離散構造とアルゴリズム
(3-1) 探索アルゴリズム

高橋篤司
東京工業大学 工学院 情報通信系

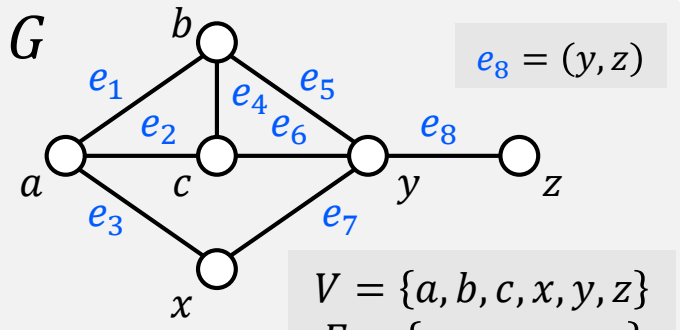
グラフ探索

- Depth-First Search (DFS)
- Breadth-First Search (BFS)

…

- Vertex Search
  - Node Search
  - Virus Search
- Edge Search
- Mixed Search

…



[DS] データ探索（確認）

■ 逐次探索

✓一つずつ順番に調べる

■ 2分探索

✓探索の範囲を狭めていく

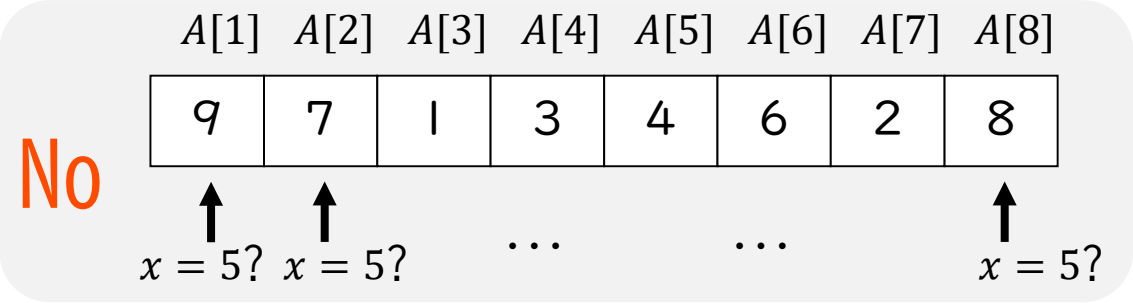
■ 電話帳

- 名前 ⟶ 番号 ： 簡単　　2分探索
- 番号 ⟶ 名前 ： 手間がかかる　逐次探索

[DS-a] 配列探索（確認）

■ ARRAY SEARCHING（判定）

- 入力 ： 配列$A[1:n]$ （整数型, 大きさn）, 整数x
- 質問 ： xは配列$A[1:n]$に含まれるか



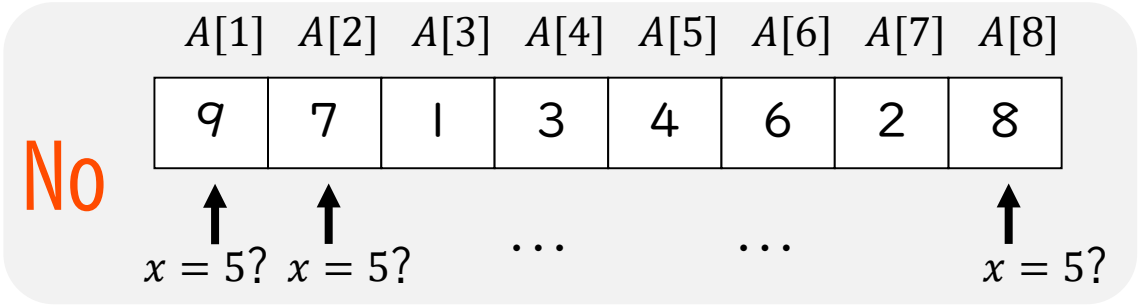
配列探索 - 逐次探索（確認）

■ ARRAY SEARCHING（判定）

- 入力 ： 配列$A[1:n]$ （整数型, 大きさn）, 整数x
- 質問 ： xは配列$A[1:n]$に含まれるか

■ Algorithm 3.a（逐次探索:Sequential Search）

Step 0: Set $i = 1$
Step 1: If $A[i] = x$ then output YES, and halt
Step 2: If $i = n$ then output NO, and halt
Step 3: Set $i = i + 1$, and return to Step 1



配列探索 - 逐次探索（確認）

■ 定理：逐次探索はARRAY SEARCHING を$O(n)$で解く

□ 証明：

- 正当性 ： trivial
- 時間計算量 ： ステップごとに解析 ： $O(n)$ Optimal!!

■ Algorithm 3.a （逐次探索:Sequential Search）

Step 0: Set $i = 1$ — $O(1)$

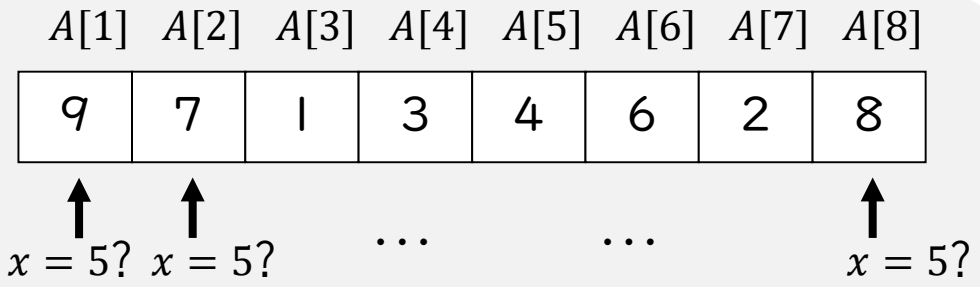
Step 1: If $A[i] = x$ then output YES, and halt — $O(1)$

Step 2: If $i = n$ then output NO, and halt — $O(1)$

Step 3: Set $i = i + 1$, and return to Step 1 — $O(1)$

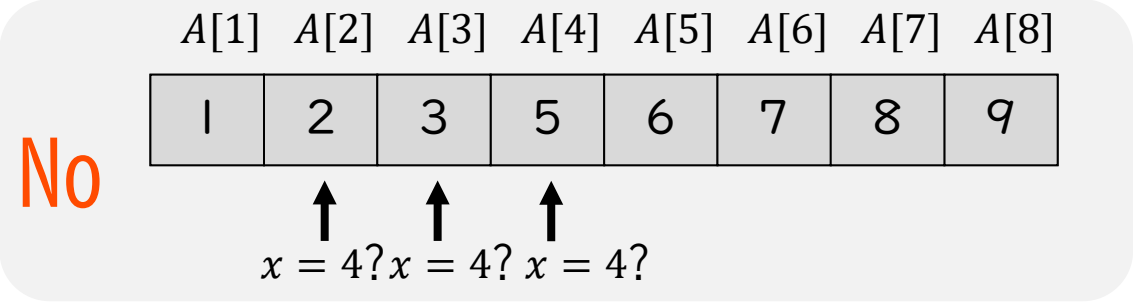
最大n回繰返し

No



[DS-b] 配列探索（確認）

- ORDERED ARRAY SEARCHING（判定）
  - 入力 ： 整列した配列$A[1:n]$（整数型, 大きさn）， 整数x
    - $A[i] < A[j]$ if $i < j$
  - 質問 ： xは配列$A[1:n]$に含まれるか



配列探索 - 2分探索（確認）

■ ORDERED ARRAY SEARCHING（判定）

- 入力 ： 整列した配列$A[1:n]$（整数型, 大きさn）， 整数x
- 質問 ： xは配列$A[1:n]$に含まれるか

■ Algorithm 3.b （2分探索:Binary Search）

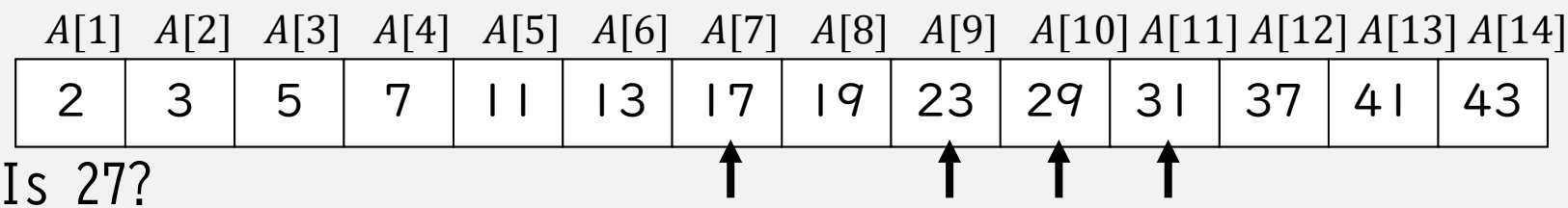
Step 0: Set $i = 1$, $j = n$

Step 1: If $i > j$ then output NO, and halt

Step 2: Set $m = \left\lfloor \frac{i+j}{2} \right\rfloor$

Step 3: If $A[m] = x$ then output YES, and halt

Step 4: If $A[m] > x$ then set $j = m - 1$
otherwise set $i = m + 1$, and return to Step 1



配列探索 - 2分探索（確認）

■ 定理：2分探索は
ORDERED ARRAY SEARCHINGを$O(\log n)$で解く

■ Algorithm 3.b （2分探索:Binary Search）

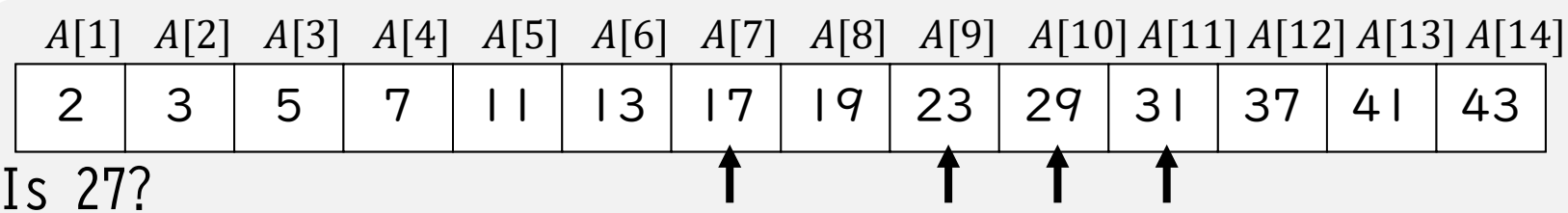
Step 0: Set $i = 1,\ j = n$

Step 1: If $i > j$ then output NO, and halt

Step 2: Set $m = \left\lfloor \frac{i+j}{2} \right\rfloor$

Step 3: If $A[m] = x$ then output YES, and halt

Step 4: If $A[m] > x$ then set $j = m - 1$
otherwise set $i = m + 1$, and return to Step 1



配列探索 - 2分探索（確認）

■ 定理：2分探索は
ORDERED ARRAY SEARCHINGを$O(\log n)$で解く

□ 証明：

- 正当性 ： trivial
- 時間計算量 ： 動作を解析

■ Algorithm 3.b （2分探索:Binary Search）

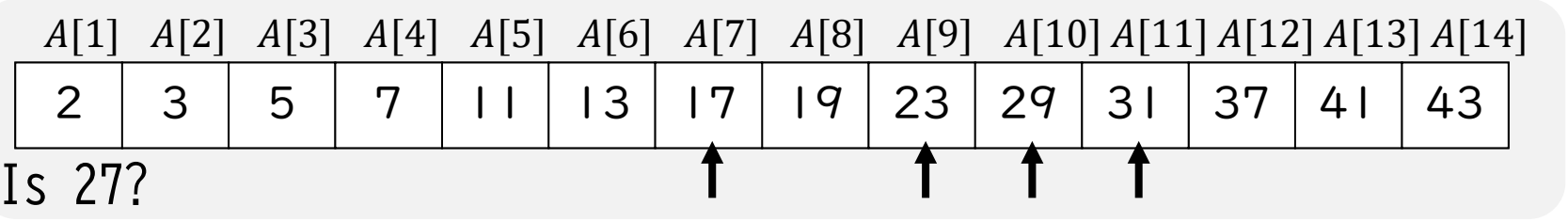
Step 0: Set $i = 1$, $j = n$

Step 1: If $i > j$ then output NO, and halt

Step 2: Set $m = \left\lfloor \frac{i+j}{2} \right\rfloor$

Step 3: If $A[m] = x$ then output YES, and halt

Step 4: If $A[m] > x$ then set $j = m - 1$
otherwise set $i = m + 1$, and return to Step 1

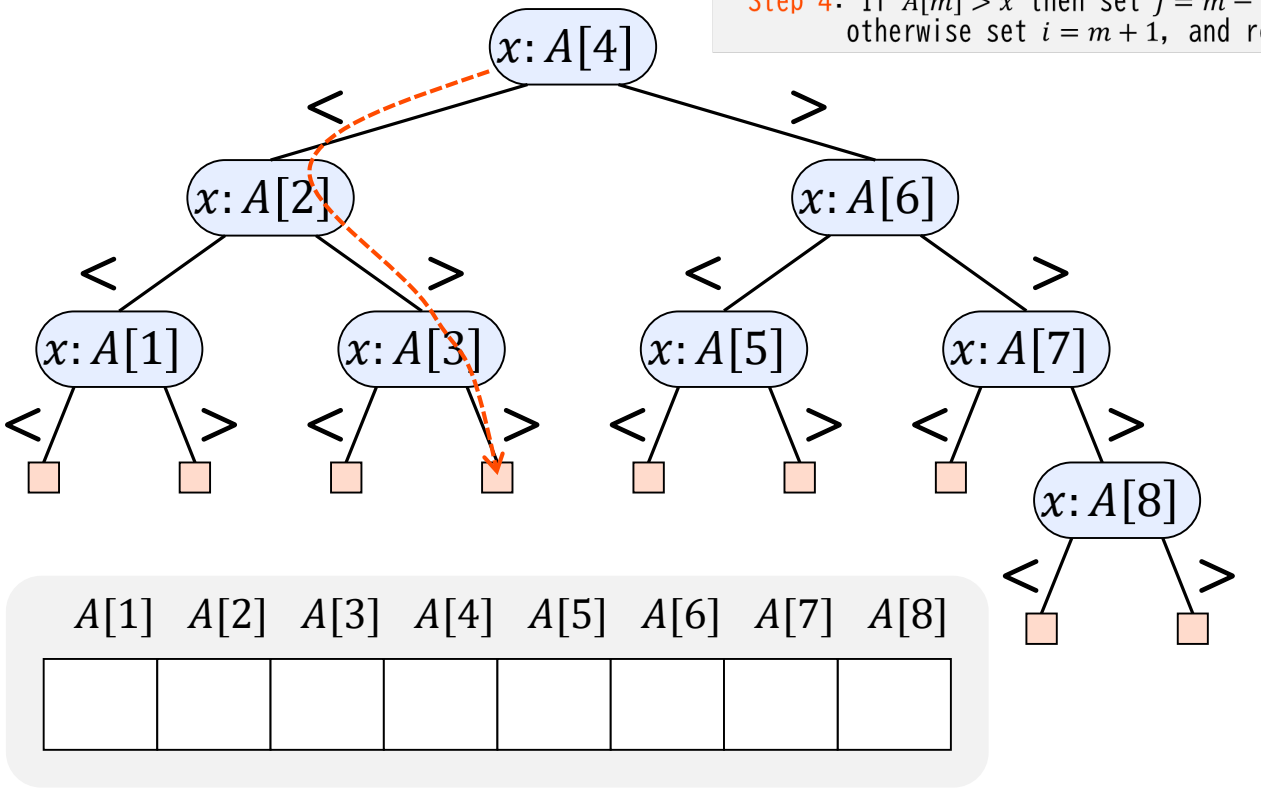


配列探索 - 2分探索（確認）

■ 2分決定木 T

- 高さ$h(T)$ = 比較回数

■ Algorithm 3.b (2分探索:Binary Search)
Step 0: Set $i=1$, $j=n$
Step 1: If $i>j$ then output NO, and halt
Step 2: Set $m=\left\lfloor\frac{i+j}{2}\right\rfloor$
Step 3: If $A[m]=x$ then output YES, and halt
Step 4: If $A[m]>x$ then set $j=m-1$
otherwise set $i=m+1$, and return to Step 1

