グラフの探索

組織的な探索アルゴリズム

水谷 正大

大東文化大学 mizutani@ic.daito.ac.jp

2014 Oct.

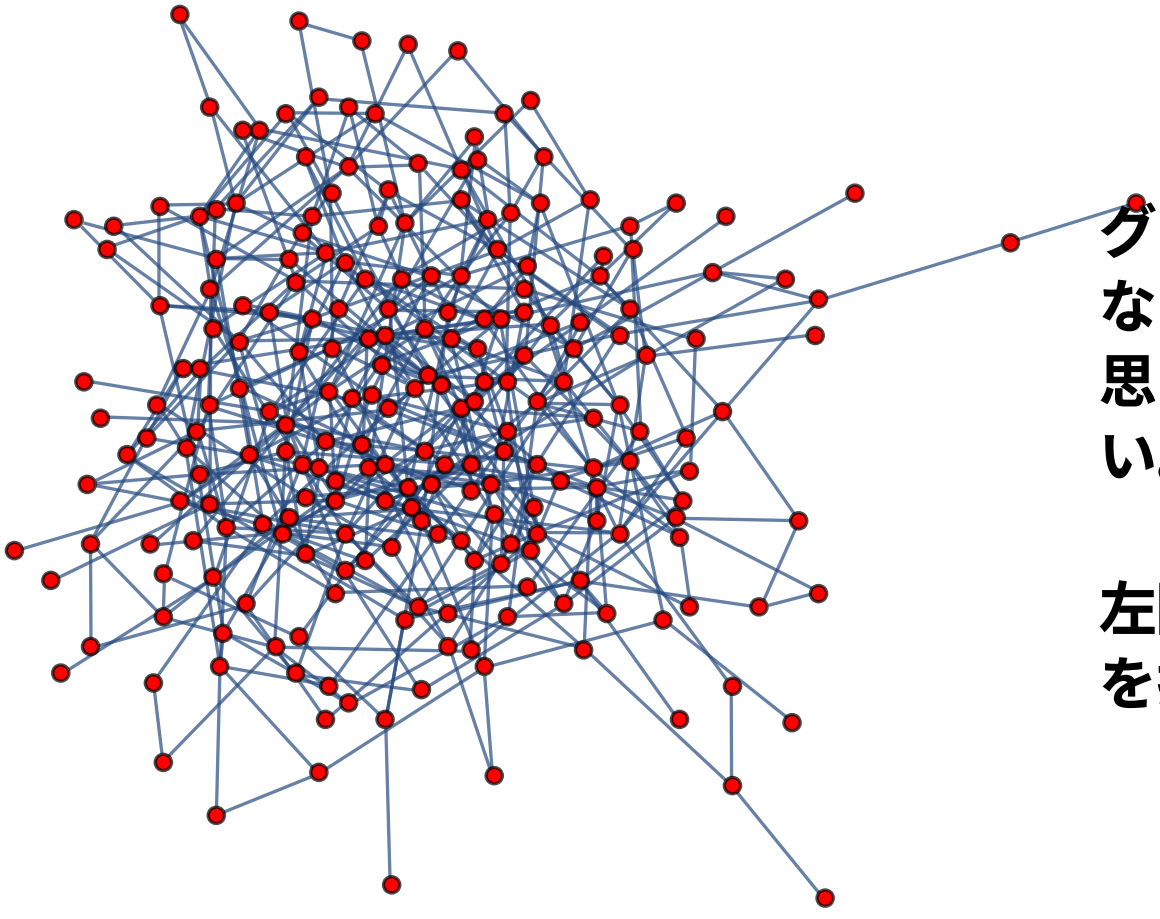
探索（serach）とは何か

メモリなどに格納しているデータの値を知るということにおいて、それらがどこに格納してあり、その場所を見て内容を調べるといった必要がある。データが格納されている場所を**漏れなくすべて訪れること**（visit）を**探索**（search）または**走査**（traversal）という。
特に、データが格納されている場所の**隣接性**が問題となるグラフ構造をなしている探索は応用上きわめて重要である。もちろん、探索では同じ場所をいくども訪れることは非効率で、何らかの方法で一度だけ全て訪れることが肝要だ。
n 個の頂点を持つグラフ $G = (V = \{v_1, v_2, \ldots, v_n\}, E)$ において、N 個を並べ替える置換（permutation）を $\sigma : \mathbb{N}_n \to \mathbb{N}_n$ としたとき、G の頂点 V を一度だけ訪れる探索は、訪れる頂点を順にならべて次のように書ける（$v_{\sigma(1)}$ が開始頂点）。

