アルゴリズムとデータ構造III
8回目：12月03日

グラフ
（動的計画法，DPマッチング，A＊アルゴリズム）

授業資料　http://ir.cs.yamanashi.ac.jp/~ysuzuki/algorithm3/index.html

次回授業

- 時間　：　12月04日（金）4時限
- 場所　：　A1-41

授業評価アンケート（中間期評価）

- CNSの授業のコミュニティに以下の項目について記入してください（匿名での記入が可能）.

1. この授業の良いところはどこですか？
2. この授業の改善してほしいところはどこですか？

授業の予定(中間試験まで)

1	10/01	スタック（後置記法で書かれた式の計算）
2	10/15	文脈自由文法，構文解析，CYK法
3	10/22	構文解析　CYK法
4	10/29	構文解析　CYK法
5	11/12	構文解析　CYK法，動的計画法
6	11/19	構文解析（チャート法），グラフ（ダイクストラ法）
7	11/26	グラフ（ダイクストラ法，DPマッチング，A*アルゴリズム）
8	12/03	グラフ（A＊アルゴリズム），前半のまとめ
9	12/04 4時限	教室：A1-41 全文検索アルゴリズム（simple search, KMP）

授業の予定(中間試験以降)

10	12/10	中間試験(8回目までの範囲)
11	12/11 4時限	教室:A1-41 全文検索アルゴリズム(BM, Aho-Corasick)
12	12/17	全文検索アルゴリズム(Aho-Corasick), データ圧縮
13	01/07	暗号(黄金虫, 踊る人形) 符号化(モールス信号, Zipfの法則, ハフマン符号)テキスト圧縮
14	01/14	テキスト圧縮 (zip), 音声圧縮 (ADPCM, MP3, CELP), 画像圧縮(JPEG)
15	01/21	期末試験

中間試験

- 中間試験日
  - 12月10日（木）
- 範囲
  - スタック
  - 文脈自由文法
  - 構文解析
  - CYK法
  - （トップダウンチャート法）
  - 動的計画法
  - ダイクストラ法
  - DPマッチング
  - A*アルゴリズム

本日のメニュー

- 動的計画法
- DPマッチング
  - アルゴリズム
  - 動作
- A*アルゴリズム
- 中間試験の範囲の説明

動的計画法（Dynamic Programming)

- 解くのに時間のかかる問題を、複数の部分問題に分割することで効率的に解くアルゴリズム

ダイクストラ法

- 動的計画法を最短経路問題に適用
- 最適経路中の部分経路もまた最適経路になっている

ダイクストラ法　アルゴリズム

1. 初期化：スタートノードの値（最小コスト候補）を0，他のノードの値を無限大に設定
2. 未確定ノードが無くなるまで以下のループを繰り返す．
   1. 確定中ノードのうち，最小の値を持つノードを見つけ，確定ノードとする．
   2. 確定ノードからのエッジに対して「確定ノードまでのコスト＋エッジのコスト」を計算し，そのノードの現在値よりも小さければ更新．

ダイクストラ法のアルゴリズム

- begin
  - for each x ∈V do begin
    - cost[x]:=w[s,x];
    - parent[x]:=‘s’;
  - end
  - U:=V-{s};
  - while U ≠ ϕ do
  - begin
    - U中のmで，cost[m]が最小となる頂点mを選ぶ；
    - U:=U-{m};
    - mから隣接する頂点の集合をD_mとする；
    - for each x ∈ D_m ∩ U do
      - If cost[m]+w[m,x]<cost[x]
      - then begin
        - Cost[x]:=cost[m]+w[m,x];
        - Parent[x:]:=m
      - end
  - end
- end

costとparentの初期化

……U（未確定ノード）の初期化

集積コストが最も小さいノードmを選んで，cost[m]とparent[m]を確定

頂点mから隣接するノードすべての集合Dmを求める．Dmの要素で且つ未確定ノードである各xについてmを経由してxに至る最短経路のコストを計算し，現在のcost[x]と比較し，小さければ更新する

ダイクストラ法の特徴

- 最短経路の見つけ方
  - ゴールノードから「どこから来たのか」調べ，さかのぼる.
- マイナスのコストを持つエッジは扱えない.
- 特定のノードからの最短距離およびその経路が全てのノードに対して求まる.

DPマッチング（例：文字列の照合）

- 2つの文字列がどのくらい似ているかを調べる．
  - takeda は nakadaiとどのくらい似ているか
  - 置換，脱落，挿入に対応
- 音声認識にも使える
  - 音声を文字列に変換した後，登録単語と比較
  - （現在主流の）HMM(Hidden Markov Model)に拡張可能
- DNAの比較にも使える
  - A（アデニン），G（グアニン），C（シトシン），T（チミン）の並び方の比較
  - ACTGAGCATTとCTGGACTACGの比較

DPマッチング（例：文字列の照合）

- 簡単に比較できる例
  - abcdef
  - abzdef
- Aに対して脱落，挿入，置換
  - A: abcdef
  - B: abdef
  - C: abccdef
  - D: abzdef

DPマッチング：脱落，挿入，置換誤りを考慮して文字列照合可能

DPマッチング（例：文字列の照合）1/8

takeda と nakadai の照合

文字が一致→ 0
文字が不一致→ 3

不一致コスト表

	n	a	k	a	d	a	i
t	3	3	3	3	3	3	3
a	3	0	3	0	3	0	3
k	3	3	0	3	3	3	3
e	3	3	3	3	3	3	3
d	3	3	3	3	0	3	3
a	3	0	3	0	3	0	3

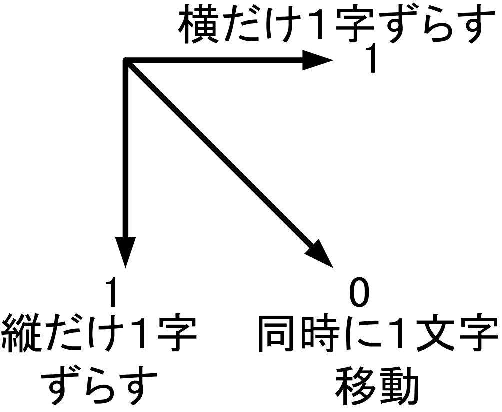
DPマッチング（例：文字列の照合）2/8

- takeda と　nakadai　　　の値を求める

不一致コスト表

	n	a	k	a	d	a	i
t	3	3	3	3	3	3	3
a	3	0	3	0	3	0	3
k	3	3	0	3	3	3	3
e	3	3	3	3	3	3	3
d	3	3	3	3	0	3	3
a	3	0	3	0	3	0	3

移動のペナルティ



不一致のペナルティ

文字が一致→ 0
文字が不一致→ 3

1文字ずらしたけれど文字が不一致：1+3=4を加算

	n	a	k	a	d	a	i
t	3 →	7	11	15	19	23	27
a	7						
k	11						
e	15						
d	19						
a	23						

DPマッチング（例：文字列の照合）3/8

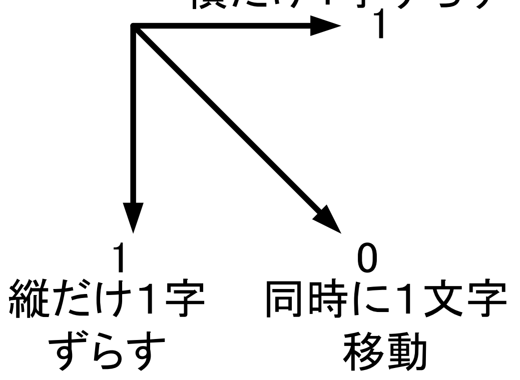
- takeda と nakadai

■ の値を求める

不一致コスト表

	n	a	k	a	d	a	i
t	3	3	3	3	3	3	3
a	3	0	3	0	3	0	3
k	3	3	0	3	3	3	3
e	3	3	3	3	3	3	3
d	3	3	3	3	0	3	3
a	3	0	3	0	3	0	3

移動のペナルティ



不一致のペナルティ

文字が一致→ 0
文字が不一致→ 3

	n	a	k	a	d	a	i
t	3	7	11	15	19	23	27
a	7	3	7	8	12	13	17
k	11						
e	15						
d	19						
a	23						

DPマッチング（例：文字列の照合）4/8

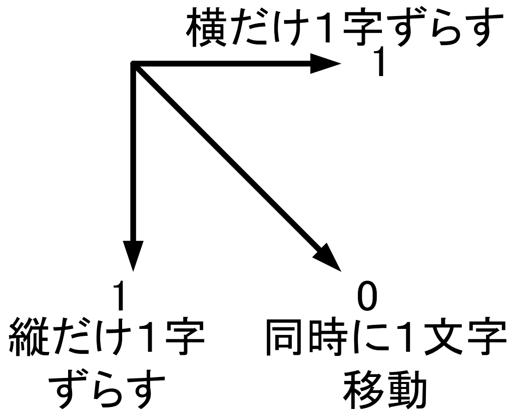
- takeda と　nakadai

（黄色のマス）の値を求める

不一致コスト表

	n	a	k	a	d	a	i
t	3	3	3	3	3	3	3
a	3	0	3	0	3	0	3
k	3	3	0	3	3	3	3
e	3	3	3	3	3	3	3
d	3	3	3	3	0	3	3
a	3	0	3	0	3	0	3

移動のペナルティ



不一致のペナルティ

文字が一致→ 0
文字が不一致→ 3

	n	a	k	a	d	a	i
t	3	7	11	15	19	23	27
a	7	3	7	8	12	13	17
k	11	7	3	7	11	15	16
e	15						
d	19						
a	23						

DPマッチング（例：文字列の照合）5/8

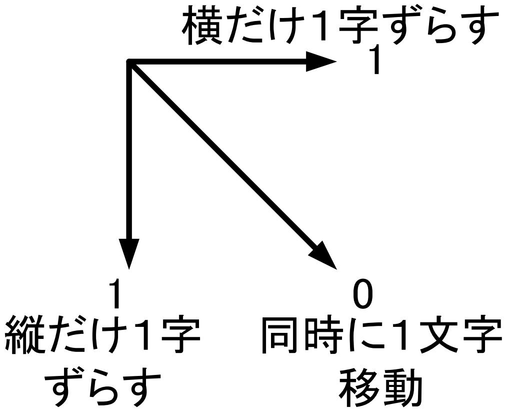
- takeda と　nakadai

（黄色の枠）の値を求める

不一致コスト表

	n	a	k	a	d	a	i
t	3	3	3	3	3	3	3
a	3	0	3	0	3	0	3
k	3	3	0	3	3	3	3
e	3	3	3	3	3	3	3
d	3	3	3	3	0	3	3
a	3	0	3	0	3	0	3

移動のペナルティ



不一致のペナルティ

文字が一致→ 0
文字が不一致→ 3

	n	a	k	a	d	a	i
t	3	7	11	15	19	23	27
a	7	3	7	8	12	13	17
k	11	7	3	7	11	15	16
e	15	11	7	6	10	14	18
d	19						
a	23						（黄色の枠）

DPマッチング（例：文字列の照合）6/8

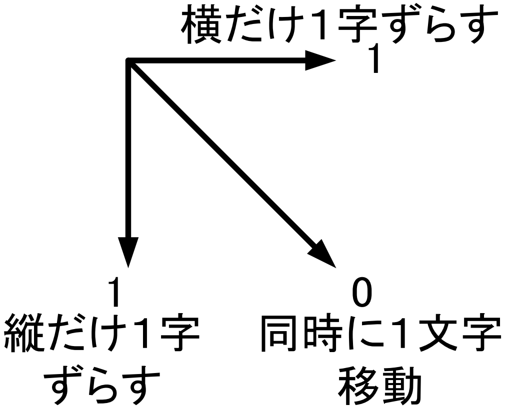
- takeda と　nakadai

（黄色のマス）の値を求める

不一致コスト表

	n	a	k	a	d	a	i
t	3	3	3	3	3	3	3
a	3	0	3	0	3	0	3
k	3	3	0	3	3	3	3
e	3	3	3	3	3	3	3
d	3	3	3	3	0	3	3
a	3	0	3	0	3	0	3

移動のペナルティ



不一致のペナルティ

文字が一致→ 0
文字が不一致→ 3

	n	a	k	a	d	a	i
t	3	7	11	15	19	23	27
a	7	3	7	8	12	13	17
k	11	7	3	7	11	15	16
e	15	11	7	6	10	14	18
d	19	15	11	10	6	10	14
a	23						

