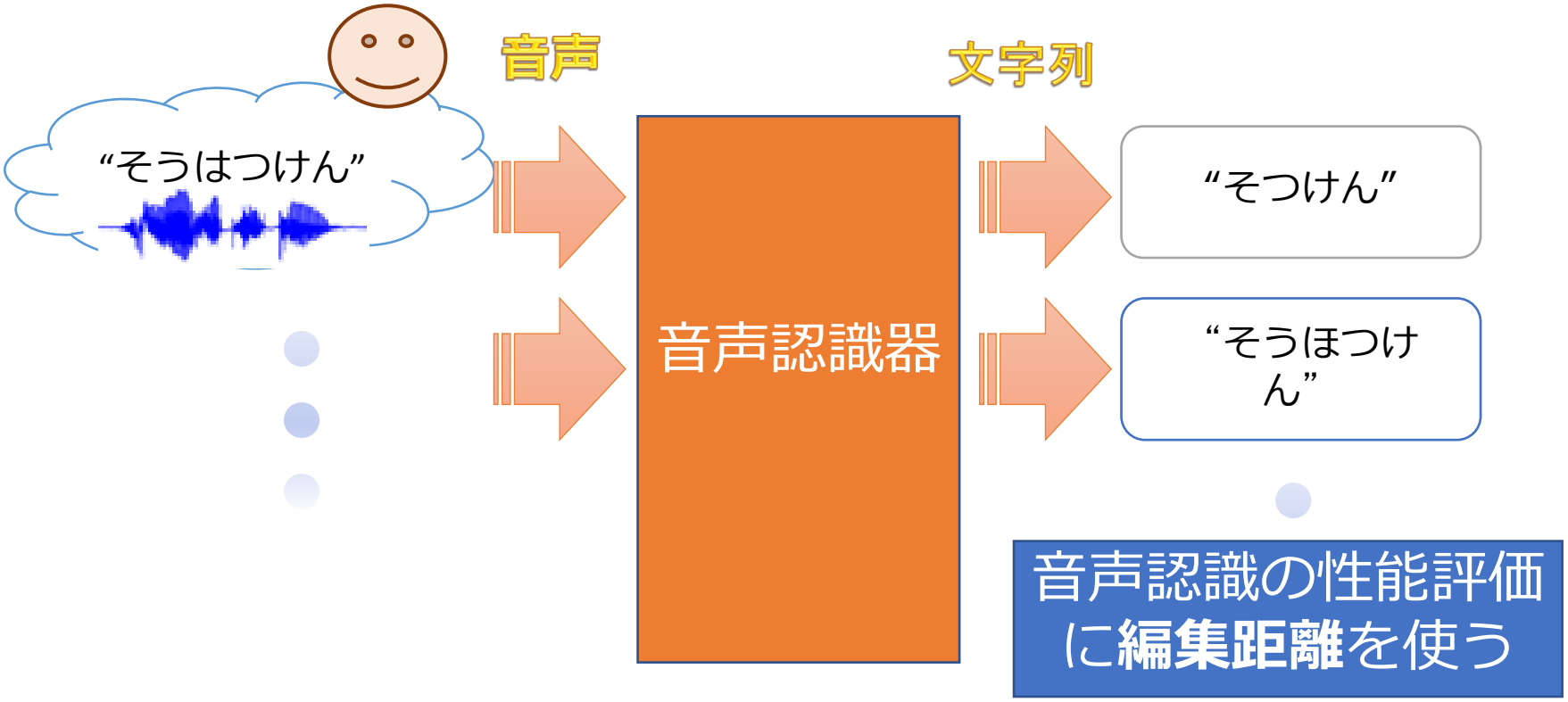
どれとどれが似てるんだ？
文字列と文字列の
距離を測りたい!!!!