$$v_{\sigma(1)} v_{\sigma(2)} \cdots v_{\sigma(n-1)} v_{\sigma(n)}$$

$\mathbb{N}_n$ の並べ替えの総数は $N1$ 個であるので、グラフの探索方法は同じく $N!$ 個ある。これらのどの探索を選ぶべきだろうか。

グラフ探索は視覚に頼れない



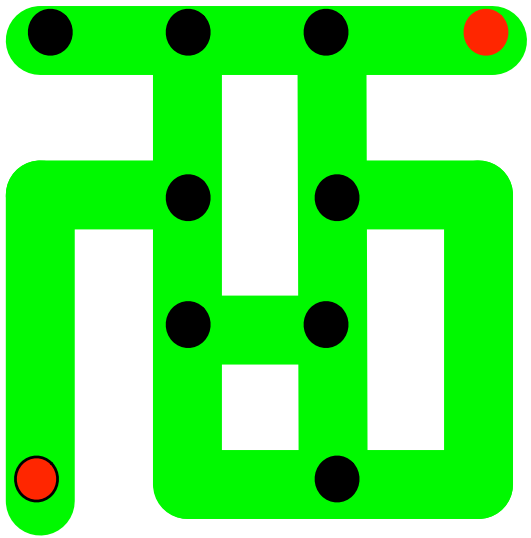
グラフの頂点が少ないと簡単なように思えるが、甘くはない。

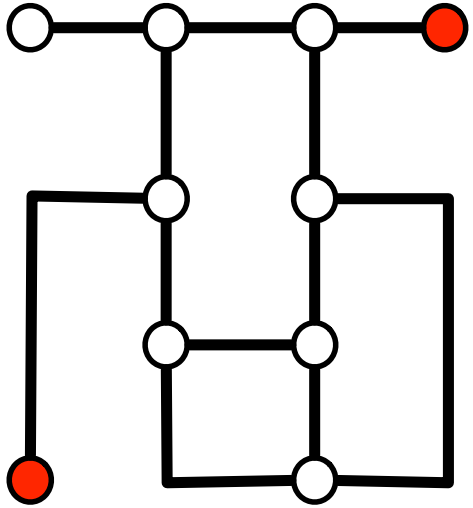
左図は 230 個の頂点を持つグラフ

探索の必要性

- （理論的にはきわめて難しい問題）すべての交差点の信号機を点検することはグラフの探索である。どのように探索すれば効率的だろう？
- **鉄道路線**では？

迷路の探索

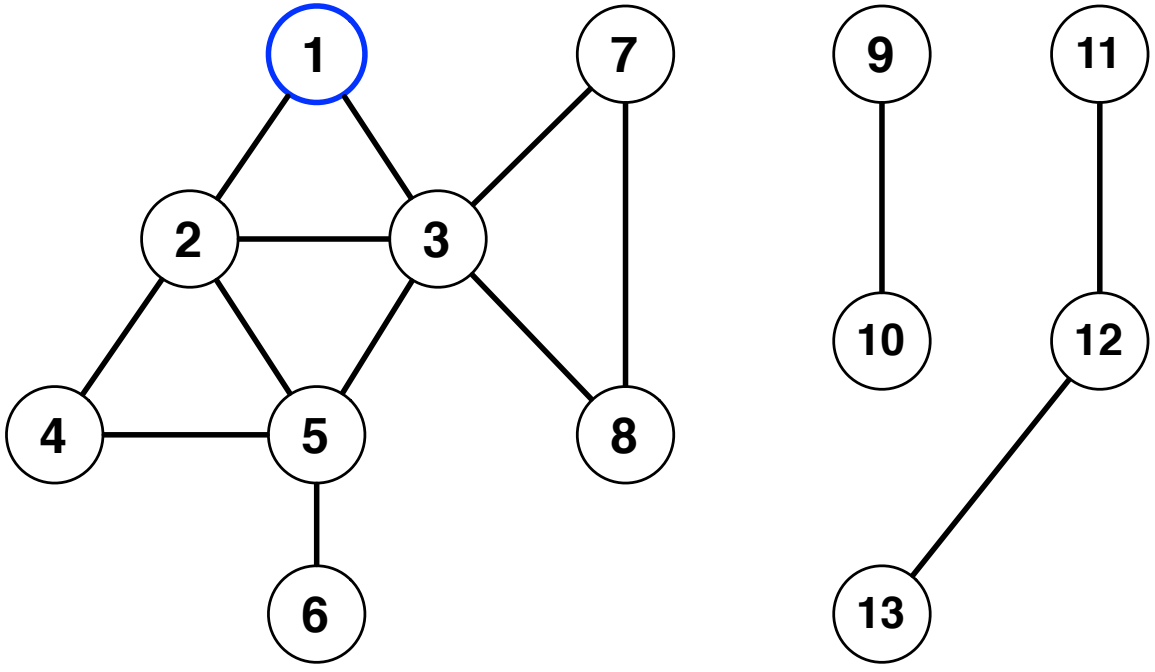




迷路の入り口から出口に至る道の発見は、交差している道を頂点とするグラフ探索問題と等価

探索方法あれこれ