DPマッチング（例：文字列の照合）7/8

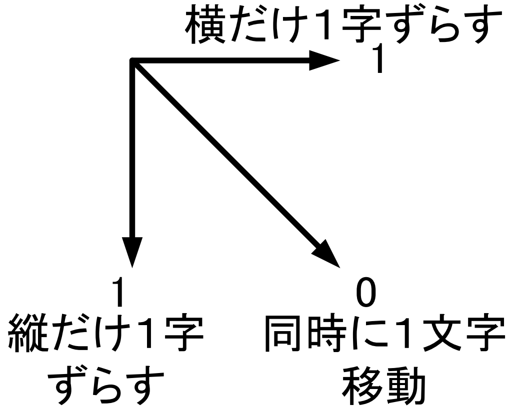
- takeda と nakadai

（黄色の枠）の値を求める

不一致コスト表

	n	a	k	a	d	a	i
t	3	3	3	3	3	3	3
a	3	0	3	0	3	0	3
k	3	3	0	3	3	3	3
e	3	3	3	3	3	3	3
d	3	3	3	3	0	3	3
a	3	0	3	0	3	0	3

移動のペナルティ



不一致のペナルティ

文字が一致→ 0
文字が不一致→ 3

	n	a	k	a	d	a	i
t	3	7	11	15	19	23	27
a	7	3	7	8	12	13	17
k	11	7	3	7	11	15	16
e	15	11	7	6	10	14	18
d	19	15	11	10	6	10	14
a	23	16	15	11	10	6	10

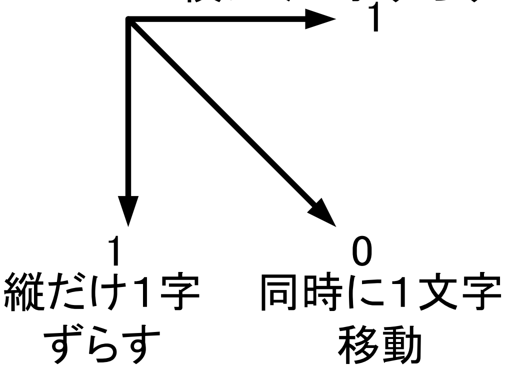
DPマッチング（例：文字列の照合）8/8

- takeda と nakadai　　（黄色のマス）の値を求める

不一致コスト表

	n	a	k	a	d	a	i
t	3	3	3	3	3	3	3
a	3	0	3	0	3	0	3
k	3	3	0	3	3	3	3
e	3	3	3	3	3	3	3
d	3	3	3	3	0	3	3
a	3	0	3	0	3	0	3

移動のペナルティ

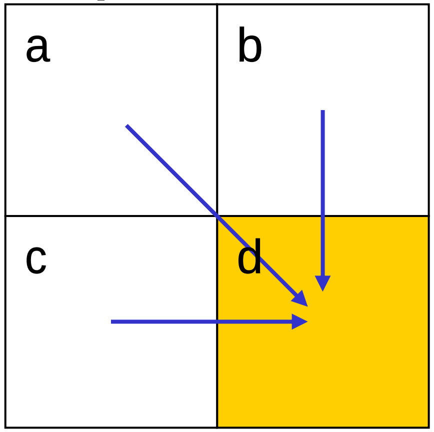


不一致のペナルティ

文字が一致→ 0
文字が不一致→ 3

	n	a	k	a	d	a	i
t	3	7	11	15	19	23	27
a	7	3	7	8	12	13	17
k	11	7	3	7	11	15	16
e	15	11	7	6	10	14	18
d	19	15	11	10	6	10	14
a	23	16	15	11	10	6	10

DPマッチング（例：文字列の照合）アルゴリズム



d へのルートは3種類

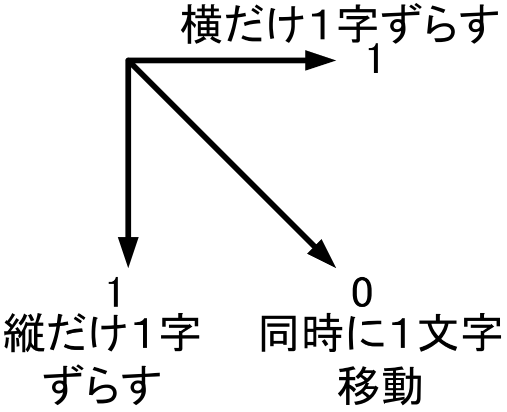
aまでの距離＋斜め移動のペナルティ＋不一致ペナルティ

bまでの距離＋下移動のペナルティ＋不一致ペナルティ

cまでの距離＋右移動のペナルティ＋不一致ペナルティ

の内の最短距離をdに書き込む

移動のペナルティ



不一致のペナルティ

文字が一致→ 0
文字が不一致→ 3

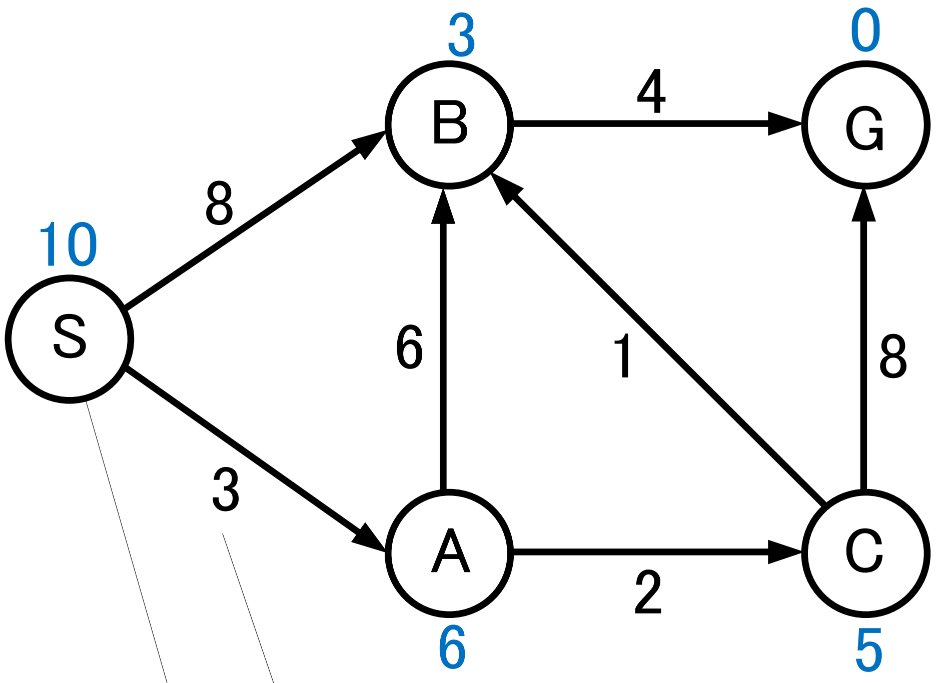
DPマッチングの応用

- DPマッチングの探索空間を制限し、探索時間を削減する方法
  - ビームサーチ
  - A*アルゴリズム
- HMM（隠れマルコフモデル）とビタビアルゴリズム
  - 音声認識手法の主流

A*アルゴリズム
最短経路探索問題

- ダイクストラ法にすこし工夫を加えた方法
- 各ノードからゴールまでの推定距離を利用
  - 0≦推定距離≦最短距離でなければならない
  - 推定距離=0なら推定していないと同じ→ダイクストラ法

