大阪電気通信大学 情報通信工学部 光システム工学科 2年次配当科目

コンピュータアルゴリズム

グラフ (1)

第9講: 平成20年12月12日 (金) 4限 E252教室

中村 嘉隆(なかむら よしたか)
奈良先端科学技術大学院大学 助教
y-nakamr@is.naist.jp
http://narayama.naist.jp/~y-nakamr/

本日の講義内容

- グラフ(graph)
  - グラフとは
  - グラフの表現
  - グラフの探索
    - 深さ優先探索
    - 幅優先探索

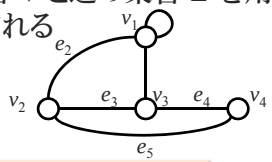
グラフとは

- グラフ: Graph
- 頂点(vertex, node, 節点)の集合と辺(edge, arc, branch, 枝)の集合からなる
- グラフ G は頂点の集合 V と辺の集合 E を用いて, $G = (V, E)$ と表される

$G = (V, E)$

$V = \{v_1, v_2, v_3, v_4\}$

$E = \{e_1, e_2, e_3, e_4, e_5\}$



頂点 v_i, v_j 間の辺 e_{ij} を 辺 (v_i, v_j) と書く

グラフの種類

- 有向グラフと無向グラフ
  - 辺に向きのあるグラフが有向グラフ
    - directed graph, 矢印付きの辺
  - 辺に向きのないグラフが無向グラフ
    - undirected graph, 矢印なしの辺

有向グラフの場合, 辺 (v_1, v_2) と辺 (v_2, v_1) は別物

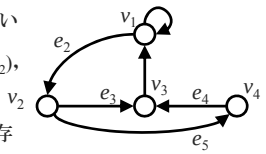
- 重みつきグラフ
  - 辺は属性として重み(weight)を持つことがある
    - コスト, 長さと呼ぶこともある
  - 例: 電車の路線図
    - 頂点は駅, 辺は区間, 辺の重みは運賃

グラフの用語

- 道(path, 路)
  - 頂点と頂点を結ぶ経路
  - 有向グラフの場合は向きも揃っていないといけない

  例: 頂点 v_1, v_4 間の道は, 辺 (v_1, v_2), (v_2, v_4) の列
- 閉路(cycle, closed path)
  - 頂点からその頂点自身への道が存在するとき, その道を閉路という

  例: 辺 (v_1, v_2), (v_2, v_3), (v_3, v_1) は閉路
- 単純路(simple path)
  - 道のうちで閉路を含まないもの, つまり同じ頂点を 2 回以上通らない道
- 木(tree)は閉路のない無向グラフ



頂点の次数

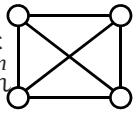
- 頂点への接続する辺の数: 次数
  - 有向グラフの場合
    - 内向き: 入次数(Inner Demi-Degree) $d_G^-(v)$
    - 外向き: 出次数(Outer Demi-Degree) $d_G^+(v)$
  - 無向グラフの場合
    - 次数(degree)

$d_G^-(v) = 2$

$d_G^+(v) = 3$

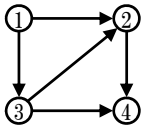
グラフの計算量

- 頂点の数：n
- 辺の数：m
- 無向グラフでは m は最小 0，最大 $\frac{n(n-1)}{2}$
- 有向グラフでは最大 $n(n-1)$
- 辺の数最大のグラフ：
  完全グラフ（Complete Graph）
  - すべての頂点間に辺がある
- グラフの計算量は，頂点数 n と辺数 m に依存し，例えば $O(m+n)$ のアルゴリズムと $O(n \log n)$ のアルゴリズムがあった場合，m が n に比べて小さければ前者を，大きければ後者を選ぶ



グラフの表現 (1)

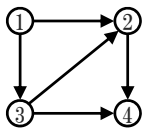
- 有向グラフ $G=(V,E)$ の表現
- 頂点を 1，…，n の番号で表現



対象とする有向グラフの例

グラフの表現 (2)

- 隣接行列：Adjacency Matrix
- n 次の正方行列
- 辺 (v_i,v_j) が存在するときに i 行 j 列の要素 a_{ij} を 1 とし，存在しない場合に 0 とする



$$\begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \end{array}\begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

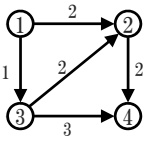
自身への辺が常にあるとする場合 (v_i, v_i) =1

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

自身への辺は 0 とする場合

グラフの表現 (3)