編集距離



じんこうちのうがいろん？
じんこうちのうがいろん？
しんこのうがいろん？
どうてきけいかくほう
どてかいかくほう？

編集距離を使用した例：音声認識誤りの評価



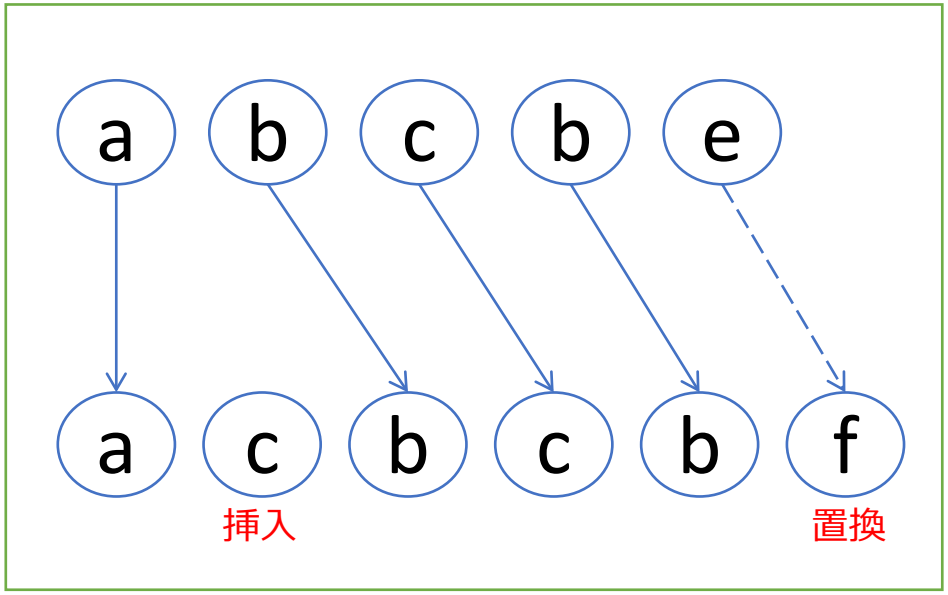
例：編集距離の計算

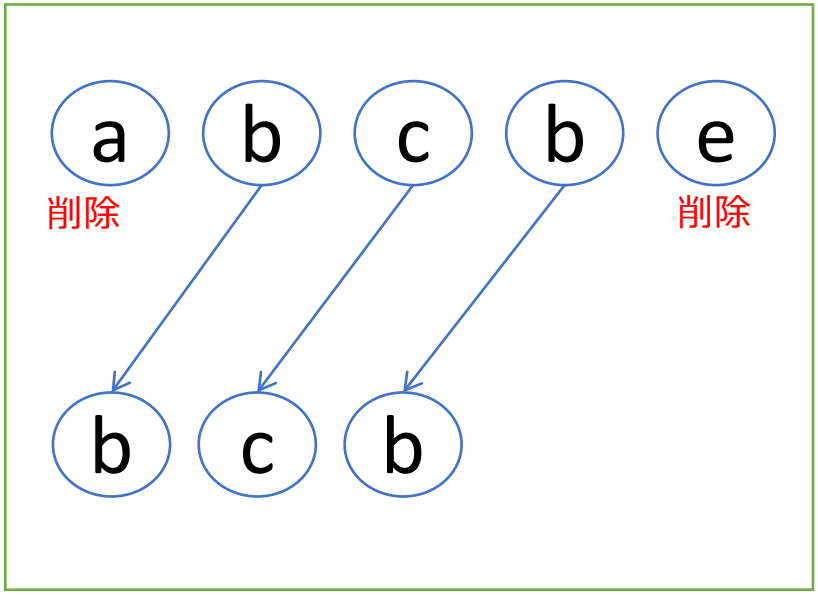
- **編集距離(edit distance)** は文字列と文字列の「**異なり具合**」を測る尺度
- **ハミング距離(Hamming distance)** では文字の置き換えには対応できるが，**文字の挿入や削除に対応できない**.
- 編集距離の計算にはストリングマッチングが必要.

