

知能科学：グラフと経路計画

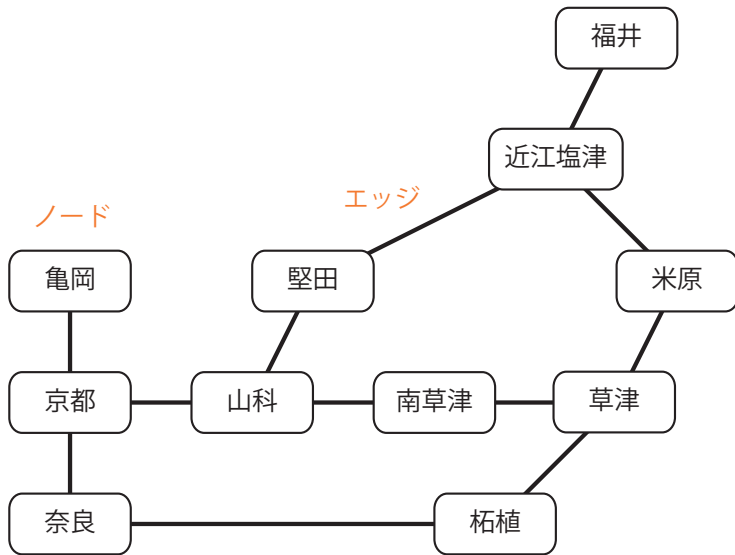
平井 慎一

立命館大学 ロボティクス学科

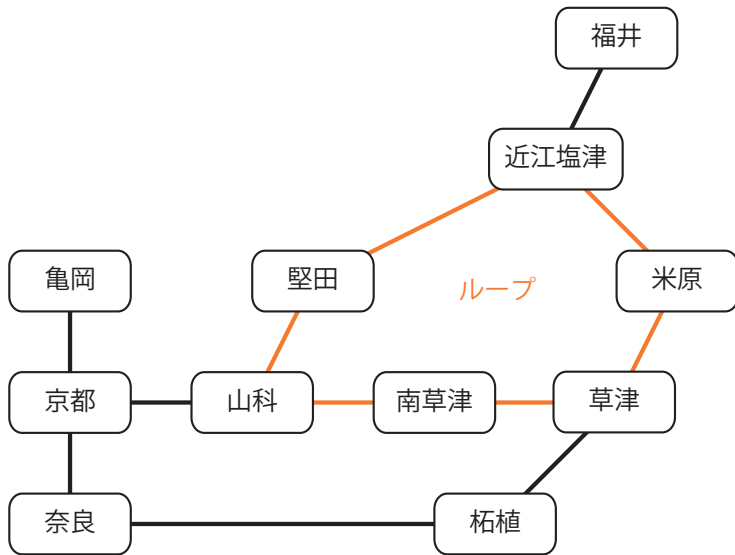
講義の流れ

- 1 グラフ
- 2 最短経路問題
- 3 ダイクストラのアルゴリズム
- 4 最大フロー問題
- 5 ゲーム木
- 6 二人零和ゲーム
- 7 まとめ

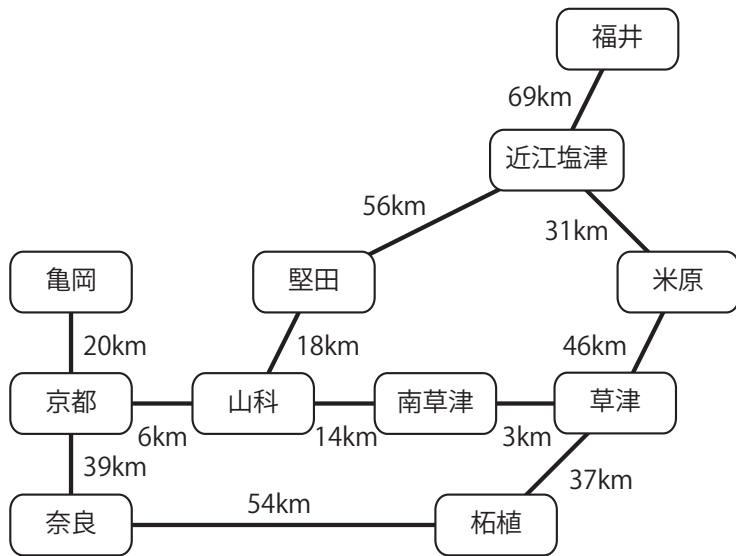
グラフ



グラフ



グラフ



MATLAB

```
names = {'南草津', '山科', '京都', ...  
         '堅田', '近江塩津', '米原', '草津', ...  
         '福井', '亀岡', '奈良', '柘植'};
```

```
snode = [ 1, 2, 2, 4, 5, 5, ...  
          6, 7, 7, 3, 3, 10 ];
```

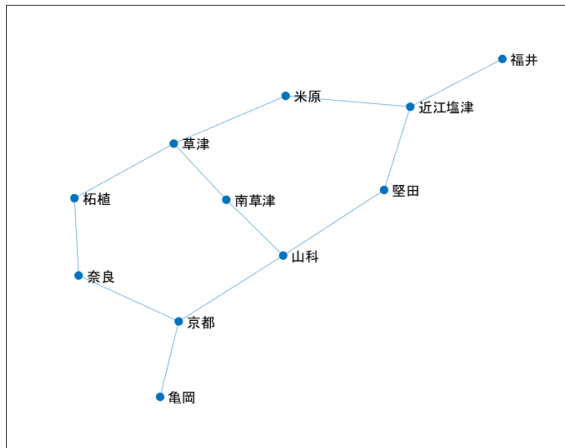
```
tnode = [ 2, 3, 4, 5, 6, 8, ...  
          7, 1, 11, 9, 10, 11 ];
```

```
dist = [ 14, 6, 18, 56, 31, 69, ...  
         46, 3, 37, 20, 39, 54 ];
```

```
g = graph(snode,tnode,dist,names);
```

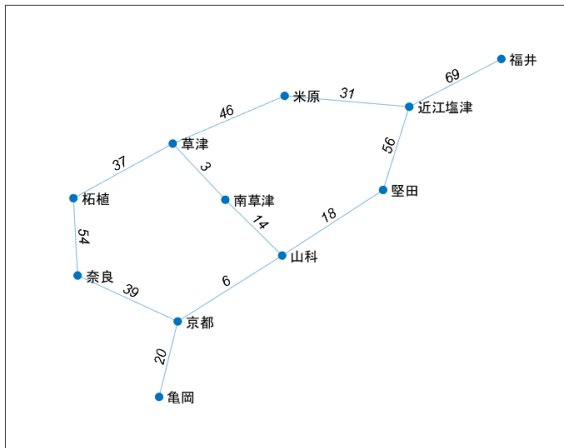
MATLAB

```
plot(g);
```



MATLAB

```
plot(g, 'EdgeLabel',g.Edges.Weight);
```

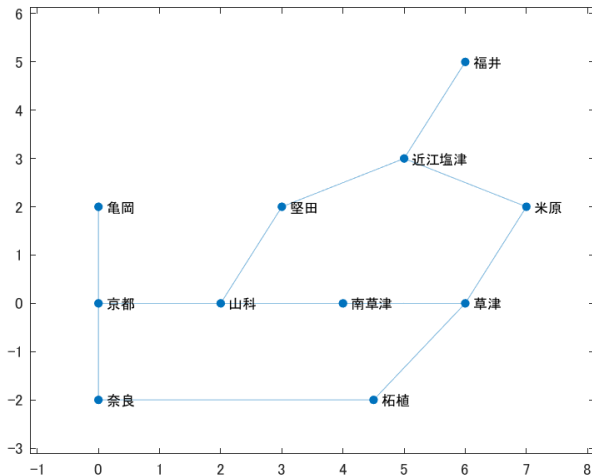


ノードの表示場所を指定

```
x = [ 4, 2, 0, 3, 5, 7, 6, 6, 0, 0, 4.5 ];  
y = [ 0, 0, 0, 2, 3, 2, 0, 5, 2, -2, -2 ];
```

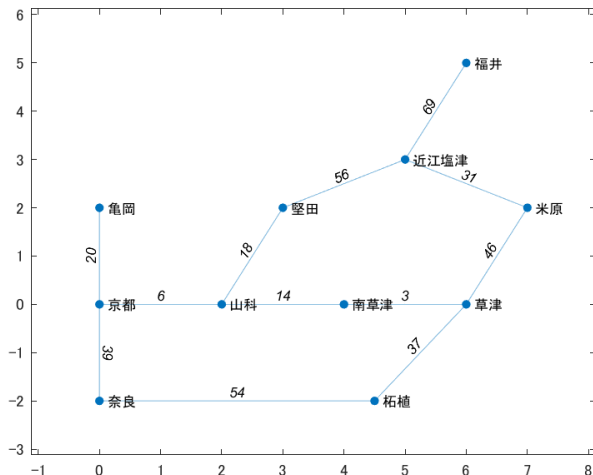
MATLAB

```
plot(g, 'XData',x,'YData',y);
```



MATLAB

```
plot(g, 'XData',x,'YData',y, 'EdgeLabel',g.Edges.Weight);
```



最短経路問題

奈良から福井へ至る最短経路 (shortest path)

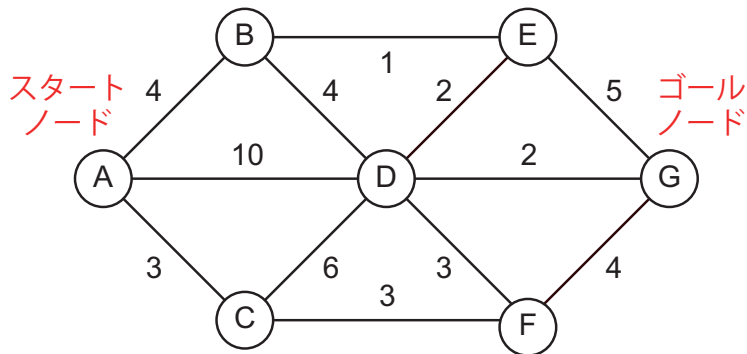
```
>> shortestpath(g, '奈良', '福井')
```

```
ans =
```

1 × 6 の cell 配列

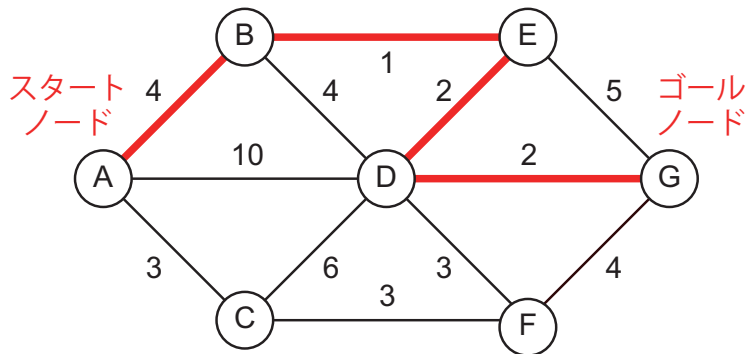
```
      '奈良',      '京都',      '山科',      '堅田',      '近江塩津',  
福井',
```

最短経路問題



ノード A から ノード G へ至る最短の経路

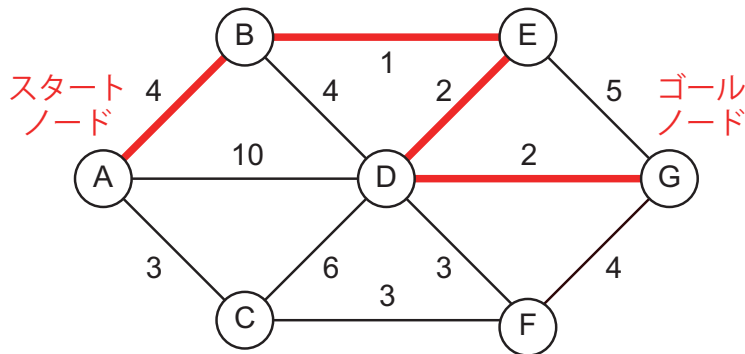
最短経路問題



最短経路 $A \rightarrow B \rightarrow E \rightarrow D \rightarrow G$

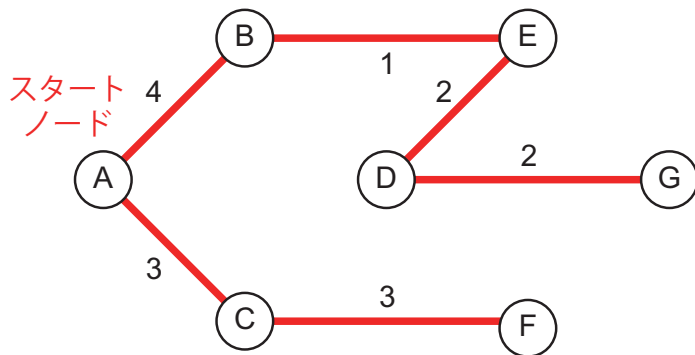
最短距離 $= 4 + 1 + 2 + 2 = 9$

最短経路問題



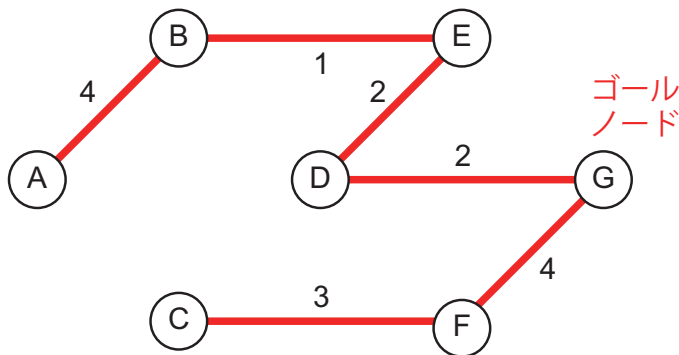
ノード A からノード B, E, D への最短経路
ノード B, E, D からノード G への最短経路
⇒ 最短経路の一部

最短経路木



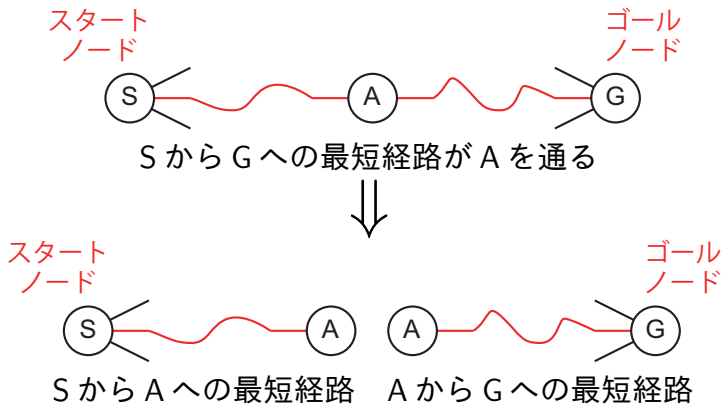
ノード A から他のノードへの最短経路
⇒ 最短経路木 (ループのないグラフ)

最短経路木



他のノードからノード G への最短経路
⇒ 最短経路木 (ループのないグラフ)

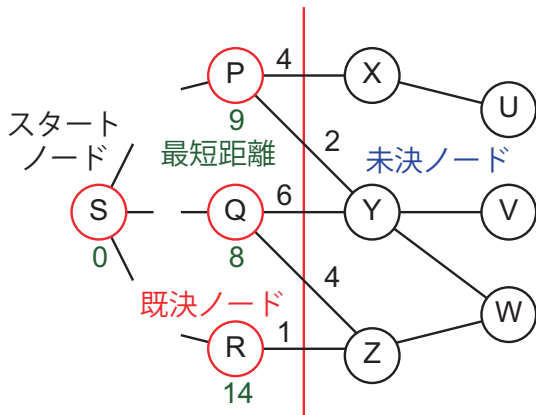
ポントリヤギンの最適性原理



ダイクストラ Edsger Dijkstra (1930 2002)