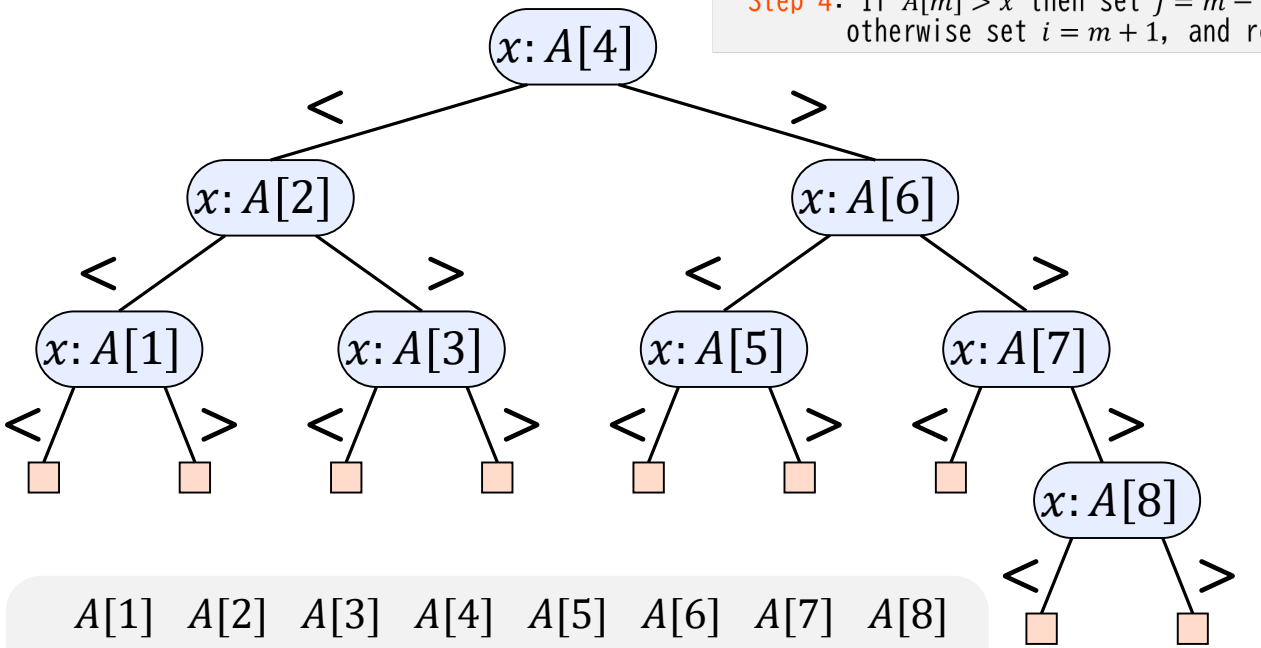


配列探索 - 2分探索（確認）

■ 補題a: 2分探索の2分決定木は正則(regular)

すべての内点の子は2

■ Algorithm 3.b (2分探索:Binary Search)
Step 0: Set $i=1,\ j=n$
Step 1: If $i>j$ then output NO, and halt
Step 2: Set $m=\left\lfloor\frac{i+j}{2}\right\rfloor$
Step 3: If $A[m]=x$ then output YES, and halt
Step 4: If $A[m]>x$ then set $j=m-1$
otherwise set $i=m+1$, and return to Step 1



配列探索 - 2分探索（確認）

■ 補題b: 2分探索の2分決定木は均衡(balanced)

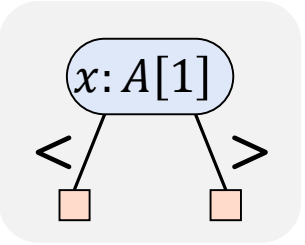
根から葉までの距離が$h(T)$ or $h(T)-1$

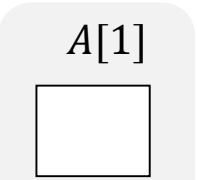
Tの高さ: $h(T)$

■ Algorithm 3.b (2分探索:Binary Search)
Step 0: Set $i=1$, $j=n$
Step 1: If $i>j$ then output NO, and halt
Step 2: Set $m=\left\lfloor\frac{i+j}{2}\right\rfloor$
Step 3: If $A[m]=x$ then output YES, and halt
Step 4: If $A[m]>x$ then set $j=m-1$
otherwise set $i=m+1$, and return to Step 1

□ 証明:

- 2分決定木Tの高さ$h(T)$に関する数学的帰納法
- 初期段階 : $h(T)=1$
  - 均衡





配列探索 - 2分探索（確認）

■ 補題b: 2分探索の2分決定木は均衡(balanced)

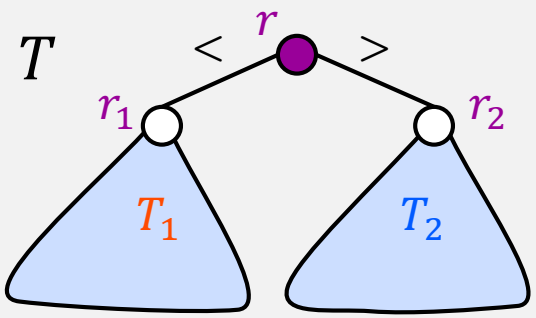
根から葉までの距離が$h(T)$ or $h(T)-1$

Tの高さ: $h(T)$

■ Algorithm 3.b (2分探索:Binary Search)
Step 0: Set $i=1$, $j=n$
Step 1: If $i>j$ then output NO, and halt
Step 2: Set $m=\left\lfloor\frac{i+j}{2}\right\rfloor$
Step 3: If $A[m]=x$ then output YES, and halt
Step 4: If $A[m]>x$ then set $j=m-1$
otherwise set $i=m+1$, and return to Step 1

□ 証明: (cont.)

- 帰納段階
  - 帰納段階の仮定
    - 高さk未満の2分決定木は均衡
  - T: 高さ$h(T)=k$の2分決定木（根:r）
    - T_1, T_2 : 2分決定木
    - $h(T_1), h(T_2)<k$
      $\Longrightarrow$ T_1, T_2 : 均衡



配列探索 - 2分探索（確認）

■ 補題b: 2分探索の2分決定木は均衡(balanced)

Tの内点数: $\mu(T)$

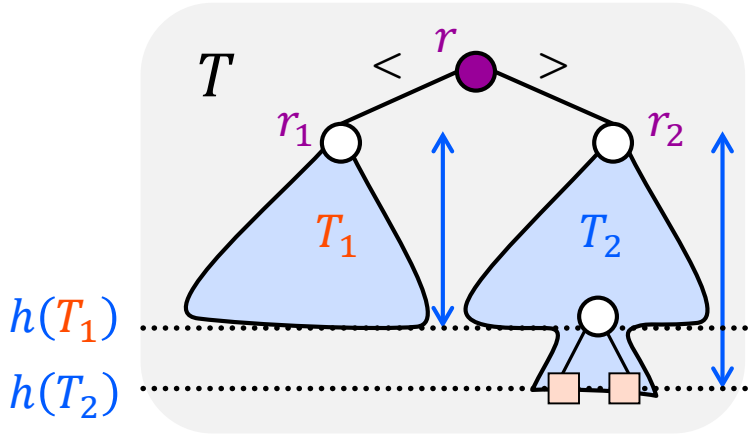
Tの高さ: $h(T)$

根から葉までの距離が$h(T)$ or $h(T)-1$

■ Algorithm 3.b (2分探索:Binary Search)
Step 0: Set $i=1,\ j=n$
Step 1: If $i>j$ then output NO, and halt
Step 2: Set $m=\left\lfloor\frac{i+j}{2}\right\rfloor$
Step 3: If $A[m]=x$ then output YES, and halt
Step 4: If $A[m]>x$ then set $j=m-1$
otherwise set $i=m+1$, and return to Step 1

□ 証明: (cont.)

- 2分探索のmの定義より
  - $\mu(T_1) \le \mu(T_2) \le \mu(T_1)+1$
- 正則均衡2分木T: $h(T)=\lfloor \log_2 \mu(T) \rfloor + 1$ （定理@[1-2]）
  - $h(T_1) \le h(T_2) \le h(T_1)+1$
- Case 1: $h(T_1)=h(T_2)$
  $\Longrightarrow$ Tは均衡
- Case 2: $h(T_1)=h(T_2)-1$
  $\Longrightarrow \mu(T_1)=\mu(T_2)-1$
  - T_1: 完全2分木 $\Longrightarrow$ Tは均衡



配列探索 - 2分探索（確認）

■ 補題c: 2分探索の2分決定木の高さは$\lfloor \log_2 n \rfloor + 1$

Tの内点数: $\mu(T)$

Tの高さ: $h(T)$

□ 証明:

- T: 2分決定木
- T: 正則均衡2分木
- $\mu(T) = n \implies h(T) = \lfloor \log_2 n \rfloor + 1$ （定理@[1-2]）

■ Algorithm 3.b (2分探索:Binary Search)

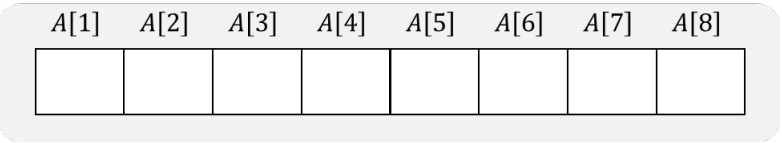
Step 0: Set $i = 1$, $j = n$

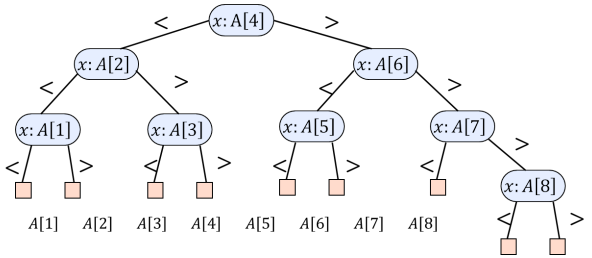
Step 1: If $i > j$ then output NO, and halt

Step 2: Set $m = \left\lfloor \frac{i+j}{2} \right\rfloor$

Step 3: If $A[m] = x$ then output YES, and halt

Step 4: If $A[m] > x$ then set $j = m - 1$
otherwise set $i = m + 1$, and return to Step 1



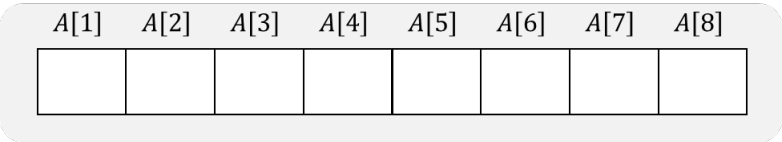


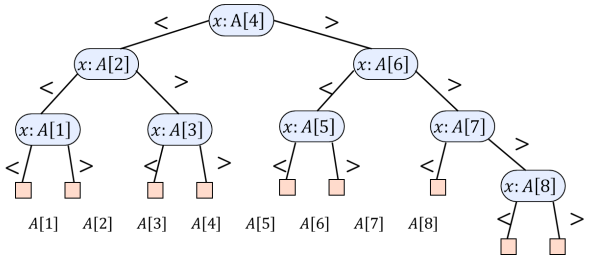
配列探索 - 2分探索（確認）

■ 系： 2分探索の時間計算量は$O(\log n)$

■ 質問
- もっとよくできるか？
- No!

■ Algorithm 3.b （2分探索:Binary Search）
Step 0: Set $i = 1$, $j = n$
Step 1: If $i > j$ then output NO, and halt
Step 2: Set $m = \left\lfloor \frac{i+j}{2} \right\rfloor$
Step 3: If $A[m] = x$ then output YES, and halt
Step 4: If $A[m] > x$ then set $j = m - 1$
otherwise set $i = m + 1$, and return to Step 1



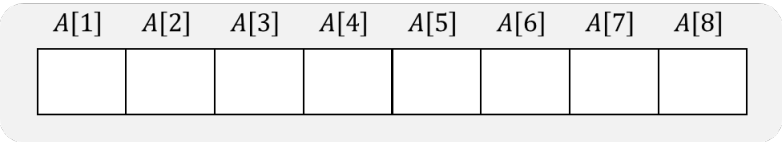


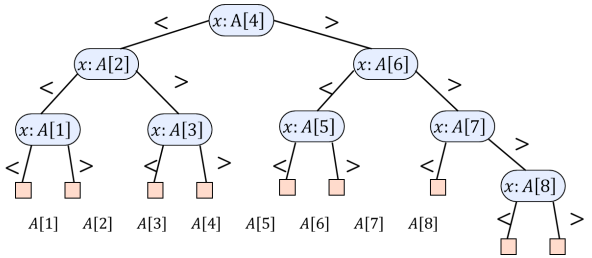
配列探索 - 2分探索（確認）

■ 定理：ORDERED ARRAY SEARCHINGの
時間計算量は少なくとも$\Omega(\log n)$

■ 2分探索
- ORDERED ARRAY SEARCHINGの最適アルゴリズム
  - 2数の比較を用いる場合

■ Algorithm 3.b（2分探索:Binary Search）
Step 0: Set $i=1,\ j=n$
Step 1: If $i>j$ then output NO, and halt
Step 2: Set $m=\left\lfloor\frac{i+j}{2}\right\rfloor$
Step 3: If $A[m]=x$ then output YES, and halt
Step 4: If $A[m]>x$ then set $j=m-1$
otherwise set $i=m+1$, and return to Step 1





配列探索 - 2分探索（確認）

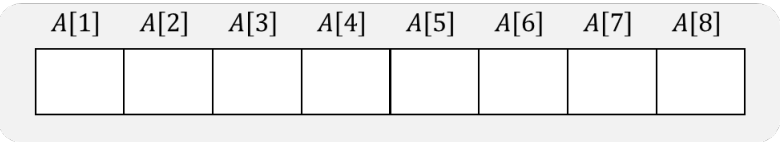
■ 定理：ORDERED ARRAY SEARCHINGの
時間計算量は少なくとも$\Omega(\log n)$

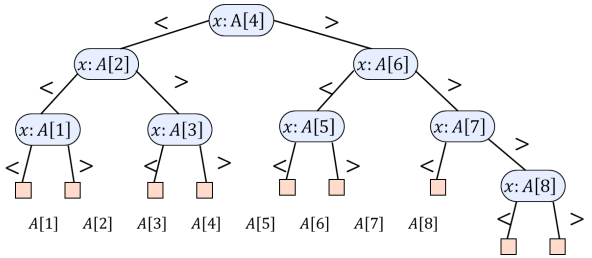
□ 証明：

- ALG：任意のアルゴリズム
- T　：ALG の2分決定木
  - $h(T)$：比較回数
- T は任意の i $(1 \leq i \leq n)$に対し内点 $x:A[i]$ を持つ
- $\mu(T) \geq n$
- $h(T) \geq \log_2(\mu(T)+1) \geq \log_2(n+1) \geq \Omega(\log n)$

定理@[1-2]

