

情報工学概論（アルゴリズムとデータ構造）

06 グラフ理論入門とグラフ探索

宮本 裕一郎
miyamoto あっと sophia.ac.jp

上智大学 理工学部 情報理工学科

目次

グラフ理論入門とグラフ探索

グラフ理論入門

グラフ探索問題とグラフ探索アルゴリズム

スタックと深さ優先探索

キューと幅優先探索

スタックと再帰

演習問題

グラフ理論入門とグラフ探索

グラフ理論入門

グラフ探索問題とグラフ探索アルゴリズム

スタックと深さ優先探索

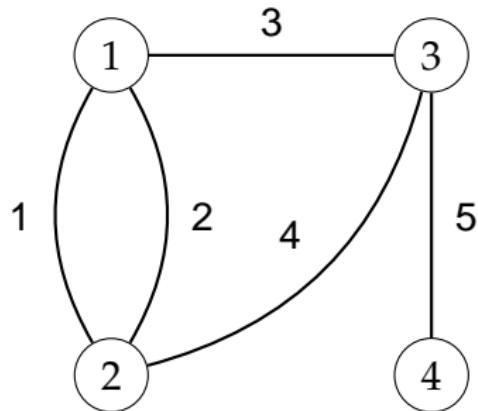
キューと幅優先探索

スタックと再帰

演習問題

グラフとは？

頂点¹ (vertex) を枝² (edge) で結んだものをグラフという .



グラフは

- ▶ 交通網
- ▶ 通信網
- ▶ 友達関係
- ▶ Web ページのリンク
- ▶ 状態遷移の関係

などの抽象化としてよく用いられる .

¹点 , 節点 , node, などと呼ばれることがある .

²辺 , リンク (link), アーク (arc) などと呼ばれることがある .

グラフの定義

定義 (グラフ)

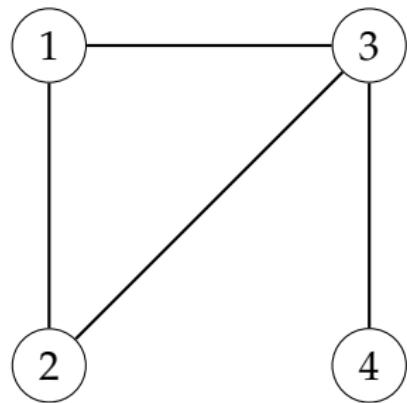
集合 V と集合 E と接続関数 $\Psi: E \rightarrow \{X \subseteq V : |X| = 2\}$ の 3 つ組 (V, E, Ψ) を **グラフ (graph)** という³。集合 V を **頂点集合 (vertices)**、集合 E を **枝集合 (edges)** という。

- ▶ グラフ G が与えられたとき、その頂点集合を $V(G)$ 、枝集合を $E(G)$ で表す。
- ▶ グラフ G において、 $\Psi(e) = \{v, w\}$ である枝 $e \in E(G)$ は頂点 v と w を **結んで (join)** いるという。このとき、 v と w はグラフ G において **隣接 (adjacent)** しているという。またこのとき、 v と w は枝 e の **端点 (end point)** であるという。
- ▶ $e, e' \in E(G)$ が $e \neq e'$ であるにもかかわらず $\Psi(e) = \Psi(e')$ であるとき、 e と e' は **parallel** であるという。
- ▶ Parallel な枝を含まないグラフを **単純グラフ (simple graph)** という。

³集合 V, E は有限集合とは限らない。しかし、アルゴリズムの文脈では多くの場合において有限集合である。

単純グラフの定義

例 (単純グラフ G の図)



- ▶ アルゴリズムやデータ構造の文脈において現れるグラフのほとんどは単純グラフである .
- ▶ 単純グラフは parallel な枝を含まないので , 接続関数の値をそのまま枝集合の要素としても正確性は失われない . よって単純グラフは簡便に以下のように定義できる .

