JOI春季セミナー 上級コース Day1－1

10章 データ構造(3)：グラフと木 2024.03.21

by **電気通信大学4年 後藤 照佳**

10章 データ構造(3)：グラフと木

グラフと知り合いになろう

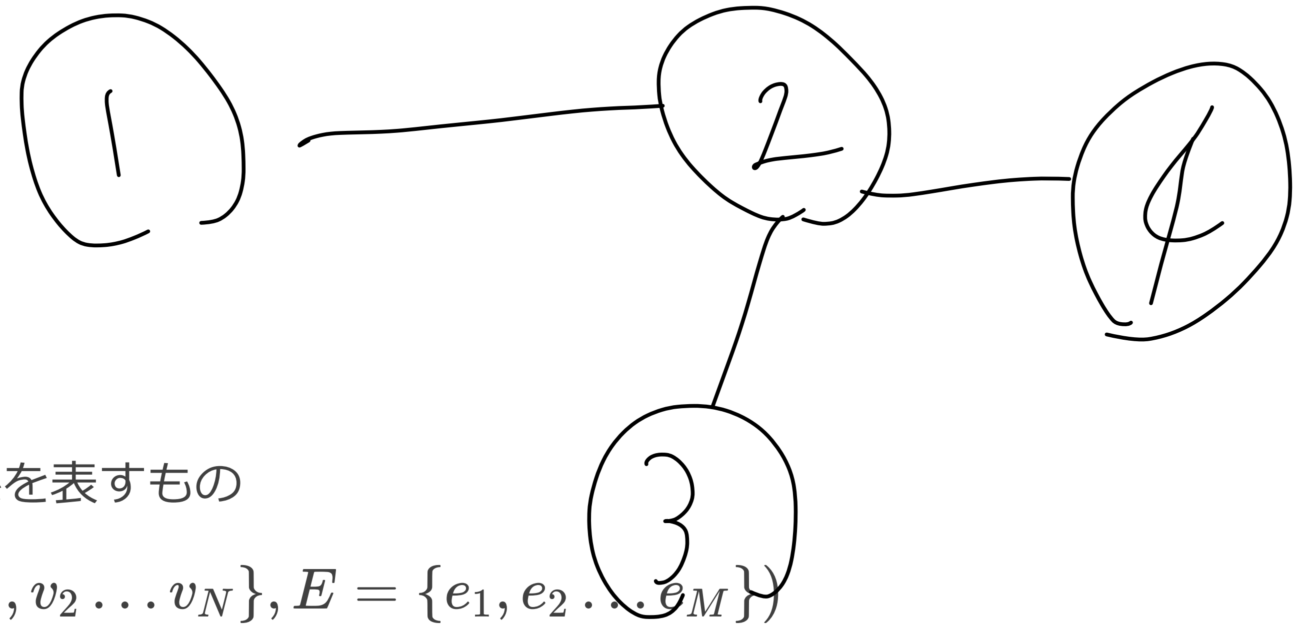
- 担当者：後藤 照佳
- 目標：競プロにおけるグラフに関する用語や基礎知識を身に着けよう

このコマでやること

以下の内容について説明します．最後は学んだことを生かして問題を解きます．

1. **グラフ** ってなに？
2. **プログラムの上での** グラフの表し方
3. **隣接行列** と **隣接リスト** のお話
4. グラフ理論における **木**
5. その他グラフを用いたデータ構造の話
6. グラフの知識を使って解ける問題を解いてみよう！

1. グラフってなに？



頂点(vertex) と **辺(edge)** を用いて関係を表すもの

- 数学的には $G=(V,E),(V=\{v_1,v_2\dots v_N\},E=\{e_1,e_2\dots e_M\})$
- $N=$ 頂点数, $M=$ 辺数 であることが多い

1. グラフってなに？(グラフの用語色々紹介コーナー)

頂点の隣接と辺の接続

- $(v_1) - e_1 - (v_2)$
  - v_1, v_2は **隣接している**
  - e_1はv_1, v_2に **接続している**

無向グラフと有向グラフ

- 矢印がついているやつとついてないやつ

重み付きグラフ

- 辺にコストがのってるやつ

単純グラフ

- 自己ループ$e = (v, v)$や多重辺(ある頂点間に複数辺がある)がないグラフ

1. グラフってなに？(グラフの用語色々紹介コーナー)