- オランダの情報工学者
- ダイクストラのアルゴリズム (1959)
- 構造化プログラミングの提唱
- 1972 年チューリング賞

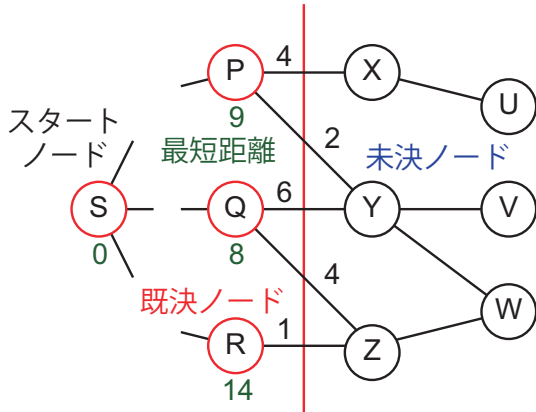
ダイクストラのアルゴリズム



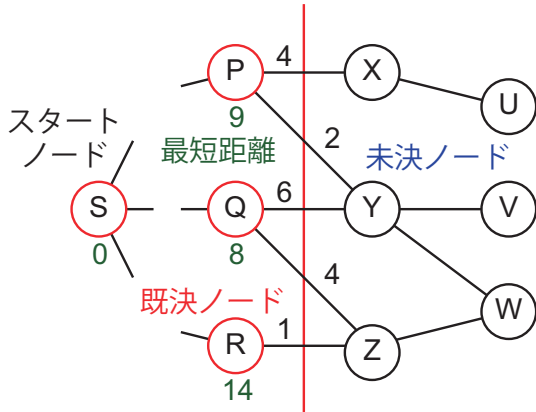
決ったノード 最短距離
が決った
決ったノードを順次求める

未決ノード 最短距離
が未決

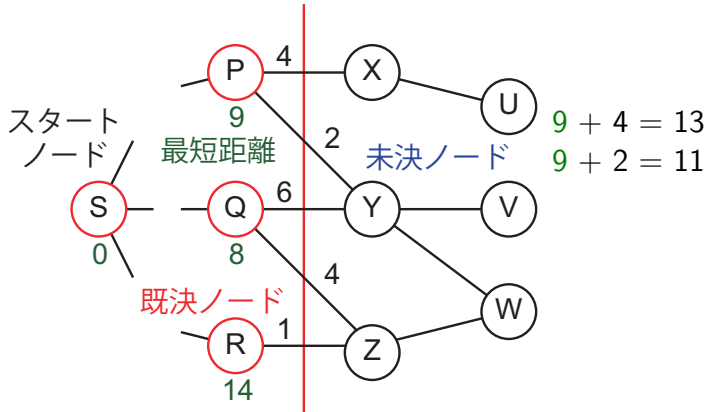
ダイクストラのアルゴリズム



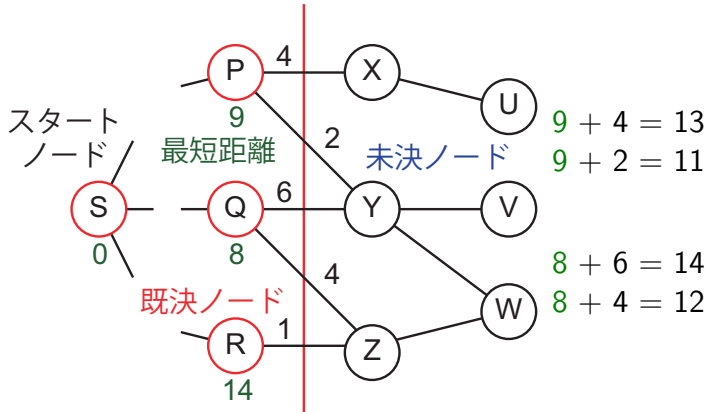
ダイクストラのアルゴリズム



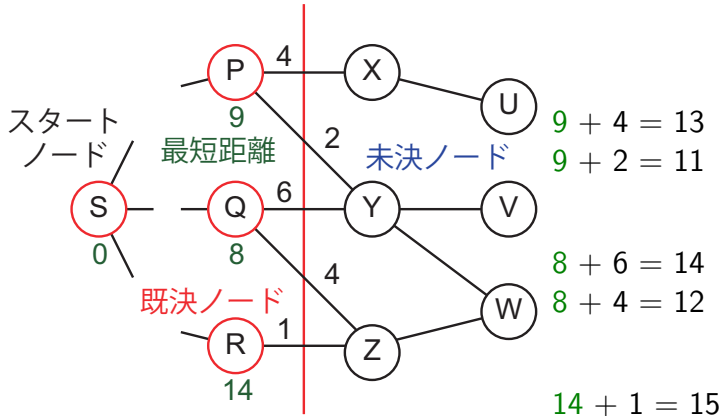
ダイクストラのアルゴリズム



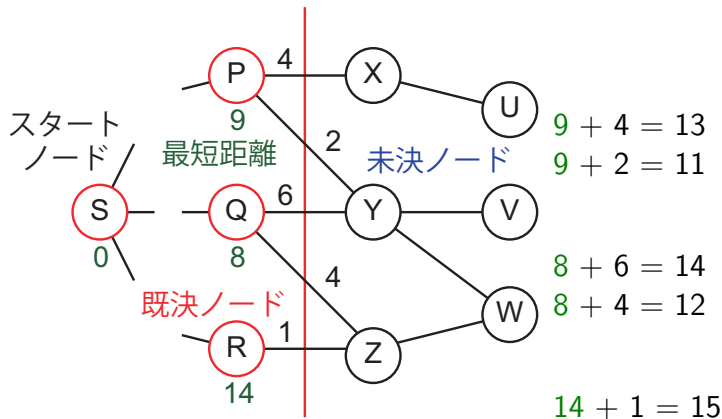
ダイクストラのアルゴリズム



ダイクストラのアルゴリズム



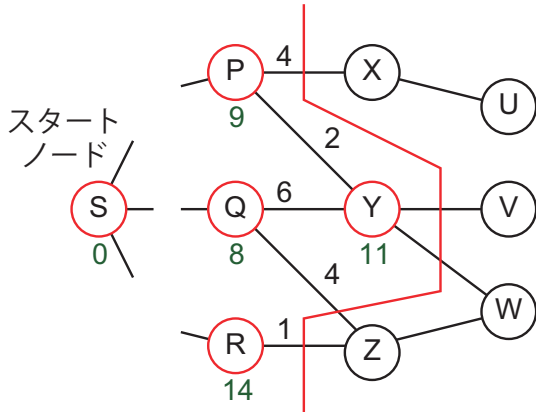
ダイクストラのアルゴリズム



$9 + 2 = 11$ が最小

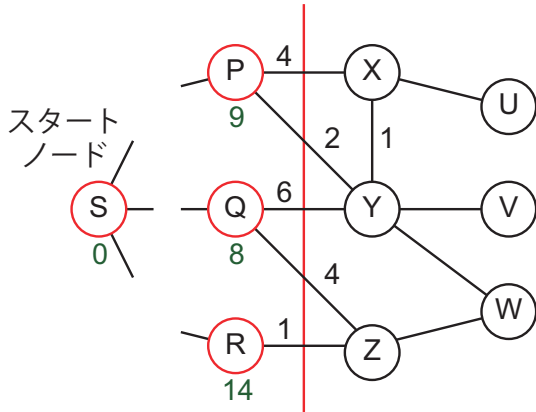
$S \rightarrow \dots \rightarrow P \rightarrow Y$ は最短経路

ダイクストラのアルゴリズム



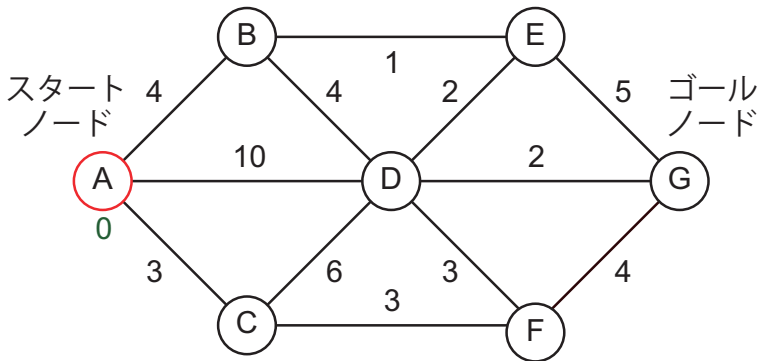
ノード Y の最短距離 = 11
ノード Y を既決とする

ダイクストラのアルゴリズム



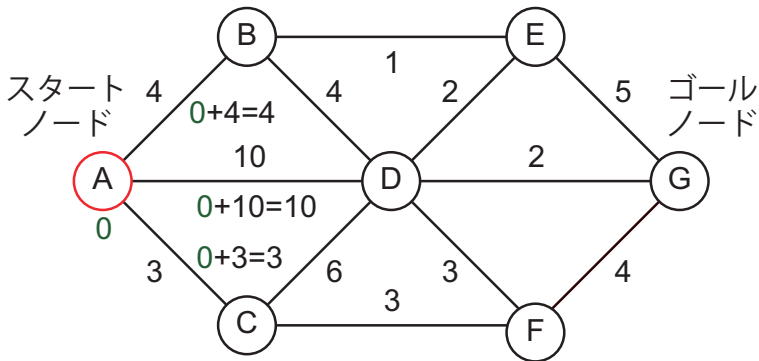
P から Y を通り X へ至る経路が最短の可能性があるので $S \rightarrow \dots \rightarrow P \rightarrow X$ は最短とは言えない

例



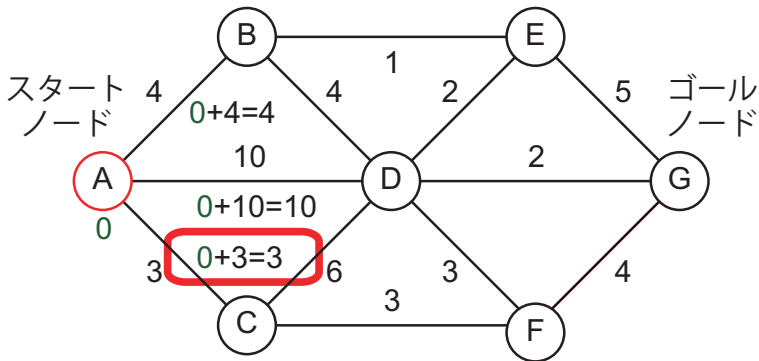
初期化: スタートノードを既決にする (最短距離 0)