定義 (単純グラフ)

単純グラフ G は頂点集合 V と枝集合 $E \subseteq \{\{v, w\} : v \in V, w \in V, v \neq w\}$ の 2 つからなる組 (V, E) である .

例 (単純グラフ G の式)

$$V(G) = \{ \quad, \quad, \quad, \quad \},$$

$$E(G) = \{ \{ \quad, \quad \}, \{ \quad, \quad \}, \{ \quad, \quad \}, \{ \quad, \quad \} \})$$

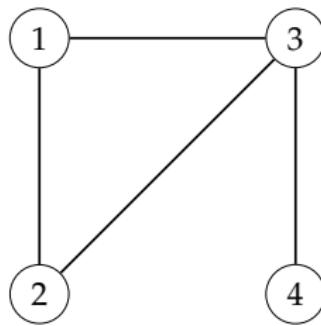
頂点隣接行列

- 単純グラフを保存する最も単純なデータ構造として、頂点隣接行列が知られている。

定義 (頂点隣接行列)

グラフ G が与えられたとき、 $\{v, w\} \in E(G)$ ならば $a_{vw} = 1$ そうでないならば $a_{vw} = 0$ となる行列 $A (= (a_{vw}))$ を **頂点隣接行列 (adjacency matrix)** という。

例 (グラフ)



例 (頂点隣接行列)

左図のグラフの頂点隣接行列は、 i 行目に対応する頂点が i 、 j 列目に対応する頂点が j とすると

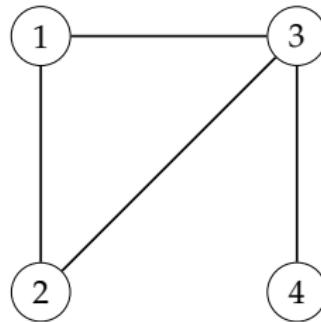
$$\begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

である。

頂点隣接行列の計算複雑度

- ▶ グラフ G の頂点数を n とすると ,
- ▶ 頂点隣接行列の空間複雑度は $O(n^2)$ である .
- ▶ 指定された頂点 $v \in V(G)$ に隣接する頂点をすべて出力する時間複雑度は $O(n)$ である .

例 (グラフ)



例 (頂点隣接行列)

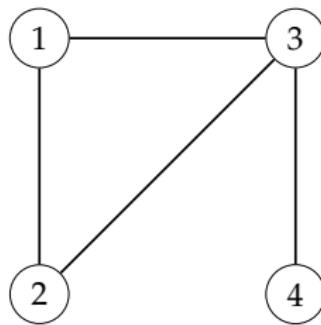
$$\begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

歩道

定義 (歩道)

グラフ G が与えられているとする。その頂点 $v_i \in V(G)$ と枝 $e_j \in E(G)$ の列 $(v_1, e_1, v_2, \dots, v_k, e_k, v_{k+1})$ は、すべての $i \in \{1, \dots, k\}$ に関して $\Psi(e_i) = \{v_i, v_{i+1}\}$ を満たすとき（あるいは G が単純グラフで $e_i = \{v_i, v_{i+1}\}$ を満たすとき）、グラフ G 上の**歩道 (walk)** であるという。

例 (グラフ G 上の歩道)



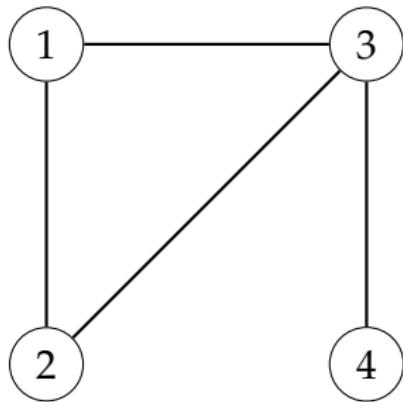
- ▶ $V(G) = \{1, 2, 3, 4\}, E(G) = \{\{1, 2\}, \{1, 3\}, \{3, 4\}, \{2, 4\}\}$ とする。
- ▶ $(1, \{1, 2\}, 1, \{1, 3\}, 1)$ はグラフ G 上の歩道である。
- ▶ $(1, \{1, 2\}, 1, \{1, 3\}, 2)$ はグラフ G 上の歩道ではない。