Tの内点数：$\mu(T)$
Tの高さ：$h(T)$



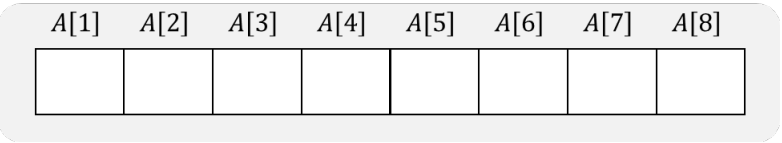


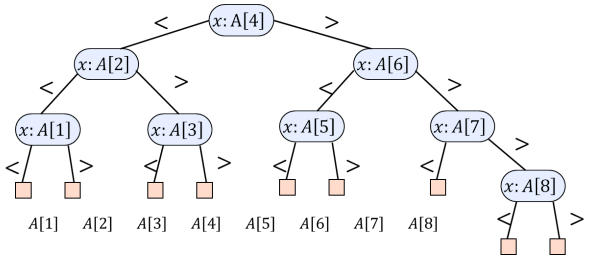
逐次探索 vs. 2分探索（確認）

探索	逐次	2分
準備	$O(n)$	$O(n \log_2 n)$
実行	$O(n)$	$O(\log_2 n)$
1回	$O(n) + O(n)$ $= O(n)$	$O(n \log_2 n) + O(\log_2 n)$ $= O(n \log_2 n)$
n回	$O(n) + n \times O(n)$ $= O(n^2)$	$O(n \log_2 n) + n \times O(\log_2 n)$ $= O(n \log_2 n)$

■ Algorithm 3.a（逐次探索:Sequential Search）
Step 0: Set $i = 1$
Step 1: If $A[i] = x$ then output YES, and halt
Step 2: If $i = n$ then output NO, and halt
Step 3: Set $i = i + 1$, and return to Step 1

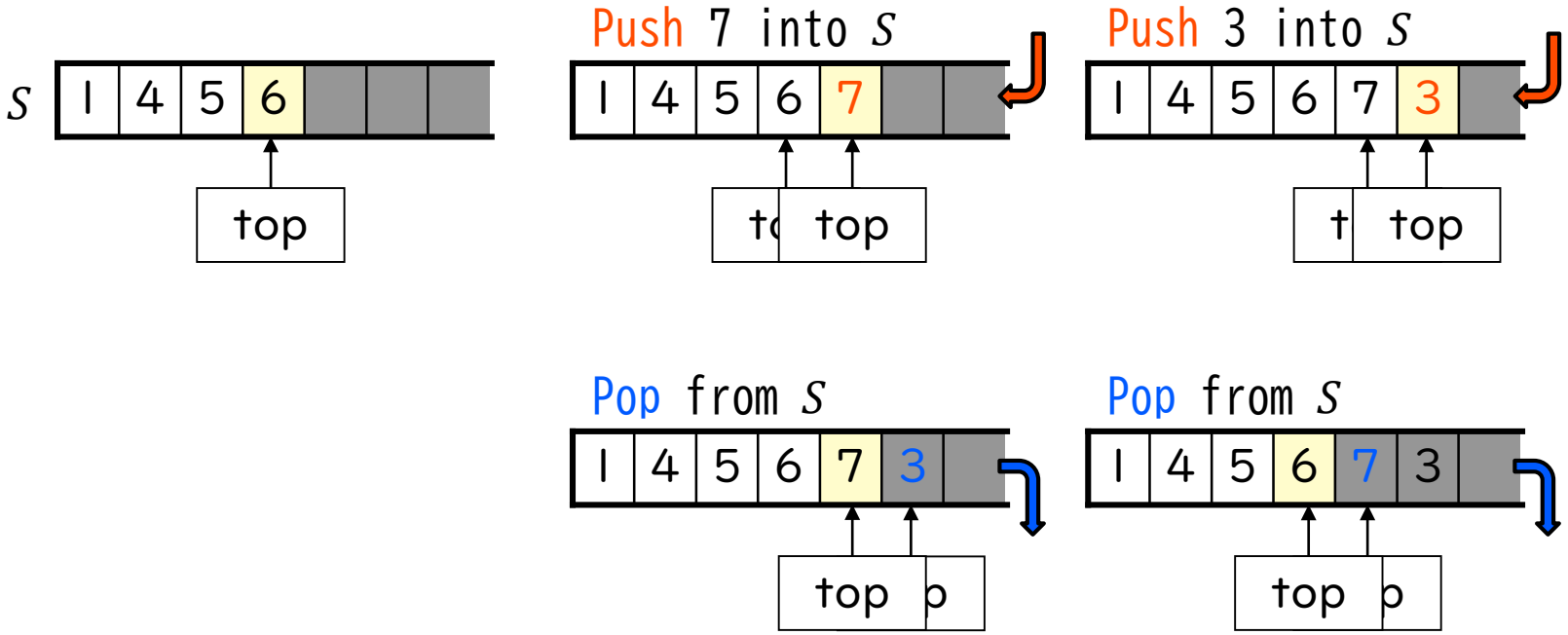
■ Algorithm 3.b（2分探索:Binary Search）
Step 0: Set $i = 1$, $j = n$
Step 1: If $i > j$ then output NO, and halt
Step 2: Set $m = \left\lfloor \frac{i+j}{2} \right\rfloor$
Step 3: If $A[m] = x$ then output YES, and halt
Step 4: If $A[m] > x$ then set $j = m - 1$
otherwise set $i = m + 1$, and return to Step 1





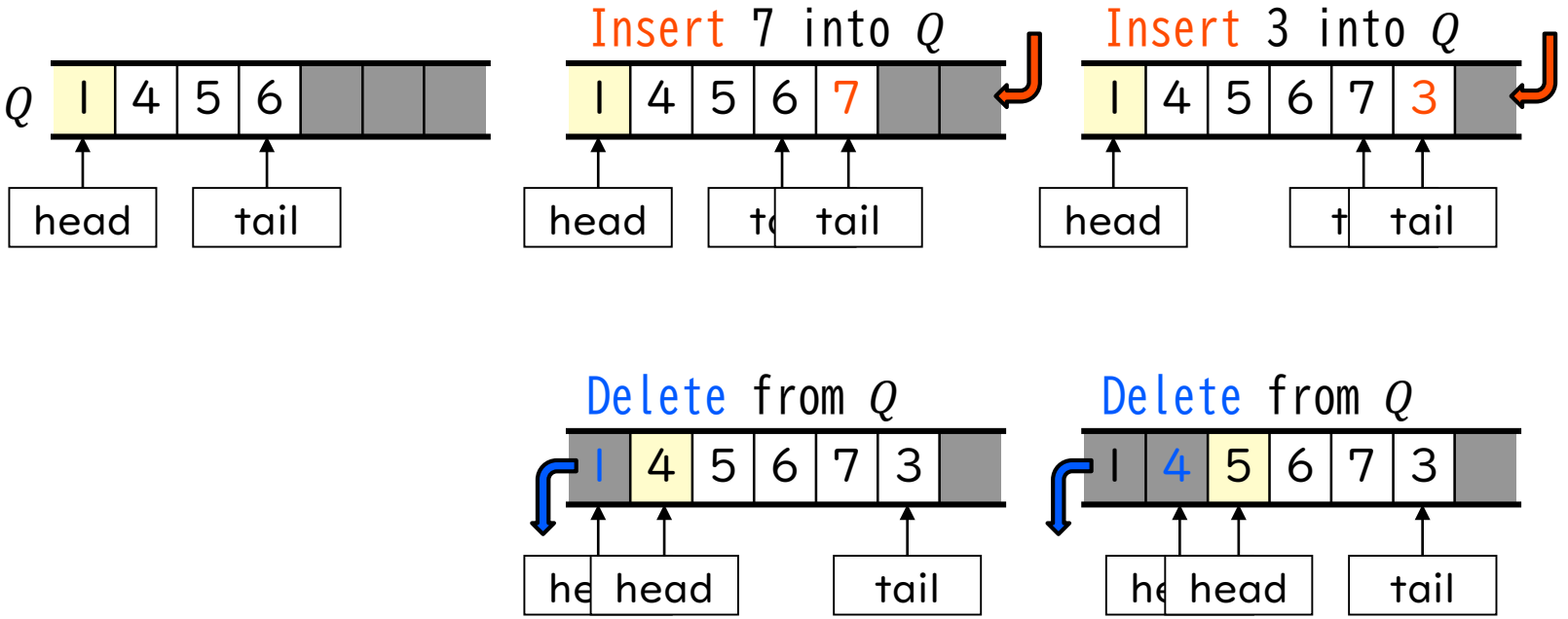
[DS-s] データ構造（確認）

- スタック (Stack)
  - LIFO (Last In First Out)
    - Push and pop operations take $O(1)$ time each



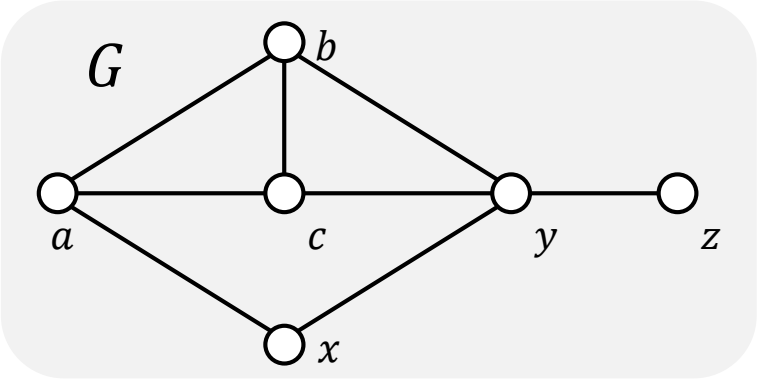
[DS-q] データ構造（確認）

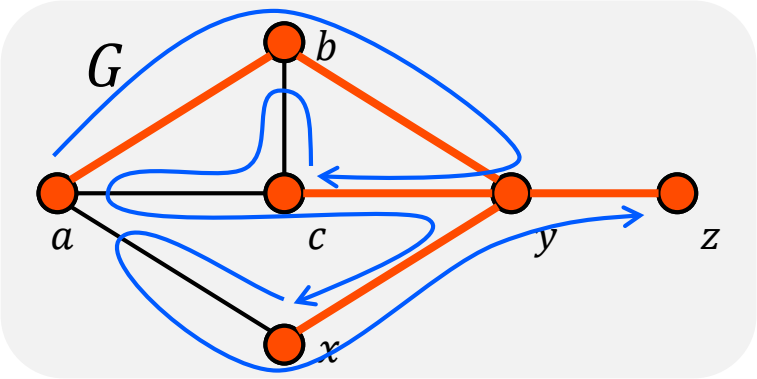
- 待ち行列 (Queue)
  - FIFO (First In First Out)
    - Insert and delete operations take O(1) time each



3-1（1）深さ優先探索アルゴリズム

- グラフの探索
  - 点辺をグラフ構造に基づき探索
  - 様々な情報処理を探索中に行なうことができる
- Depth-First-Search（DFS）
  - 縦型探索
  - とにかく前に進む，突き当ったら戻って別の方向へ





深さ優先探索 (depth-first search)

■ Algorithm 3.1 (DFS)

- 入力：グラフ$G = (V, E)$, 始点$r \in V$
- 出力：変数$f(v)$ $(\forall v \in V)$, 辺の集合X $(\subseteq E)$

Step 0: Set $f(v) \coloneqq 0, p(v) \coloneqq v, \forall v \in V$ and set $X \coloneqq \emptyset, Y \coloneqq E$

Step 1: Set $s \coloneqq r$

Step 2: Set $f(s) \coloneqq 1$

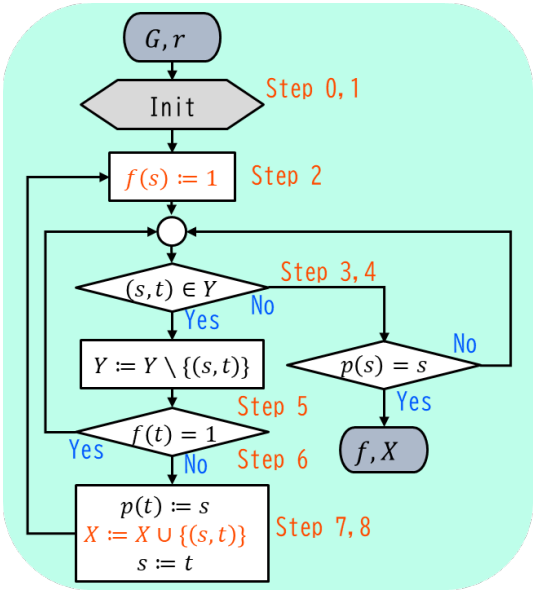
Step 3,4: If no incident edge of s in Y then if $p(s) = s$ then output f and X, and halt else set $s \coloneqq p(s)$ and return to Step 3

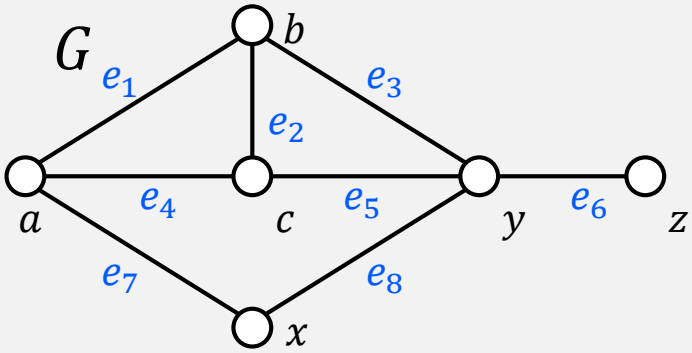
Step 5: Pick an arbitrary edge (s,t) in Y and set $Y \coloneqq Y \setminus \{(s,t)\}$

Step 6: If $f(t) = 1$ then return to Step 3

Step 7: Set $p(t) \coloneqq s$, $X \coloneqq X \cup \{(s,t)\}$

Step 8: Set $s \coloneqq t$ and return to Step 2





深さ優先探索 (depth-first search)

■ Algorithm 3.1 (DFS)

- 入力：グラフ$G=(V,E)$，始点$r \in V$
- 出力：変数$f(v)$ $(\forall v \in V)$，辺の集合X $(\subseteq E)$

⇨ Step 0: Set $f(v) \coloneqq 0, p(v) \coloneqq v, \forall v \in V$ and set $X \coloneqq \emptyset, Y \coloneqq E$

⇨ Step 1: Set $s \coloneqq r$

⇨ Step 2: Set $f(s) \coloneqq 1$

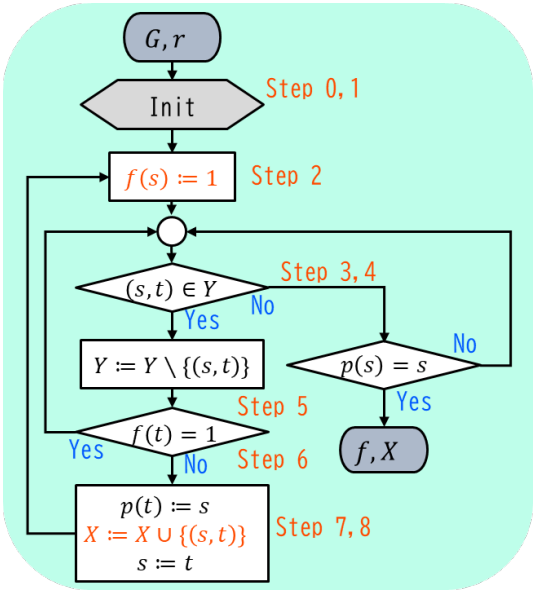
Step 3,4: If no incident edge of s in Y then if $p(s)=s$ then output f and X, and halt else set $s \coloneqq p(s)$ and return to Step 3