例



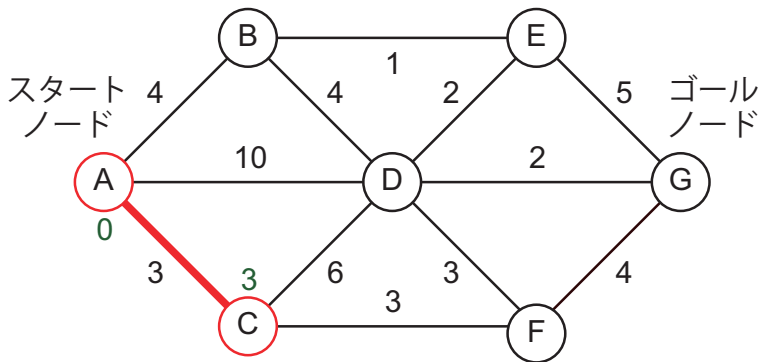
既決ノードを通り未決ノードへ至る経路の距離を計算

例



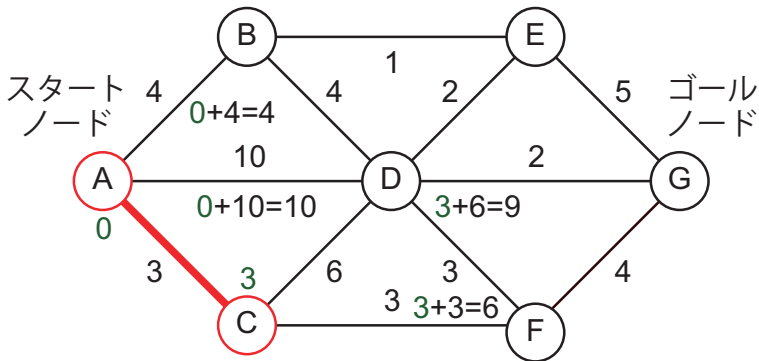
最小の経路を選ぶ

例



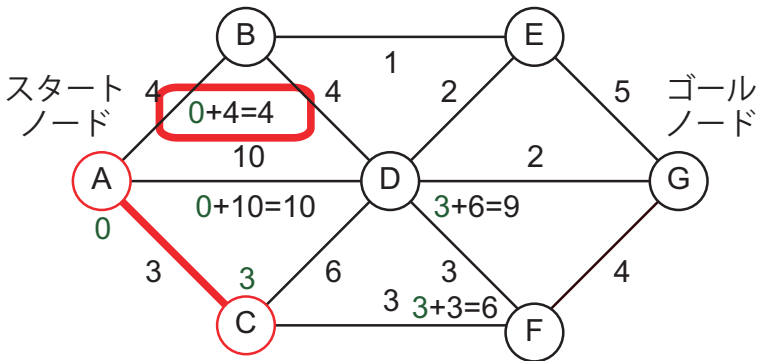
ノード C を既決にする

例



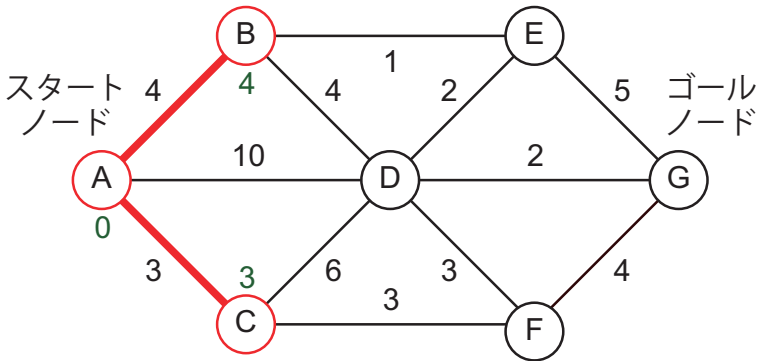
既決ノードを通り未決ノードへ至る経路の距離を計算

例



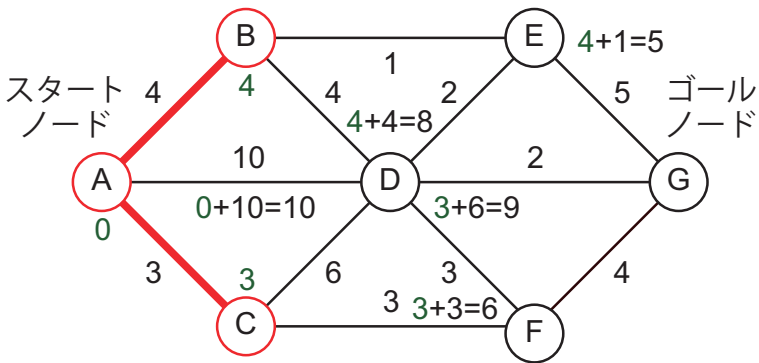
最小の経路を選ぶ

例



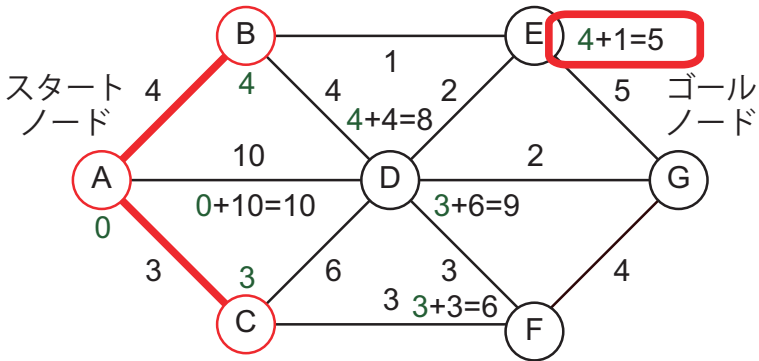
ノード B を既決にする

例



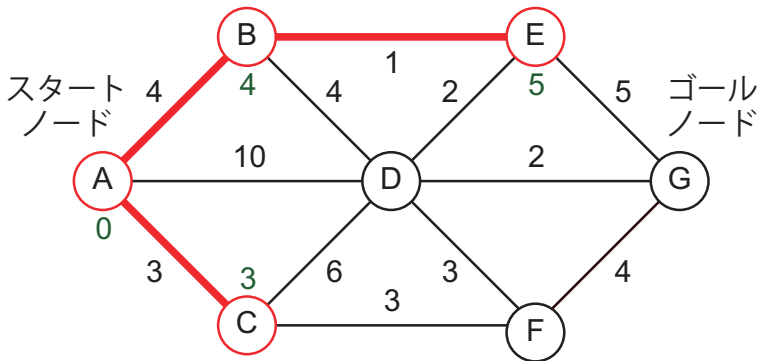
既決ノードを通り未決ノードへ至る経路の距離を計算

例

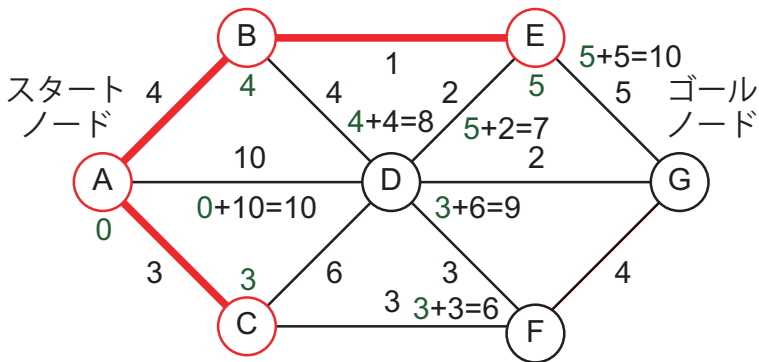


最小の経路を選ぶ

例

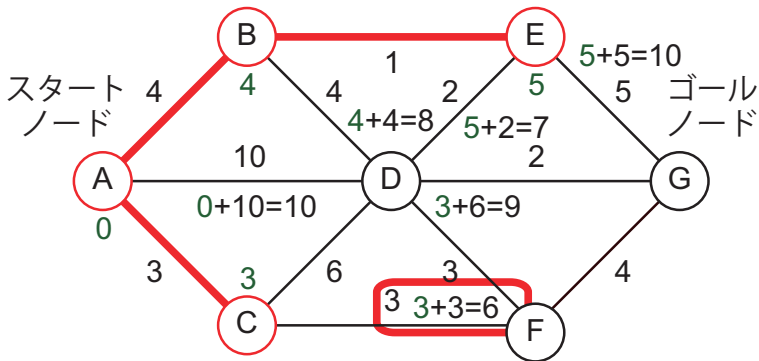


例



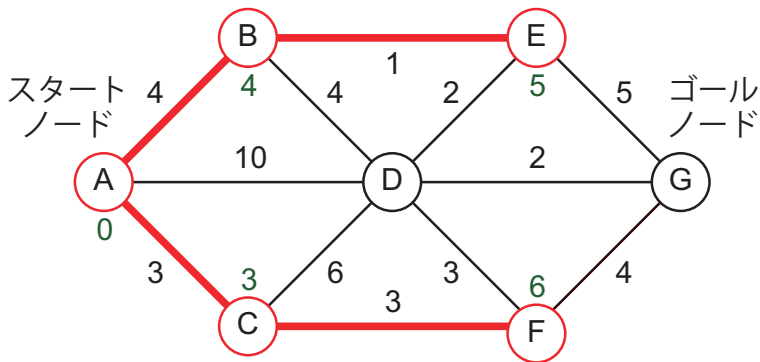
既決ノードを通り未決ノードへ至る経路の距離を計算

例



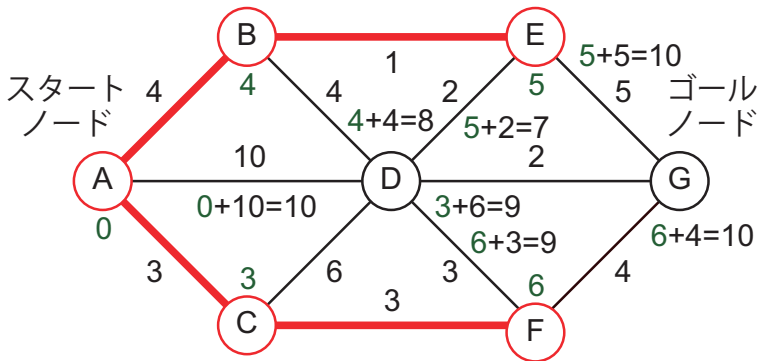
最小の経路を選ぶ

例



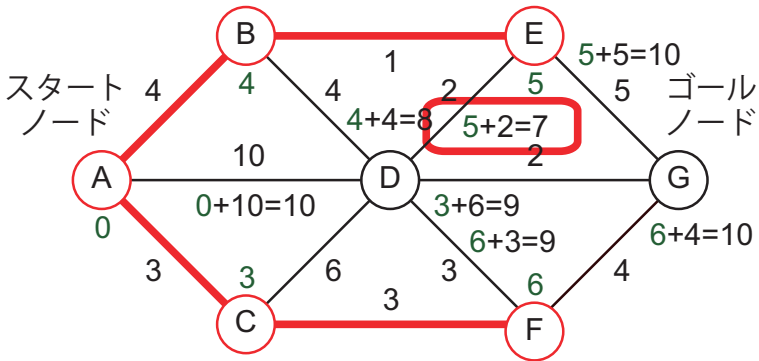
ノード F を既決にする

例



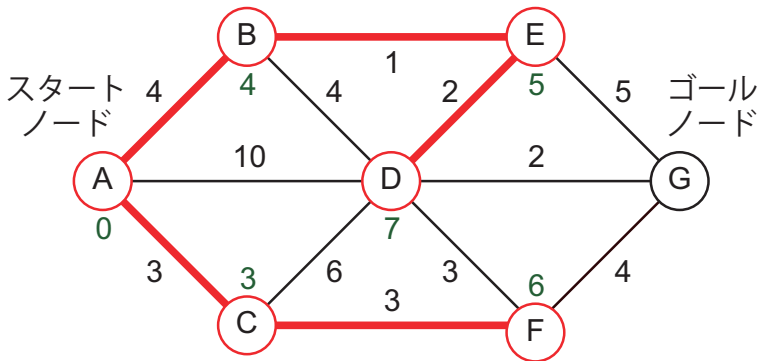
既決ノードを通り未決ノードへ至る経路の距離を計算

例

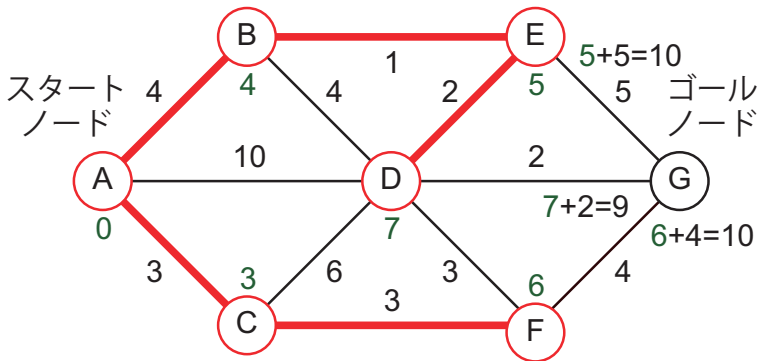


最小の経路を選ぶ

例

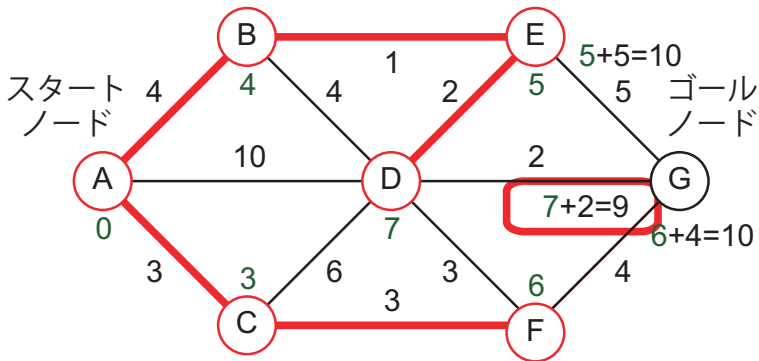


例



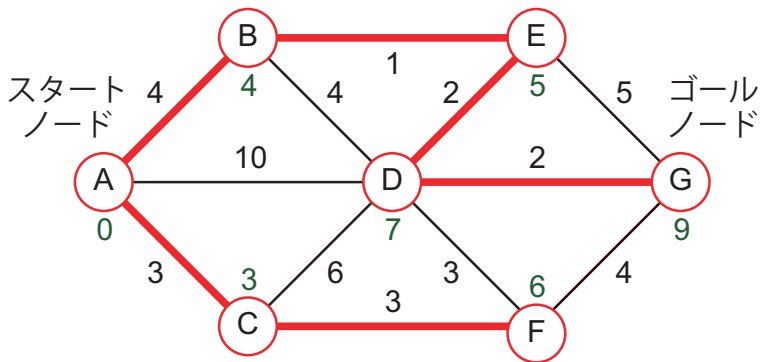
既決ノードを通り未決ノードへ至る経路の距離を計算

例



最小の経路を選ぶ

例



ゴールノードを既決にする
停止

ダイクストラのアルゴリズム

距離の計算の重複を避けるためにラベルを導入する

ラベル

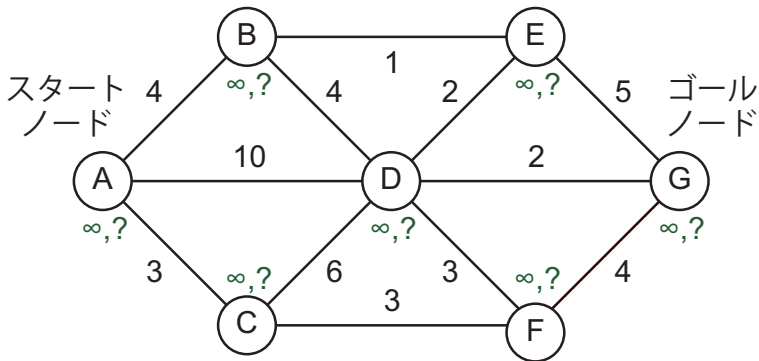
各ノードに付ける

距離, ノード の対

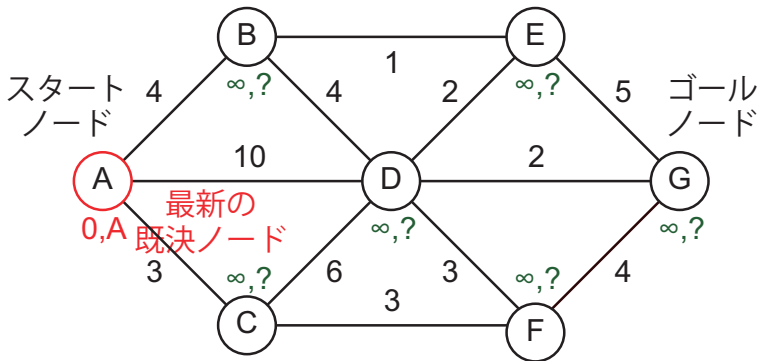
距離 これまでに判った最短距離
 さらに短い距離が得られた場合は更新する

ノード どのノードからの経路か
 (最短経路木における親ノード)