部分グラフ

- 元のグラフの一部を切り取ったもの
  - $G' = (V', E')\ (V' \subseteq V, E' \subseteq E)$

歩道(walk) s-t路

- $s, t \in V$について，sからtへ到達可能である時，その経路のこと
- 始点と終点が同じ歩道を **閉路(サイクル)** といったりする

道，パス(path)

- 同じ頂点を2度以上通らない歩道のこと

連結である

- 任意の2頂点間にパスが存在する時グラフが連結であるという

1. グラフってなに？(グラフの用語色々紹介コーナー)

次数

- ある頂点からでている辺の数のこと
- 有向グラフのとき，入ってくる辺の数を **入次数** ，出ていく辺の数を **出次数** という
- 関連：握手補題
  - すべての頂点の次数の和は辺の2倍である

橋

- 無向連結グラフで、削除したときに連結成分が増える辺のこと
  - ないとバラバラになっちゃうやつのことね

ここでちょっと，問題！

このグラフ，なーんだ！

1行目は頂点数と辺数を表す$N\ M$が並ぶ．残りの行は2つの頂点を結ぶ辺が存在することを示す．

```
7 7
1 3
2 7
3 4
4 5
4 6
5 6
6 7
```

皆さんはこのグラフ，描けますか？

ここでちょっと，問題！

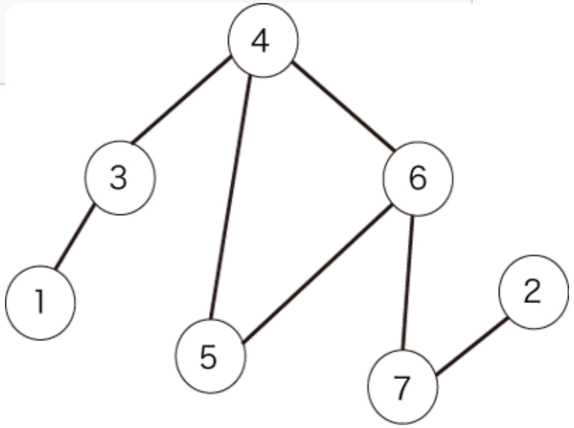
このグラフ，なーんだ！ (出典ABC054 C)

1行目は頂点数と辺数を表すN Mが並ぶ．残りの行は2つの頂点を結ぶ辺が存在することを示す．

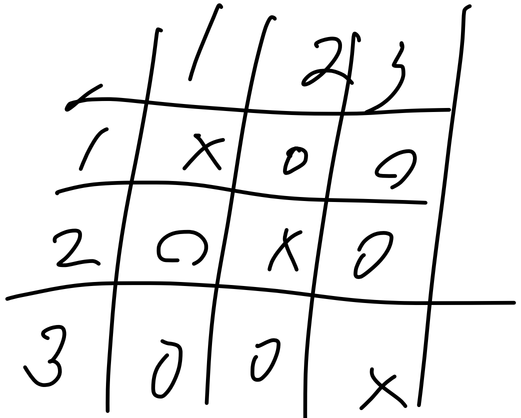
```
7 7
1 3
2 7
3 4
4 5
4 6
5 6
6 7
```

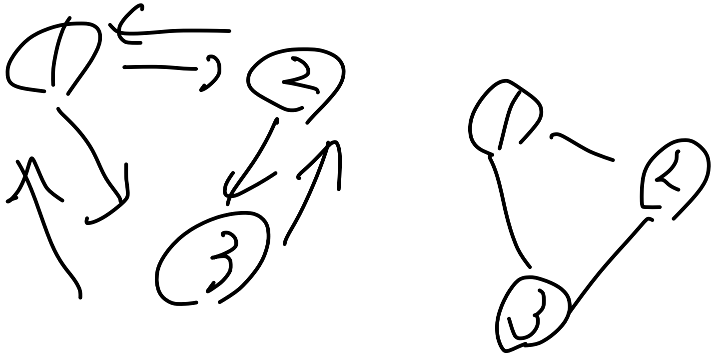
皆さんはこのグラフ，描けますか？

答え→



2. プログラムの上での グラフの表し方





手でかけてもプログラム内でどう表現すればいいかわからない！ あるあるだよね〜

どうやったら表現できるかな？考えてみよー
(知ってる人もいるみたいだけど)

3. 隣接行列 と 隣接リスト のお話

重要キーワードの説明に入ります．頂点数はNとします

隣接行列(けんちょん本省略)

- $N \times N$ 行列Gを考える．$G_{i,j} = 1$のとき，v_i, v_j間に辺があることを表現する
- 重み付きの場合は=1ではなく，重みにすることも