まずはAアルゴリズム



Aを経由するルートの推定コスト評価値

$$\hat{f}(A)=\hat{g}(S)+w(S,A)+\hat{h}(A)$$

$$=0+3+6=9$$

AからGまでの推定コスト

$\forall n, 0 \le \hat{h}(n) \le h(n)$

を満たすとき

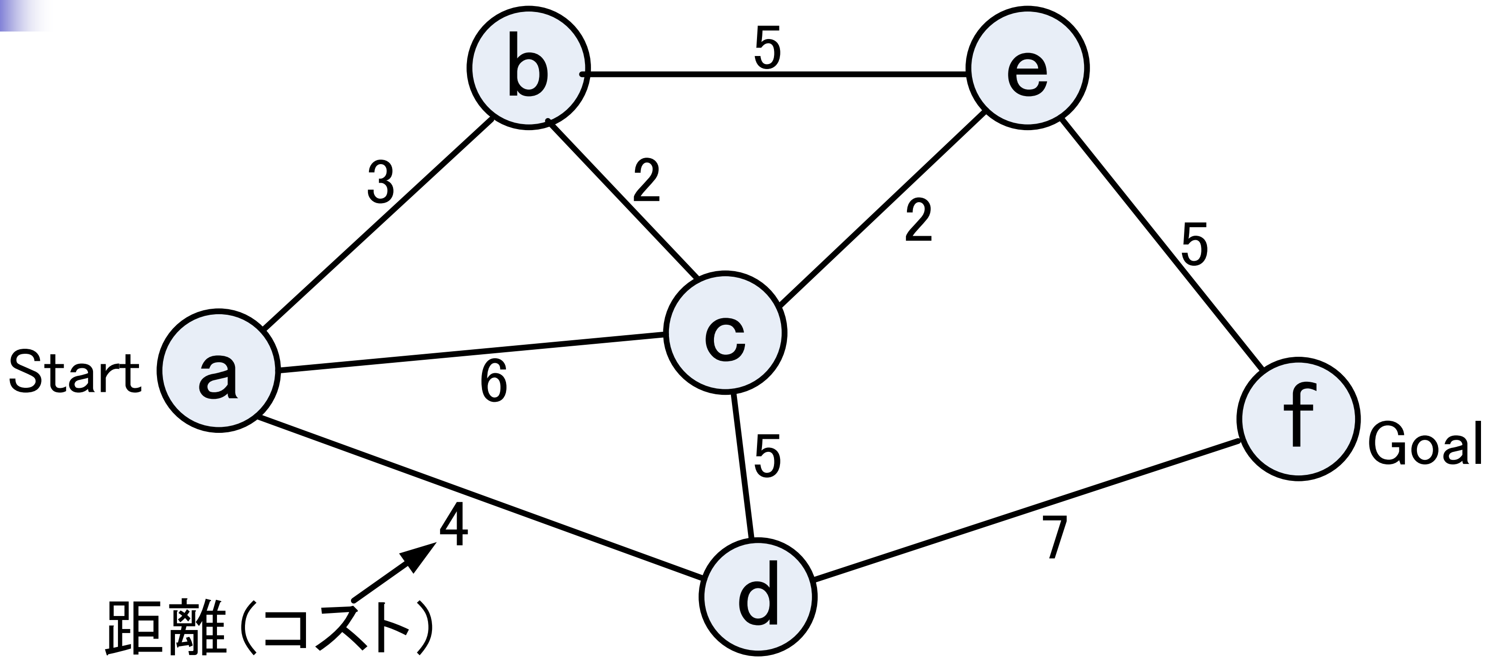
A^*アルゴリズム

但し，nは節点番号，

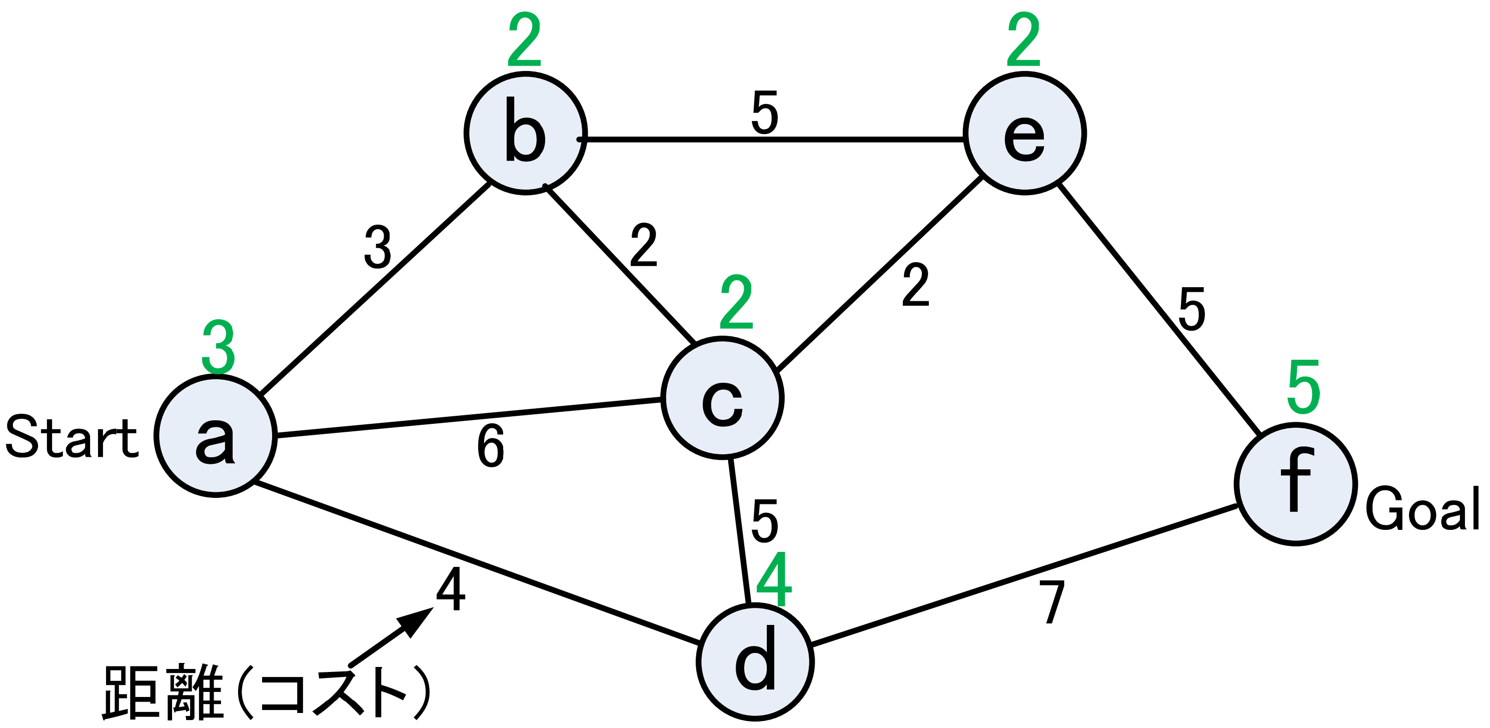
$h(n)$は節点nから

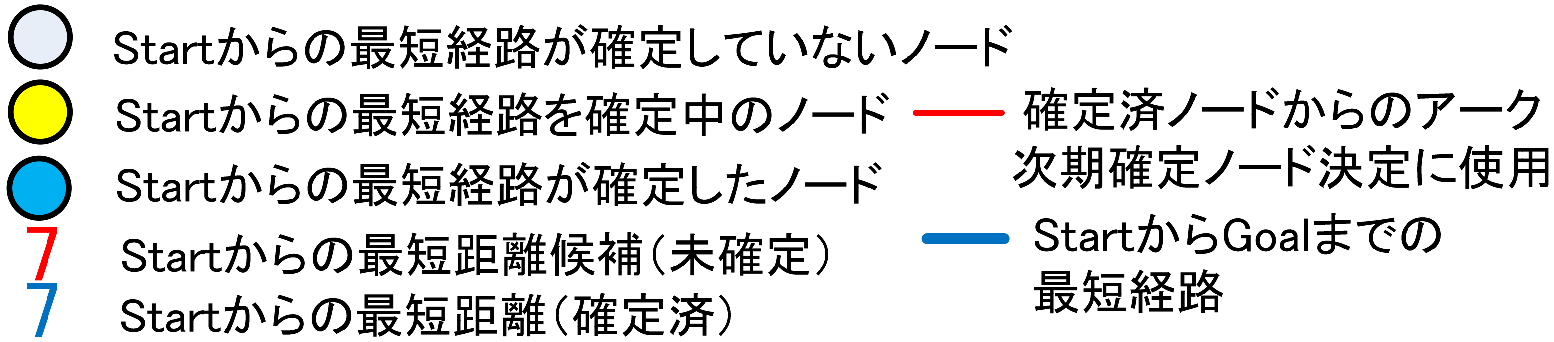
ゴールまでのコスト

A*アルゴリズム　動作例

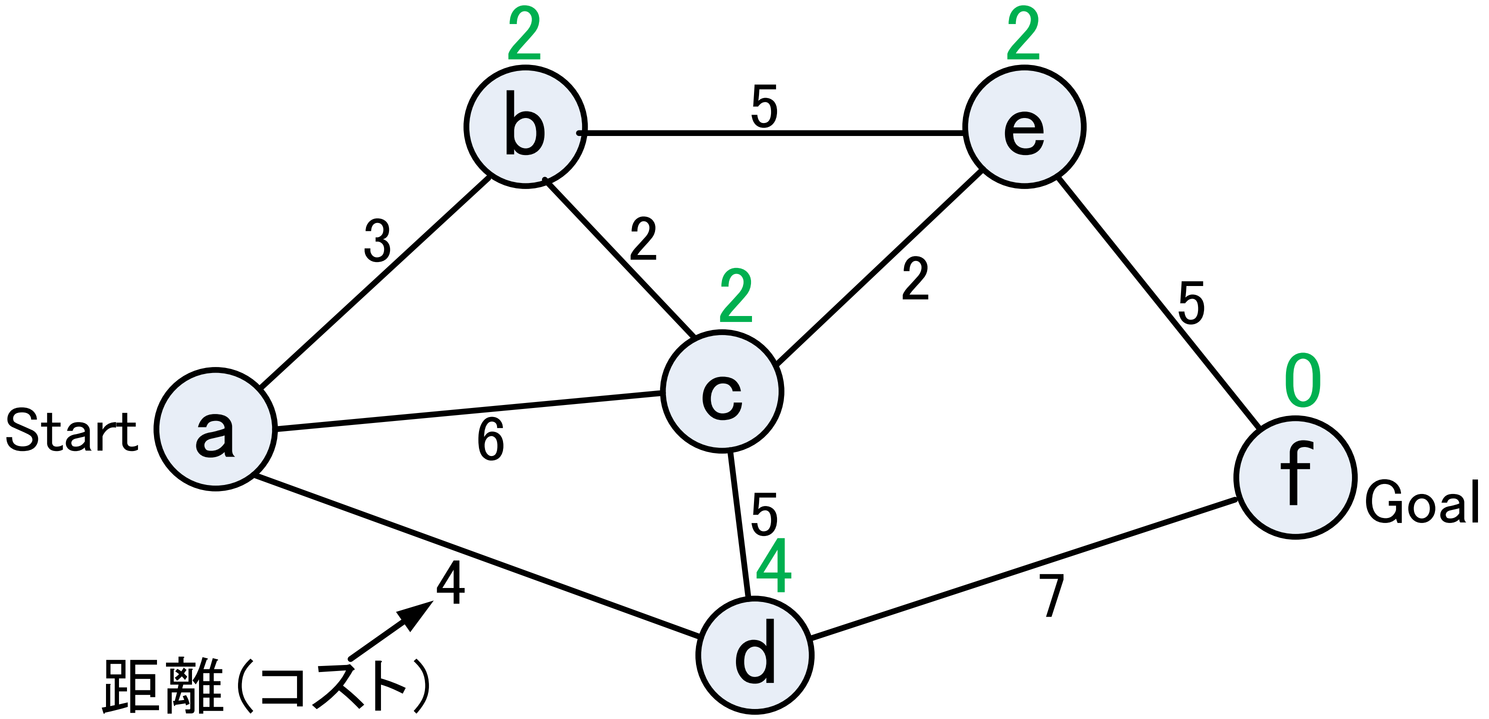


A*アルゴリズム　動作例 1/14



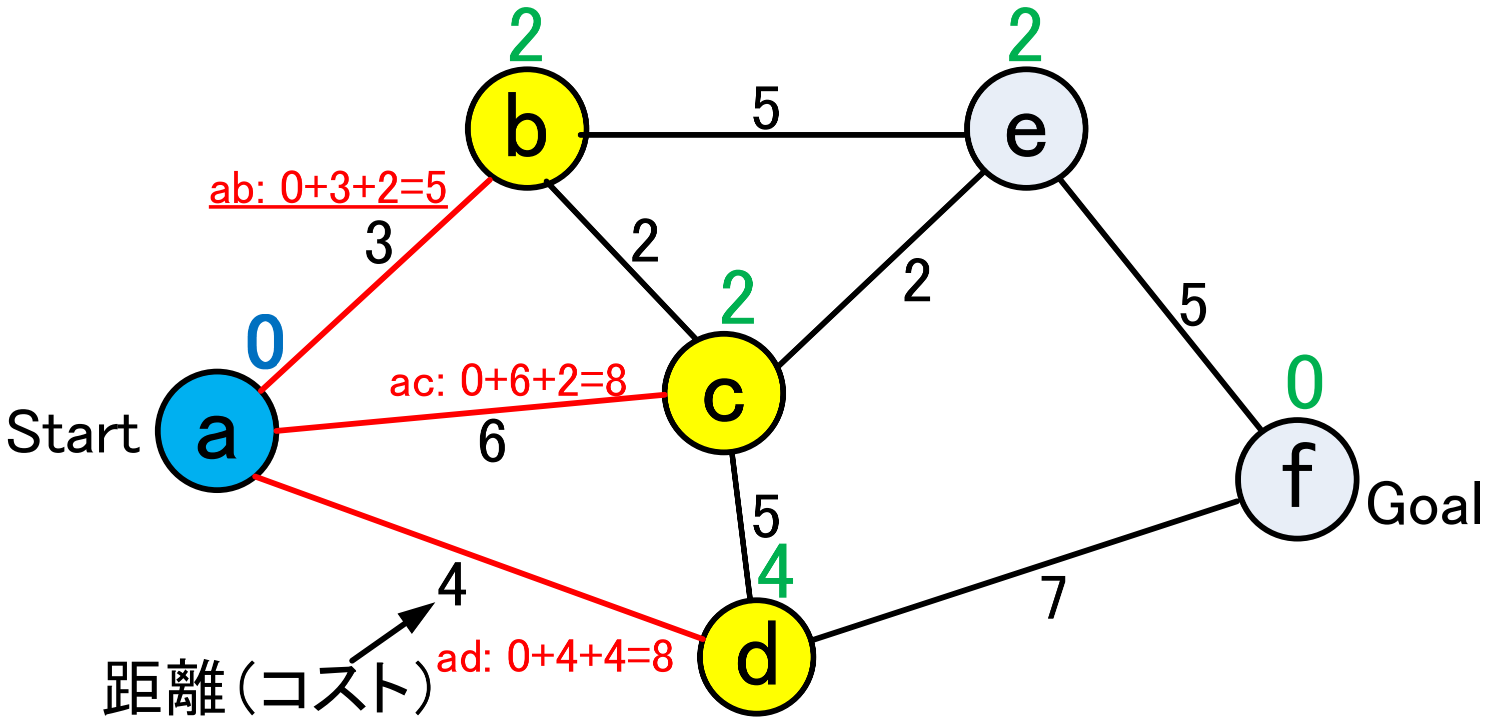


A*アルゴリズム　動作例 2/14



- Startからの最短経路が確定していないノード
- Startからの最短経路を確定中のノード
- Startからの最短経路が確定したノード
- 7 Startからの最短距離候補（未確定）
- 7 Startからの最短距離（確定済）
- 確定済ノードからのアーク 次期確定ノード決定に使用
- StartからGoalまでの最短経路

A*アルゴリズム　動作例 3/14



Startからの最短経路が確定していないノード

Startからの最短経路を確定中のノード

Startからの最短経路が確定したノード

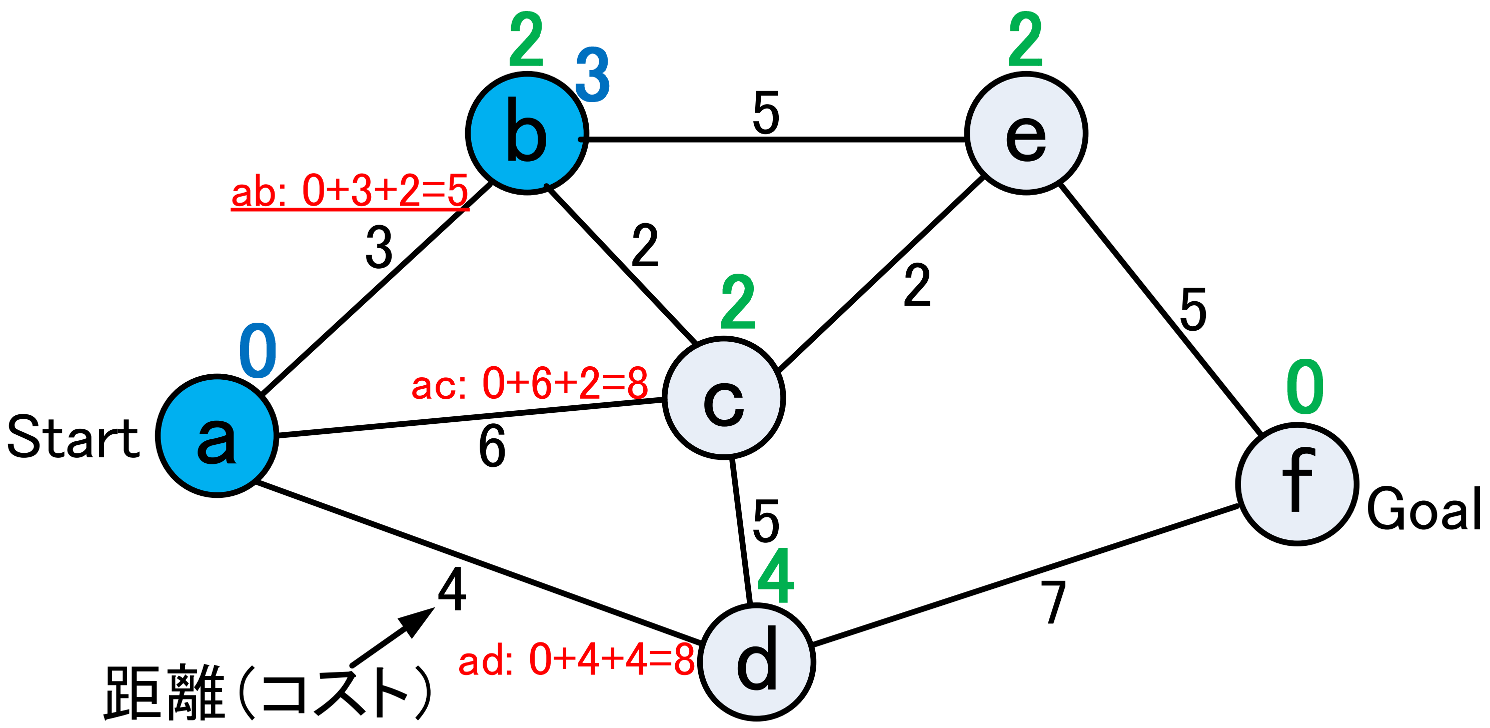
7 Startからの最短距離候補(未確定)