Step 5: Pick an arbitrary edge (s,t) in Y and set $Y \coloneqq Y \setminus \{(s,t)\}$

Step 6: If $f(t)=1$ then return to Step 3

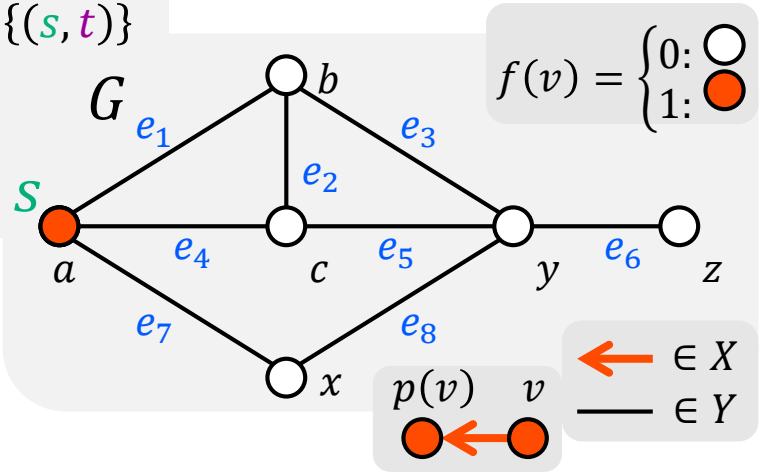
Step 7: Set $p(t) \coloneqq s$, $X \coloneqq X \cup \{(s,t)\}$

Step 8: Set $s \coloneqq t$ and return to Step 2



$X = \{$ $\}$

$Y = \{ e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8 \}$



深さ優先探索 (depth-first search)

■ Algorithm 3.1 (DFS)

- 入力：グラフ$G=(V,E)$，始点$r \in V$
- 出力：変数$f(v)$ $(\forall v \in V)$，辺の集合X $(\subseteq E)$

Step 0: Set $f(v) \coloneqq 0, p(v) \coloneqq v, \forall v \in V$ and set $X \coloneqq \emptyset, Y \coloneqq E$

Step 1: Set $s \coloneqq r$

Step 2: Set $f(s) \coloneqq 1$

⇨ Step 3,4: If no incident edge of s in Y then if $p(s)=s$ then output f and X, and halt else set $s \coloneqq p(s)$ and return to Step 3

Step 5: Pick an arbitrary edge (s,t) in Y and set $Y \coloneqq Y \setminus \{(s,t)\}$

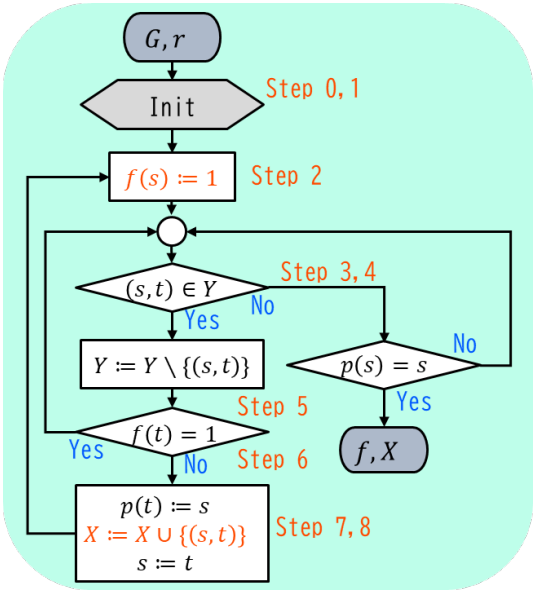
Step 6: If $f(t)=1$ then return to Step 3

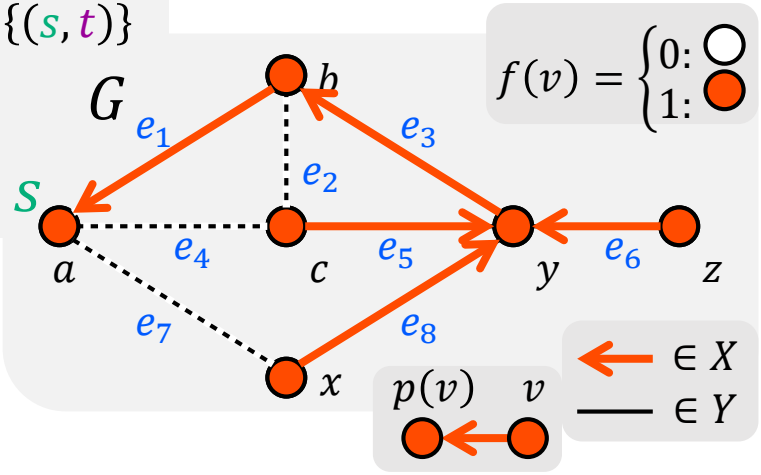
Step 7: Set $p(t) \coloneqq s$, $X \coloneqq X \cup \{(s,t)\}$

Step 8: Set $s \coloneqq t$ and return to Step 2

$X = \{ e_1, e_3, e_5, e_6, e_8 \}$

$Y = \{$ $\}$





深さ優先探索 (depth-first search)

■ 定理(Theorem3.1): DFSの時間計算量は$O(n+m)$

□ 証明

$n = |V|, m = |E|$

■ Algorithm 3.1 (DFS)

- 入力:グラフ$G = (V, E)$, 始点$r \in V$
- 出力:変数$f(v)$ $(\forall v \in V)$, 辺の集合X $(\subseteq E)$

Step 0: Set $f(v) := 0, p(v) := v, \forall v \in V$ and set $X := \emptyset, Y := E$ — $O(n+m)$

Step 1: Set $s := r$ — $O(1)$

Step 2: Set $f(s) := 1$ — $O(1)$

Step 3,4: If no incident edge of s in Y then — $O(1)$
  if $p(s) = s$ then output f and X, and halt
  else set $s := p(s)$ and return to Step 3

Step 5: Pick an arbitrary edge (s,t) in Y and set $Y := Y \setminus \{(s,t)\}$ — $O(1)$

Step 6: If $f(t) = 1$ then return to Step 3 — $O(1)$

Step 7: Set $p(t) := s$, $X := X \cup \{(s,t)\}$ — $O(1)$

Step 8: Set $s := t$ and return to Step 2 — $O(1)$

高々n回

各点v回に対し高々$\deg_G(v)+1$回

全部で$O(m)$回

出力:$O(n)$ 1回

高々m回

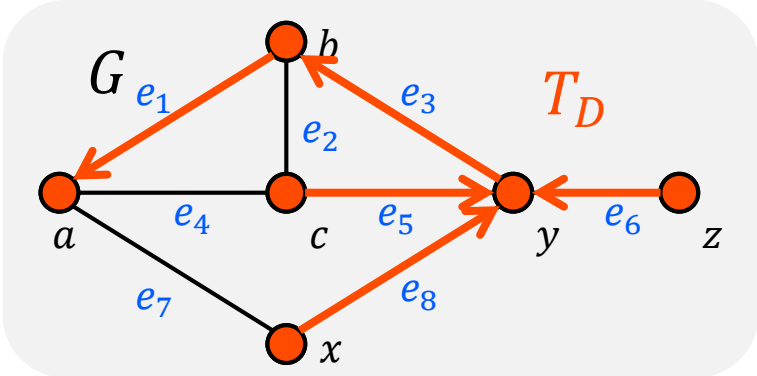
深さ優先探索 (depth-first search)

■ 定理(Theorem3.2)：DFSの出力$T_D(=G[X])$は木

□ 証明

- 任意の2点は連結
  - 任意の点$v \in V(T_D)$は始点rとT_Dで連結
    - $p(v)$をたどると始点に到達：$(r, \dots, p(p(v)), p(v), v)$は$(r, v)$-路
    - $\forall u, v \in V(T_D)$, rを経由する(u, v)-ウォーク $\Rightarrow$ (u, v)-路
- $|E(T_D)| = |V(T_D)| - 1$
  - 各辺を有向辺と考えたとき，$v \in V(T_D) \setminus \{r\}$を始点とする辺は1本
  - $\forall v \in V(T_D) \setminus \{r\}, (v, p(v)) \in E(T_D)$

✓ T_Dは木 ∵木の性質@[1-2] ■



➢ T_D ： DFS木

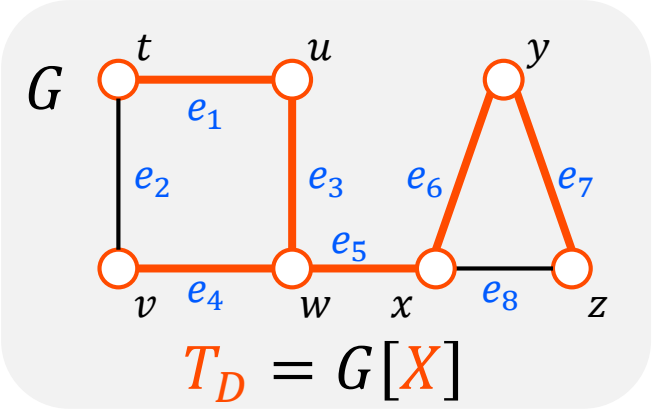
連結性判定（connectivity）

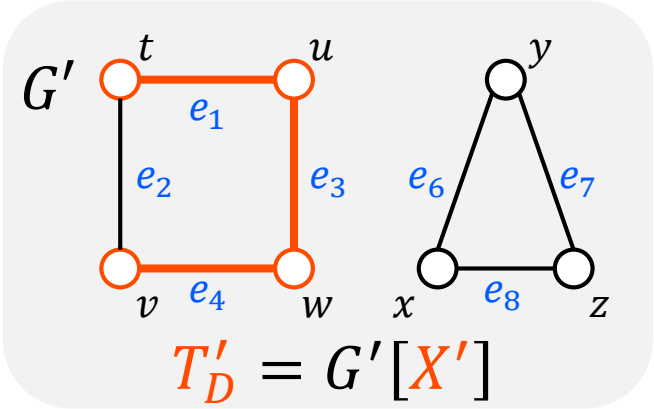
■ 定理：グラフGが連結である ⇔
DFSがすべての点vに対して$f(v) = 1$と出力する

■ 定理：連結グラフであるか否か線形時間で判定できる

□ 証明：
- DFSで判定できる
- DFSの時間計算量は$O(n + m)$ ■

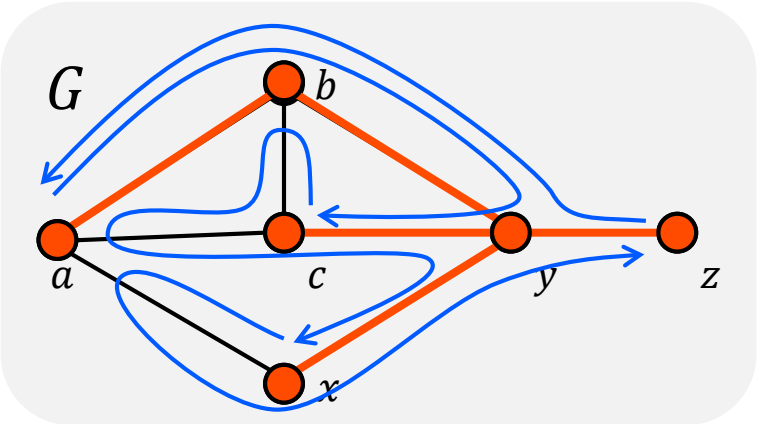
✓ グラフG連結ならばT_Dは全域木





深さ優先探索(DFS)の応用

- 探索目的に応じ，探索中に様々な情報処理
  - 探索中に各点は何度か訪問される
- グラフ構造に基づく点の順序づけ，処理の順序
  - 前順序(pre-order)
    - 最初の訪問の早い順
      - 先に進みながら各点に情報伝達
  - 後順序(post-order)
    - 最後の訪問の早い順
      - 根に戻りながら情報収集や情報処理



前順序・後順序 (pre- and post-order numbering)

■ Algorithm 3.3 (pre-post-order)

- 入力：グラフ$G = (V, E)$, 始点$r \in V$
- 出力：前順序$k(v)$, 後順序$k'(v)$

Step 0: Set $\forall v \in V, k(v) \coloneqq k'(v) \coloneqq 0, p(v) \coloneqq v$ and set $Y \coloneqq E, i \coloneqq j \coloneqq 1$

Step 1: Set $s \coloneqq r$

Step 2: Set $k(s) \coloneqq i, i \coloneqq i + 1$

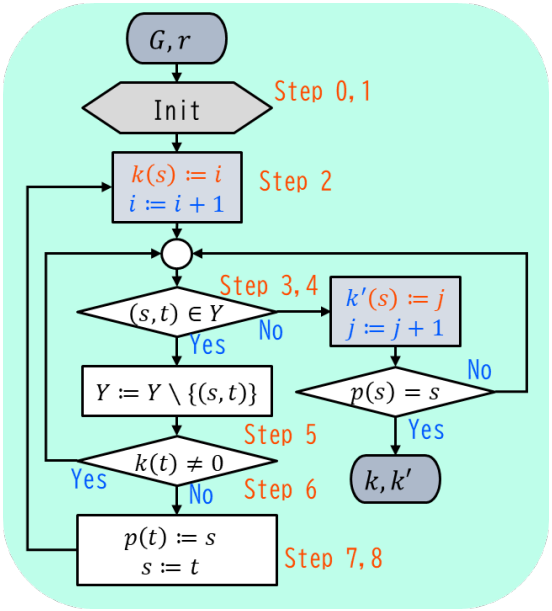
Step 3,4: If no incident edge of s in Y then set $k'(s) \coloneqq j, j \coloneqq j + 1$, and if $p(s) = s$ then output k and k', and halt else set $s \coloneqq p(s)$ and return to Step 3

Step 5: Pick an arbitrary edge (s,t) in Y and set $Y \coloneqq Y \setminus \{(s,t)\}$

Step 6: If $k(t) \neq 0$ then return to Step 3

Step 7: Set $p(t) \coloneqq s$

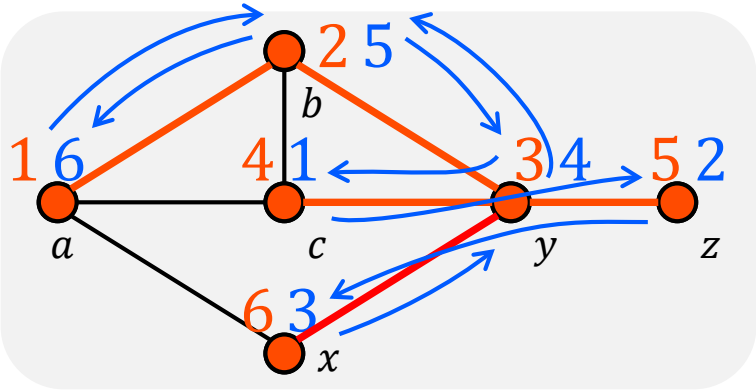
Step 8: Set $s \coloneqq t$ and return to Step 2



■ 定理 (Theorem 3.4): Algorithm3.3の時間計算量は$O(n + m)$

前順序・後順序 (pre- and post-order numbering)

■ 順序付け例



前順序k : $a\ b\ y\ c\ z\ x$

後順序k' : $c\ z\ x\ y\ b\ a$

■ Algorithm 3.3 (pre-post-order)

- 入力：グラフ$G=(V,E)$，始点$r \in V$
- 出力：前順序$k(v)$，後順序$k'(v)$

Step 0: Set $\forall v \in V, k(v) := k'(v) := 0, p(v) := v$ and set $Y := E, i := j := 1$