例

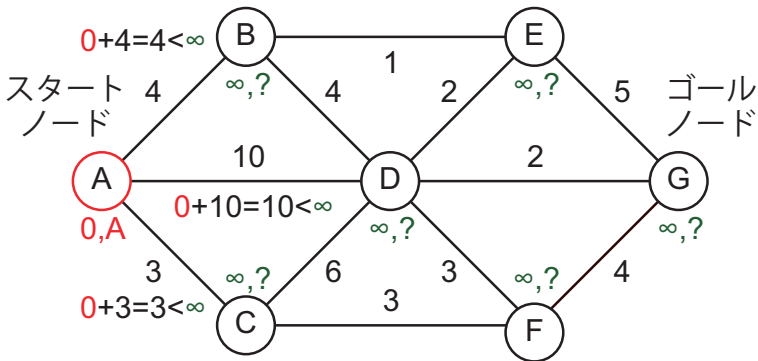


例



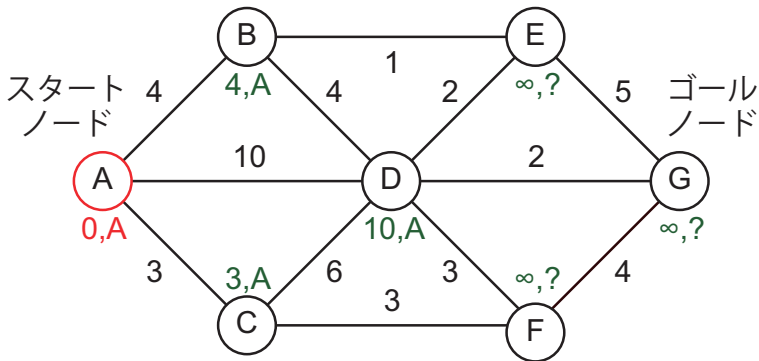
スタートノード A にラベル 0, A を与える
ノード A を既決にする

例

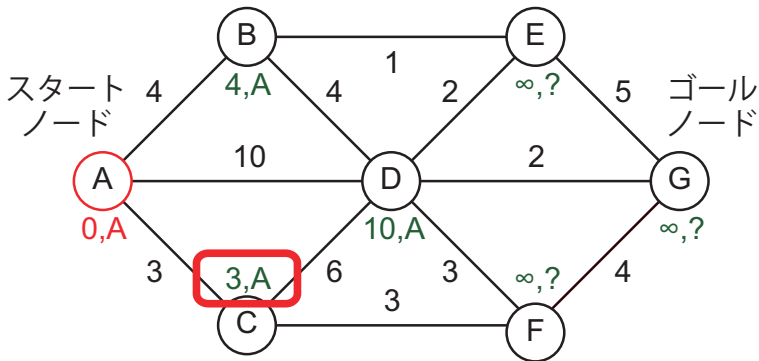


最新の既決ノードから未決ノードへの距離を計算
未決ノードのラベルの距離と比較

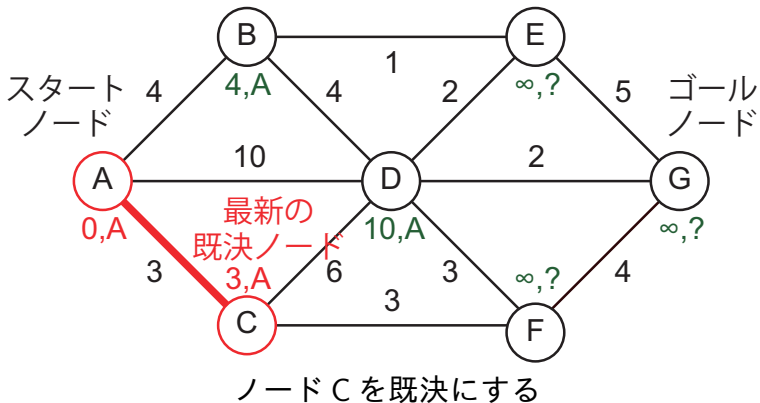
例



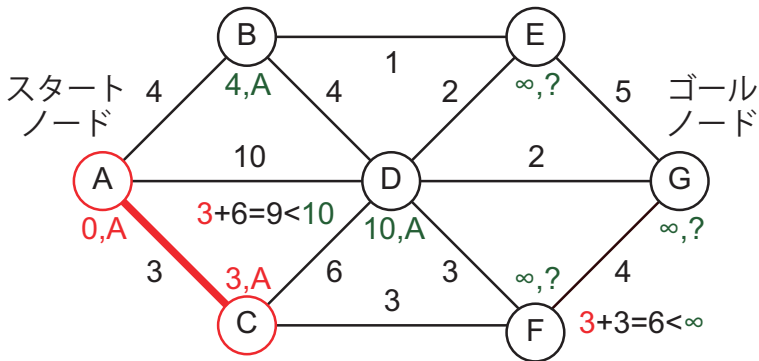
例



例

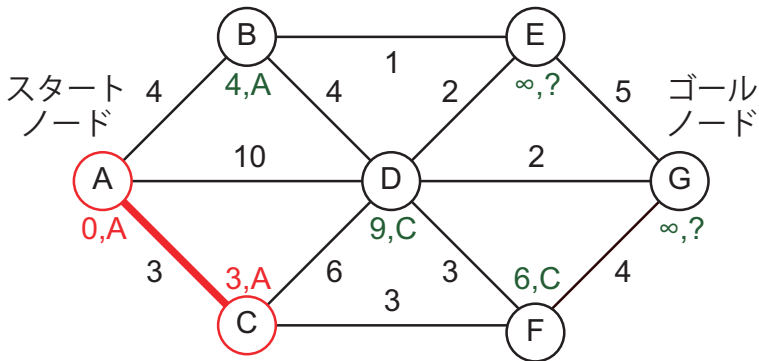


例

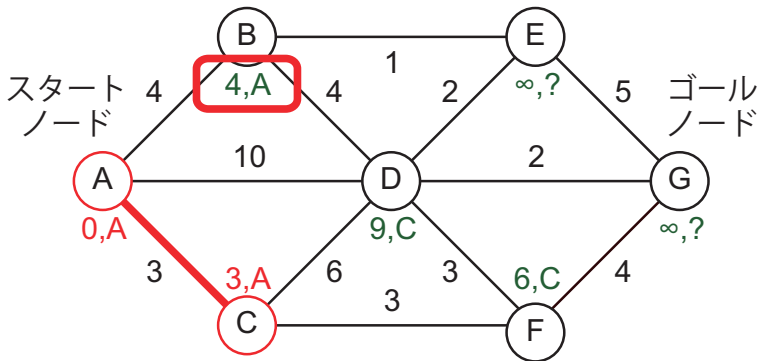


最新の既決ノードから未決ノードへの距離を計算
未決ノードのラベルの距離と比較

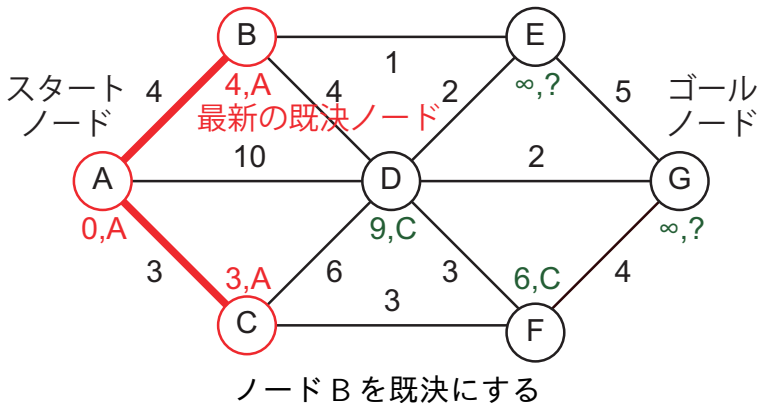
例



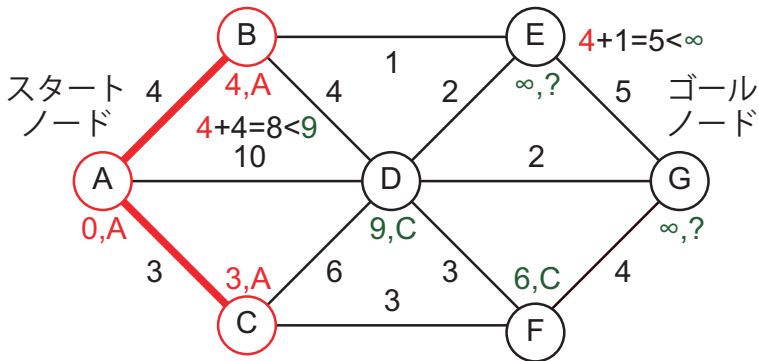
例



例

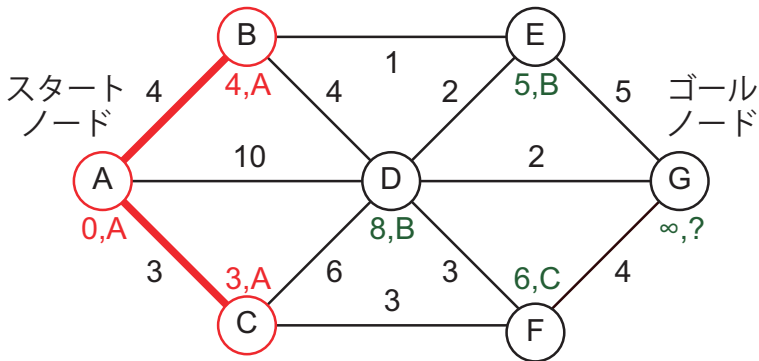


例



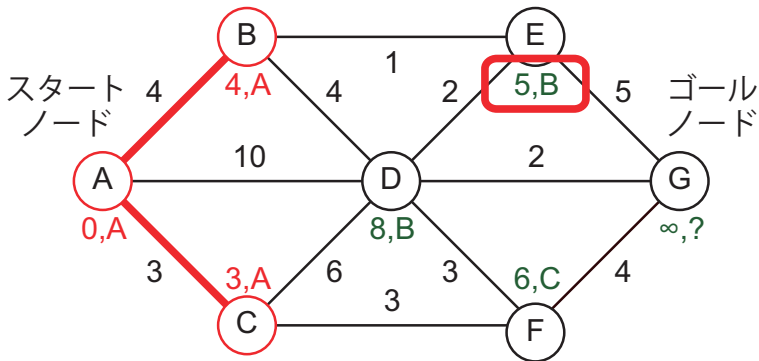
最新の既決ノードから未決ノードへの距離を計算
未決ノードのラベルの距離と比較

例

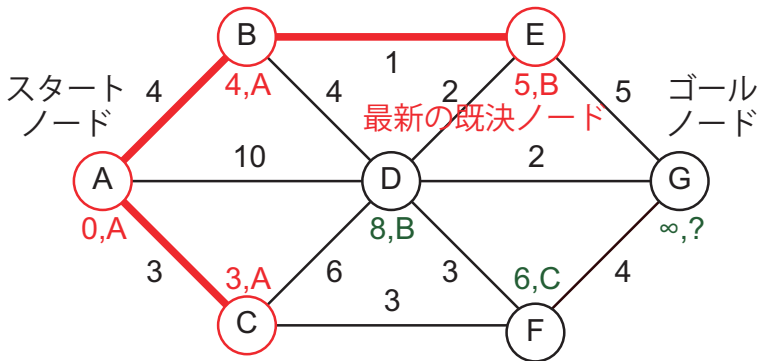


計算した距離 < ラベルの距離ならば，ラベルを更新

例

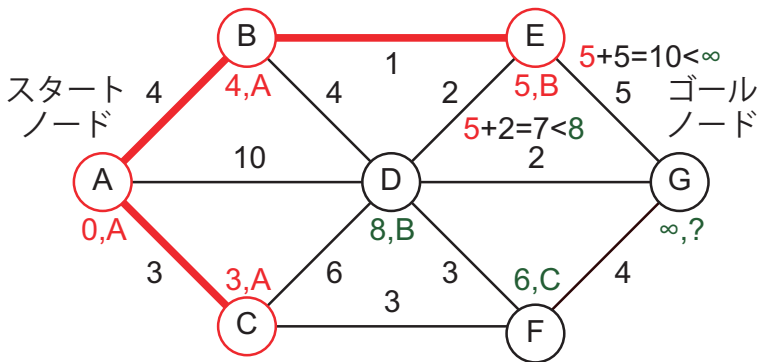


例



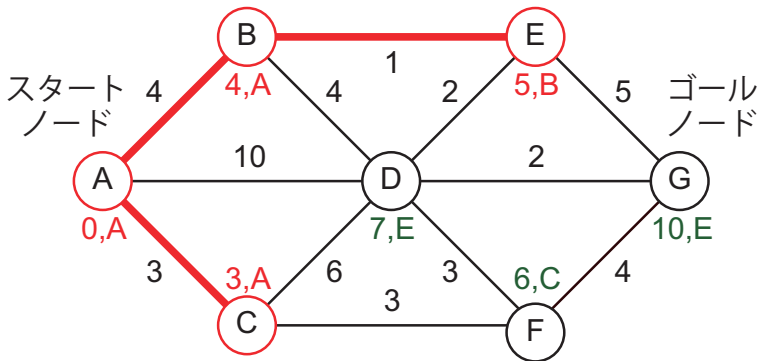
ノード E を既決にする

例



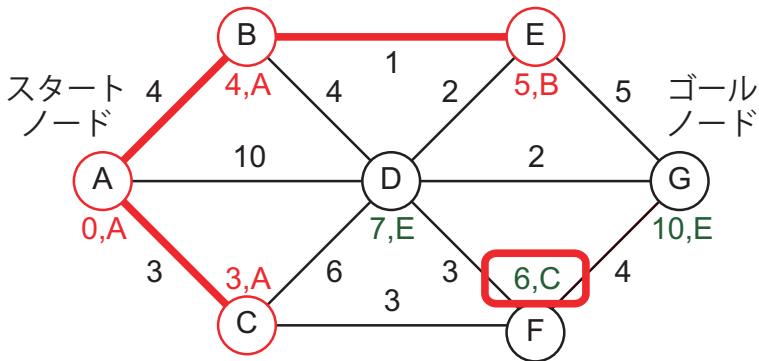
最新の既決ノードから未決ノードへの距離を計算
未決ノードのラベルの距離と比較

例

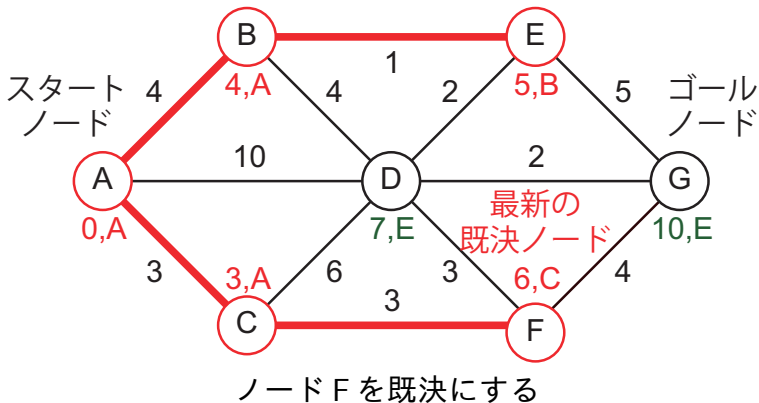


計算した距離 < ラベルの距離ならば，ラベルを更新

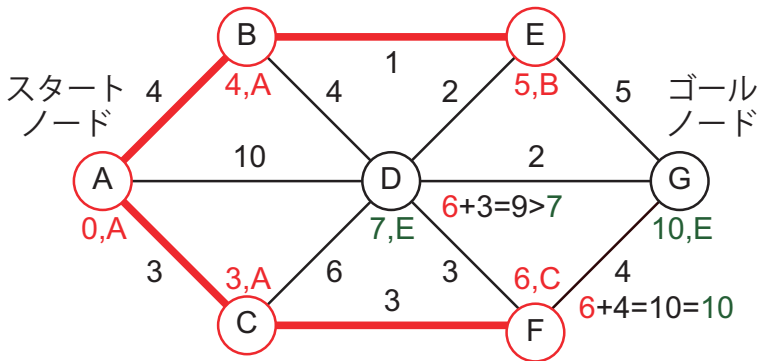
例



例

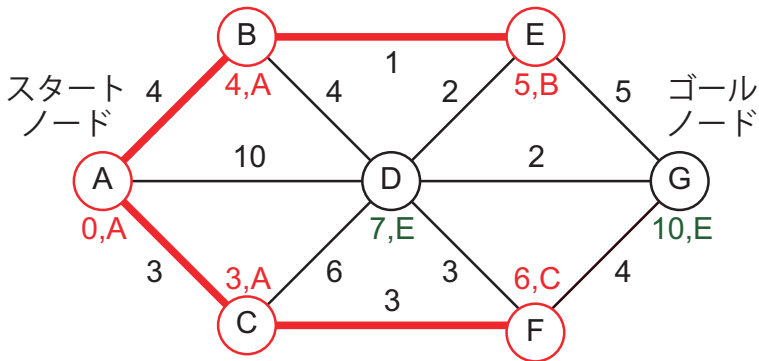


例

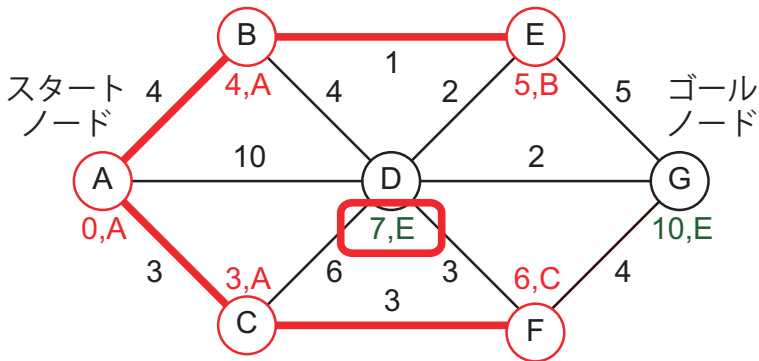


最新の既決ノードから未決ノードへの距離を計算
未決ノードのラベルの距離と比較

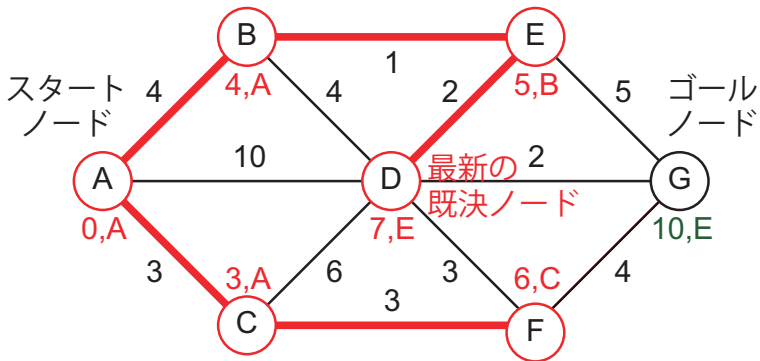
例



例

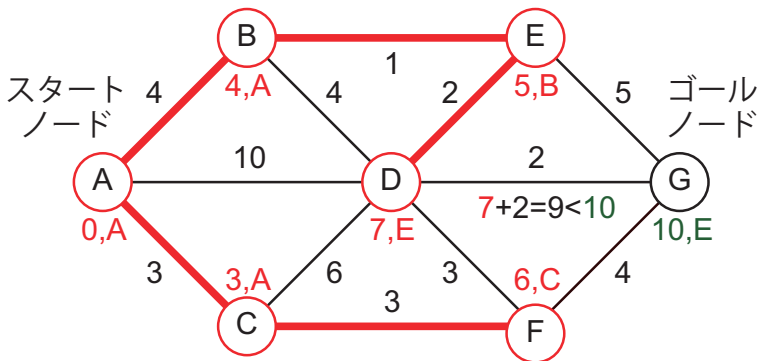


例



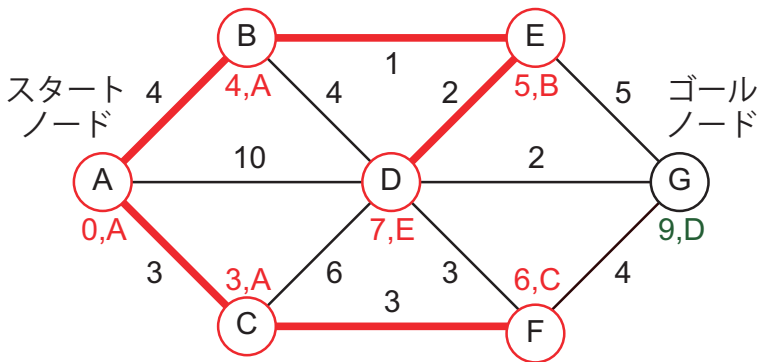
ノード D を既決にする

例

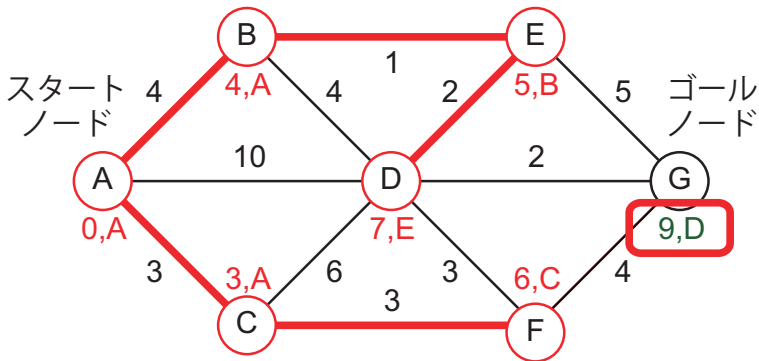


最新の既決ノードから未決ノードへの距離を計算
未決ノードのラベルの距離と比較

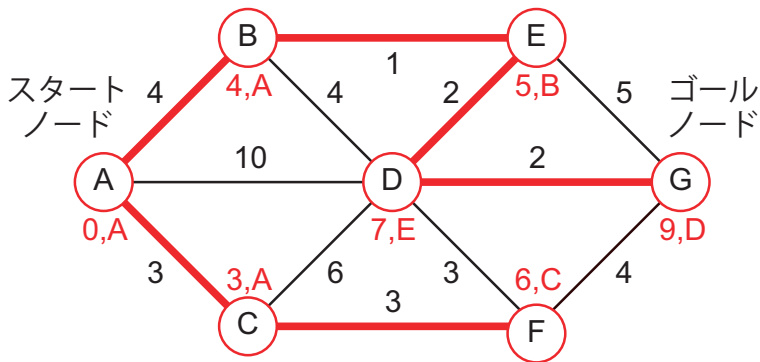
例



例

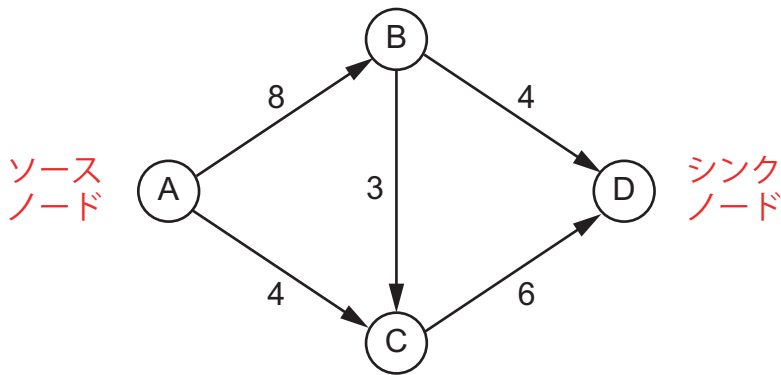


例



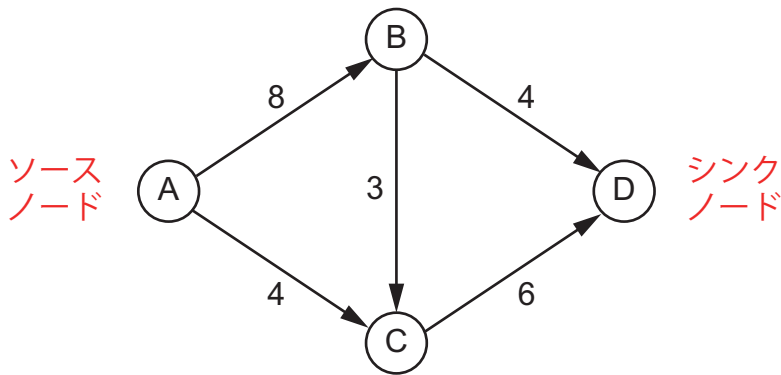
ノード G を既決にする
停止

最大フロー問題



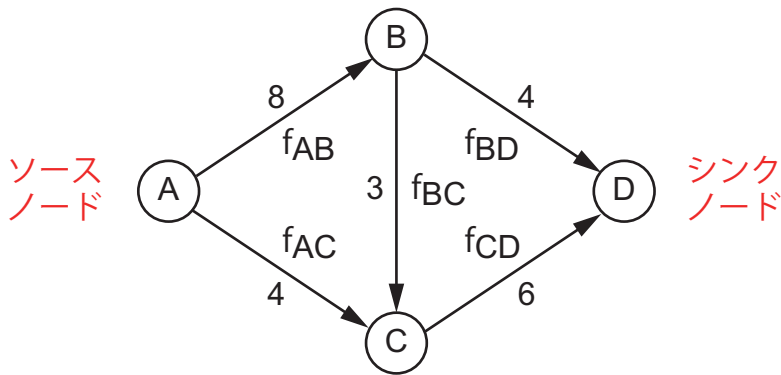
ソースノードからシンクノードに流れが生じる。
エッジに沿って一方向に流れる（有向グラフ）。

最大フロー問題



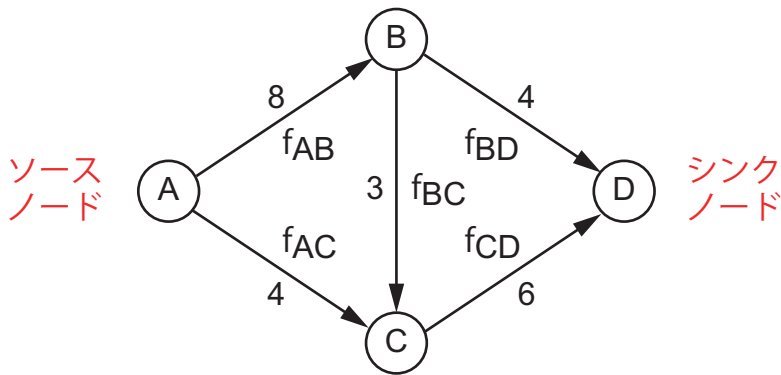
各エッジには流量の上限がある（容量）．
ソースノードからシンクノードへの最大流量を求める．
（電力，水道，通信，…）

最大フロー問題



ノード P から Q への流量を f_{PQ} と表す.

最大フロー問題

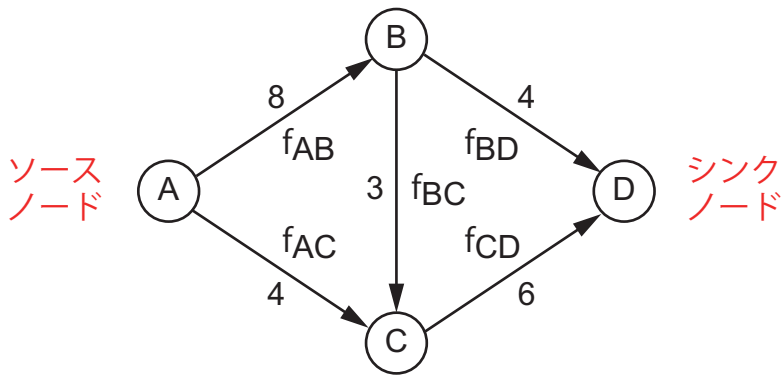


途中のノードでは，流入量と流出量が等しい．

ノード B $f_{AB} = f_{BC} + f_{BD}$

ノード C $f_{AC} + f_{BC} = f_{CD}$

最大フロー問題



容量制限

$$0 \leq f_{AB} \leq 8$$

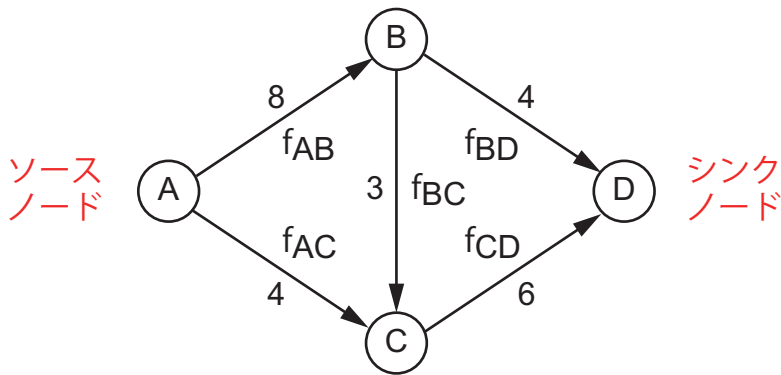
$$0 \leq f_{AC} \leq 4$$

$$0 \leq f_{BC} \leq 3$$

$$0 \leq f_{BD} \leq 4$$

$$0 \leq f_{CD} \leq 6$$

最大フロー問題



$$\text{流量 flow} = f_{BD} + f_{CD}$$

最大フロー問題

$$\begin{aligned} &\text{maximize} \quad \text{flow} = f_{BD} + f_{CD} \\ &\text{subject to} \quad f_{AB} - f_{BC} - f_{BD} = 0 \\ &\quad \quad \quad f_{AC} + f_{BC} - f_{CD} = 0 \\ &\quad \quad \quad 0 \leq f_{AB} \leq 8 \\ &\quad \quad \quad 0 \leq f_{AC} \leq 4 \\ &\quad \quad \quad 0 \leq f_{BC} \leq 3 \\ &\quad \quad \quad 0 \leq f_{BD} \leq 4 \\ &\quad \quad \quad 0 \leq f_{CD} \leq 6 \end{aligned}$$

最大フロー問題

変数ベクトルを導入

$$\mathbf{x} = \begin{bmatrix} f_{AB} \\ f_{AC} \\ f_{BC} \\ f_{BD} \\ f_{CD} \end{bmatrix}$$

flow を最大化 $\equiv$ $-\text{flow}$ を最小化

$$\text{minimize} \quad -\text{flow} = -f_{BD} - f_{CD} = \mathbf{c}^\top \mathbf{x}$$

$$\mathbf{c} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -1 \\ -1 \end{bmatrix}$$