7 Startからの最短距離(確定済)

確定済ノードからのアーク
次期確定ノード決定に使用

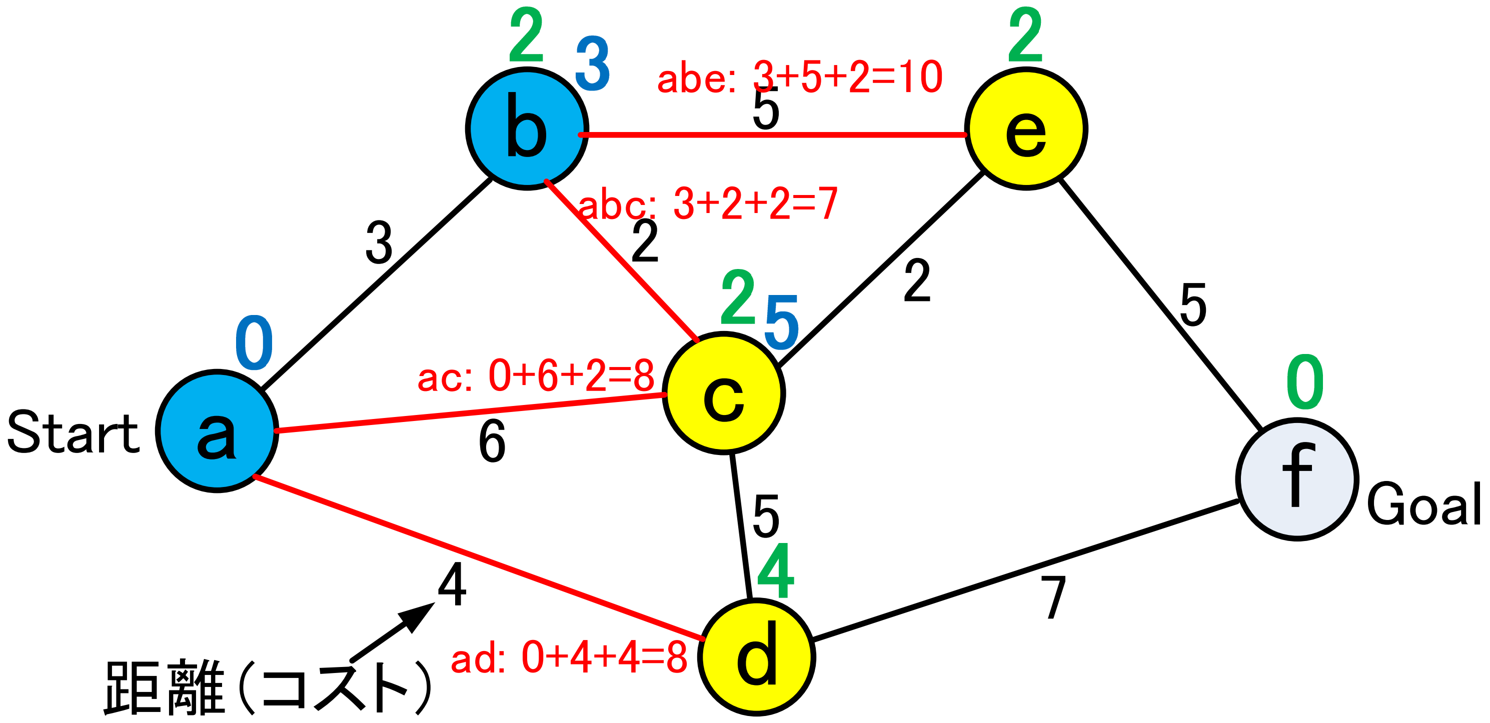
StartからGoalまでの
最短経路

A*アルゴリズム　動作例 4/14



- Startからの最短経路が確定していないノード
- Startからの最短経路を確定中のノード
- Startからの最短経路が確定したノード
- 7 Startからの最短距離候補(未確定)
- 7 Startからの最短距離(確定済)
- 確定済ノードからのアーク 次期確定ノード決定に使用
- StartからGoalまでの最短経路

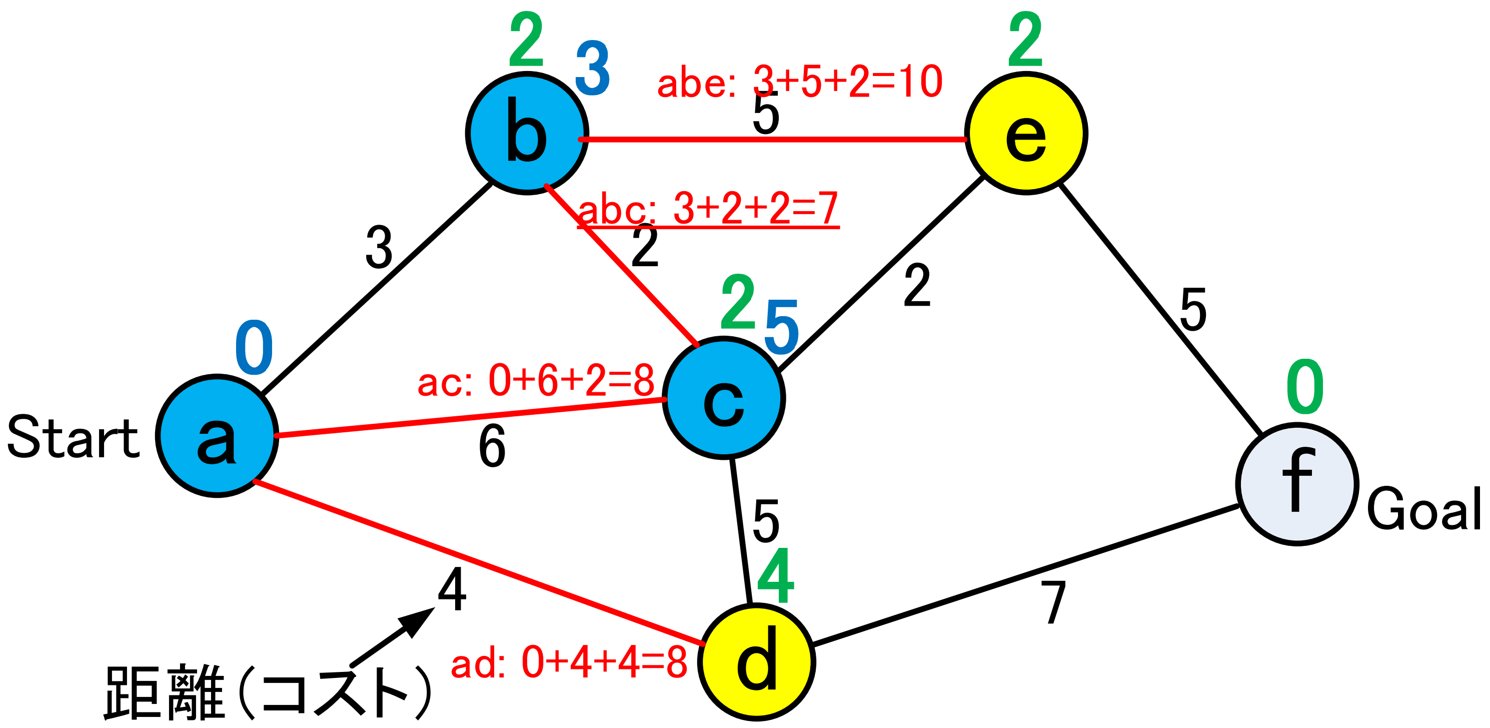
A*アルゴリズム　動作例 5/14



- Startからの最短経路が確定していないノード
- Startからの最短経路を確定中のノード
- Startからの最短経路が確定したノード
- 7 Startからの最短距離候補(未確定)
- 7 Startからの最短距離(確定済)