連結グラフ

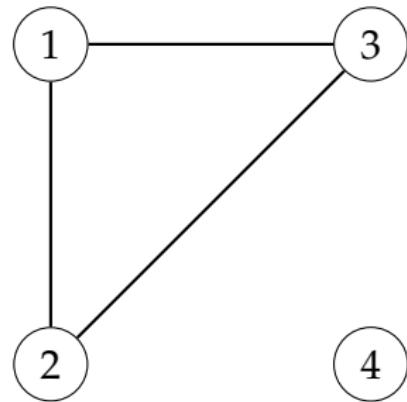
定義 (連結グラフ)

すべての頂点の間に歩道があるグラフを連結グラフ (connected graph) という。あるいは、そのグラフは連結である、という。

例 (連結グラフ)



例 (連結でないグラフ)

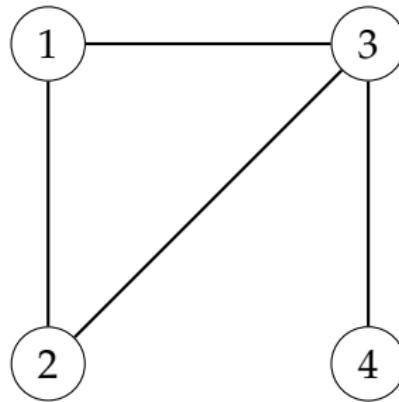


部分グラフ

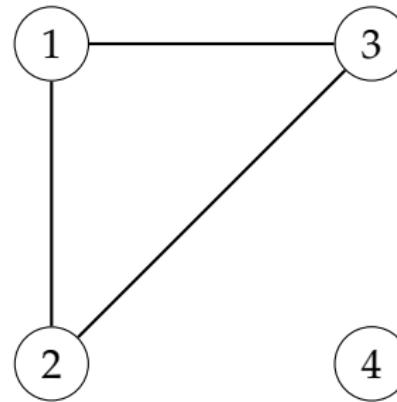
定義 (部分グラフ)

グラフ G に対して, $V(H) \subseteq V(G)$ かつ $E(H) \subseteq E(G)$ となるグラフ H を G の部分グラフ (subgraph) という.

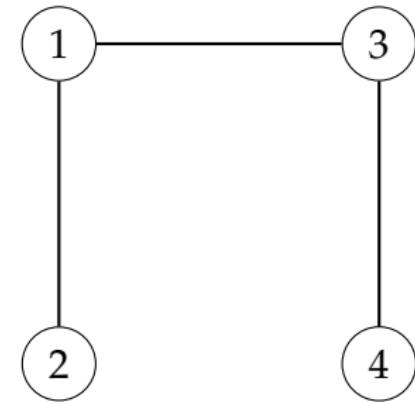
例 (グラフ G)



例 (G の部分グラフ H)



例 (G の部分グラフ H')

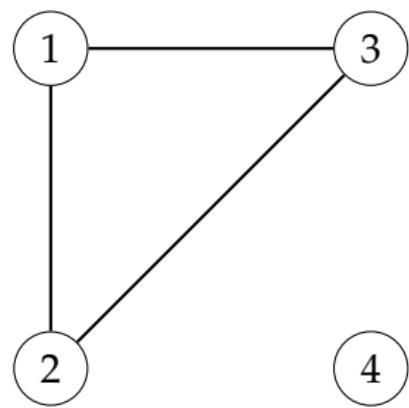


連結成分

定義 (連結成分)