最大フロー問題

$$f_{AB} - f_{BC} - f_{BD} = 0$$

$$f_{AC} + f_{BC} - f_{CD} = 0$$

↓

$$A_{eq} \mathbf{x} = \mathbf{b}_{eq} \quad \text{線形条件}$$

$$A_{eq} = \begin{bmatrix} 1 & 0 & -1 & -1 & 0 \\ 0 & 1 & 1 & 0 & -1 \end{bmatrix}$$

$$\mathbf{b}_{eq} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

最大フロー問題

$$0 \leq f_{AB} \leq 8$$

$$0 \leq f_{AC} \leq 4$$

$$0 \leq f_{BC} \leq 3$$

$$0 \leq f_{BD} \leq 4$$

$$0 \leq f_{CD} \leq 6$$



$$l_b \leq \mathbf{x} \leq \mathbf{u}_b$$

下限 (lower bound) と上限 (upper bound)

$$\mathbf{l}_b = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix},$$

$$\mathbf{u}_b = \begin{bmatrix} 8 \\ 4 \\ 3 \\ 4 \\ 6 \end{bmatrix}$$

最大フロー問題

$$\begin{array}{ll}\text{minimize} & \mathbf{c}^\top \mathbf{x} \\ \text{subject to} & A_{eq} \mathbf{x} = \mathbf{b}_{eq} \\ & \mathbf{l}_b \leq \mathbf{x} \leq \mathbf{u}_b\end{array}$$



目的関数と制約式がすべて線形



線形計画問題 (linear programming, 略して LP)
MATLAB の `linprog` を用いて解くことができる

最大フロー問題

$[x, fmin] = \text{linprog}(c, A_{ineq}, b_{ineq}, A_{eq}, b_{eq}, lb, ub)$

$$\begin{aligned} & \text{minimize } \mathbf{c}^\top \mathbf{x} \\ & \text{subject to } A_{ineq} \mathbf{x} \leq \mathbf{b}_{ineq} \\ & \quad A_{eq} \mathbf{x} = \mathbf{b}_{eq} \\ & \quad \mathbf{l}_b \leq \mathbf{x} \leq \mathbf{u}_b \end{aligned}$$

を解く.

maximize g は minimize $f = -g$ とする.

不等式制約がない場合は $A_{ineq} = []$, $b_{ineq} = []$ とする.

等式制約がない場合は $A_{eq} = []$, $b_{eq} = []$ とする.

上限や下限の指定がない場合は $lb = []$ あるいは $ub = []$ とする.

最大フロー問題

ファイル max_flow_LP.m

% 目的関数

```
c = [ 0;  
      0;  
      0;  
     -1;  
     -1 ];
```

% 線形条件

```
Aeq = [ 1, 0, -1, -1, 0;  
        0, 1, 1, 0, -1 ];  
beq = [ 0;  
        0 ];
```

最大フロー問題

ファイル max_flow_LP.m

% 下限と上限

lb = zeros(5,1);

ub = [8;
4;
3;
4;
6];

% 線形計画法

[x,fmin] = linprog(c, [],[], Aeq,beq, lb,ub);

x

flow = -fmin;

flow

最大フロー問題

実行結果

```
>> max_flow_LP
```

最適解が見つかりました。

```
x =
```

7

3

3

4

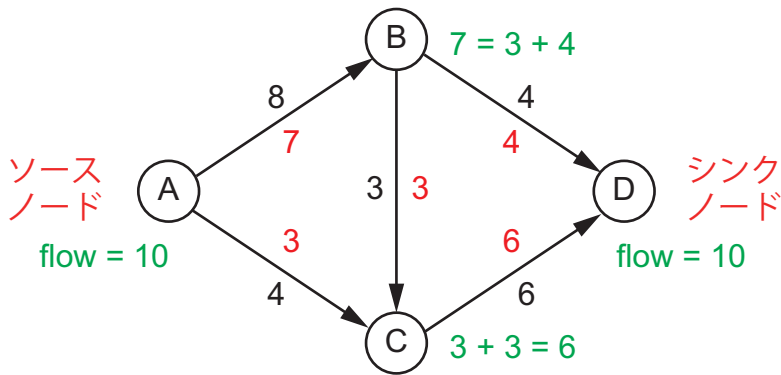
6

```
flow =
```

10

```
>>
```

最大フロー問題



$$\begin{bmatrix} f_{AB} \\ f_{AC} \\ f_{BC} \\ f_{BD} \\ f_{CD} \end{bmatrix} = \begin{bmatrix} 7 \\ 3 \\ 3 \\ 4 \\ 6 \end{bmatrix}$$

最大フロー問題

下限と上限を不等式条件で表す

$$0 \leq f_{AB} \leq 8 \rightarrow -f_{AB} \leq 0, f_{AB} \leq 8$$

$$0 \leq f_{AC} \leq 4 \rightarrow -f_{AC} \leq 0, f_{AC} \leq 4$$

...

$$\begin{bmatrix} -1 & & & & & \\ & -1 & & & & \\ & & -1 & & & \\ & & & -1 & & \\ & & & & -1 & \\ 1 & & & & & \\ & 1 & & & & \\ & & 1 & & & \\ & & & 1 & & \\ & & & & 1 & \end{bmatrix} \begin{bmatrix} f_{AB} \\ f_{AC} \\ f_{BC} \\ f_{BD} \\ f_{CD} \end{bmatrix} \leq \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 8 \\ 4 \\ 3 \\ 4 \\ 6 \end{bmatrix}$$

最大フロー問題

ファイル max_flow_LP_2.m

% 線形不等式

```
Aineq = [ -eye(5); eye(5) ];
```

```
bineq = [ zeros(5,1); 8; 4; 3; 4; 6 ];
```

% 線形計画法

```
[x,fmin] = linprog(c, Aineq,bineq, Aeq,beq, [],[]);
```

```
x
```

```
flow = -fmin;
```

```
flow
```