- 確定済ノードからのアーク
次期確定ノード決定に使用
- StartからGoalまでの
最短経路

A*アルゴリズム　動作例 6/14



Startからの最短経路が確定していないノード

Startからの最短経路を確定中のノード

Startからの最短経路が確定したノード

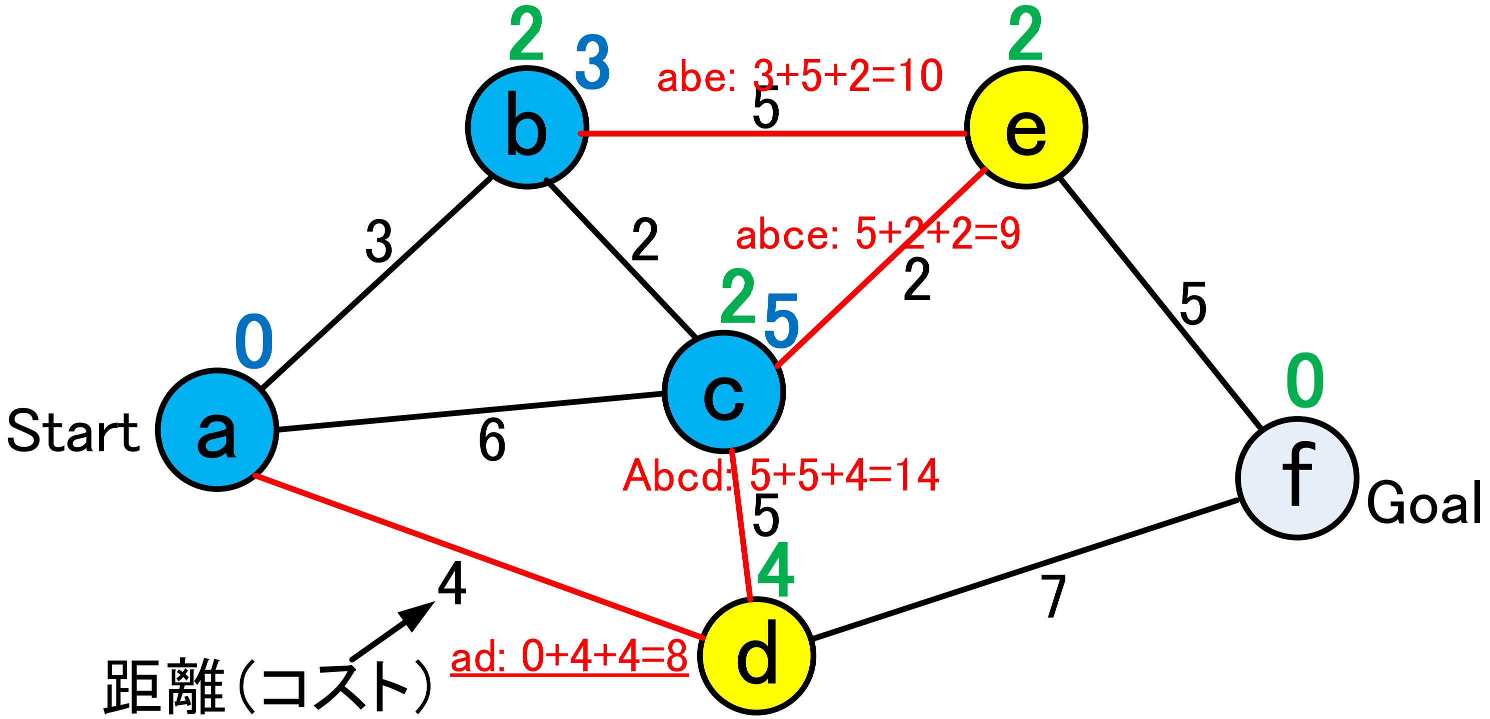
7 Startからの最短距離候補（未確定）

7 Startからの最短距離（確定済）

確定済ノードからのアーク
次期確定ノード決定に使用

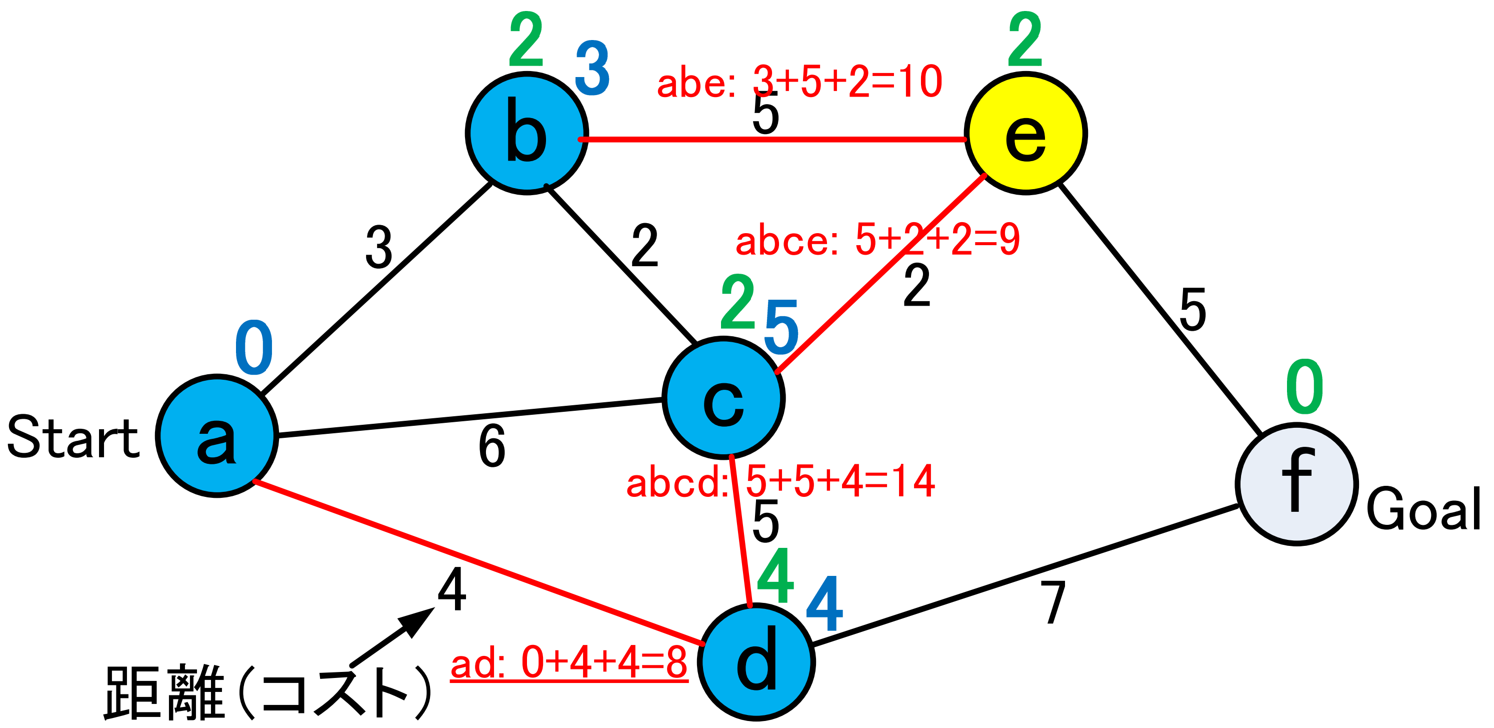
StartからGoalまでの
最短経路

A*アルゴリズム　動作例 7/14



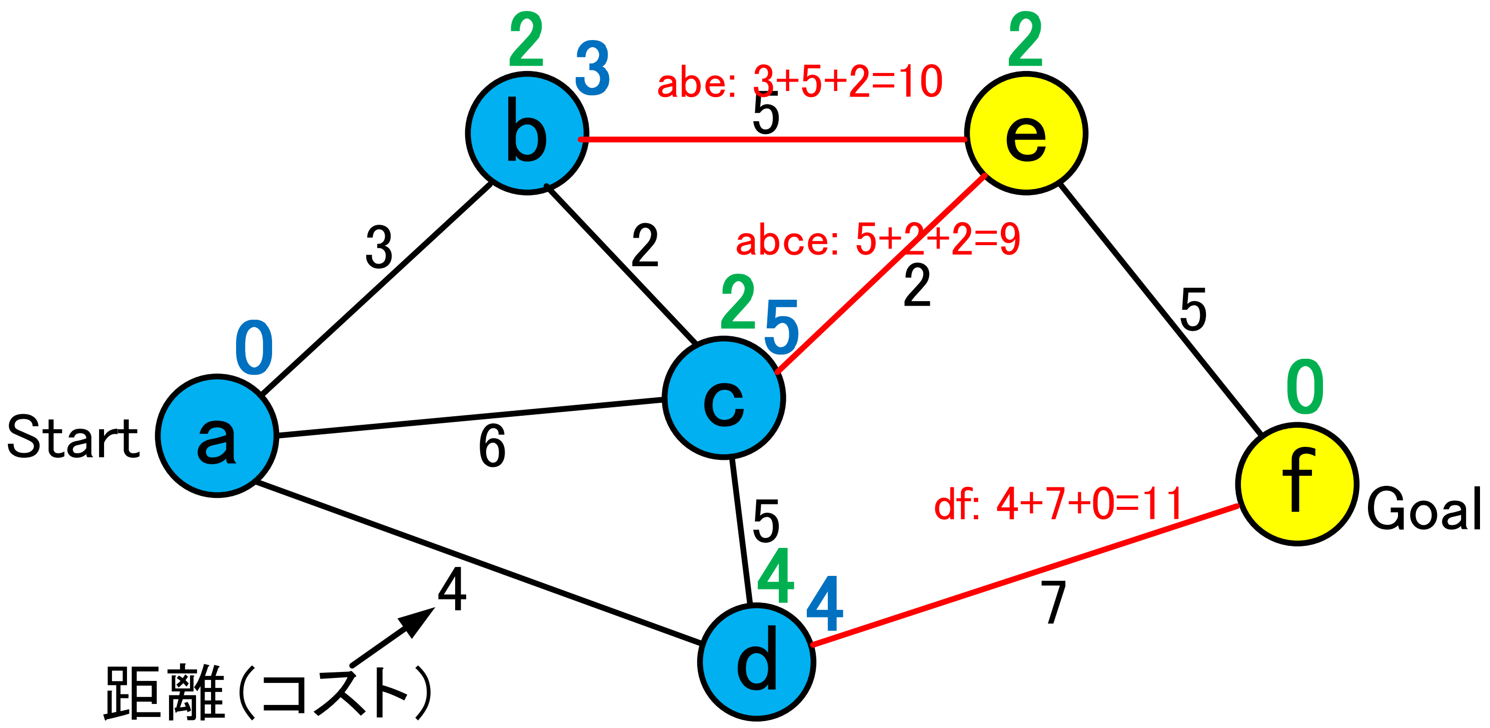
- Startからの最短経路が確定していないノード
- Startからの最短経路を確定中のノード
- Startからの最短経路が確定したノード
- 7 Startからの最短距離候補(未確定)
- 7 Startからの最短距離(確定済)
- 確定済ノードからのアーク 次期確定ノード決定に使用
- StartからGoalまでの最短経路

A*アルゴリズム　動作例 8/14



- Startからの最短経路が確定していないノード
- Startからの最短経路を確定中のノード
- Startからの最短経路が確定したノード
- 7 Startからの最短距離候補（未確定）
- 7 Startからの最短距離（確定済）
- 確定済ノードからのアーク
  次期確定ノード決定に使用
- StartからGoalまでの
  最短経路