グラフの極大な連結部分グラフを **連結成分 (connected component(s))** という。

例 (グラフ G の連結成分)



- ▶ $V(G) = \{1, 2, 3, 4\}$,
 $E(G) = \{\{1, 2\}, \{1, 4\}, \{2, 4\}\}$
 とする。
- ▶ グラフ G の連結成分は
 $(\{1, 2, 4\}, \{3\})$
 と
 $(\{3\}, \{\})$
 である。

グラフ理論入門とグラフ探索

グラフ理論入門

グラフ探索問題とグラフ探索アルゴリズム

スタックと深さ優先探索

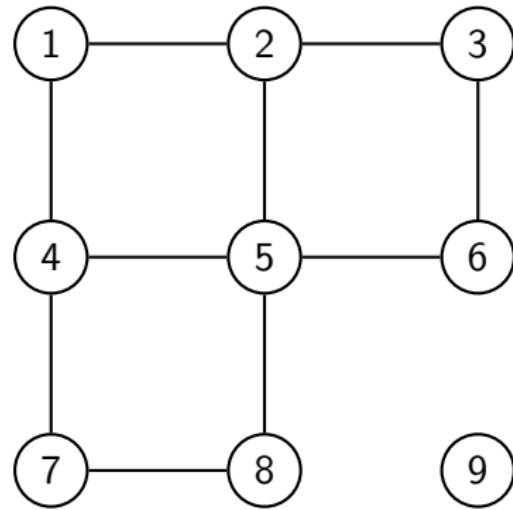
キューと幅優先探索

スタックと再帰

演習問題

グラフ探索問題

例 (入力)



$s =$

問題 (グラフ探索)

入力 グラフ G , 始点 $s \in V(G)$

出力 グラフ G の連結成分のうち,
始点 s を含むものの頂点集合 C

例 (出力)

$C = \{ \quad , \quad \}$

グラフ探索アルゴリズムの直感的記述

- ▶ グラフ探索問題に対するアルゴリズムとして、すぐに思いつきそうな方法を以下に直感的に記す。

グラフ探索アルゴリズム：

- ▶ 最初は始点 s にいるとする。「これから訪れたい頂点の集合」として始点 s に隣接する頂点を覚える。
- ▶ 「これから訪れたい頂点」がある限り、以下を繰り返す。
 - ▶ 「これから訪れたい頂点」のうち 1 つに移動し、そこは「これから訪れたい頂点」から消す。
 - ▶ 現在いる頂点に隣接する頂点のうち、まだ訪れていない頂点を「これから訪れたい頂点」に加える。
- ▶ 訪れた頂点のすべてを出力しておわり。

グラフ探索アルゴリズムのより厳密な記述

- ▶ グラフ探索アルゴリズムを GraphScanning として関数っぽく以下に記す .
- ▶ 参照 , 代入 , 四則演算を基本演算としている .
- ▶ 以下では , 頂点部分集合 C が「これまでに訪れた頂点」に対応し , 頂点部分集合 S が「これから訪れたい頂点の集合」に対応する .

GraphScanning(G, s):

Step 1 頂点部分集合 C を $\{s\}$ とする . 頂点部分集合 S を $\{s\}$ とする .

Step 2 頂点部分集合 S が空集合でない間 , 以下を繰り返す .

 Step 2-1 頂点部分集合 S から要素を 1 つ取り出し , それを v とする .

 Step 2-2 v の隣接頂点 $w \in V(G)$ それぞれについて ,
 もし $w \notin C$ ならば , 頂点部分集合 C と S に w を加える .

Step 3 頂点部分集合 C を出力して終了する .