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.

ストリングマッチング：文字列照合（String matching）



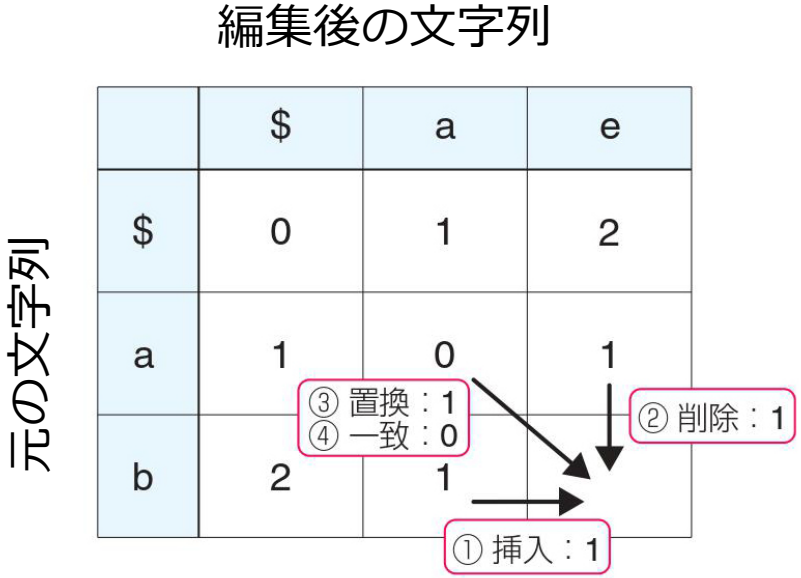


編集距離を計算時の各移動コスト

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.