A*アルゴリズム　動作例 9/14



Startからの最短経路が確定していないノード

Startからの最短経路を確定中のノード

Startからの最短経路が確定したノード

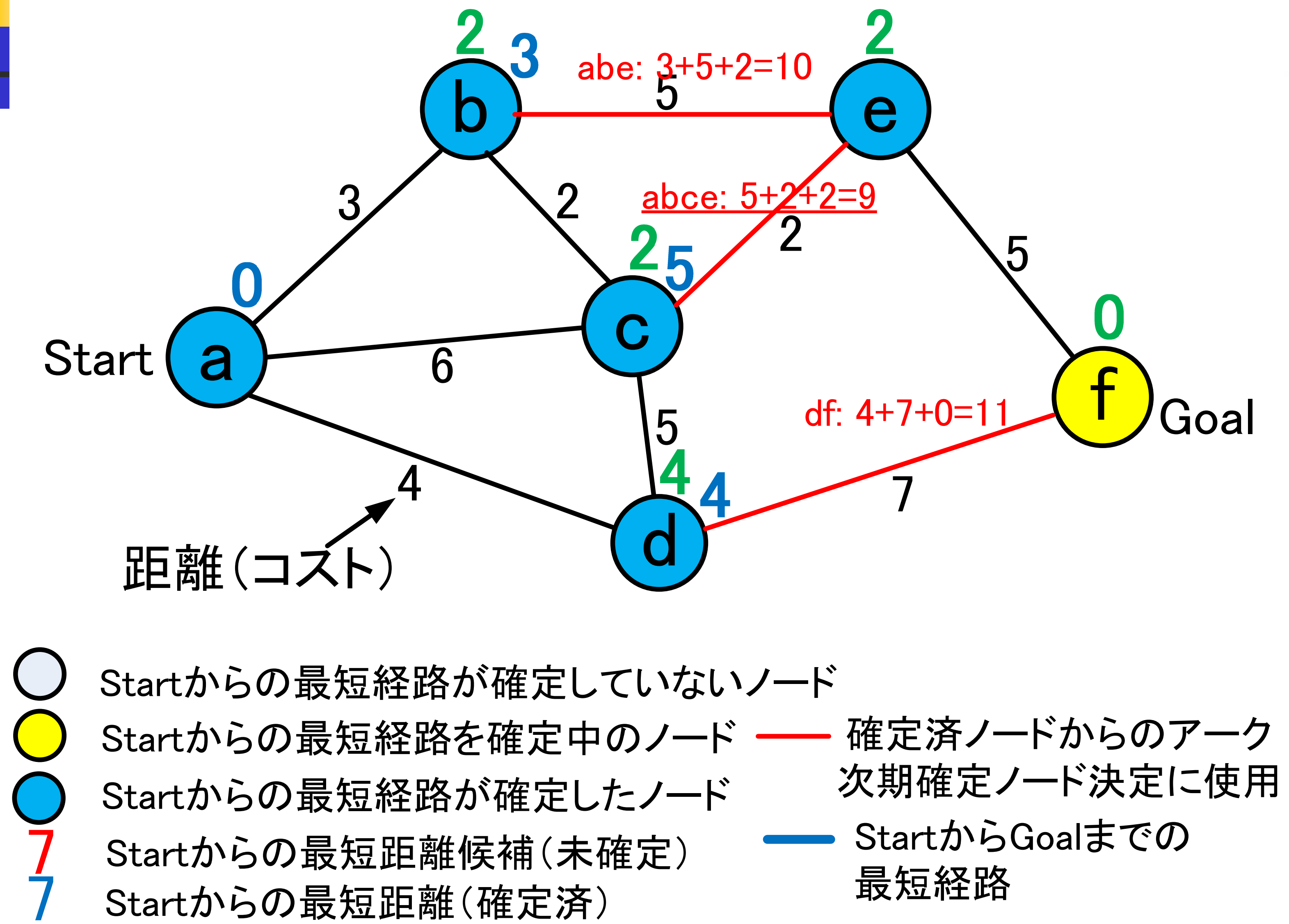
7 Startからの最短距離候補（未確定）

7 Startからの最短距離（確定済）

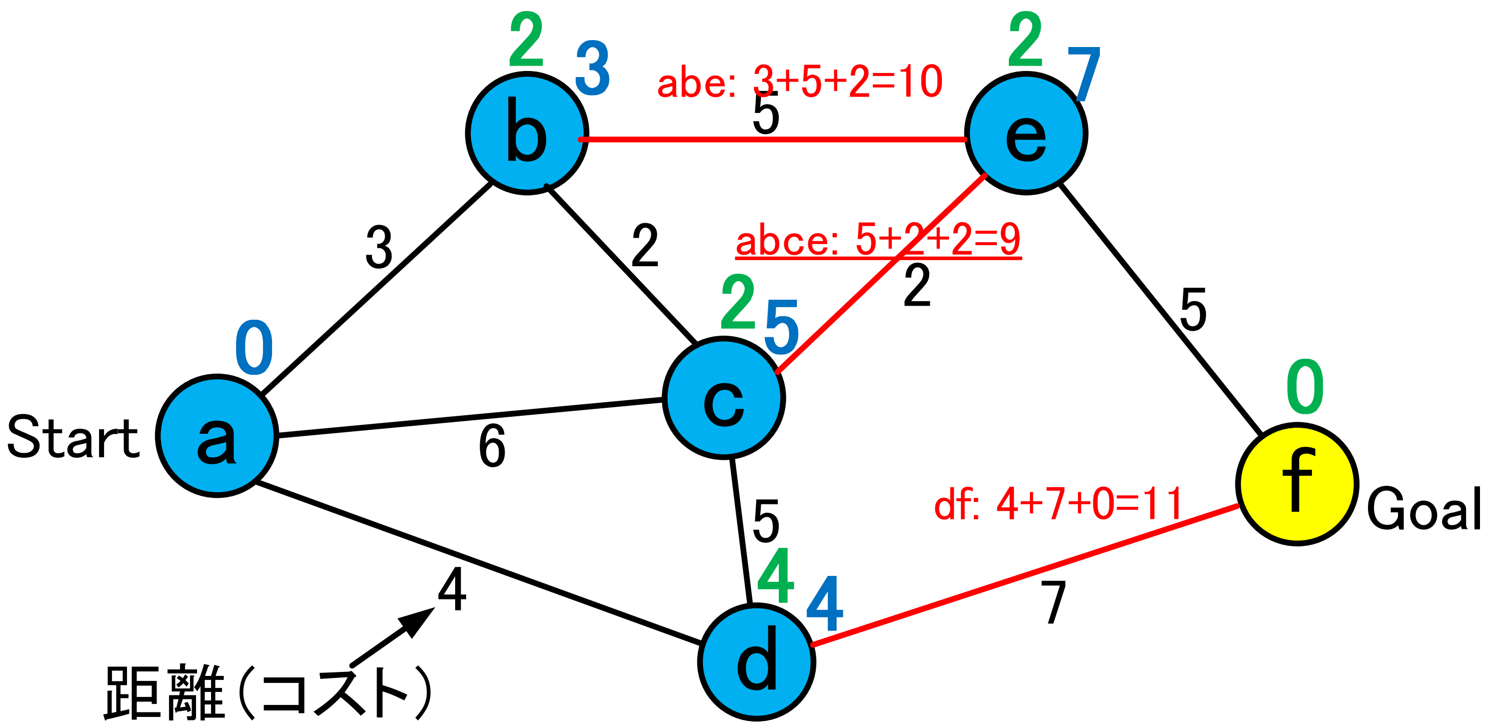
確定済ノードからのアーク
次期確定ノード決定に使用

StartからGoalまでの
最短経路

A*アルゴリズム　動作例 10/14

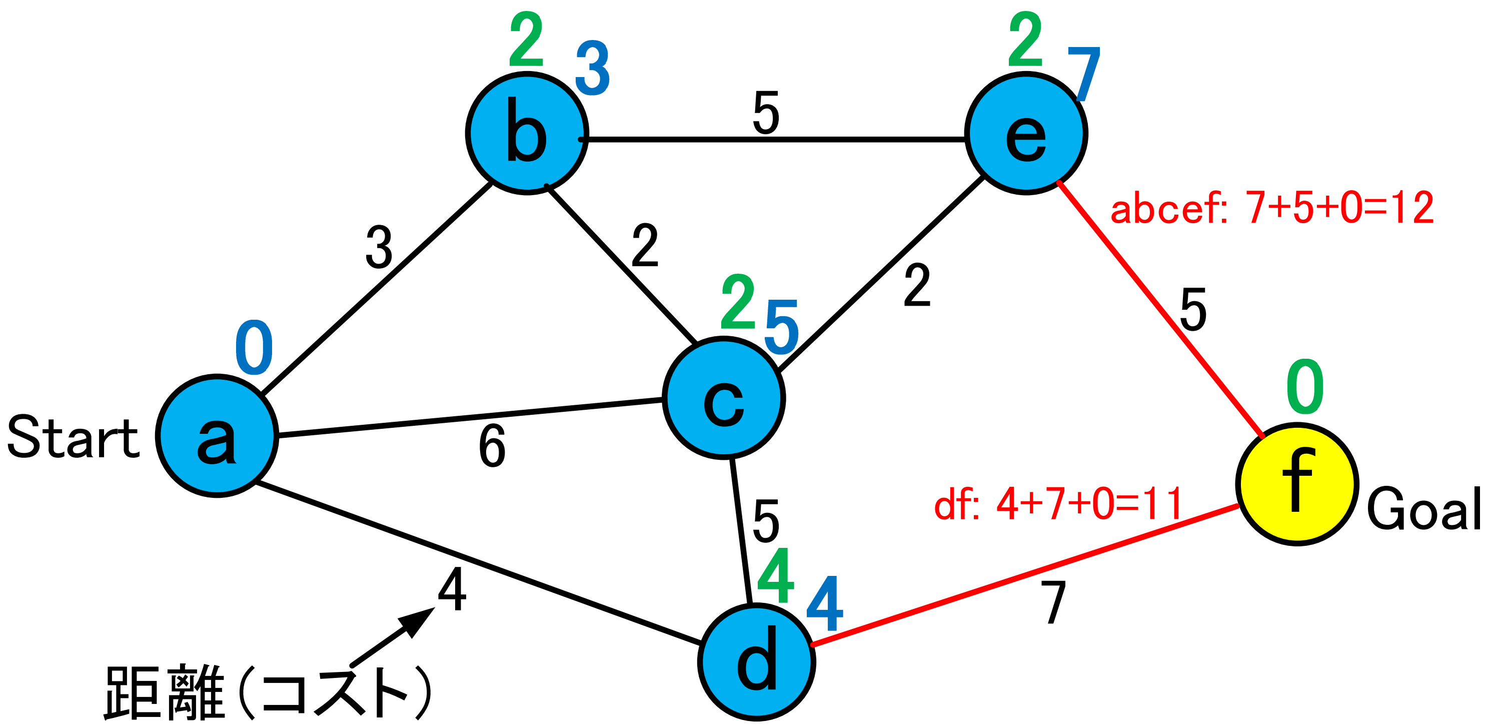


A*アルゴリズム　動作例 11/14



- Startからの最短経路が確定していないノード
- Startからの最短経路を確定中のノード
- Startからの最短経路が確定したノード
- 7 Startからの最短距離候補(未確定)
- 7 Startからの最短距離(確定済)
- 確定済ノードからのアーク
次期確定ノード決定に使用
- StartからGoalまでの
最短経路

A*アルゴリズム　動作例 12/14



Startからの最短経路が確定していないノード

Startからの最短経路を確定中のノード

Startからの最短経路が確定したノード

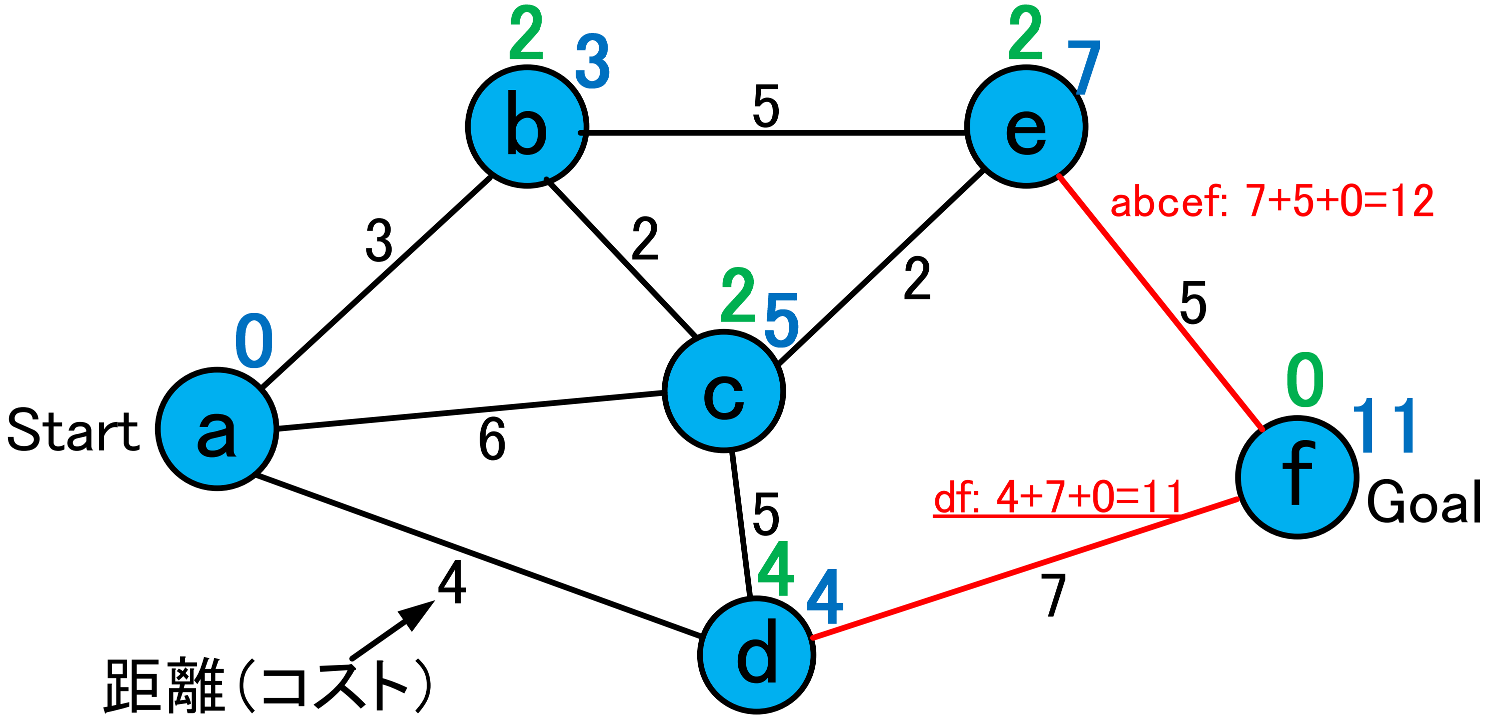
7 Startからの最短距離候補(未確定)

7 Startからの最短距離(確定済)

確定済ノードからのアーク
次期確定ノード決定に使用

StartからGoalまでの
最短経路

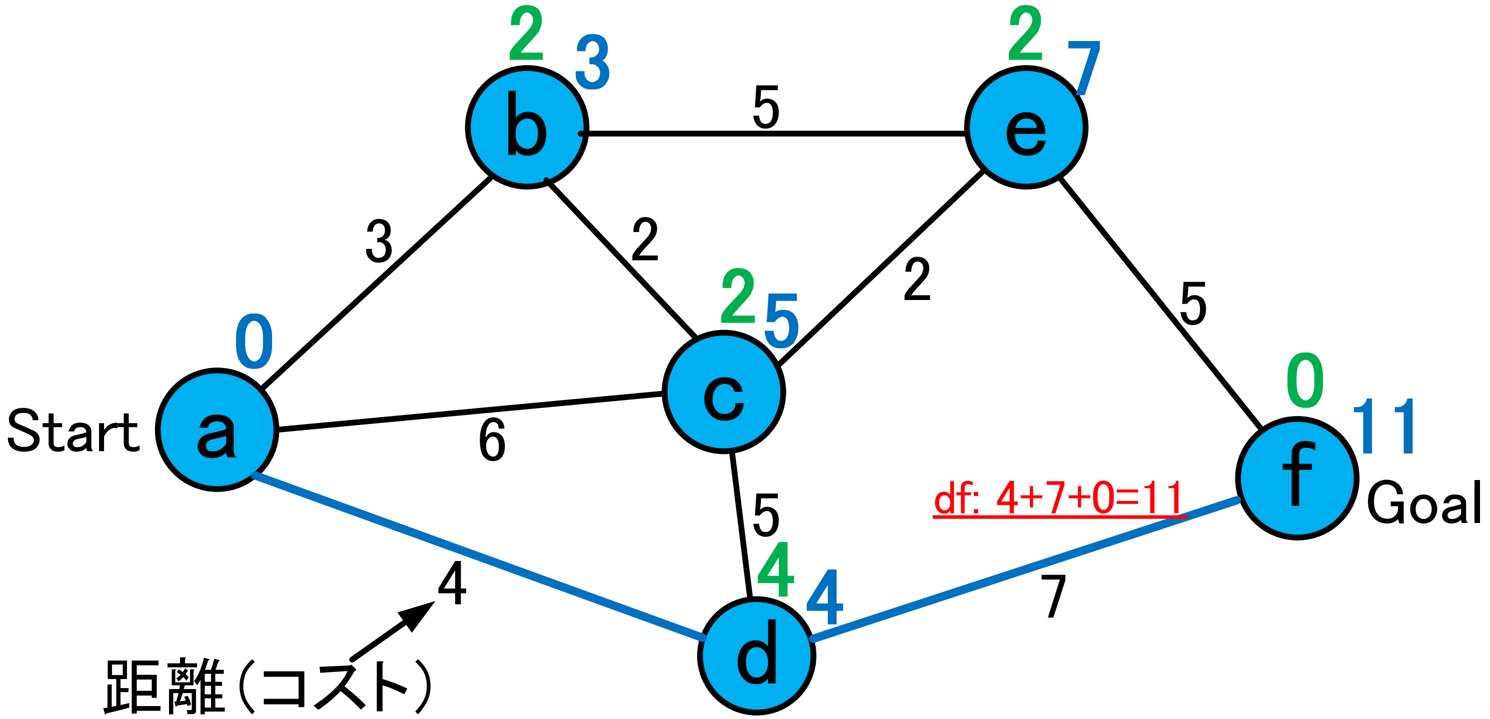
A*アルゴリズム　動作例 13/14



- Startからの最短経路が確定していないノード
- Startからの最短経路を確定中のノード
- Startからの最短経路が確定したノード
- 7 Startからの最短距離候補(未確定)
- 7 Startからの最短距離(確定済)