グラフ探索アルゴリズムのデータ構造の工夫

- ▶ グラフ探索アルゴリズム GraphScanning の頂点部分集合 S を保存するデータ構造を工夫すると、何らかの意味でアルゴリズムの使い勝手が良くなる。
- ▶ 以降では、最も基本的なデータ構造としてスタックとキューを紹介する。
- ▶ グラフ探索アルゴリズムにおいてスタックを S として用いると、 S の記憶容量が比較的少なくて済む。
- ▶ グラフ探索アルゴリズムにおいてキューを S として用いると、グラフ探索アルゴリズムを最短路問題（いずれ紹介する）に適用できる。

【再掲】GraphScanning(G, s):

Step 1 頂点部分集合 C を $\{s\}$ とする。頂点部分集合 S を $\{s\}$ とする。

Step 2 頂点部分集合 S が空集合でない間、以下を繰り返す。

Step 2-1 頂点部分集合 S から要素を 1 つ取り出し、それを v とする。

Step 2-2 v の隣接頂点 $w \in V(G)$ それぞれに関して、

もし $w \notin C$ ならば、頂点部分集合 C と S に w を加える。

Step 3 頂点部分集合 C を出力して終了する。

グラフ理論入門とグラフ探索

グラフ理論入門

グラフ探索問題とグラフ探索アルゴリズム

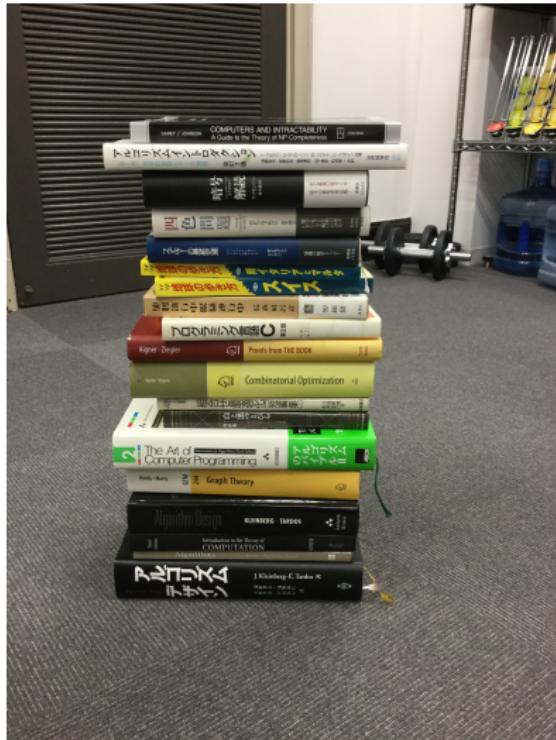
スタックと深さ優先探索

キューと幅優先探索

スタックと再帰

演習問題

スタックとそのイメージ



- ▶ グラフ探索アルゴリズムは、
LIFO スタック (LIFO-stack あるいは単に stack) を用いると、
少し見通しが良くなる。

スタック【stack】

データ構造の一種。最後に入れた
データが最初に取り出せるもの。
LIFO (last-in first-out) ともいう。
待ち行列

「広辞苑（第六版）」より

スタックの定義

【再掲】スタック【stack】

データ構造の一種。最後に入れたデータが最初に取り出せるもの。LIFO (last-in first-out) ともいう。待ち行列

「広辞苑（第六版）」より

- ▶多くのプログラミング言語では、配列（あるいはリスト）を用いればスタックを実現できる。
- ▶スタックをクラスなどとして提供しているプログラミング言語も多い。

定義（データ構造スタック）

- ▶スタックには、要素を1つずつ時間複雑度 $O(1)$ で加えられる。
- ▶スタックから要素を取り出す際には、（残っている要素のうち）最後に加えたもののみを時間複雑度 $O(1)$ で取り出せる。
- ▶スタックの要素数は時間複雑度 $O(1)$ でわかる。

深さ優先探索アルゴリズム

- ▶ スタックを用いたグラフ探索アルゴリズムは**深さ優先探索 (Depth First Search 略して DFS)**ともよばれる .
- ▶ 深さ優先探索アルゴリズムを StackDFS として関数っぽく以下に記す .
- ▶ StackDFS では , 頂点部分集合 C が「これまでに訪れた頂点」に対応し , スタック S が「これから訪れたい頂点の集合」に対応する .