非連結グラフで探索してみる：頂点 1 からの経路は頂点 2 から 8 まではあるが、頂点 9 から 13 まではない。



グラフの構造を反映させる代表的な 2 つの方法

- **深さ優先探索**（depth first search）。開始点 s から到達可能な点を求める迷路解決的方法で、可能な限り奥深く探索していき、それが不可能になる場合にだけ最小限戻ってこれを繰り返す。
- **横幅優先探索**（breadth first search）

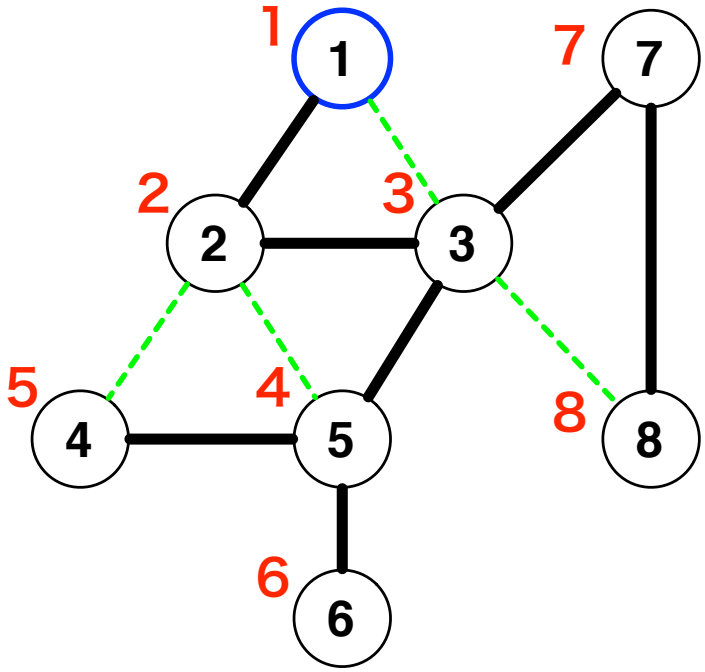
深さ優先探索 DFS のアルゴリズム（再帰版）

s から出ている辺を 1 本を辿り、未訪の v（$\mathrm{IsVisited}(v) =$ 未訪）に到達したとする。次の v からせている辺をたどり行き止まりにならない限り、これを繰り返して奥深く進む。ある点で、そこから出ている辺が既に訪れた点に向かっているときは“行き止まり”で、これまで来た経路を逆にたどり、新しい点に向かう辺を持つ点にまで戻り、先の方法を繰り返す。