- 赤線：確定済ノードからのアーク 次期確定ノード決定に使用
- 青線：StartからGoalまでの最短経路

A*アルゴリズム　動作例 14/14



Startからの最短経路が確定していないノード

Startからの最短経路を確定中のノード

Startからの最短経路が確定したノード

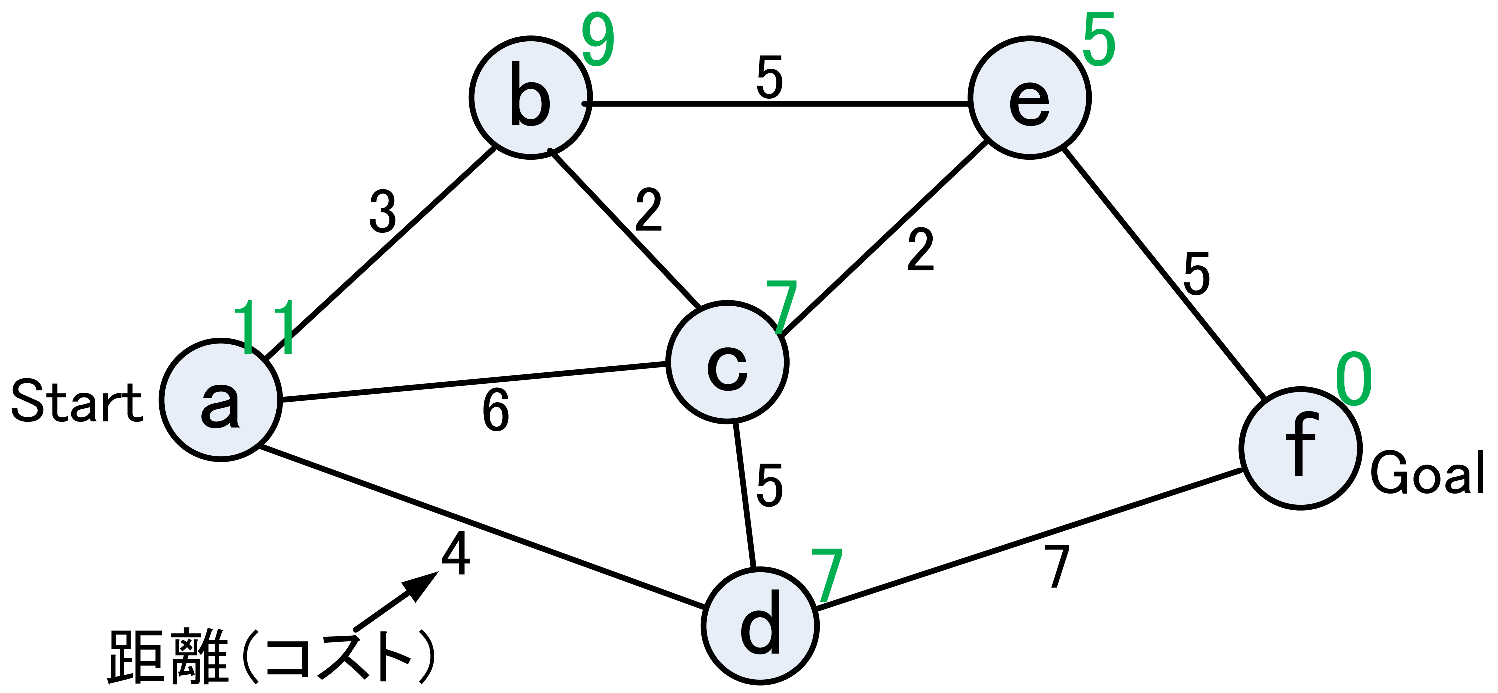
7 Startからの最短距離候補(未確定)

7 Startからの最短距離(確定済)

確定済ノードからのアーク 次期確定ノード決定に使用

StartからGoalまでの最短経路

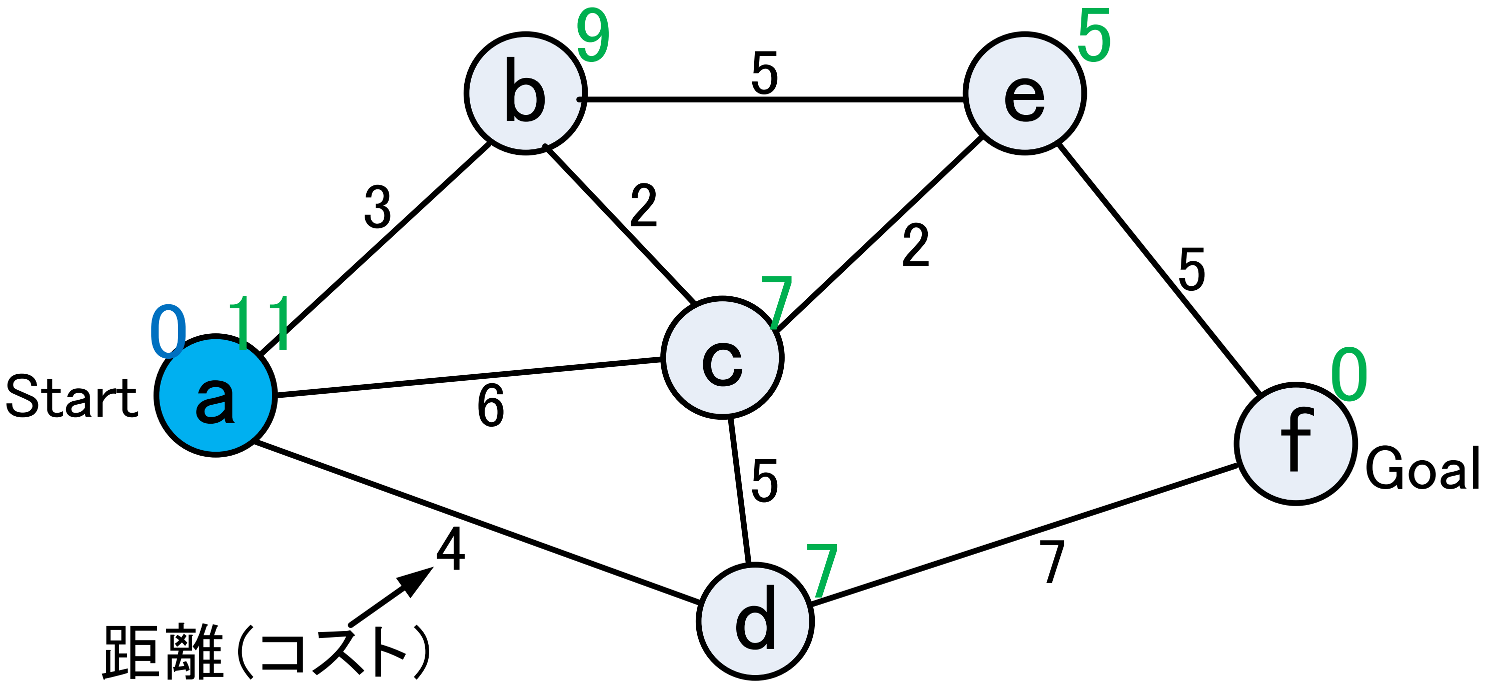
A*アルゴリズムその2 (最良のh(n))　1/6



- Startからの最短経路が確定していないノード
- Startからの最短経路を確定中のノード
- Startからの最短経路が確定したノード
- 7 Startからの最短距離候補（未確定）
- 7 Startからの最短距離（確定済）

- 確定済ノードからのアーク 次期確定ノード決定に使用
- StartからGoalまでの最短経路

A*アルゴリズム　その2　2/6

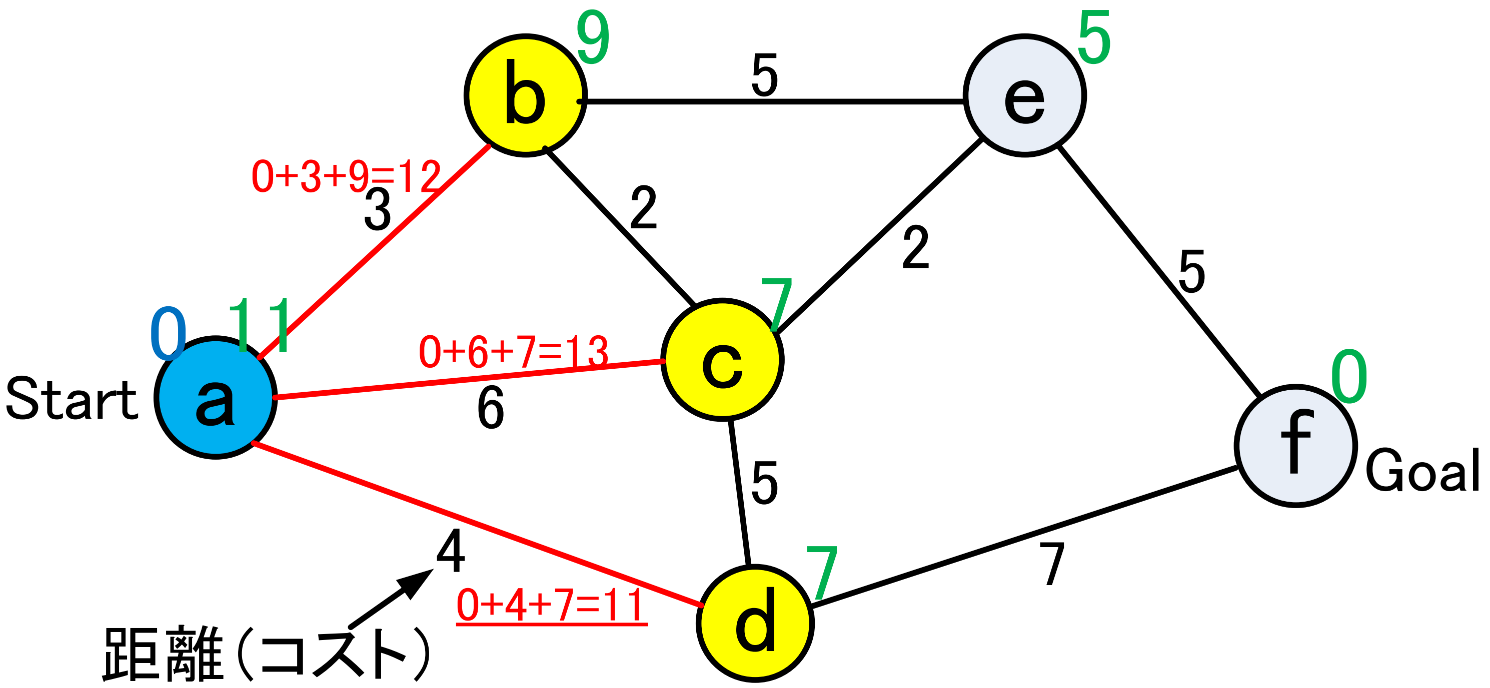


- Startからの最短経路が確定していないノード
- Startからの最短経路を確定中のノード
- Startからの最短経路が確定したノード
- 7　Startからの最短距離候補（未確定）
- 7　Startからの最短距離（確定済）