最大フロー問題

実行結果

```
>> max_flow_LP_2
```

最適解が見つかりました。

```
x =
```

```
7
```

```
3
```

```
3
```

```
4
```

```
6
```

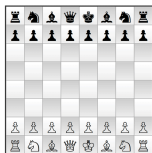
```
flow =
```

```
10
```

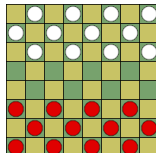
```
>>
```

歴史

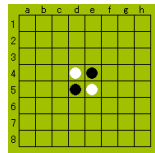
- 1949 クロード・シャノン
「チェスをするコンピュータのプログラミング」
- 1996 ディープ・ブルー (コンピュータ) 1 勝
ガルリ・カスパロフ 3 勝 2 引き分け
- 2007 チェッカーの完全解析 (引き分け)
- 2008 6×6 リバーシの完全解析 (後手必勝)



チェス



チェッカー



リバーシ
(オセロ)

叡王戦/電王戦

叡王戦で優勝した棋士がコンピュータプログラムと対戦

2016 年

コンピュータ 2 勝 0 敗

2017 年

名人がコンピュータと対局

コンピュータ 2 勝 0 敗

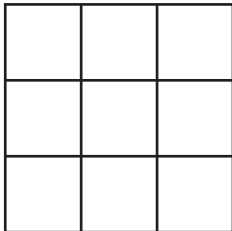
囲碁

2015 年 アルファ碁がプロ棋士（二段）に勝利
2016 年 アルファ碁がプロ棋士（九段）に勝利
(世界戦優勝経験棋士)

モンテカルロ探索と深層学習を使用

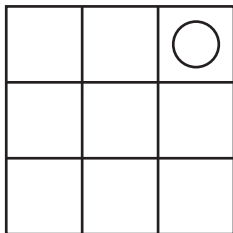
ゲーム

三目並べ



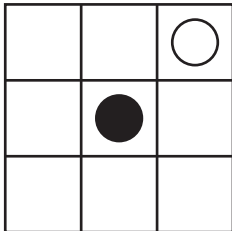
ゲーム

三目並べ



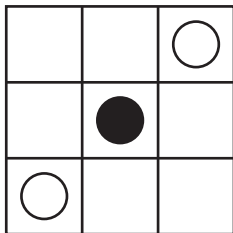
ゲーム

三目並べ



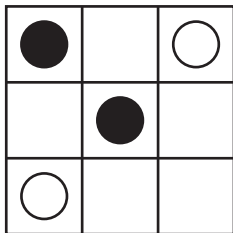
ゲーム

三目並べ



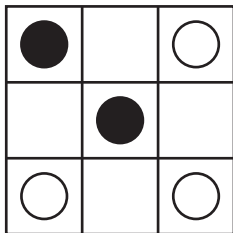
ゲーム

三目並べ



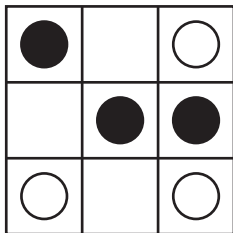
ゲーム

三目並べ



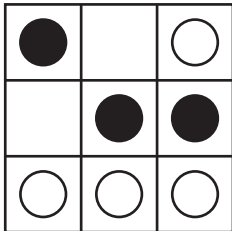
ゲーム

三目並べ



ゲーム

三目並べ



ゲーム

三目並べ, 五目並べ, 囲碁, 将棋,
チェス, チェッカー, リバーシ (オセロ)

二人 two-person

零和 zero-sum

有限 finite ゲーム

確定 deterministic

完全情報 perfect information

コントラクトブリッジ	確定, 不完全情報
バックギャモン	不確定, 完全情報,
マージャン	確定, 不完全情報, 多人数 (四人)
ダイヤモンドゲーム	多人数 (三人)

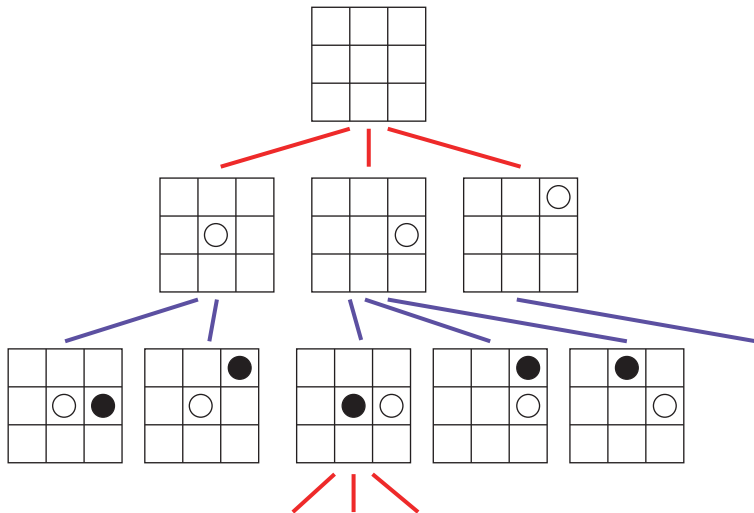
ゲーム

二人	two-person	
零和	zero-sum	
有限	finite	ゲーム
確定	deterministic	
完全情報	perfect information	



双方のプレイヤーが最善手
⇒ 先手必勝，後手必勝，引き分け

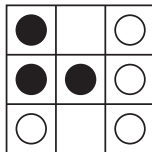
ゲーム木



局面の評価

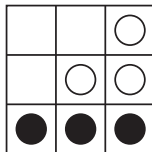
局面に数値を対応させる

先手が勝ち



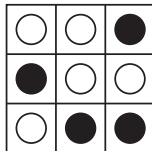
+1

後手が勝ち



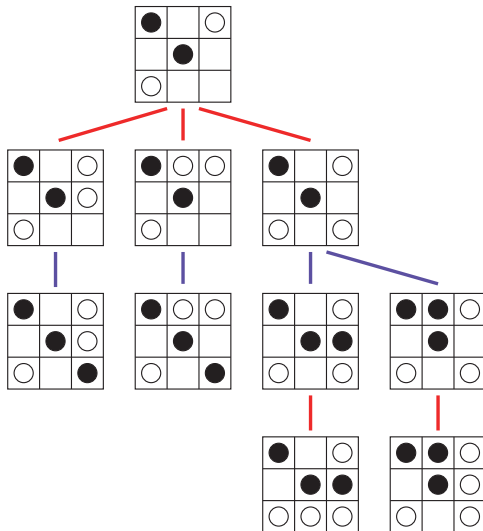
-1

引き分け

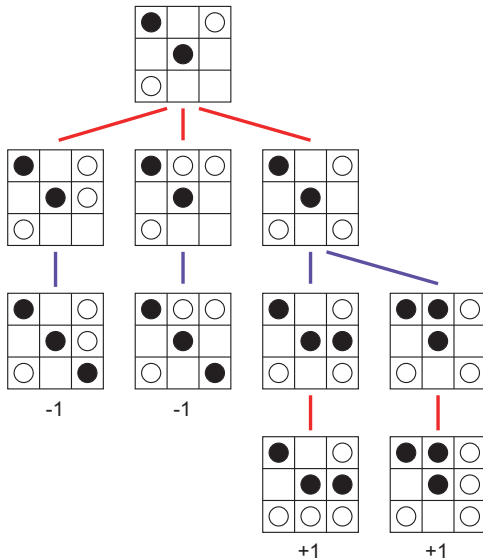


0

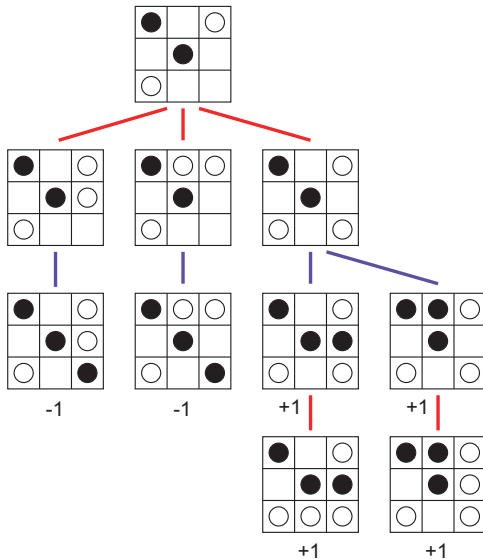
局面の評価



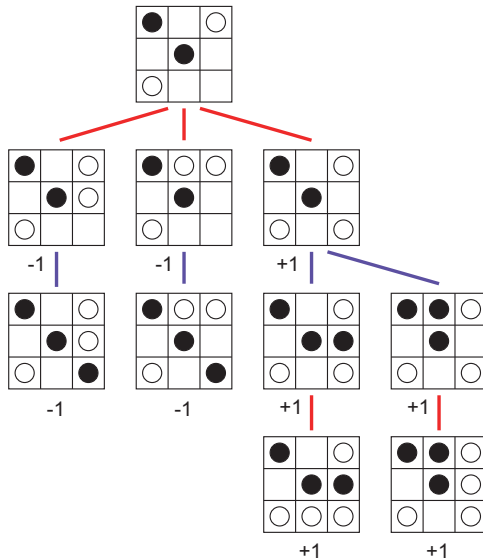
局面の評価



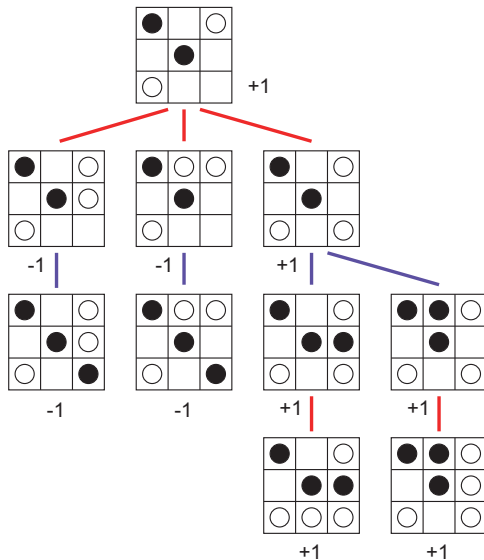
局面の評価



局面の評価



局面の評価



局面の評価

局面に数値を対応させる

先手が有利	正の数 (+)
後手が有利	負の数 (-)

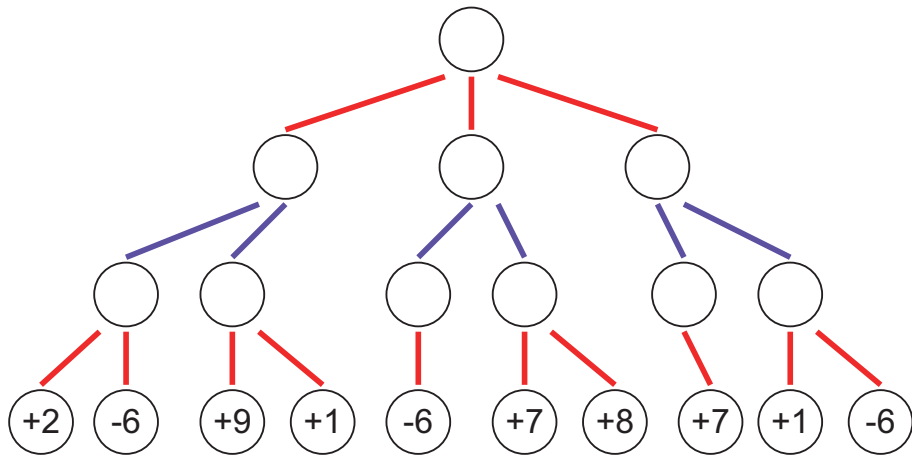
将棋の場合

- 駒の損得
(e.g. 飛 19 点, 角 17 点, 金 11 点, 銀 10 点, ...)
- 駒の配置 (駒の接続 玉の防御)
- ...