Require: 全ての頂点 v について $\mathrm{IsVisited}(v) =$ 未訪
Ensure: $\mathrm{DFS}(s)$
  $\mathrm{IsVisited}(s) =$ 既訪;
  s に関する処理（‘s を訪問’ と印刷）
  for s に接続する各辺 (s, w) **do**
    if $\mathrm{IsVisited}(w) ==$ 未訪 **then**
      $\mathrm{DFS}(w)$
    end if
  end for

深さ優先の探索木 DFS 木（DFS tree）

- DFS は開始点 s を含む G の連結成分上の根付き木 T_D を構成（DFS 木）。
- s を T_D の根とし、u が w の発見に責任があるとき（DFS(u) の実行中で DFS(w) が呼び出されるとき）、辺 $[u, w]$ が T_D に加えられていく（u は w の親頂点）。
- G の辺で緑破線辺は T_D には属さない。
- DFS の最初の実行で 1,2,3,4 からなる経路が構成。4 に至って新しい点を見いだせずに行き止まり。5 まで戻って 6 を見つけ、3 まで戻って 7,8 を見つける。



赤の数字が訪問の順番
（ある頂点に隣接する頂点の調べた方は、隣接リストの逆順とした場合）

DFS アルゴリズムの実装（スタック版）

Require: 全ての頂点 v について $\mathrm{IsVisited}(v) = \mathrm{False}$（未訪）
Ensure: $\mathrm{DFS}(s)$
  $S \leftarrow [s]$（スタック S は要素 s だけからなる）
  $T_D \leftarrow \phi$（DFS 木は空）
  while $S \neq \phi$（スタック S が空でない） **do**
    $u = \mathrm{pop}(S)$（S から要素を取り出して u とする）
    if $\mathrm{isVisited}(u) == \mathrm{False}$ **then**
      $\mathrm{isVisited}(u) = \mathrm{True}$（既訪に）
      $\mathrm{append}(T_D, [\mathrm{parent}(u), u])$（ただし $u \neq s$）
        u に関する処理（‘u を訪問’ と印刷）
      for u に接続する各辺 $[u, w]$ **do**
        $\mathrm{push}(S, w)$（S に要素 w を加える）
        $\mathrm{parent}[w] = u$（w の親頂点は u）
      end for
    end if
  end while

DFS スタック版の動作

頂点 u に隣接する頂点 w を調べてスタックに追加する順番（for 文の順番）は、隣接リストの逆順とする。T_D を構成する木の辺が追加されていく。

vertex		隣接リスト
1	→	(2,3)
2	→	(1,3,4,5)
3	→	(1,2,5,7,8)
4	→	(2,5)
5	→	(2,3,4,6)
6	→	(5)
7	→	(3,8)
8	→	(3,7)

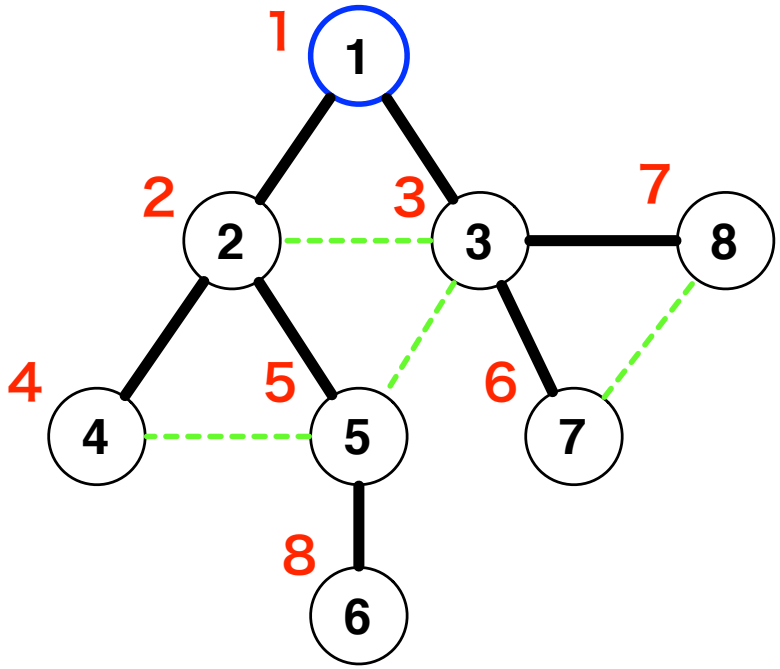
訪問点	スタック	追加辺
	[1]	
1:	2,3(	
2:	1,3,4,5,3(	+[1,2]
3:	1,2,5,7,8,4,5,3(	+[2,3]
5:	2,3,4,6,7,8,4,5,3(	+[3,5]
4:	2,5,6,7,8,4,5,3(	+[5,4]
6:	5,7,8,4,5,3(	+[5,6]
7:	3,8,8,4,5,3(	+[3,7]
8:	3,7,8,4,5,3(	+[7,8]