確定済ノードからのアーク
次期確定ノード決定に使用

StartからGoalまでの
最短経路

A*アルゴリズム　その2　3/6



Startからの最短経路が確定していないノード

Startからの最短経路を確定中のノード

Startからの最短経路が確定したノード

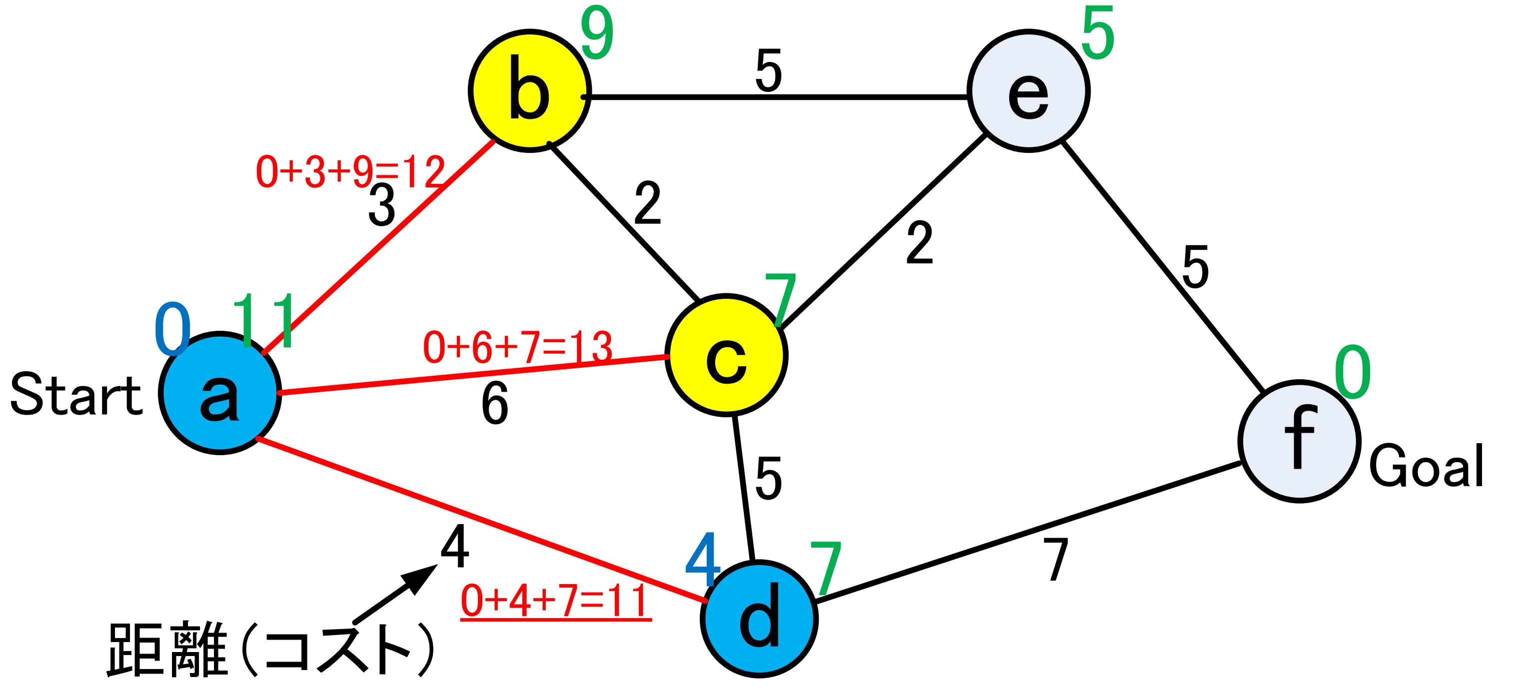
7 Startからの最短距離候補(未確定)

7 Startからの最短距離(確定済)

確定済ノードからのアーク
次期確定ノード決定に使用

StartからGoalまでの
最短経路

A*アルゴリズム　その2　4/6



Startからの最短経路が確定していないノード

Startからの最短経路を確定中のノード

Startからの最短経路が確定したノード

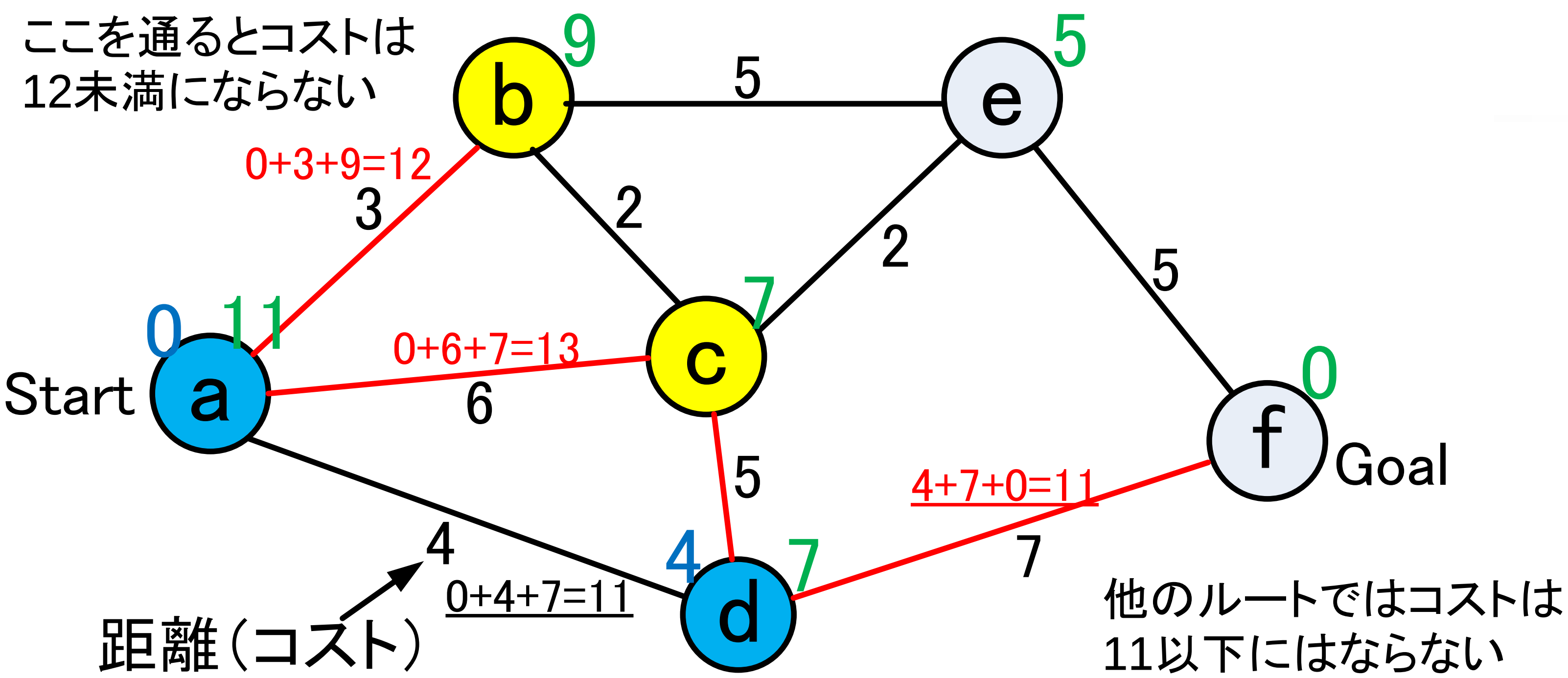
7　Startからの最短距離候補（未確定）

7　Startからの最短距離（確定済）

確定済ノードからのアーク
次期確定ノード決定に使用

StartからGoalまでの
最短経路

A*アルゴリズム　その2　5/6



Startからの最短経路が確定していないノード

Startからの最短経路を確定中のノード

Startからの最短経路が確定したノード

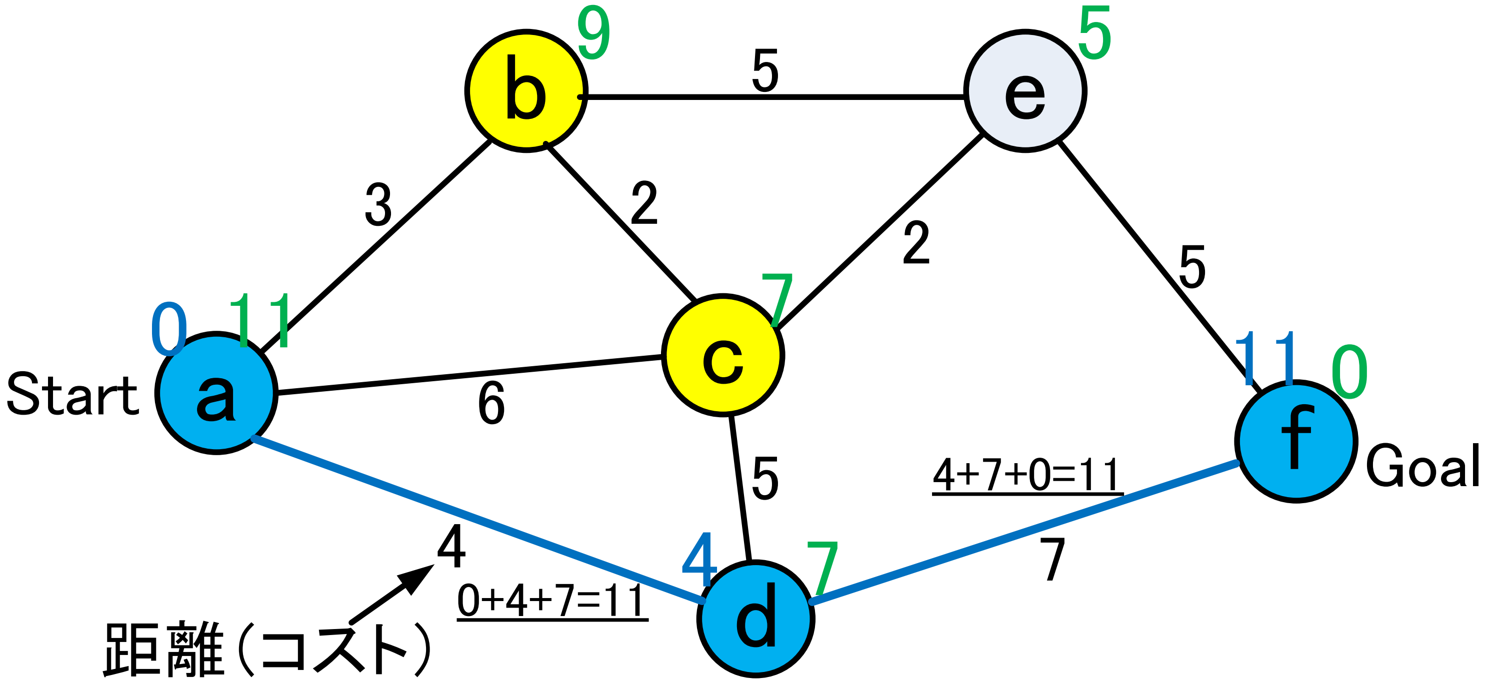
7　Startからの最短距離候補(未確定)

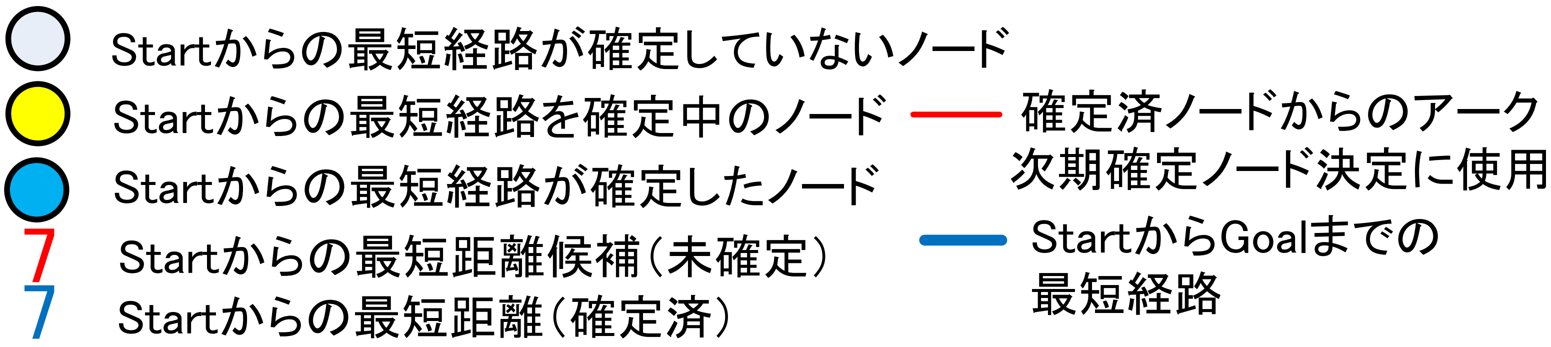
7　Startからの最短距離(確定済)

確定済ノードからのアーク
次期確定ノード決定に使用

StartからGoalまでの
最短経路

A*アルゴリズム　その2　6/6





Aアルゴリズム，A*アルゴリズム ダイクストラ法

- f(n)=g(n)+h(n)
  - n:節点番号，
  - f(n):節点nを通りスタートからゴールまでの距離
  - g(n):スタートから節点nまでの距離
  - h(n):節点nからゴールまでの距離
- Aアルゴリズム
  - f*(n)=g(n)+h*(n)を利用してスタートからゴールまでの距離を調べる
    - f*(n): 節点nからゴールまでの距離の評価値
    - h*(n): 節点nからゴールまでの距離の評価値
- A*アルゴリズム
  - Aアルゴリズムのf*(n)=g(n)+h*(n)のh*(n)が0≦h*(n)≦h(n)の時
  - 最初に見つけたルートが最短ルートであることが保証されている
- ダイクストラ法
  - Aアルゴリズムのf*(n)=g(n)+h*(n)のh*(n)が0の時
  - つまりf*(n)=g(n)

12月10日は中間試験

- 試験範囲は今日の授業まで
  - スタック
  - 文脈自由文法
  - 構文解析
  - CYK法
  - (トップダウンチャート法)
  - ダイクストラ法
  - DPマッチング
  - A*アルゴリズム
- 試験問題の傾向はWebに掲載している去年の試験問題を見てください