複数の項目を評価し, それらから一つの評価値を計算

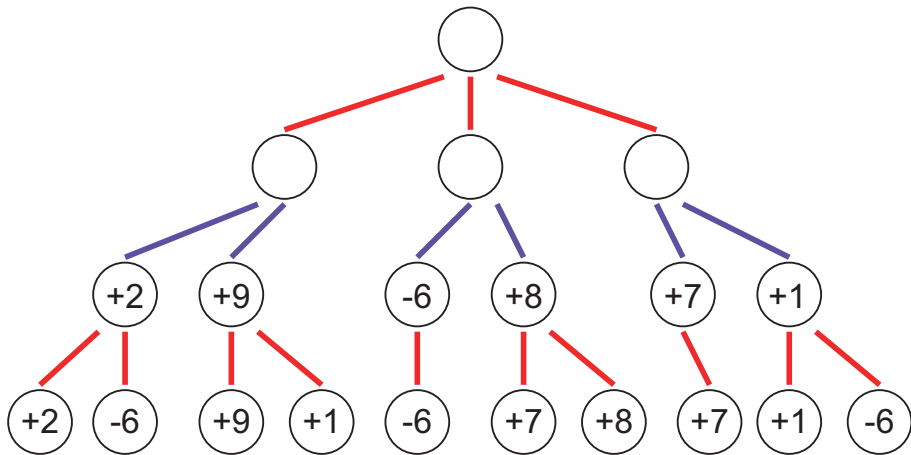
局面の評価

先手の番



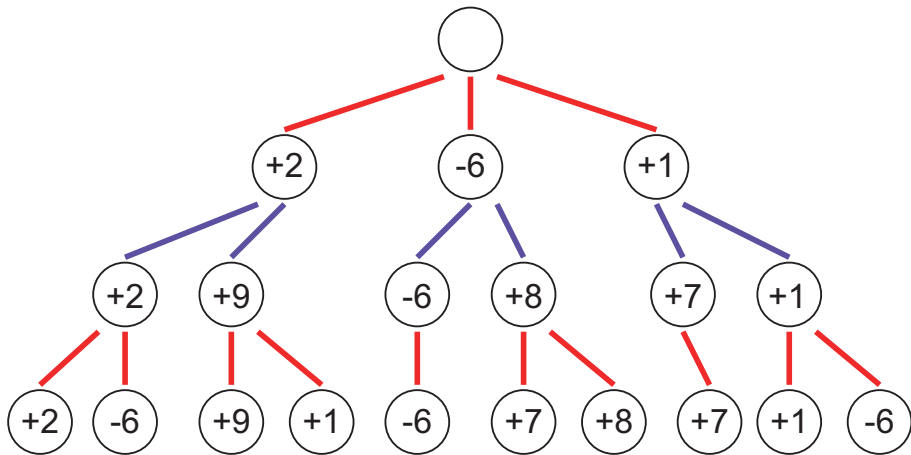
局面の評価

先手の番



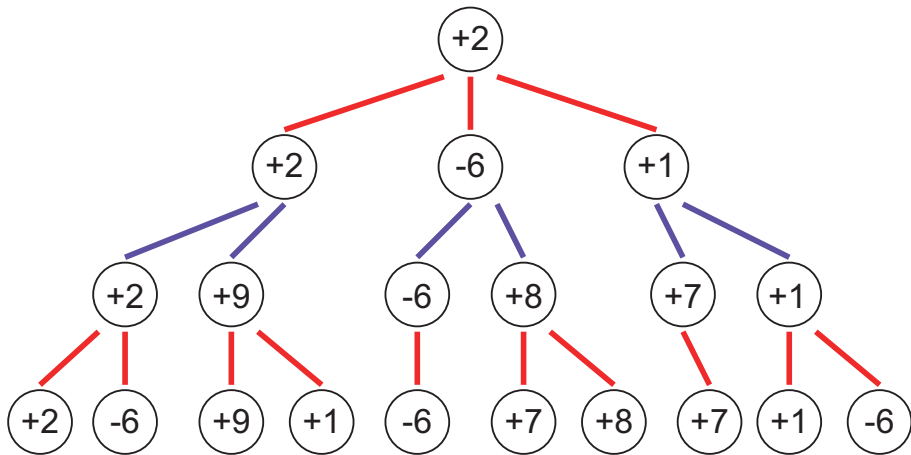
局面の評価

先手の番



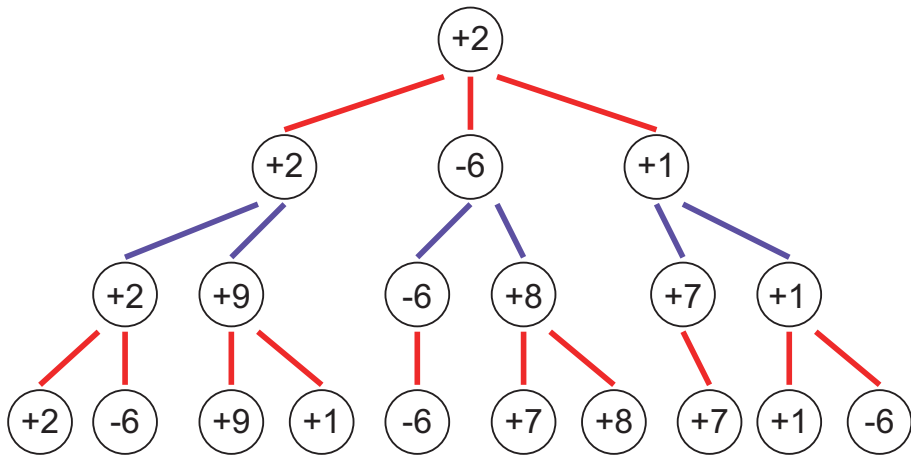
局面の評価

先手の番



局面の評価

先手の番

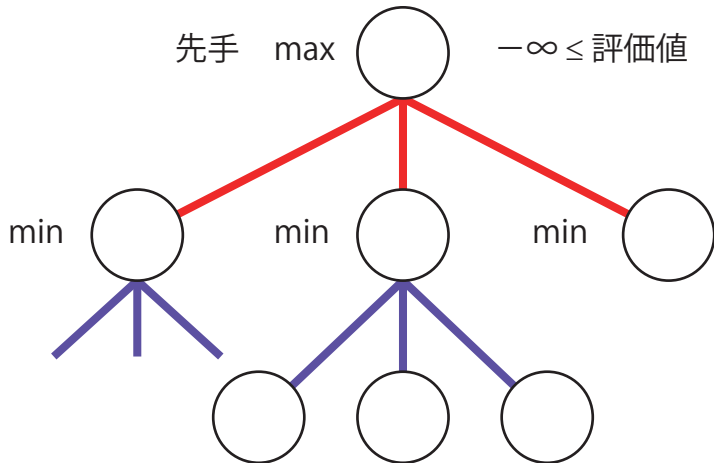


ミニマックス (minimax) 法

ミニマックス法

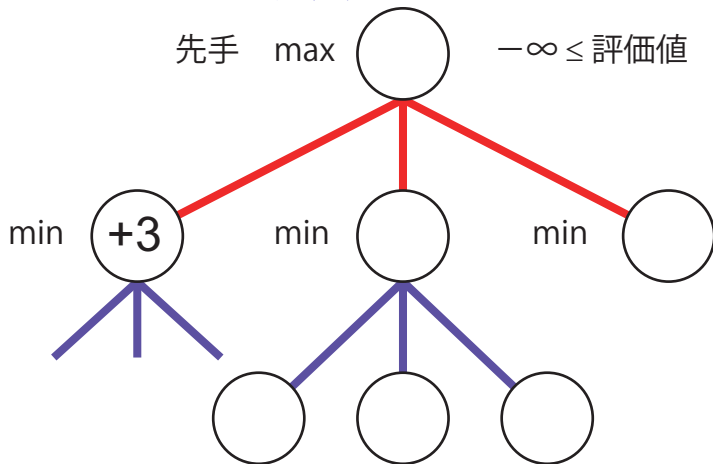
```
score = minimax (node, depth)
  if (depth == 0) return (node の評価値);
  if node が先手の局面
    max =  $-\infty$ 
    foreach node のすべての子ノード child
      評価値 = minimax(child, depth-1);
      if 評価値 > max ならば max = 評価値;
    return max;
  if node が後手の局面
    min =  $\infty$ 
    foreach node のすべての子ノード child
      評価値 = minimax(child, depth-1);
      if 評価値 < min ならば min = 評価値;
    return min;
end
```

アルファベータ法



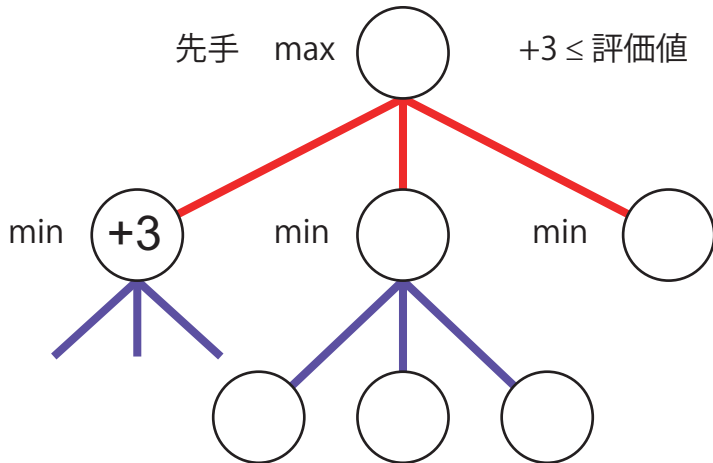
先手番で評価値を求める

アルファベータ法



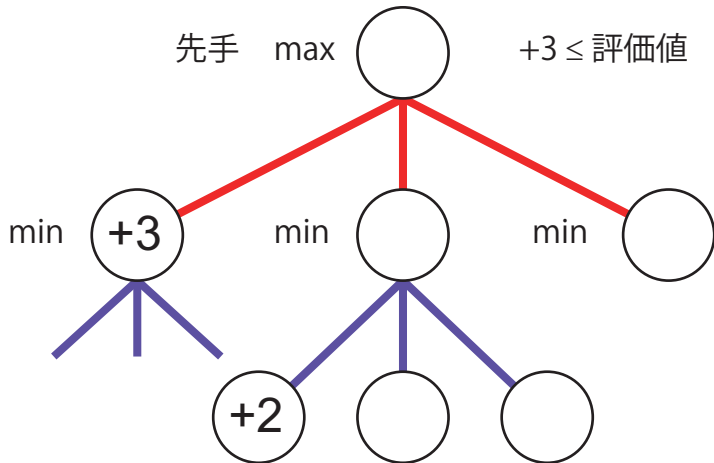
一つめの子ノードで評価値 +3 を得た

アルファベータ法



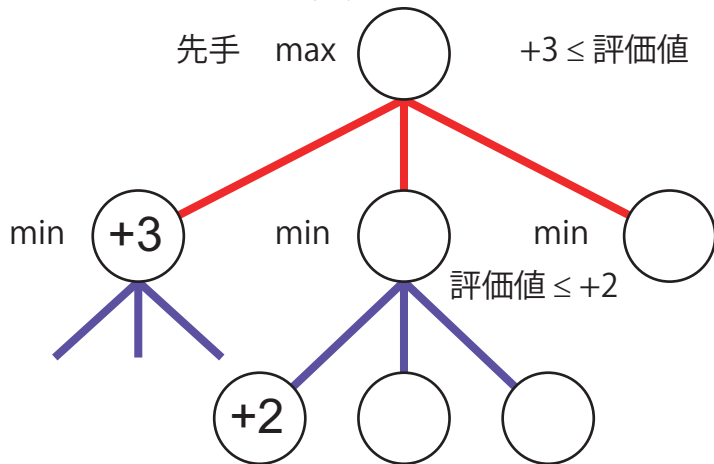
親ノードの評価値は +3 以上（評価値の下限）

アルファベータ法



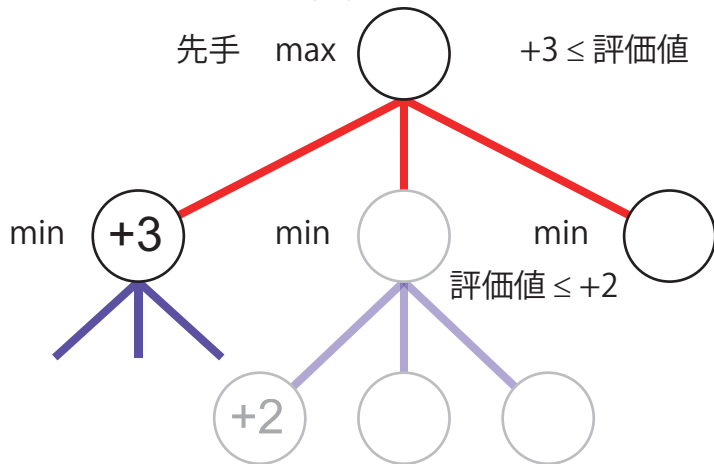
二つめの子ノードで評価値を計算. 孫ノードで評価値 $+2$ を得た

アルファベータ法



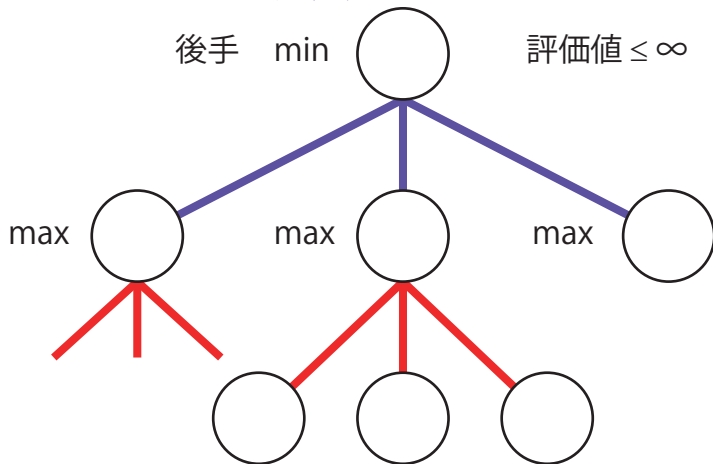
二つ目の子ノードの評価値は +2 以下（評価値の上限）

アルファベータ法



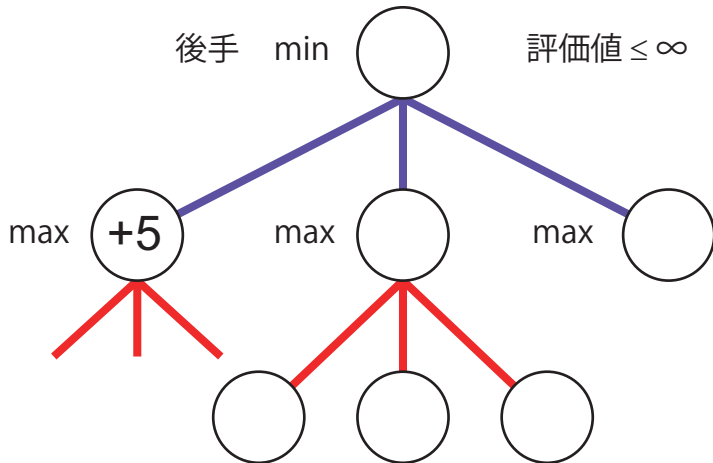
二つ目の子ノードを評価する必要はないので、カットする
(ベータカット)

アルファベータ法



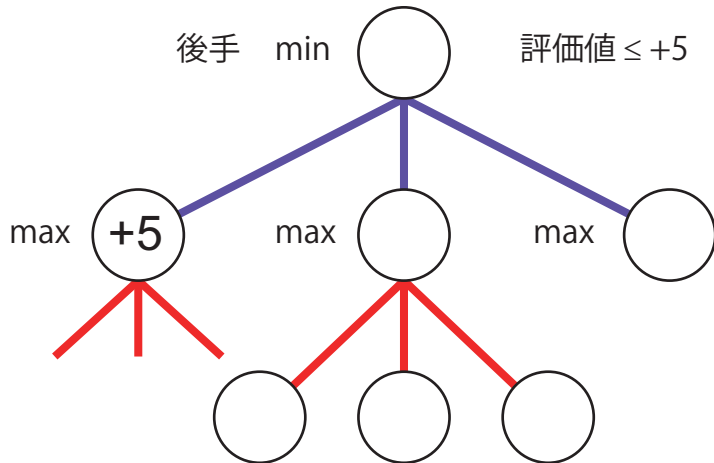
後手番で評価値を求める

アルファベータ法



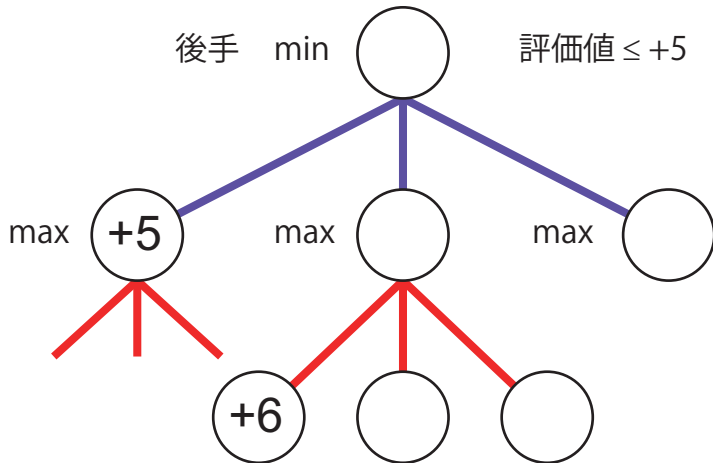
一つめの子ノードで評価値 +5 を得た

アルファベータ法



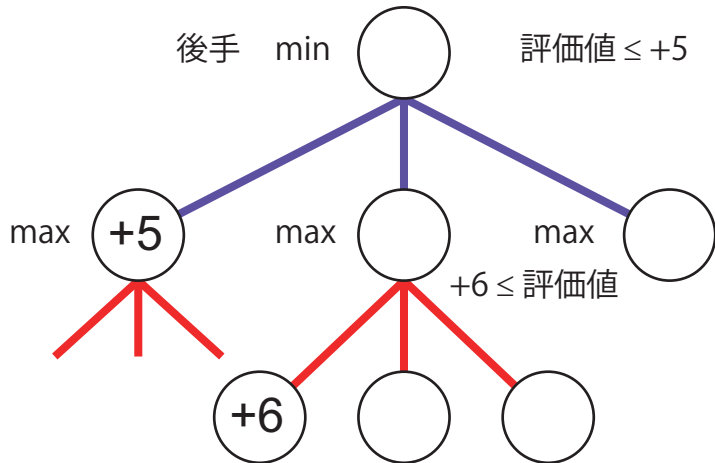
親ノードの評価値は $+5$ 以下（評価値の上限）

アルファベータ法



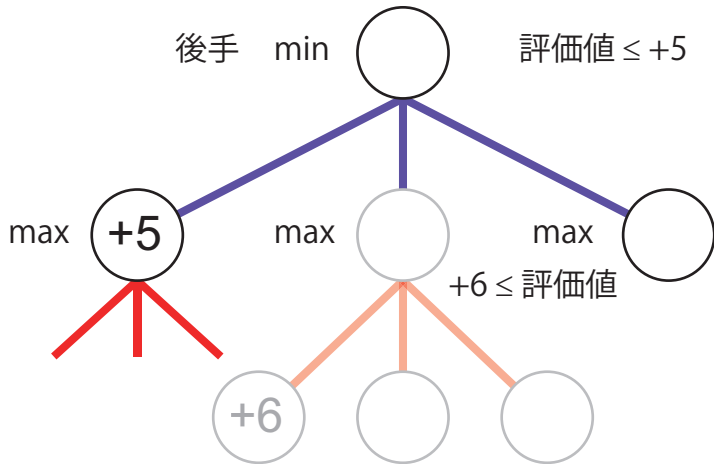
二つめの子ノードで評価値を計算. 孫ノードで評価値 +6 を得た

アルファベータ法



二つ目の子ノードの評価値は +6 以上（評価値の下限）

アルファベータ法



二つ目の子ノードを評価する必要はないので、カットする
(アルファカット)