Step 1: Set $s := r$

Step 2: Set $k(s) := i, i := i+1$

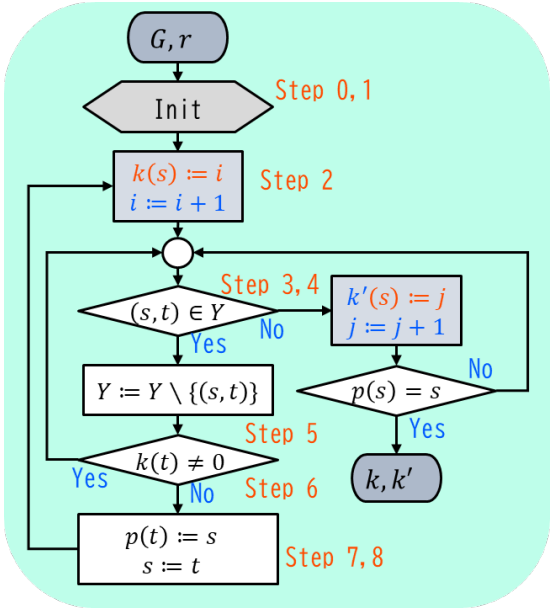
Step 3,4: If no incident edge of s in Y then set $k'(s) := j, j := j+1$, and if $p(s) = s$ then output k and k', and halt else set $s := p(s)$ and return to Step 3

Step 5: Pick an arbitrary edge (s,t) in Y and set $Y := Y \setminus \{(s,t)\}$

Step 6: If $k(t) \neq 0$ then return to Step 3

Step 7: Set $p(t) := s$

Step 8: Set $s := t$ and return to Step 2



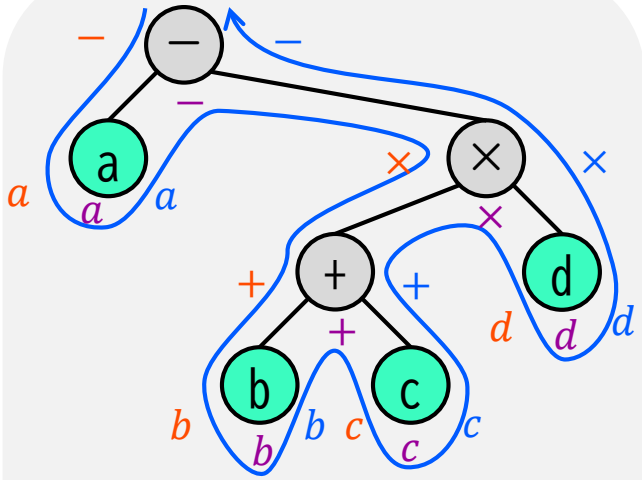
二項演算の表現

- 中置記法(in-order)
  - a operator b ： 演算子(operator)を被演算子(operand)の間に置く
    - 演算順序を明確にするためには括弧が必要
- 前置記法(pre-order)(ポーランド記法)
  - operator a b ： 演算子を被演算子の前に置く
- 後置記法(post-order)(逆ポーランド記法)
  - a b operator ： 演算子を被演算子の後に置く

◆ 二項演算からなる式の2分木表現
- 演算子:内点
- 被演算子:葉

➢ 深さ優先探索で点ラベルを出力
- ✓ 根から左子,右子の順
  - 前順序:前置記法(1回目の訪問:親から探索が到達)
  - 後順序:後置記法(3回目(最後)の訪問:親に戻る)
  - 中順序:中置記法(2回目の訪問:左子から戻り右子を探索前)

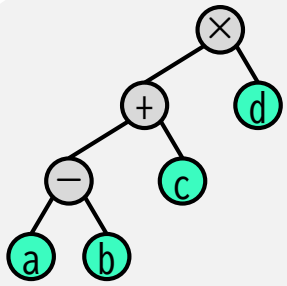
✓ 前置・後置記法は括弧不要で演算順序を規定
- ✓ 2分木は一意に定まる



➢ $\big(a-\big((b+c)\times d\big)\big)$
- ✓ 前置: $-a\times+bcd$
- ✓ 後置: $abc+d\times-$
- ✓ 中置: $a-b+c\times d$
- 一般的な表現
  - $a-(b+c)\times d$

二項演算と2分木

■ 演算数3の場合



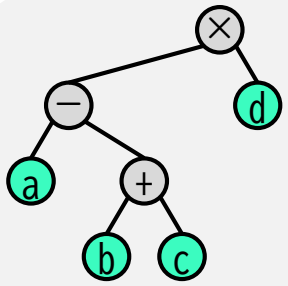
$\big(((a-b)+c)\times d\big)$

$\times + - abcd$

$ab - c + d \times$

$((()))$

$()()()$



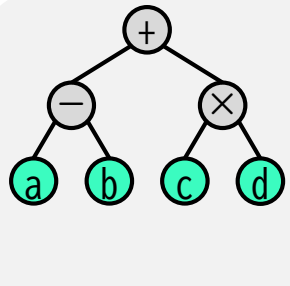
$\big((a-(b+c))\times d\big)$

$\times - a + bcd$

$abc + - d \times$

$(()())$

$(())()$



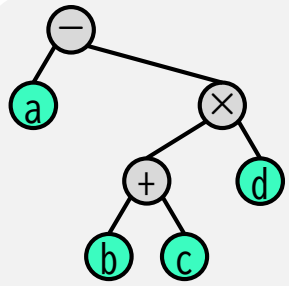
$((a-b)+(c\times d))$

$+ - ab \times cd$

$ab - cd \times +$

$(())()$

$()(())$



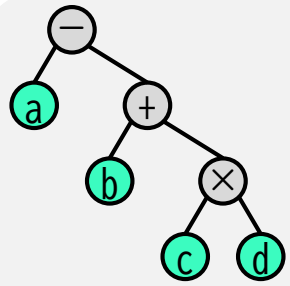
$\big(a-((b+c)\times d)\big)$

$-a \times + bcd$

$abc + d \times -$

$()(())$

$((())())$



$\big(a-(b+(c\times d))\big)$

$-a + b \times cd$

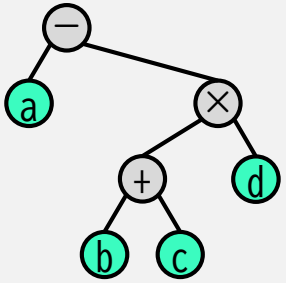
$abcd \times + -$

$()()()$

$((()))$

カタラン数(Catalan number)(おまけ)

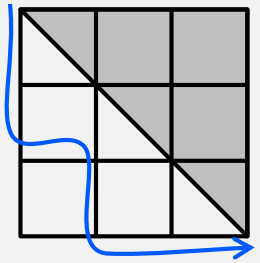
- $C_n = \frac{1}{n+1}\binom{2n}{n} = \binom{2n}{n} - \binom{2n}{n-1}$
  - 内点数nの正則2分木の総数
  - n個の二項演算の式の総数
  - n部屋のスライス構造フロアプランの総数
  - $n \times n$格子上の対角間の下三角最短経路の総数
  - n組の括弧の正しい並べ方の総数



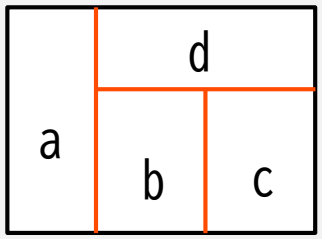
(()())

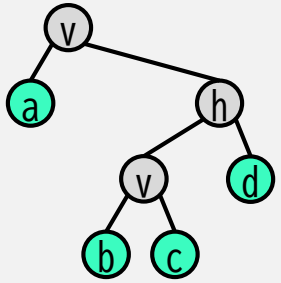
開開閉開閉閉

())(()はダメ



下下右下右右

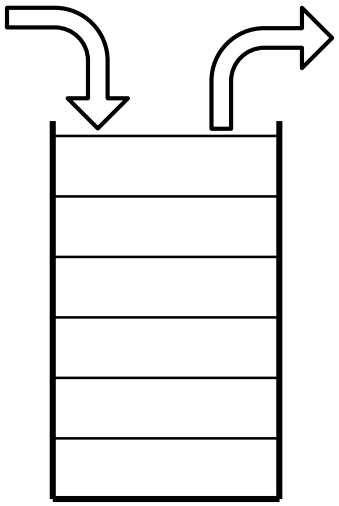


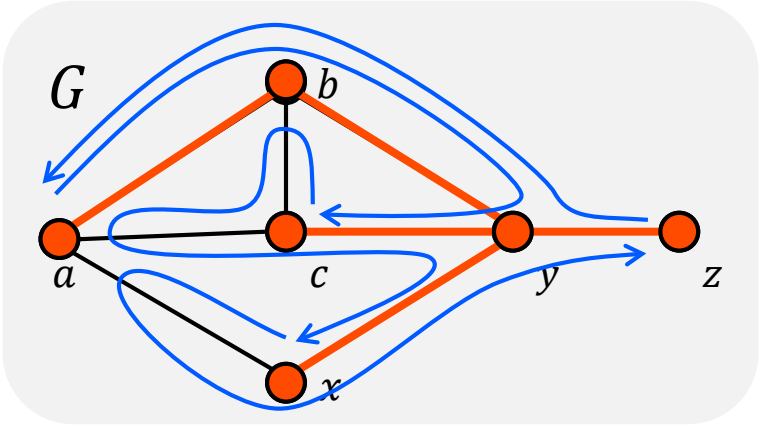


深さ優先探索(DFS)の実現法

■ スタックの利用

- LIFO (Last In First Out)





深さ優先探索(スタック利用)

■ Algorithm 3.2 (stack)

- 入力:グラフ$G=(V,E)$, 始点$r \in V$
- 出力:変数$f(v)$ $(\forall v \in V)$, 辺の集合X $(\subseteq E)$

Step 0: Set $f(v) \coloneqq 0, \forall v \in V$, and set $X \coloneqq \emptyset, Y \coloneqq E$

Step 1: Set $s \coloneqq r$

Step 2: Set $f(s) \coloneqq 1$

Step 3: For all edge $(s,v) \in Y$,
Push$(S,(s,v))$ and set $Y \coloneqq Y \setminus \{(s,v)\}$

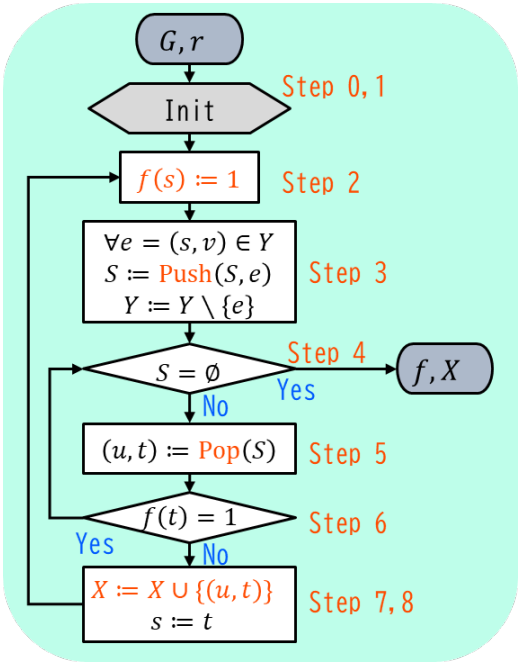
Step 4: If $S=\emptyset$ then output f and X, and halt

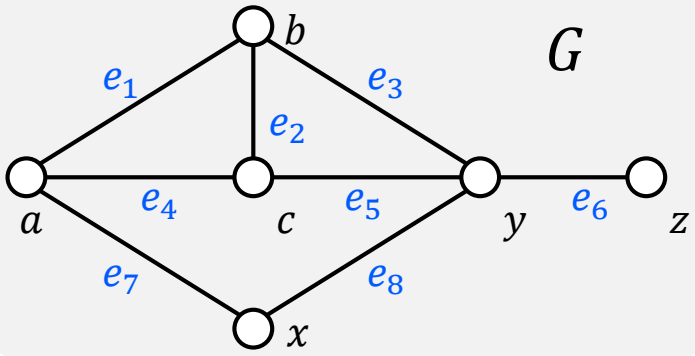
Step 5: Set $(u,t) \coloneqq \mathrm{Pop}(S)$

Step 6: If $f(t)=1$ then return to Step 4

Step 7: Set $X \coloneqq X \cup \{(u,t)\}$

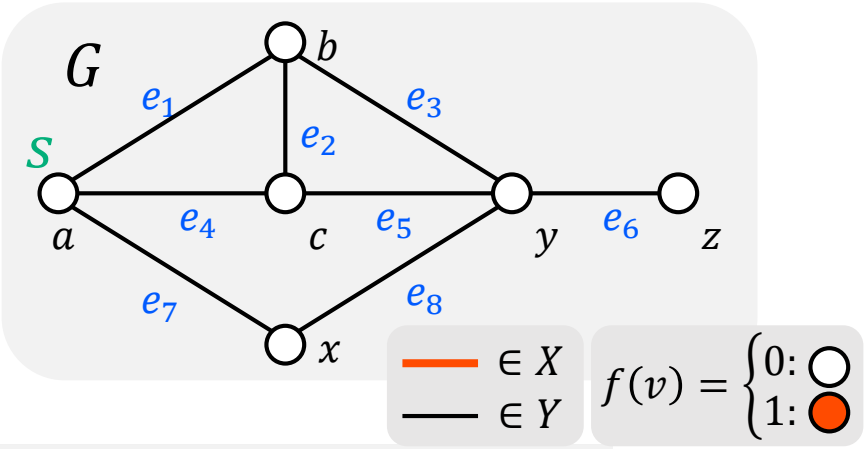
Step 8: Set $s \coloneqq t$ and return to Step 2

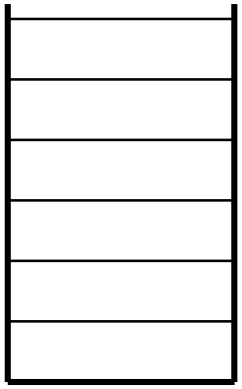




深さ優先探索(スタック利用)

■ DFS(stack) 探索例

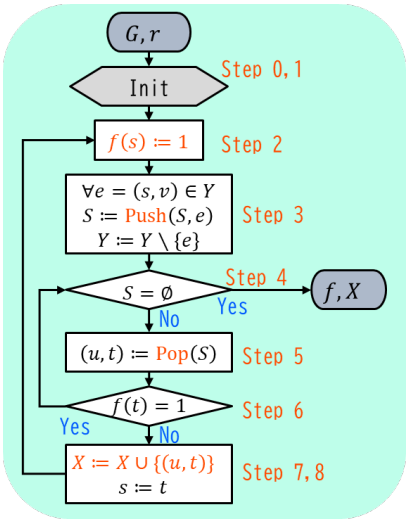




■ Algorithm 3.2 (stack)