隣接リスト

- 「その頂点に接続された辺のもう一つの端点を記録した配列」をすべての頂点に対して用意する
  - 例：G[1]={2,3} → (1,2)，(1,3)の辺が1にはつながっている

3. 隣接行列と隣接リストのお話

- 隣接行列は頂点数が大きいと空間計算量が足らなくなる
- 隣接リストのほうが少ない空間で表現可能
- グリット上は隣接行列を使うと簡単かも
    - メモリに余裕がある時ね

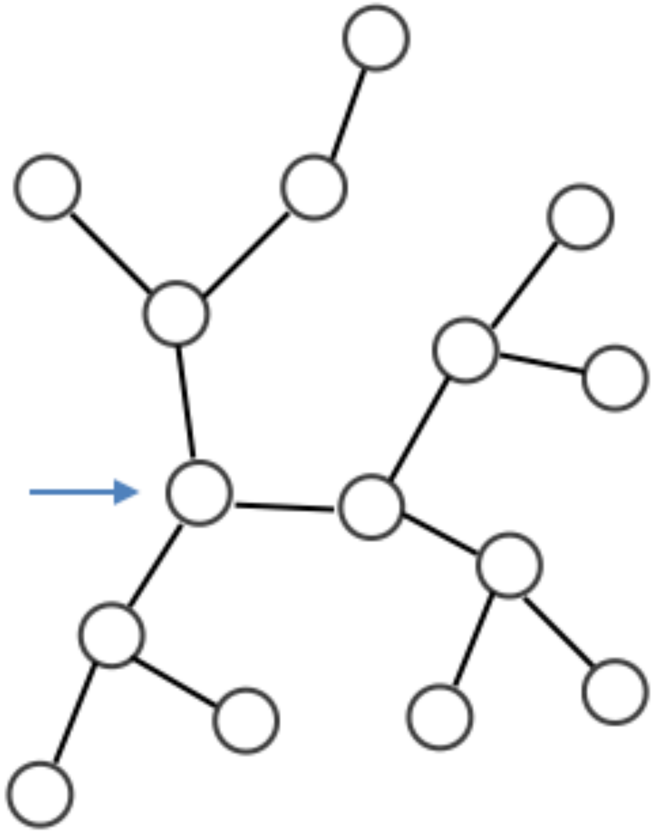
探索方法は明日やる予定.

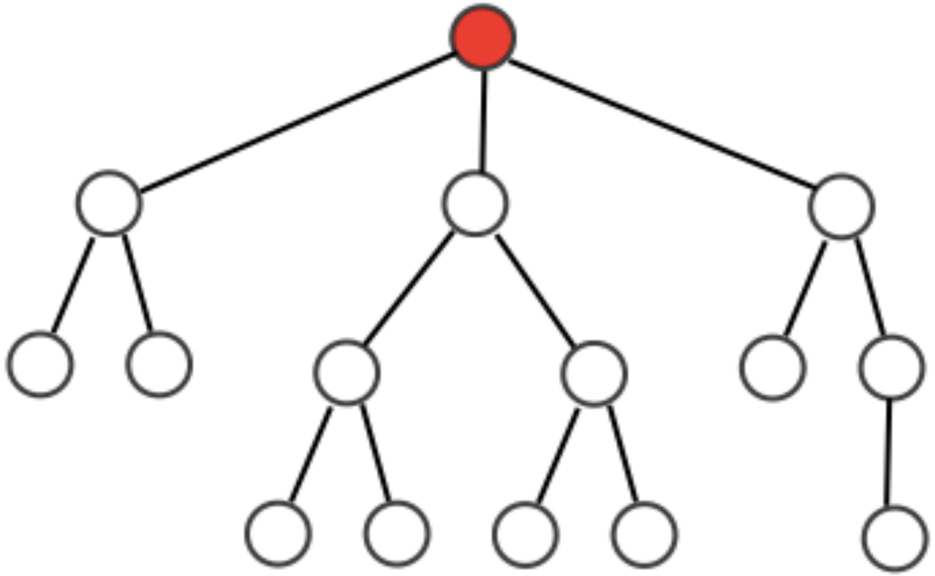
4. グラフ理論における 木

木です.