横幅優先探索 BFS のアルゴリズム（待ち行列版）

Require: 全ての頂点 v について $\mathrm{IsVisited}(v) = \mathrm{False}$（未訪）
Ensure: $\mathrm{BFS}(s)$
  $Q \leftarrow [s]$（待ち行列 Q は要素 s だけからなる）
  $T_B \leftarrow \phi$（DFS 木は空）
  while $Q \neq \phi$（待ち行列 Q が空でない） **do**
    $u = \mathrm{removeQueue}(Q)$（$Q$ から要素を取り出して u とする）
    if $\mathrm{isVisited}(u) == \mathrm{False}$ **then**
      $\mathrm{isVisited}(u) = \mathrm{True}$（既訪に）
      $\mathrm{append}(T_B, [\mathrm{parent}(u), u])$（但し $u \neq s$）
        u に関する処理（‘u を訪問’ と印刷）
      for u に接続する各辺 $[u, w]$ **do**
        $\mathrm{addQueue}(Q, w)$（Q に要素 w を加える）
        $\mathrm{parent}[w] = u$（w の親頂点は u）
      end for
    end if
  end while

横幅優先の探索木 BFS 木（BFS tree）

- BFS は開始点 s を含む G の連結成分上の根付き木 T_B を構成（BFS 木）。
- s を T_B の根として、待ち行列から取り出して u が未踏であるとき、辺 $[w$ の親$, w]$ が T_B に加えられていく。
- G の緑破線辺は T_B には属さない。
- BFS の最初の実行で第 1 層に属する頂点 1 とに至る経路 [1,2],[1,3] が見つかる。
- 第 2 層（出発点から距離 2 ある）の頂点集合は、第 1 層の頂点 2 及び 3 からの未踏頂点への経路として見つかる。



赤の数字が訪問の順番
（ある頂点に隣接する頂点の調べた方は、隣接リスト順とした場合）

BFS 待ち行列版の動作

頂点 u に隣接する頂点 w を調べて待ち行列に追加する順番（for 文の順番）は、隣接リスト順とする。T_B を構成する木の辺が追加されていく。

vertex		隣接リスト
1	→	(3, 2)
2	→	(1,3,4,5)
3	→	(1,2,5,7,8)
4	→	(2,5)
5	→	(2,3,4,6)
6	→	(5)
7	→	(3,8)
8	→	(3,7)

訪問点	待ち行列	追加辺
	<1-	
1:	<2,3-	
2:	<3,1,3,4,5-	+[1,2]
3:	<1,3,4,5,1,2,5,7,8-	+[1,3]
4:	<5,1,2,5,7,8,2,5-	+[2,4]
5:	<1,2,5,7,8,2,5,2,3,4,6-	+[2,5]
7:	<8,2,5,2,3,4,6,3,8-	+[3,7]
8:	<2,5,2,3,4,6,3,8,3,7-	+[3,8]
6:	<3,8,3,7,5-	+[5,6]

深さ優先 DFS と横幅優先探索 BFS の関係