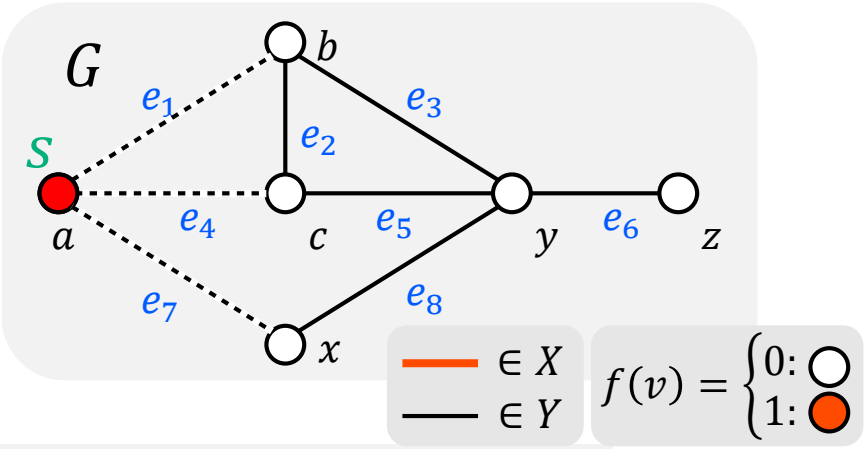
- 入力:グラフ$G = (V, E)$, 始点$r \in V$
- 出力:変数$f(v)$ $(\forall v \in V)$, 辺の集合X $(\subseteq E)$

Step 0: Set $f(v) \coloneqq 0, \forall v \in V$, and set $X \coloneqq \emptyset, Y \coloneqq E$
Step 1: Set $s \coloneqq r$
Step 2: Set $f(s) \coloneqq 1$
Step 3: For all edge $(s, v) \in Y$,
Push$(S, (s, v))$ and set $Y \coloneqq Y \setminus \{(s, v)\}$
Step 4: If $S = \emptyset$ then output f and X, and halt
Step 5: Set $(u, t) \coloneqq \mathrm{Pop}(S)$
Step 6: If $f(t) = 1$ then return to Step 4
Step 7: Set $X \coloneqq X \cup \{(u, t)\}$
Step 8: Set $s \coloneqq t$ and return to Step 2



深さ優先探索(スタック利用)

■ DFS(stack) 探索例

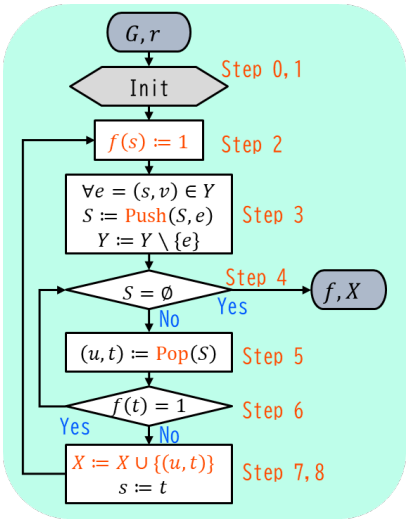


$e_1 = (a, b)$
$e_4 = (a, c)$
$e_7 = (a, x)$

■ Algorithm 3.2 (stack)

- 入力:グラフ$G = (V, E)$, 始点$r \in V$
- 出力:変数$f(v)$ $(\forall v \in V)$, 辺の集合X $(\subseteq E)$

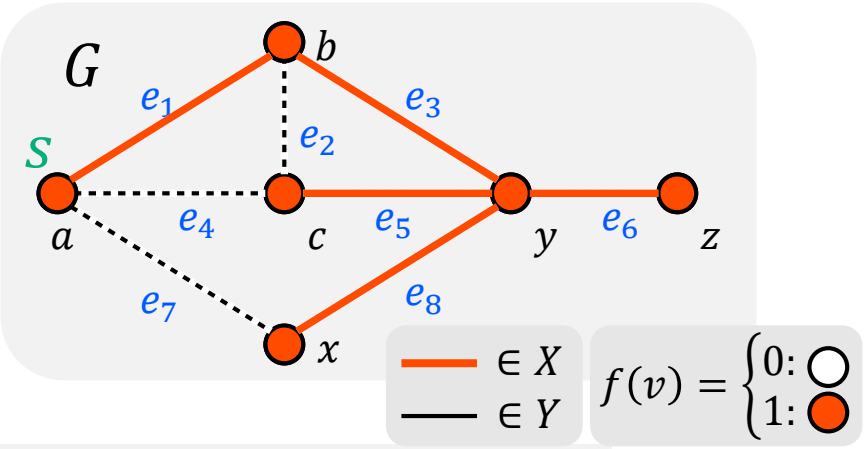
Step 0: Set $f(v) \coloneqq 0, \forall v \in V$, and set $X \coloneqq \emptyset, Y \coloneqq E$
Step 1: Set $s \coloneqq r$
Step 2: Set $f(s) \coloneqq 1$
Step 3: For all edge $(s, v) \in Y$,
$\mathrm{Push}(S, (s, v))$ and set $Y \coloneqq Y \setminus \{(s, v)\}$
Step 4: If $S = \emptyset$ then output f and X, and halt
Step 5: Set $(u, t) \coloneqq \mathrm{Pop}(S)$
Step 6: If $f(t) = 1$ then return to Step 4
Step 7: Set $X \coloneqq X \cup \{(u, t)\}$
Step 8: Set $s \coloneqq t$ and return to Step 2

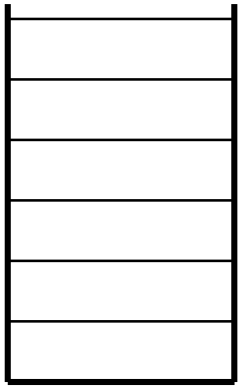


深さ優先探索(スタック利用)

■ DFS(stack) 探索例

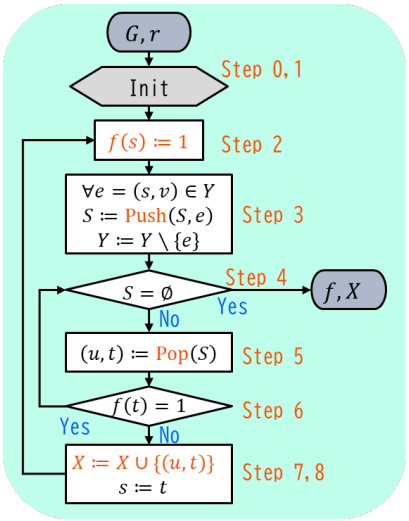
$e_8 = (y, x)$
$e_7 = (a, x)$ $e_6 = (y, z)$
$e_4 = (a, c)$ $e_5 = (y, c)$
$e_2 = (b, c)$ $e_3 = (b, y)$
$e_1 = (a, b)$





■ Algorithm 3.2 (stack)
- 入力:グラフ$G = (V, E)$, 始点$r \in V$
- 出力:変数$f(v)$ $(\forall v \in V)$, 辺の集合X $(\subseteq E)$

Step 0: Set $f(v) \coloneqq 0, \forall v \in V$, and set $X \coloneqq \emptyset, Y \coloneqq E$
Step 1: Set $s \coloneqq r$
Step 2: Set $f(s) \coloneqq 1$
Step 3: For all edge $(s, v) \in Y$,
Push$(S, (s, v))$ and set $Y \coloneqq Y \setminus \{(s, v)\}$
⇨ Step 4: If $S = \emptyset$ then output f and X, and halt
Step 5: Set $(u, t) \coloneqq \mathrm{Pop}(S)$
Step 6: If $f(t) = 1$ then return to Step 4
Step 7: Set $X \coloneqq X \cup \{(u, t)\}$
Step 8: Set $s \coloneqq t$ and return to Step 2



深さ優先探索(スタック利用)

■ 定理(Theorem 3.3):DFS(stack)の時間計算量は$O(n+m)$

□ 証明

$n = |V|, m = |E|$

■ Algorithm 3.2 (stack)

- 入力:グラフ$G = (V, E)$, 始点$r \in V$
- 出力:変数$f(v)$ $(\forall v \in V)$, 辺の集合X $(\subseteq E)$

Step 0: Set $f(v) \coloneqq 0, \forall v \in V$, and set $X \coloneqq \emptyset, Y \coloneqq E$ $O(n+m)$

Step 1: Set $s \coloneqq r$ $O(1)$

Step 2: Set $f(s) \coloneqq 1$ $O(1)$

Step 3: For all edge $(s, v) \in Y$, $O(\deg_G(v))$

Push$(S, (s, v))$ and set $Y \coloneqq Y \setminus \{(s, v)\}$

Step 4: If $S = \emptyset$ then output f and X, and halt $O(1)$

Step 5: Set $(u, t) \coloneqq \mathrm{Pop}(S)$ $O(1)$

Step 6: If $f(t) = 1$ then return to Step 4 $O(1)$

Step 7: Set $X \coloneqq X \cup \{(u, t)\}$ $O(1)$

Step 8: Set $s \coloneqq t$ and return to Step 2 $O(1)$

高々n回

各点v回に対し高々1回

全部で$O(m)$

出力:$O(n)$

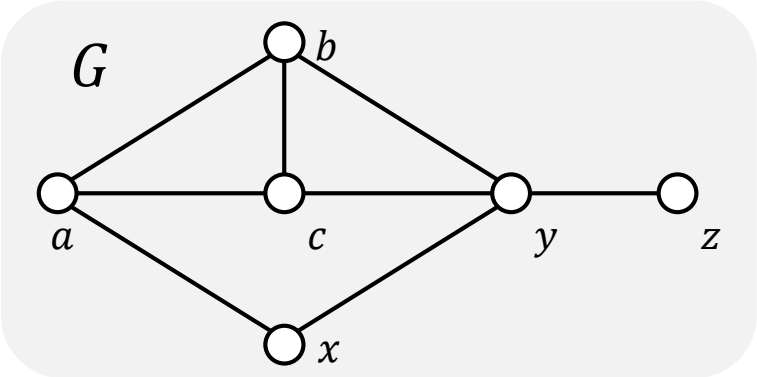
高々1回

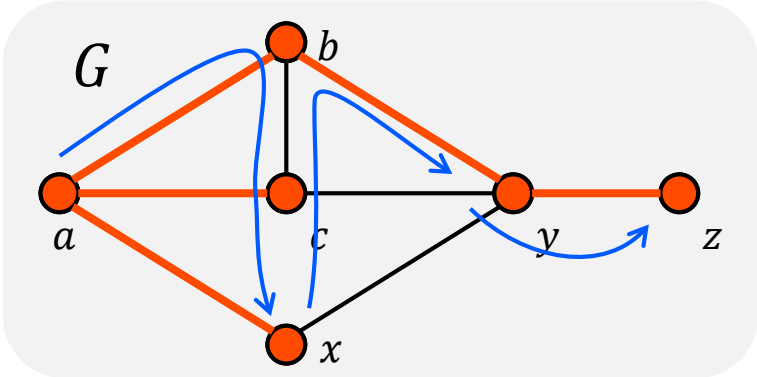
高々$m+1$回

高々m回

3-1（2）幅優先探索アルゴリズム

- グラフの探索
  - 点辺をグラフ構造に基づき探索
  - 様々な情報処理を探索中に行なうことができる
- Breadth-First-Search（BFS）
  - 横型探索
  - 近くから順に探索





幅優先探索 (breadth-first search)

■ Algorithm 3.4 (BFS)

- 入力:グラフ$G=(V,E)$, 始点$r \in V$
- 出力:変数$f(v)$ $(\forall v \in V)$

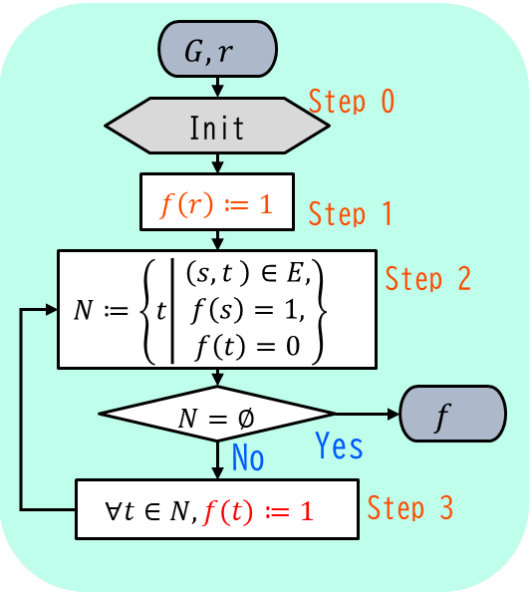
Step 0: Set $f(v) \coloneqq 0, \forall v \in V$

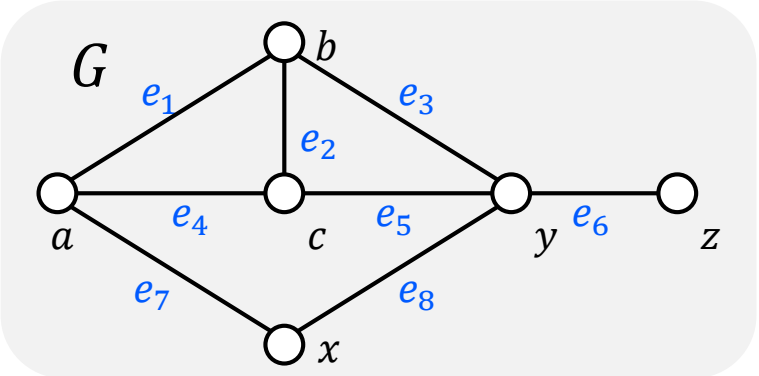
Step 1: Set $f(r) \coloneqq 1$

Step 2: Set $N = \{t | (s,t) \in E, f(s) = 1, f(t) = 0\}$

Step 2.1: If $N = \emptyset$ then output f, and halt

Step 3: Set $f(t) = 1$, $\forall v \in N$ and return to Step 2





幅優先探索 (breadth-first search)

■ Algorithm 3.4 (BFS)

- 入力:グラフ$G=(V,E)$, 始点$r \in V$
- 出力:変数$f(v)$ $(\forall v \in V)$

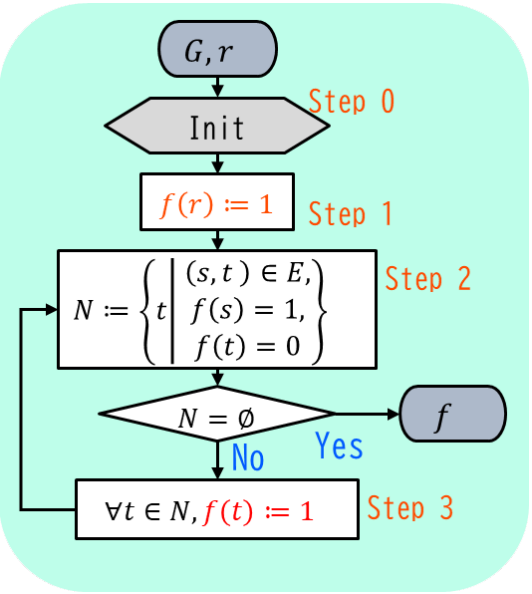
Step 0: Set $f(v) \coloneqq 0, \forall v \in V$