StackDFS(G, s):

Step 1 頂点部分集合 C を $\{s\}$ とする . スタック S を (s) とする .

Step 2 スタック S が空集合でない間 , 以下を繰り返す .

Step 2-1 スタック S から要素を 1 つ取り出し , それを v とする .

Step 2-2 v の隣接頂点 $w \in V(G)$ それぞれに関して ,
もし $w \notin C$ ならば , 頂点部分集合 C とスタック S に w を加える .

Step 3 頂点部分集合 C を出力して終了する .

グラフ探索アルゴリズムと深さ優先探索アルゴリズムの比較

- ▶ 念のため，グラフ探索アルゴリズムと深さ優先探索アルゴリズムを並べて見てみる．
- ▶ S をスタックとしているか否かの違いだけであることがわかる．

【再掲】 $\text{GraphScanning}(G, s)$:

Step 1 頂点部分集合 C を $\{s\}$ とする．頂点部分集合 S を $\{s\}$ とする．

Step 2 頂点部分集合 S が空集合でない間，以下を繰り返す．

Step 2-1 頂点部分集合 S から要素を 1 つ取り出し，それを v とする．

Step 2-2 v の隣接頂点 $w \in V(G)$ それぞれに関して，もし $w \notin C$ ならば，頂点部分集合 C と S に w を加える．

Step 3 頂点部分集合 C を出力して終了する．

【再掲】 $\text{StackDFS}(G, s)$:

Step 1 頂点部分集合 C を $\{s\}$ とする．スタック S を (s) とする．

Step 2 スタック S が空集合でない間，以下を繰り返す．

Step 2-1 スタック S から要素を 1 つ取り出し，それを v とする．

Step 2-2 v の隣接頂点 $w \in V(G)$ それぞれに関して，もし $w \notin C$ ならば，頂点部分集合 C とスタック S に w を加える．

Step 3 頂点部分集合 C を出力して終了する．

グラフ理論入門とグラフ探索

グラフ理論入門

グラフ探索問題とグラフ探索アルゴリズム

スタックと深さ優先探索

キューと幅優先探索

スタックと再帰

演習問題

キューとそのイメージ



キュー【queue】

「待ち行列」に同じ。

「広辞苑（第六版）」より

まちぎょうれつ【待ち行列】

データ構造の一種。最初に入れたデータが最初に取り出せるもの。FIFO (first-in first-out) ともいう。

スタック

「広辞苑（第六版）」より

キュー

キュー【queue】

「待ち行列」に同じ。

「広辞苑（第六版）」より

まちぎょうれつ【待ち行列】

データ構造の一種。最初に入れたデータが最初に取り出せるもの。FIFO (first-in first-out)ともいう。　スタック

「広辞苑（第六版）」より

スタック【stack】

データ構造の一種。最後に入れたデータが最初に取り出せるもの。LIFO (last-in first-out)ともいう。　待ち行列

「広辞苑（第六版）」より

データ構造キュー

- ▶ スタックと対照的なデータ構造として FIFO-キュー (FIFO-queue あるいは単に queue) がある .
- ▶ キューも , スタックと同様に , 配列を用いれば実現できる . ただし , 配列を循環的に利用するなど若干の工夫が必要である .

定義 (データ構造キュー)

- ▶ キューには , 要素を 1 つずつ時間複雑度 $O(1)$ で加えられる .
- ▶ キューから要素を取り出す際には ,(残っている要素のうち) 最初に 加えたものののみを時間複雑度 $O(1)$ で取り出せる .
- ▶ キューの要素数は時間複雑度 $O(1)$ でわかる .