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

編集後の文字列

元の文字列	$	a	e	b	c
$	0	1	2	3	4
a	1				
b	2				
c	3				
d	4				Goal

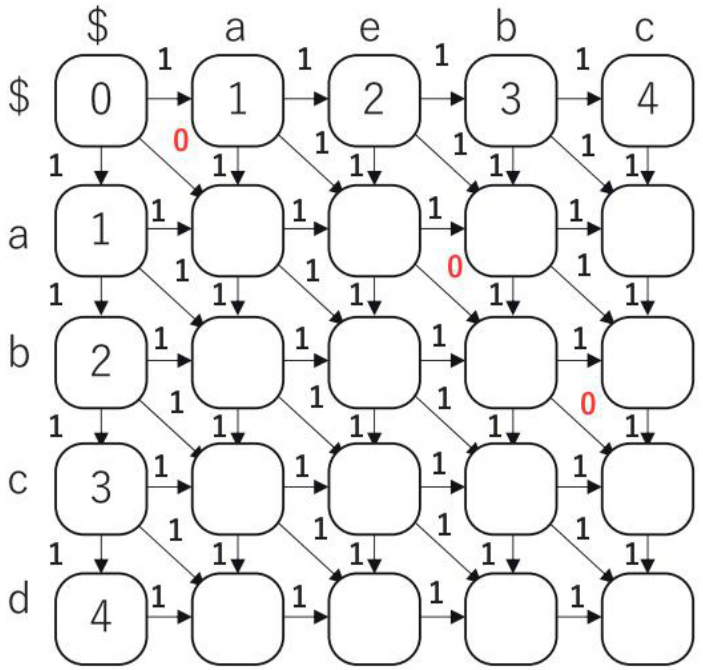


図 5.12 編集距離を計算するための表

Goalの値＝編集距離

編集距離の計算結果

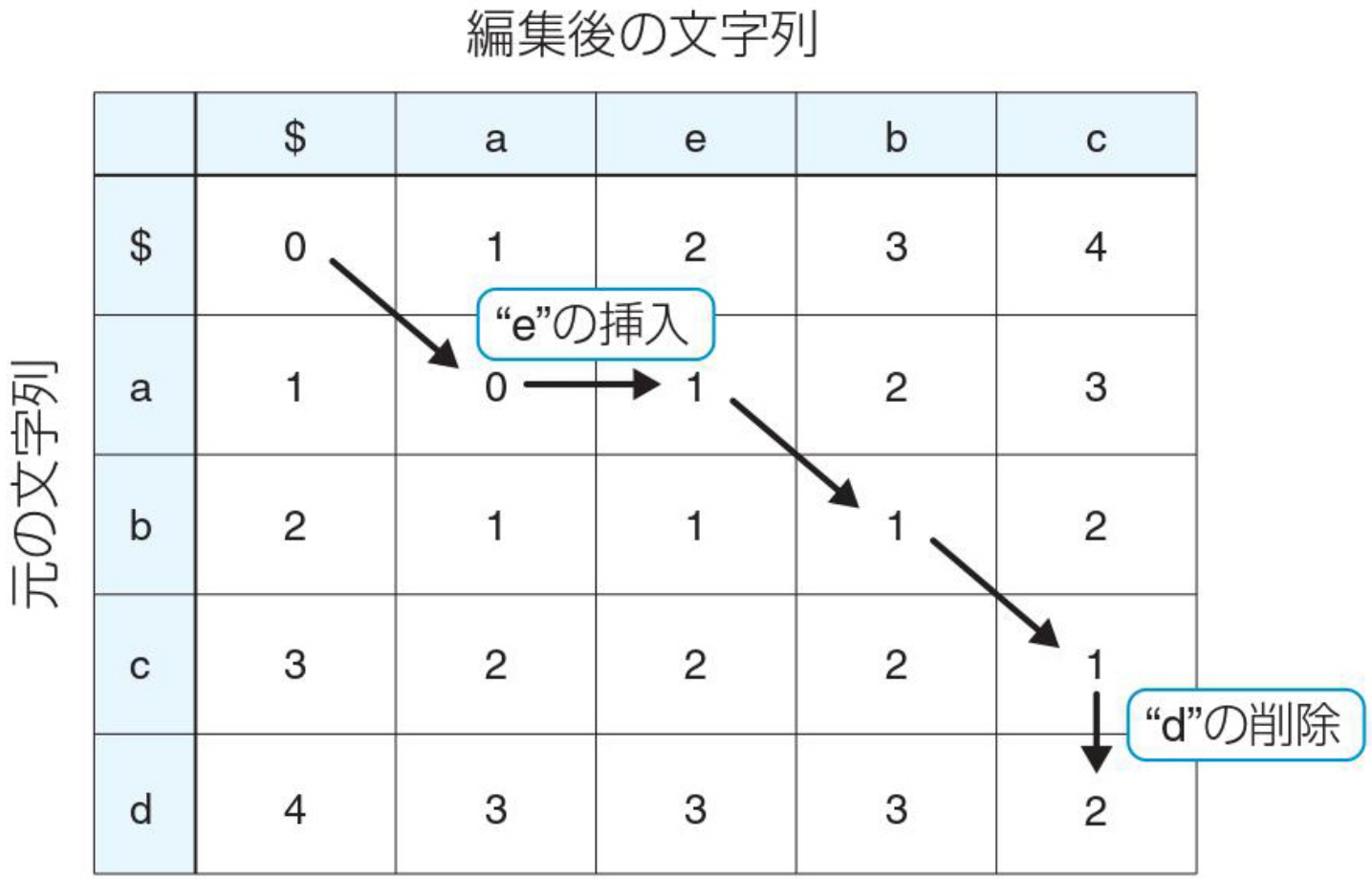


図 5.14 編集距離の計算結果

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

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，ストリングマッチングと編集距離の計算方法について学んだ.

- 次回：「確率モデル（１）」
  - 確率の基本定理，ベイズの定理，期待値，確率分布の推定