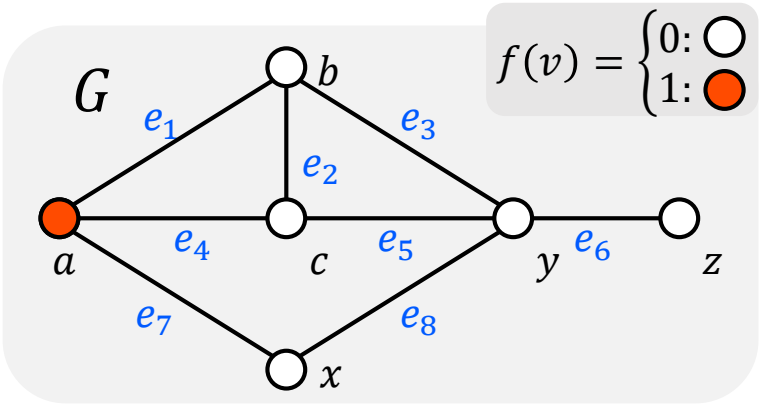
⇨ Step 1: Set $f(r) \coloneqq 1$

Step 2: Set $N = \{t | (s,t) \in E, f(s) = 1, f(t) = 0\}$

Step 2.1: If $N = \emptyset$ then output f, and halt

Step 3: Set $f(t) = 1$, $\forall v \in N$ and return to Step 2

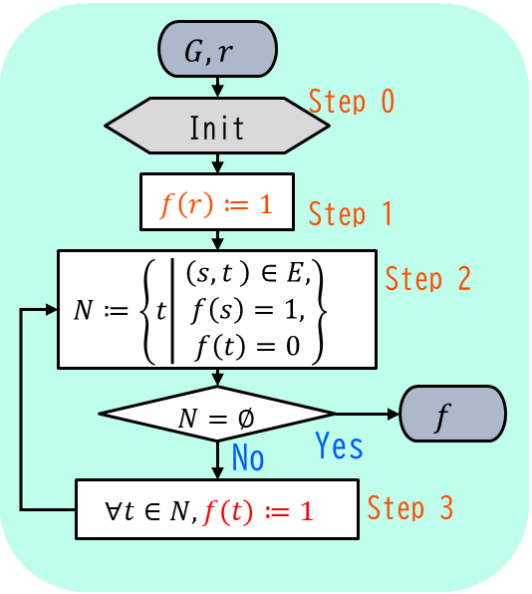




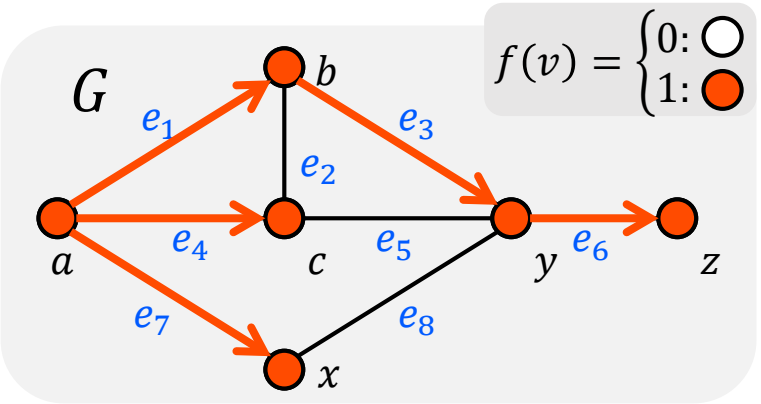
幅優先探索 (breadth-first search)

■ Algorithm 3.4 (BFS)
- 入力：グラフ$G=(V,E)$，始点$r \in V$
- 出力：変数$f(v)$ $(\forall v \in V)$

Step 0: Set $f(v) \coloneqq 0, \forall v \in V$
Step 1: Set $f(r) \coloneqq 1$
Step 2: Set $N = \{t | (s,t) \in E, f(s)=1, f(t)=0\}$
⇨ Step 2.1: If $N = \emptyset$ then output f, and halt
Step 3: Set $f(t)=1$, $\forall v \in N$ and return to Step 2



1: $N = \{b, c, x\}$
2: $N = \{y\}$
3: $N = \{z\}$
4: $N = \{\}$



■ Step 2
- 1回あたりの計算量は？
- 全体の計算量は？
- $O(n+m)$を達成できないとDFSより遅いが…

➢ BFSの時間計算量は$O(n+m)$

幅優先探索(待ち行列利用)

■ Algorithm 3.5 (queue)

- 入力:グラフ$G=(V,E)$, 始点$r \in V$
- 出力:変数$f(v)$ $(\forall v \in V)$, 辺の集合X $(\subseteq E)$

Step 0: Set $f(v) \coloneqq 0, \forall v \in V$, and set $X \coloneqq \emptyset, Y \coloneqq E$

Step 1: Set $s \coloneqq r$

Step 2: Set $f(s) \coloneqq 1$

Step 3: For all edge $(s,v) \in Y$,
Insert$(Q,(s,v))$ and set $Y \coloneqq Y \setminus \{(s,v)\}$

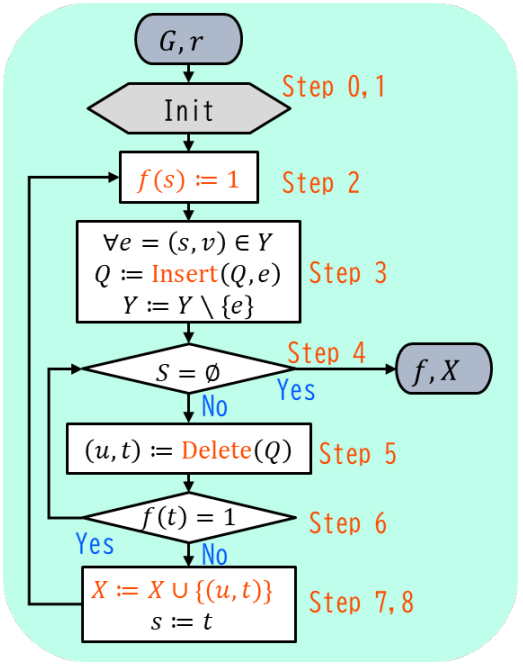
Step 4: If $Q=\emptyset$ then output f and X, and halt

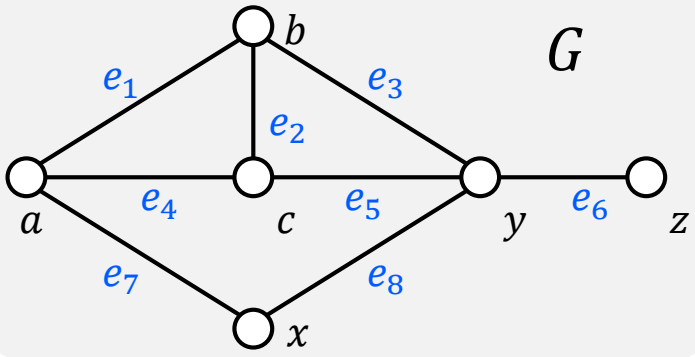
Step 5: Set $(u,t) \coloneqq$ Delete(Q)

Step 6: If $f(t)=1$ then return to Step 4

Step 7: Set $X \coloneqq X \cup \{(u,t)\}$

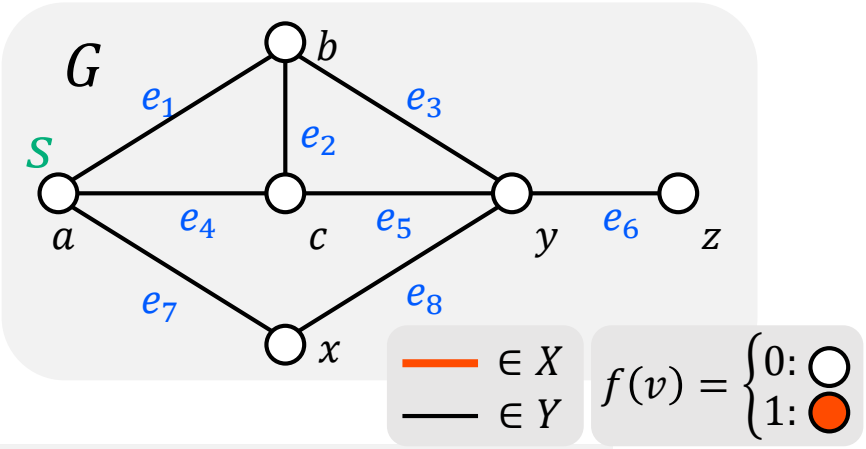
Step 8: Set $s \coloneqq t$ and return to Step 2

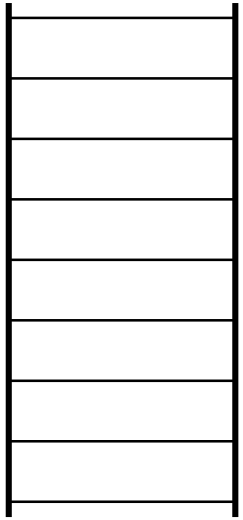




幅優先探索(待ち行列利用)

■ BFS(queue) 探索例

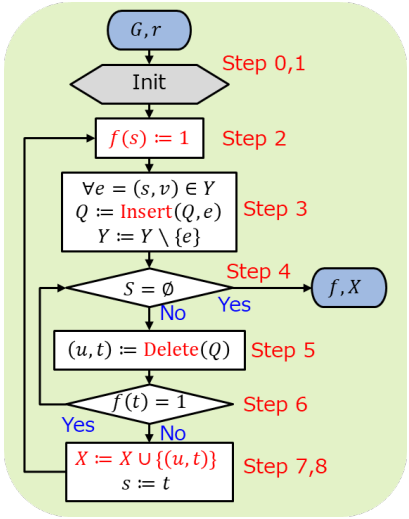




■ Algorithm 3.5 (queue)

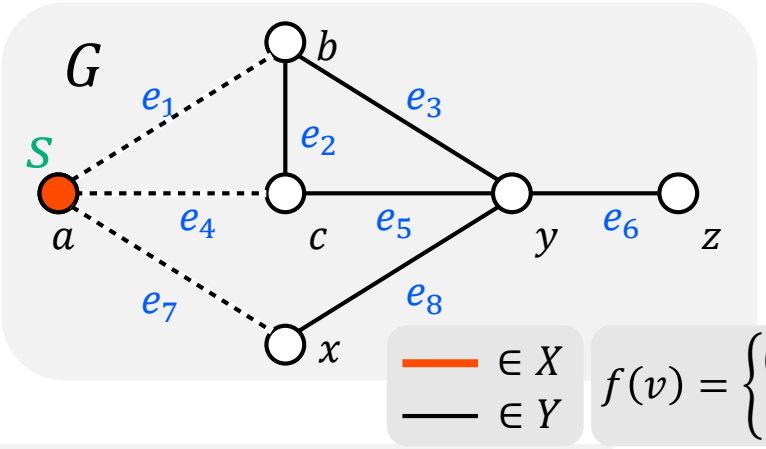
- 入力:グラフ$G = (V, E)$, 始点$r \in V$
- 出力:変数$f(v)$ $(\forall v \in V)$, 辺の集合X $(\subseteq E)$

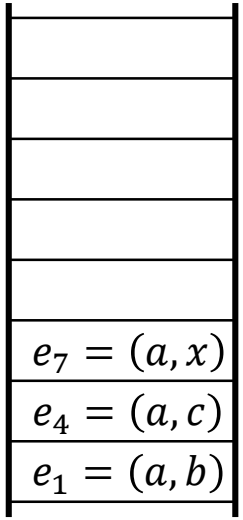
Step 0: Set $f(v) \coloneqq 0, \forall v \in V$, and set $X \coloneqq \emptyset, Y \coloneqq E$
Step 1: Set $s \coloneqq r$
Step 2: Set $f(s) \coloneqq 1$
Step 3: For all edge $(s, v) \in Y$,
Insert$(Q, (s, v))$ and set $Y \coloneqq Y \setminus \{(s, v)\}$
Step 4: If $Q = \emptyset$ then output f and X, and halt
Step 5: Set $(u, t) \coloneqq$ Delete(Q)
Step 6: If $f(t) = 1$ then return to Step 4
Step 7: Set $X \coloneqq X \cup \{(u, t)\}$
Step 8: Set $s \coloneqq t$ and return to Step 2



幅優先探索(待ち行列利用)

■ BFS(queue) 探索例

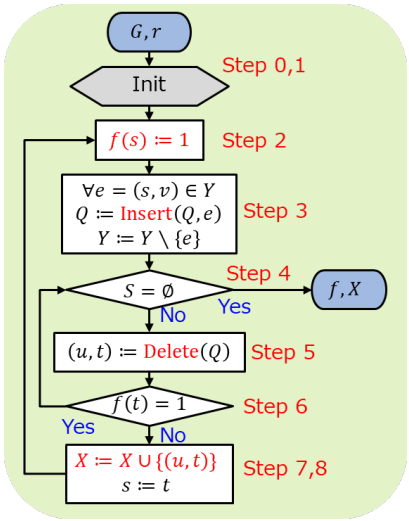




■ Algorithm 3.5 (queue)

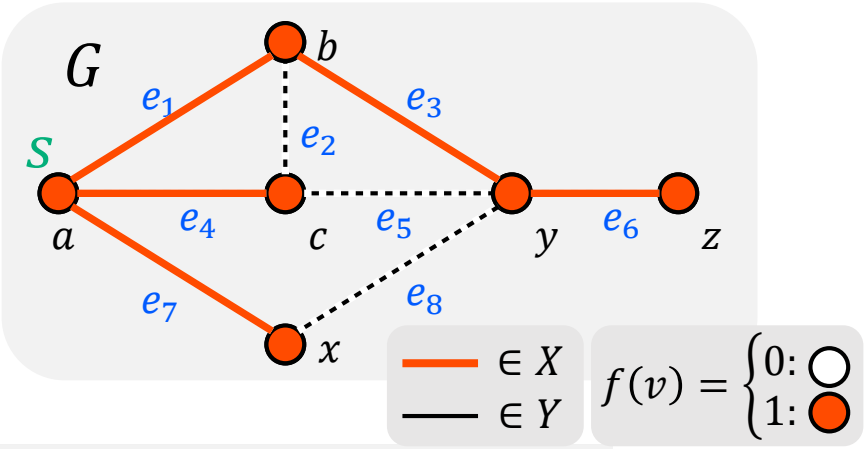
- 入力:グラフ$G = (V, E)$, 始点$r \in V$
- 出力:変数$f(v)$ $(\forall v \in V)$, 辺の集合X $(\subseteq E)$

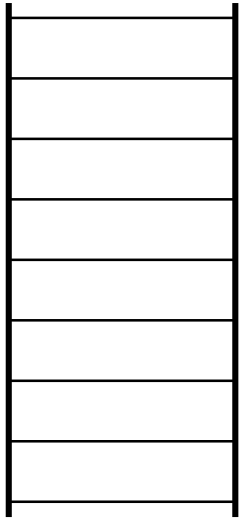
Step 0: Set $f(v) \coloneqq 0, \forall v \in V$, and set $X \coloneqq \emptyset, Y \coloneqq E$
Step 1: Set $s \coloneqq r$
Step 2: Set $f(s) \coloneqq 1$
Step 3: For all edge $(s, v) \in Y$,
Insert$(Q, (s, v))$ and set $Y \coloneqq Y \setminus \{(s, v)\}$
Step 4: If $Q = \emptyset$ then output f and X, and halt
Step 5: Set $(u, t) \coloneqq$ Delete(Q)
Step 6: If $f(t) = 1$ then return to Step 4
Step 7: Set $X \coloneqq X \cup \{(u, t)\}$
Step 8: Set $s \coloneqq t$ and return to Step 2



幅優先探索(待ち行列利用)

■ BFS(queue) 探索例

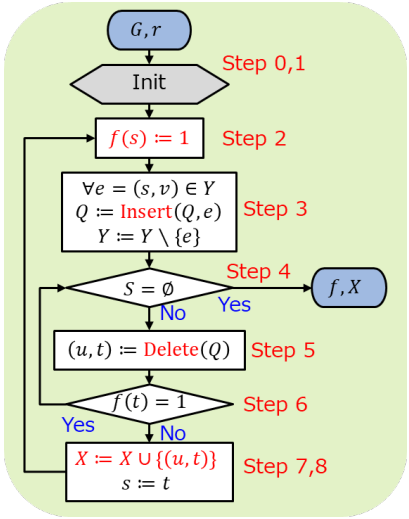




■ Algorithm 3.5 (queue)

- 入力:グラフ$G=(V,E)$, 始点$r \in V$
- 出力:変数$f(v)$ $(\forall v \in V)$, 辺の集合X $(\subseteq E)$