幅優先探索アルゴリズム

- ▶ キューを用いたグラフ探索アルゴリズムは幅優先探索 (Breadth First Search 略して BFS) ともよばれる .
- ▶ 幅優先探索アルゴリズムを BFS として関数っぽく以下に記す .
- ▶ 以下では , 頂点部分集合 C が「これまでに訪れた頂点」に対応し , キュー S が「これから訪れたい頂点の集合」に対応する .

$\text{BFS}(G, s)$:

Step 1 頂点部分集合 C を $\{s\}$ とする . キュー S を (s) とする .

Step 2 キュー S が空集合でない間 , 以下を繰り返す .

Step 2-1 キュー S から要素を 1 つ取り出し , それを v とする .

Step 2-2 v の隣接頂点 $w \in V(G)$ それぞれに関して ,
もし $w \notin C$ ならば , 頂点部分集合 C とキュー S に w を
加える .

Step 3 頂点部分集合 C を出力して終了する .

グラフ理論入門とグラフ探索

グラフ理論入門

グラフ探索問題とグラフ探索アルゴリズム

スタックと深さ優先探索

キューと幅優先探索

スタックと再帰

演習問題

再帰による深さ優先探索

- ▶ グラフ探索アルゴリズムとして、再帰を用いるものも考えられる。
- ▶ 再帰を用いたグラフ探索アルゴリズムを RecursiveDFS として簡略化以下に記す。
- ▶ 名前から想像がつく通り、RecursiveDFS はスタックを用いた深さ優先探索とほぼ同じ動きをする。
- ▶ グラフ探索問題の入力としてグラフ G と始点 $s \in V(G)$ が与えられたら、 $\text{RecursiveDFS}(G, s, \{\})$ を実行すればよい。

$\text{RecursiveDFS}(G, v, C)$:

Step 1 頂点部分集合 C に v を加える。すなわち C に $C \cup \{v\}$ を代入する。

Step 2 頂点 v の隣接頂点 $w \in V(G)$ それぞれに関して以下を行う。

Step 2-1 $w \notin C$ ならば、

C に $\text{RecursiveDFS}(G, w, C)$ の出力結果を代入する。

Step 3 C を出力して終了する。

再帰とスタックと Stack overflow

- ▶ 再帰アルゴリズムはスタックの利用によって機械的に反復計算に変換できる。
- ▶ 深さ優先探索では、後で探索するつもりの頂点をスタックに保存した。一般には、何をスタックに保存するか決めるのは難しいので、計算に関わるもろもろをすべてスタックに保存している。
- ▶ 再帰が深く（多く）なるとスタックに保存するものが多くなり、メモリ容量が足りなくなることが予想される。
- ▶ よって、多くのプログラミング言語ではスタックの高さ（あるいは深さ）の限界を予め決めておいて、それを超えたら処理を中断する。それが stack overflow である。
- ▶ 多くの処理系において、再帰はスタックで実現されている。よって、再帰における関数呼び出しを基本演算とすることは、スタックの使用、さらには参照、代入、四則演算を基本演算とすることと等しい。以降、本講義では、参照、代入、四則演算を基本演算としている場合には（断りなく）再帰も使用可能であることとする。

グラフ理論入門とグラフ探索

グラフ理論入門

グラフ探索問題とグラフ探索アルゴリズム

スタックと深さ優先探索

キューと幅優先探索

スタックと再帰

演習問題

深さ優先探索の時間複雑度

問題 入力として単純グラフ（と始点）が与えられた場合の，深さ優先探索アルゴリズム StackDFS の時間複雑度をグラフの頂点数 n の関数で表わせ．なお，グラフは頂点隣接行列で保存されているとする．

幅優先探索の時間複雑度

問題 入力として単純グラフ（と始点）が与えられた場合の、幅優先探索アルゴリズム QueueBFS の時間複雑度をグラフの頂点数 n の関数で表わせ。なお、グラフは頂点隣接行列で保存されているとする。