アルファベータ法

```
score = alphabeta (node, depth,  $\alpha$ ,  $\beta$ )
  if (depth == 0) return (node の評価値);
  if node が先手の局面
    foreach node のすべての子ノード child
       $\alpha$  = max ( $\alpha$ , alphabeta(child, depth-1,  $\alpha$ ,  $\beta$ ));
      if  $\alpha \geq \beta$  ならば break;
    return  $\alpha$ ;
  if node が後手の局面
    foreach node のすべての子ノード child
       $\beta$  = min ( $\beta$ , alphabeta(child, depth-1,  $\alpha$ ,  $\beta$ ));
      if  $\alpha \geq \beta$  ならば break;
    return  $\beta$ ;
end

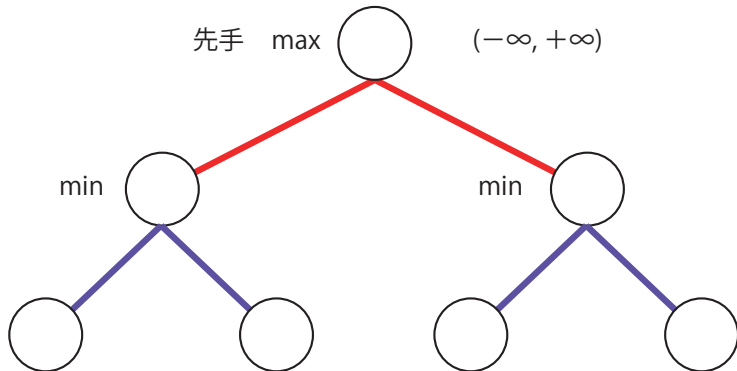
alphabeta(現在の局面, depth,  $-\infty$ ,  $+\infty$ ) で実行
```

アルファベータ法

先手

max

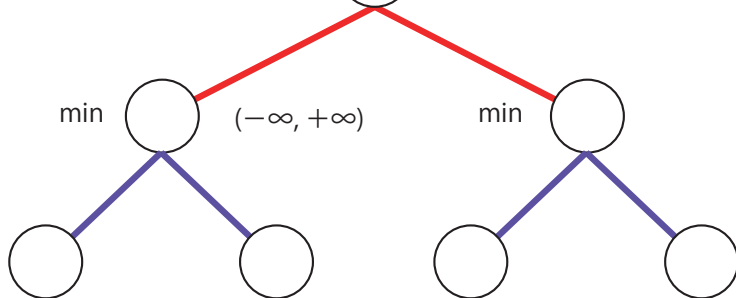
$(-\infty, +\infty)$



先手番で評価値を求める. $(\alpha, \beta) = (-\infty, +\infty)$

アルファベータ法

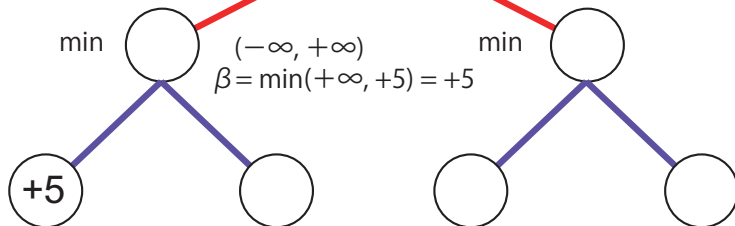
先手 max $(-\infty, +\infty)$



一つめの子ノードで評価値を求める

アルファベータ法

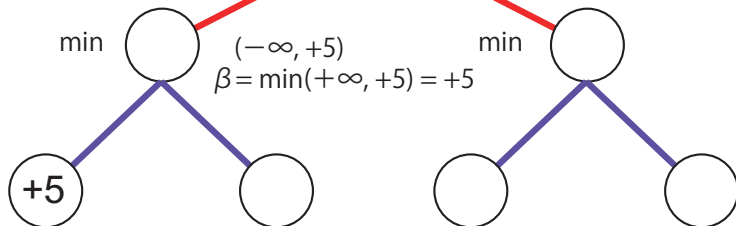
先手 max $(-\infty, +\infty)$



評価値 +5 を得た. β の値を計算

アルファベータ法

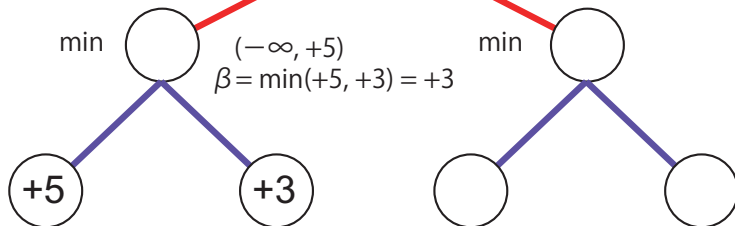
先手 max $(-\infty, +\infty)$



$$(\alpha, \beta) = (-\infty, +5)$$

アルファベータ法

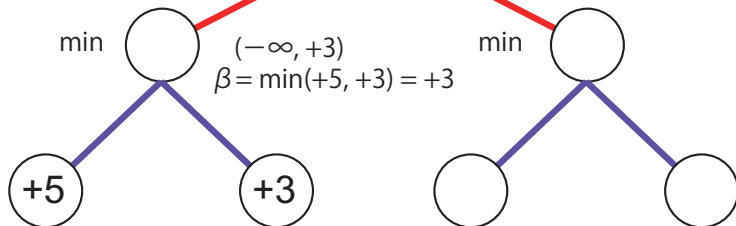
先手 max $(-\infty, +\infty)$



評価値 +3 を得た. β の値を計算

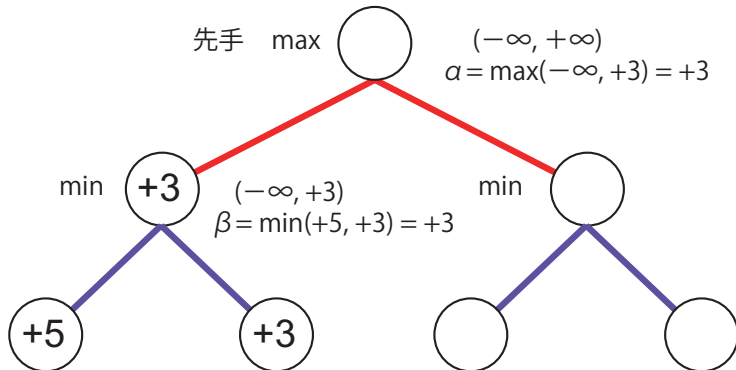
アルファベータ法

先手 max $(-\infty, +\infty)$



$$(\alpha, \beta) = (-\infty, +3)$$

アルファベータ法



評価値 +3 を得た. α の値を計算

アルファベータ法

先手 max

$(+3, +\infty)$

$$\alpha = \max(-\infty, +3) = +3$$

min

+3

$(-\infty, +3)$

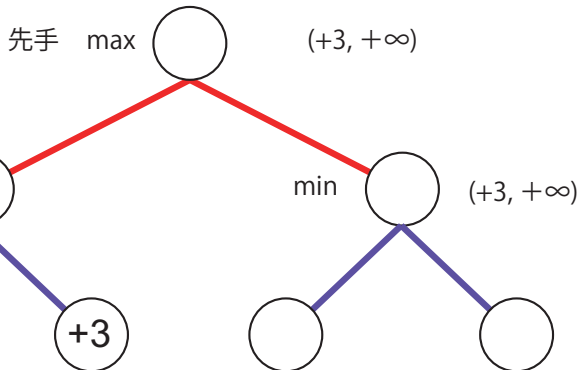
min

+5

+3

$$(\alpha, \beta) = (+3, +\infty)$$

アルファベータ法

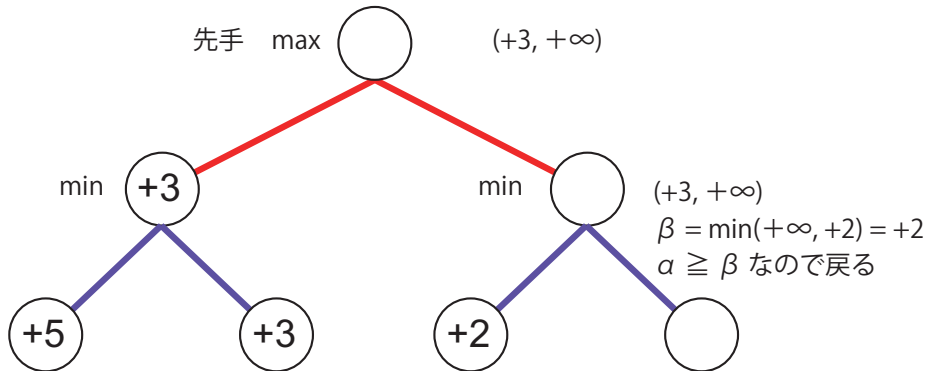


二つめの子ノードで評価値を求める

アルファベータ法

先手 max

$(+3, +\infty)$



$(+3, +\infty)$

$\beta = \min(+\infty, +2) = +2$

$\alpha \geq \beta$ なので戻る

評価値 $+2$ を得た. β の値を計算. $\alpha \geq \beta$

アルファベータ法

先手 max

$$(+3, +\infty)$$
$$\alpha = \max(+3, +2) = +3$$

min

+3

+5

+3

min

$$(+3, +\infty)$$

$$\beta = \min(+\infty, +2) = +2$$

$\alpha \geq \beta$ なので戻る

+2

二つ目の子ノードにおける計算を打ち切る. α の値を計算

アルファベータ法

先手 max

+3

$(+3, +\infty)$

$$\alpha = \max(+3, +2) = +3$$

min

+3

min

$(+3, +\infty)$

$$\beta = \min(+\infty, +2) = +2$$

$\alpha \geq \beta$ なので戻る

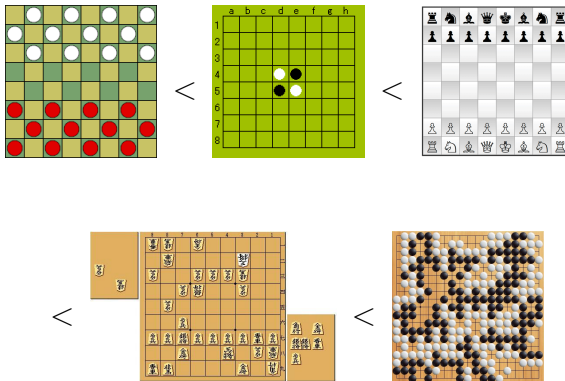
+5

+3

+2

評価値 +3 を得た.

局面における場合の数



局面の評価

評価関数を人が試行錯誤し作成



評価関数をデータから自動的に作成（学習）

評価関数をコンピュータが指した結果から自動的に作成
（モンテカルロ学習）

最終的な結果（勝ち負け）を規範とする

二人零和ゲーム

プレイヤー A と B が対戦

各プレイヤーは手 P, Q を選択する.

プレイヤー A の利得

		B	
		P	Q
A	P	+5	-8
	Q	-2	+5

A が手 P, B が手 P → A が +5, B が -5

A が手 P, B が手 Q → A が -8, B が +8

A が手 Q, B が手 P → A が -2, B が +2

A が手 Q, B が手 Q → A が +5, B が -5

二人零和ゲーム

各プレイヤーは確率的に手を選ぶ

A が手 P を選ぶ確率 x $\rightarrow$ A が手 Q を選ぶ確率 $1 - x$

B が手 P を選ぶ確率 y $\rightarrow$ B が手 Q を選ぶ確率 $1 - y$

プレイヤー A の利得

$$G_A = (+5)xy + (-8)x(1 - y) \\ + (-2)(1 - x)y + (+5)(1 - x)(1 - y)$$

極値を計算し平衡点を求める．平衡点は鞍点 (saddle point)

$$\frac{\partial G_A}{\partial x} = (+5)y + (-8)(1 - y) + (-2)(-1)y + (+5)(-1)(1 - y) = 0$$

$$\frac{\partial G_A}{\partial y} = (+5)x + (-8)x(-1) + (-2)(1 - x) + (+5)(1 - x)(-1) = 0$$

$\Downarrow$

$$x = 0.35, \quad y = 0.65, \quad G_A = 0.45$$

二人零和ゲーム

プレイヤー A の利得

		B	
		P	Q
A	P	g_{PP}	g_{PQ}
	Q	g_{QP}	g_{QQ}

平衡点における利得

$$G_A^* = \frac{g_{PP}g_{QQ} - g_{PQ}g_{QP}}{g_{PP} - g_{PQ} - g_{QP} + g_{QQ}}$$

⇓

公平な勝負であるための条件 $G_A^* = 0$

$$g_{PP}g_{QQ} - g_{PQ}g_{QP} = 0$$

二人零和ゲーム

プレイヤー A の利得

		B	
		P	Q
A	P	+4	-8
	Q	-2	+4



$$x = 0.3333, \quad y = 0.6667, \quad G_A = 0$$

社会的ジレンマ

囚人のジレンマ

2名の囚人 A, B に検事が提案する.

2名の囚人はたがいに相談することはできない.

		B	
		黙秘	自白
A	黙秘	(1, 1)	(10, 0)
	自白	(0, 10)	(5, 5)

A が黙秘, B が黙秘 → A が 1 年, B が 1 年

A が黙秘, B が自白 → A が 10 年, B が 0 年 (釈放)

A が自白, B が黙秘 → A が 0 年 (釈放), B が 10 年

A が自白, B が自白 → A が 5 年, B が 5 年

二人非零和ゲーム (双方ともに得する/損する可能性)

⇒ 協調するという行動があり得る

社会的ジレンマ

囚人のジレンマ

2名の囚人 A, B に検事が提案する.

2名の囚人はたがいに相談することはできない.

		B	
		黙秘	自白
A	黙秘	(1, 1)	(10, 0)
	自白	(0, 10)	(5, 5)

A の判断

		B	
		黙秘	自白
A	黙秘	(1, 1)	(10, 0)
	自白	(0, 10)	(5, 5)

B が黙秘するならば, A (自分) は自白する方が有利

社会的ジレンマ

囚人のジレンマ

2名の囚人 A, B に検事が提案する.

2名の囚人はたがいに相談することはできない.

		B	
		黙秘	自白
A	黙秘	(1, 1)	(10, 0)
	自白	(0, 10)	(5, 5)

A の判断

		B	
		黙秘	自白
A	黙秘	(1, 1)	(10, 0)
	自白	(0, 10)	(5, 5)

B が自白するならば, A (自分) は自白する方が有利

社会的ジレンマ

囚人のジレンマ

2名の囚人 A, B に検事が提案する.

2名の囚人はたがいに相談することはできない.

		B	
		黙秘	自白
A	黙秘	(1, 1)	(10, 0)
	自白	(0, 10)	(5, 5)

A の判断

		B	
		黙秘	自白
A	黙秘	(1, 1)	(10, 0)
	自白	(0, 10)	(5, 5)

B の行動に関わらず, A (自分) は自白する方が有利

社会的ジレンマ

囚人のジレンマ

2名の囚人 A, B に検事が提案する.

2名の囚人はたがいに相談することはできない.

		B	
		黙秘	自白
A	黙秘	(1, 1)	(10, 0)
	自白	(0, 10)	(5, 5)

A の判断 B の行動に関わらず, 自白する方が有利

B の判断 A の行動に関わらず, 自白する方が有利

A, B ともに自白 (5, 5)



両方に有利な行動 A, B ともに黙秘 (1, 1)

社会的ジレンマ

囚人のジレンマ

2名の囚人 A, B に検事が提案する.

2名の囚人はたがいに相談することはできない.

		B	
		黙秘	自白
A	黙秘	(1, 1)	(10, 0)
	自白	(0, 10)	(5, 5)

各個人の最適な選択 (ナッシュ均衡; Nash equilibrium)
(ナッシュは米国の数学者. 映画「ビューティフル・マインド」)



全体で最適な選択 (パレート最適; Paretian optimum)
(パレート (Pareto) はイタリアの社会学者. 80:20 の法則)

まとめ

グラフ

ノードとエッジから構成される

最短経路問題

最短経路木

最適性原理 ダイクストラのアルゴリズム

最大フロー問題

LP 法

ゲーム

ゲーム木 ミニマックス法 アルファベータ法 二人零和ゲーム

レポート (MATLAB Grader)

MATLAB grader 「グラフと経路計画」
締切：11月18日（月曜）午前1時