- 重みつきグラフの場合
  - 通常は隣接行列の他に重み行列が必要
  - 辺が存在しないことを無限大などの特別な値で表現すれば隣接行列のみで表現可能

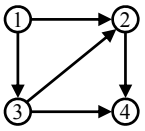


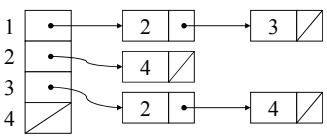
$$\begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \end{array}\begin{array}{c} \begin{array}{cccc} v_1 & v_2 & v_3 & v_4 \end{array} \\ \begin{bmatrix} 0 & 2 & 1 & \infty \\ \infty & 0 & \infty & 2 \\ \infty & 2 & 0 & 3 \\ \infty & \infty & \infty & 0 \end{bmatrix} \end{array}$$

- 自分自身への辺の重みは 0
- 辺がない場合の重みは ∞

グラフの表現 (4)

- リスト表現
  - それぞれの頂点を始点とする辺のリストを頂点毎に作成
- リスト 1 本 1 本を隣接リストと呼ぶ





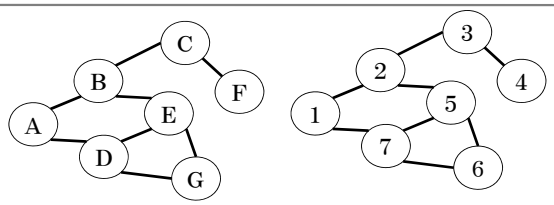
グラフの探索

- グラフの探索 (search)
  - グラフ全体を組織的に調べ，すべての頂点や辺を辿るアルゴリズム
  - 前回までの探索と違って，目的の頂点を探すわけではなく，すべての頂点を調べることである
  - 探索の過程でそれぞれの頂点を調べに行くことをその頂点の訪問 (visit) と呼ぶ
- 探索法の種類
  - 深さ優先探索 (Depth First Search)
  - 幅優先探索 (Breadth First Search)

深さ優先探索（Depth First Search，縦型探索）

- 1 つの道を選んでいけるところまで行き，進めなくなったら引き返して別の道を選ぶ探索法
- 手順
  1. 頂点を 1 つ選び出発点とする
  2. そこから出る辺を 1 つ選びその先の頂点を訪問
  3. この頂点からも同様に辺を選び先の頂点を訪問
  4. 以下，同様に行き止まりまで進む
     - 行き止まり ＝ 訪問済み頂点，辺のない頂点
  5. 行き止まりになったら引き返して調べてない辺の先の頂点を訪問
  6. これを繰り返し，すべての頂点から出る辺を探索すれば終了