Step 0: Set $f(v) \coloneqq 0, \forall v \in V$, and set $X \coloneqq \emptyset, Y \coloneqq E$
Step 1: Set $s \coloneqq r$
Step 2: Set $f(s) \coloneqq 1$
Step 3: For all edge $(s,v) \in Y$,
Insert$(Q,(s,v))$ and set $Y \coloneqq Y \setminus \{(s,v)\}$
⇨ Step 4: If $Q = \emptyset$ then output f and X, and halt
Step 5: Set $(u,t) \coloneqq$ Delete(Q)
Step 6: If $f(t) = 1$ then return to Step 4
Step 7: Set $X \coloneqq X \cup \{(u,t)\}$
Step 8: Set $s \coloneqq t$ and return to Step 2



$e_6 = (y,z)$
$e_8 = (y,x)$
$e_5 = (y,c)$
$e_3 = (b,y)$
$e_2 = (b,c)$
$e_7 = (a,x)$
$e_4 = (a,c)$
$e_1 = (a,b)$

幅優先探索(待ち行列利用)

■ 定理(Theorem 3.5):BFS(queue)の時間計算量は$O(n+m)$

□ 証明

$n = |V|, m = |E|$

■ Algorithm 3.5 (queue)

- 入力:グラフ$G = (V, E)$, 始点$r \in V$
- 出力:変数$f(v)$ $(\forall v \in V)$, 辺の集合X $(\subseteq E)$

Step 0: Set $f(v) \coloneqq 0, \forall v \in V$, and set $X \coloneqq \emptyset, Y \coloneqq E$ $O(n+m)$

Step 1: Set $s \coloneqq r$ $O(1)$

Step 2: Set $f(s) \coloneqq 1$ $O(1)$

Step 3: For all edge $(s, v) \in Y$, $O(\deg_G(v))$

Insert$(Q, (s, v))$ and set $Y \coloneqq Y \setminus \{(s, v)\}$

Step 4: If $Q = \emptyset$ then output f and X, and halt $O(1)$ 出力:$O(n)$

Step 5: Set $(u, t) \coloneqq$ Delete(Q) $O(1)$

Step 6: If $f(t) = 1$ then return to Step 4 $O(1)$

Step 7: Set $X \coloneqq X \cup \{(u, t)\}$ $O(1)$

Step 8: Set $s \coloneqq t$ and return to Step 2 $O(1)$

高々n回

各点v回に対し高々1回

全部で$O(m)$

高々1回

高々$m+1$回

高々m回

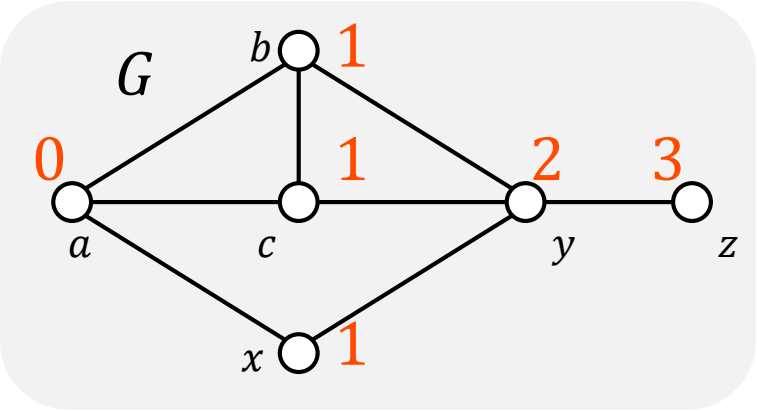
■ DFS(stack)と計算量は同じ(スタックの代わりに待ち行列)

幅優先探索(BFS)の応用

■ 始点から各点までの距離

■ 例

- 点aからの距離$\mathbf{dis}_G(a, v)$



距離ラベル付け (graph-distance)

■ Algorithm 3.6 (graph-distance)

- 入力:グラフ$G=(V,E)$, 始点$r \in V$
- 出力:変数$\lambda(v)$ $(\forall v \in V)$

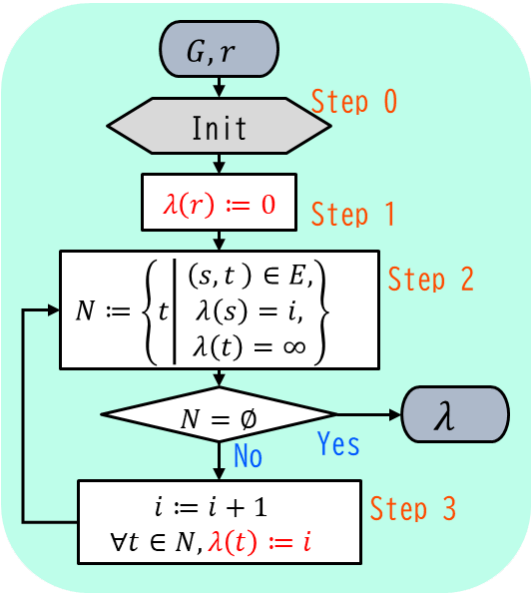
Step 0: Set $\lambda(v)=\infty, \forall v \in V$ and $i=0$

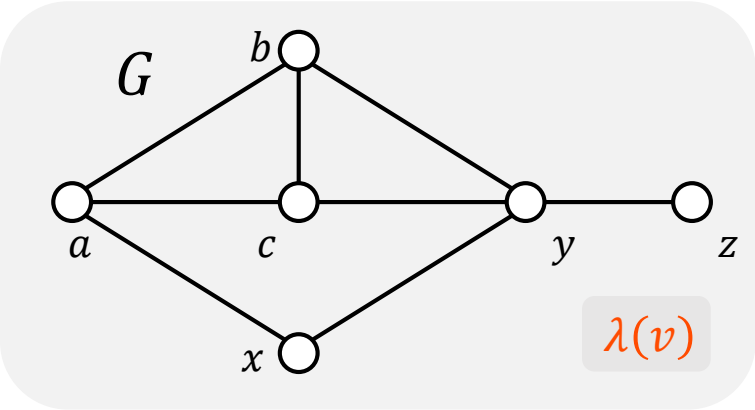
Step 1: Set $\lambda(r)=0$

Step 2: Set $N=\{t|(s,t)\in E, \lambda(s)=i, \lambda(t)=\infty\}$

Step 2.1: If $N=\emptyset$ then output λ and halt

Step 3: Set $i=i+1$, and set $\lambda(t)=i, \forall v \in N$, and return to Step 2

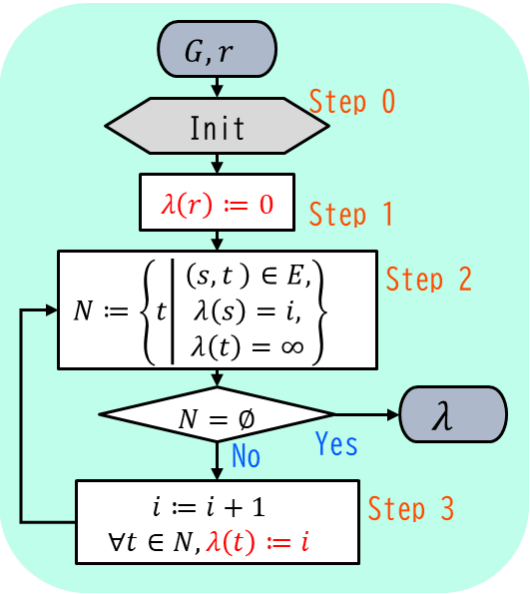


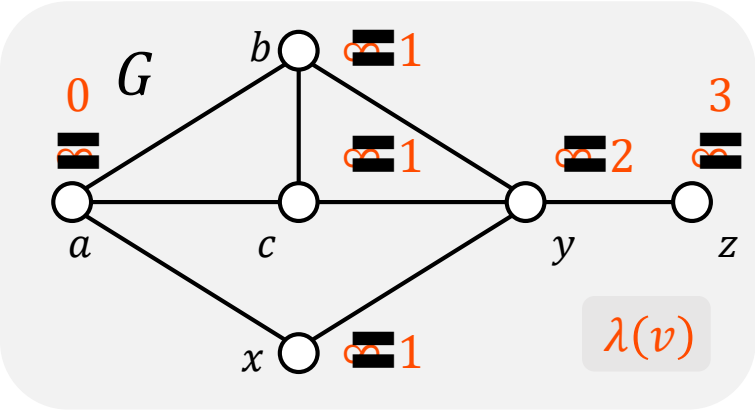


距離ラベル付け (graph-distance)

■ Algorithm 3.6 (graph-distance)
- 入力:グラフ$G=(V,E)$, 始点$r \in V$
- 出力:変数$\lambda(v)$ $(\forall v \in V)$

⇨ Step 0: Set $\lambda(v)=\infty, \forall v \in V$ and $i=0$
⇨ Step 1: Set $\lambda(r)=0$
Step 2: Set $N=\{t|(s,t)\in E, \lambda(s)=i, \lambda(t)=\infty\}$
Step 2.1: If $N=\emptyset$ then output λ and halt
⇨ Step 3: Set $i=i+1$, and set $\lambda(t)=i, \forall v \in N$, and return to Step 2





Step	a	b	c	x	y	z
0. (initial)	∞	∞	∞	∞	∞	∞
1. (start)	0	∞	∞	∞	∞	∞
3. (update)		1	1	1	∞	∞
3. (update)					2	∞
3. (update)						3

距離ラベル付け (graph-distance)

■ Algorithm 3.6 (graph-distance)

- 入力:グラフ$G=(V,E)$, 始点$r \in V$
- 出力:変数$\lambda(v)$ $(\forall v \in V)$

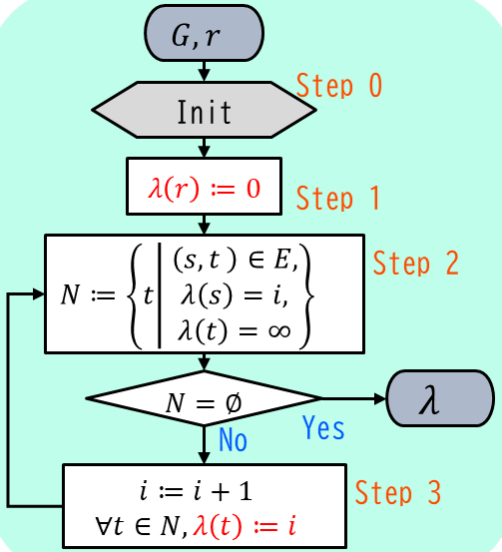
Step 0: Set $\lambda(v)=\infty, \forall v \in V$ and $i=0$

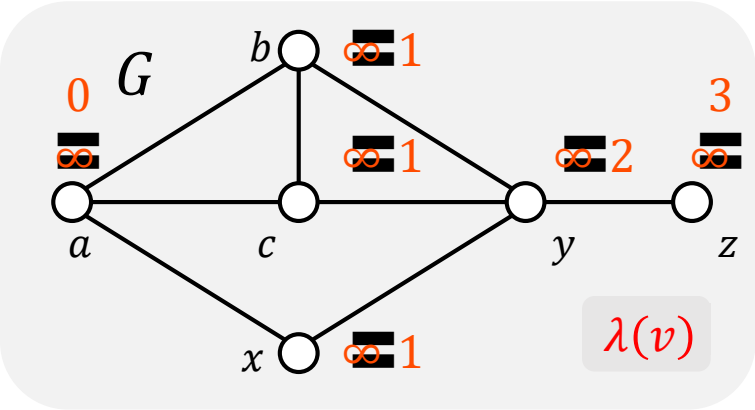
Step 1: Set $\lambda(r)=0$

Step 2: Set $N=\{t|(s,t)\in E, \lambda(s)=i, \lambda(t)=\infty\}$

Step 2.1: If $N=\emptyset$ then output λ and halt

Step 3: Set $i=i+1$, and set $\lambda(t)=i, \forall v \in N$, and return to Step 2





Step	a	b	c	x	y	z
0.(initial)	∞	∞	∞	∞	∞	∞
1.(start)	0	∞	∞	∞	∞	∞
3.(update)		1	1	1	∞	∞
3.(update)					2	∞
3.(update)						3

■ 定理 (Theorem 3.6): Algorithm3.6の時間計算量は$O(n+m)$

距離ラベル付け (graph-distance)

■ 定理 (Theorem 3.7): $\lambda(v) = \mathbf{dis}_G(r, v), \quad \forall v \in V$

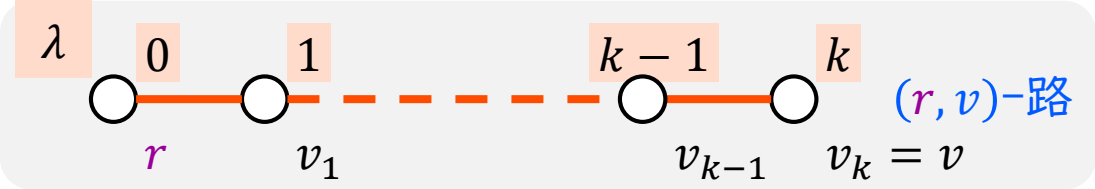
□ 証明 (Algorithm 3.6)

➢ $\lambda(v) = \infty \implies \mathbf{dis}_G(r, v) = \infty$

- (r, v)-路が存在するならば、(r, v)-路上
  - 有限ラベルの点xと無限大ラベルの点yが隣接
  - Step 2で$y \in N$となり, Step 3でyに有限ラベルが付けられるはずで矛盾
- (r, v)-路は存在せず$\mathbf{dis}_G(r, v) = \infty$

➢ $\lambda(v) < \infty$

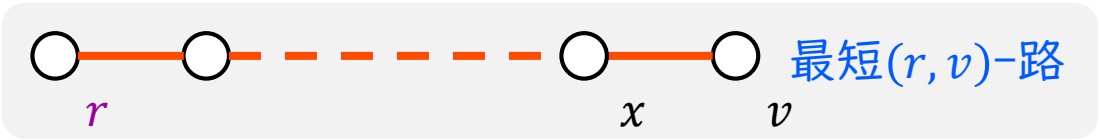
- Claim 1: $\lambda(v) \geq \mathbf{dis}_G(r, v)$
  - 長さkの(r, v)路が存在



- Claim 2: $\lambda(v) = \mathbf{dis}_G(r, v)$
  - 距離$\mathbf{dis}_G(r, v)$に関する数学的帰納法

距離ラベル付け (graph-distance)

- 定理 (Theorem 3.7): $\lambda(v) = \mathbf{dis}_G(r, v),\ \forall v \in V$
- 証明 (cont.) (Algorithm 3.6)
  - Claim 2: $\lambda(v) = \mathbf{dis}_G(r, v)$
    - 距離$\mathbf{dis}_G(r, v)$に関する数学的帰納法
    - 初期段階: $\mathbf{dis}_G(r, v) = 0$
      - $v = r$
      - $\lambda(r) = 0 \implies \lambda(r) = 0 = \mathbf{dis}_G(r, r)$
    - 帰納段階
      - 帰納法の仮定:$\lambda(v') = \mathbf{dis}_G(r, v')$ if $\mathbf{dis}_G(r, v') < d$
      - $\mathbf{dis}_G(r, v) = d$の場合
        - 最短(r, v)路においてvに隣接する点x



      - $\mathbf{dis}_G(r, x) = d - 1 \implies \lambda(x) = d - 1$ (帰納法の仮定)
        - $\lambda(v) \geq d$ (Claim 1)
        - $i = d - 1$のとき $\lambda(v) = \infty \implies \lambda(v)$は$d = (d - 1) + 1$に設定される

$$\therefore \lambda(v) = d = \mathbf{dis}_G(r, v)$$