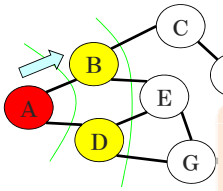
深さ優先探索



- A から始める
- 同じ深さの場合は早いアルファベットから処理

処理順：{A, B, C, F, E, G, D}

深さ優先探索

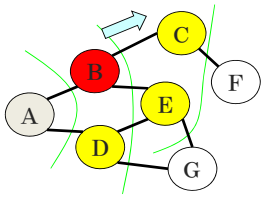


深い頂点（いま訪問した頂点の子）が優先なので，スタックを使う
（スタック ＝ 後から入れたもの優先）

処理中：「A」　スタック：{B, D}
処理済：ϕ

次は B

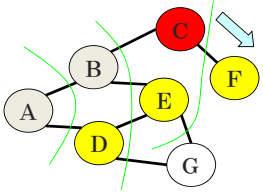
深さ優先探索



処理中：「B」　スタック：{C, E, D}
処理済：{A}

次は C

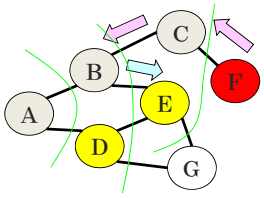
深さ優先探索



処理中：「C」　スタック：{F, E, D}
処理済：{A, B}

次は F

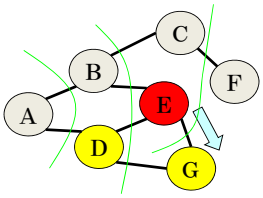
深さ優先探索



処理中：「F」　スタック：{E, D}
処理済：{A, B, C}

次は E

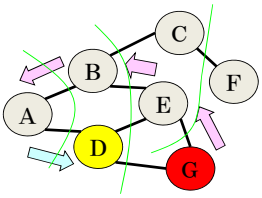
深さ優先探索



処理中:「E」　スタック:{G, D}
処理済:{A, B, C, F}

次は G

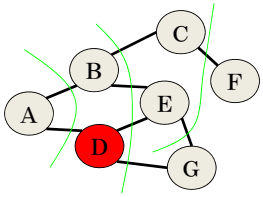
深さ優先探索



処理中:「G」　スタック:{D}
処理済:{A, B, C, F, E}

次は D

深さ優先探索



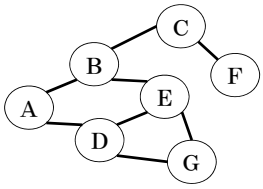
処理中:「D」　スタック:φ
処理済:{A, B, C, F, E, G}

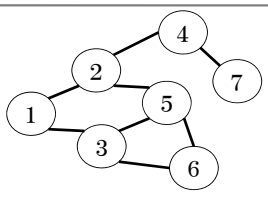
次がないので終了

幅優先探索 (Breadth First Search, 横型探索)

- 手順
  1. 最初の頂点を訪問
  2. この頂点から到達可能な頂点(レベル 1 頂点)をすべて訪問
  3. レベル 1 頂点のいずれかから到達可能な頂点(レベル 2 頂点)を訪問
  4. 以上を繰り返す
  5. 一度訪問した頂点は二度と訪問しない

幅優先探索

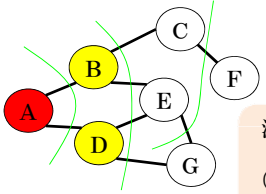




- A から始める
- 同じ深さの場合は早いアルファベットから処理

処理順:{A, B, D, C, E, G, F}

幅優先探索

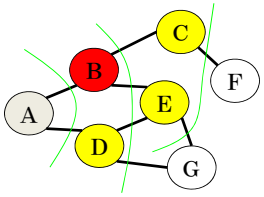


浅い頂点(いま訪問した頂点の兄弟)が優先なので, キューを使う(キュー ＝ 先に入れたもの優先)

処理中:「A」　キュー:{B, D}
処理済:φ

次は B

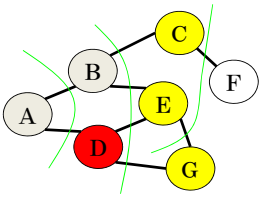
幅優先探索



処理中:「B」　キュー: {D, C, E}
処理済: {A}

次は D

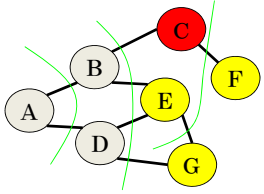
幅優先探索



処理中:「D」　キュー: {C, E, G}
処理済: {A, B}

次は C

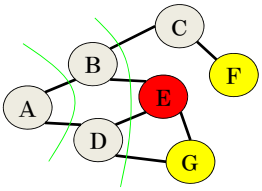
幅優先探索



処理中:「C」　キュー: {E, G, F}
処理済: {A, B, D}

次は E

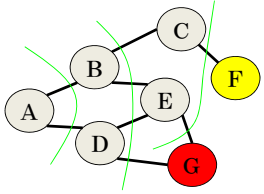
幅優先探索



処理中:「E」　キュー: {G, F}
処理済: {A, B, D, C}

次は G

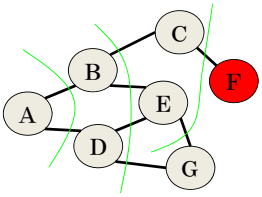
幅優先探索



処理中:「G」　キュー: {F}
処理済: {A, B, D, C, E}

次は F

幅優先探索

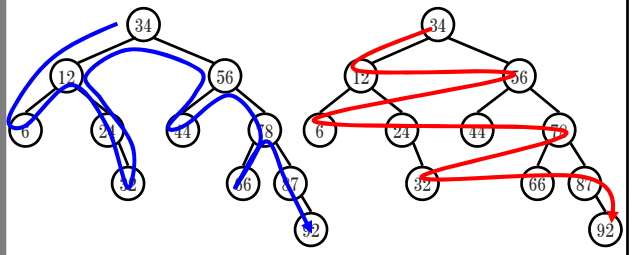


処理中:「F」　キュー: ϕ
処理済:{ A, B, D, C, E, G}

キューが空なので終了

比較

➢深さ優先探索　と　幅優先探索



第 9 講のまとめ

➢グラフ(graph)
- ➢グラフとは
- ➢グラフの表現
- ➢グラフの探索
  - ➢深さ優先探索
  - ➢幅優先探索