4. グラフ理論における 木

木です．(グラフ理論)





画像引用元：https://qiita.com/drken/items/a803d4fc4a727e02f7ba

4. グラフ理論における **木**

用語説明

- 根付き木
  - 一個の頂点を一番「上」にあると考えて上下関係を考える木のこと
  - **根** ：一番「上」の頂点
- v_1, v_2が辺で結ばれているとき
  - 根に近い方を **親** ,遠い方を **子** と呼ぶ.
- 共通の親を持つとき， **兄弟** という
- 根付き木上の点v_1, v_2について
  - v_2と根の間にv_1があるとき，v_1はv_2の先祖であり，v_2はv_1の子孫である.
- 根以外の頂点で，その頂点に接続している辺が1本しかないものを **葉** と呼ぶ
- 頂点vから根まで道の長さをvの **深さ** という．各頂点の深さの最大値を木の **高さ** という.
  - 便宜上，根の深さは0

4. グラフ理論における 木

木の性質

- 連結で，$N-1$ 本の辺を持つ
- 閉路はなく，$N-1$ 本の辺を持つ
- 連結で，すべての辺は橋である
- 任意の2点を結ぶ道がちょうど1つある
- 新しい辺を付け加えると閉路が必ず1つできる

証明はうちの大学の講義だと3年で出てくる．ひとまずそういうものなんだと思っておけばOK

木の直径(オマケ)

- 木の中の2頂点を結ぶ道のうち一番長い道の長さを直径という．

スペース余ったし...

5. その他グラフを用いたデータ構造の話(紹介程度)

ヒープ

- 二分木を用いる構造
- C++でいうと std::priority_queue
- 値の挿入と削除が$O(\log N)$でできる
- 最大値(or最小値)を$O(1)$で取得するための構造
- 値xをキーに持つ要素を検索しろ！→苦手($O(N)$かかる)

平衡二分探索木

- C++でいうと std::set,std::map
- 挿入，削除，検索(xはあるか？)が全部O($\log N$)！
- ヒープが得意な最大値の取得が$O(\log N)$でできる
- 定数が重め...

6. グラフの知識を使って解ける問題を解いてみよう！

練習問題一覧です．(解かれた数が多い順に修正しました)

1. ABC262 B - Triangle (Easier)
2. ABC054 C - One Stroke Path
3. 木の直径練習問題(AOJ)
4. 典型90 003 - Longest Circular Road（★4）

ごめん！ほか思いつかん...(後半2つはグラフ探索のの先取り)

例題1

1. $G_1 = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$ が表すグラフを図示しなさい
2. G_1を隣接リストで表現しなさい

例題2 問題概要

- 頂点数 N，辺数 M の有向グラフの隣接リストを以下の形式で出力しなさい
- 入力の頂点は **1-indexed** だが，出力では **0-indexed** であることに注意！
- 頂点番号[出次数]: <頂点リスト(空白区切り)>

入力形式

```
N,M
A_1 B_1
...
A_M B_M
```

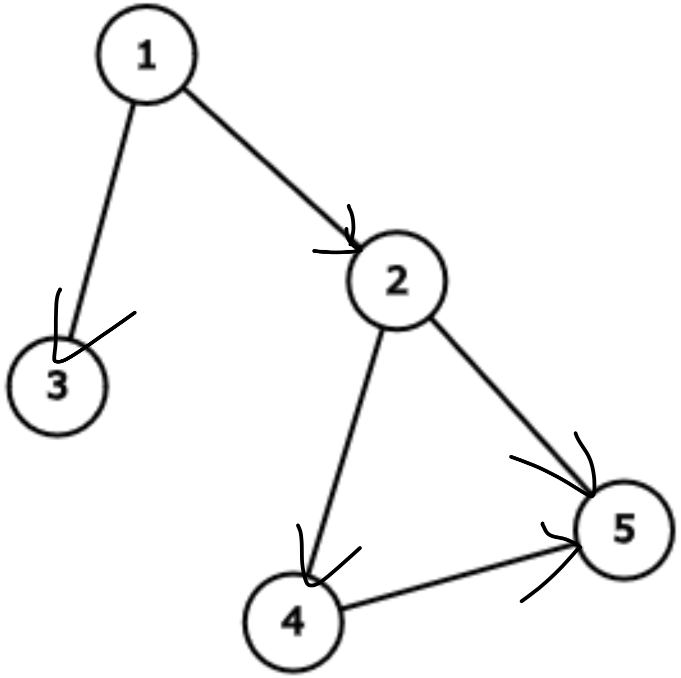
例題2 入出力例

入力

```
5 5
1 2
1 3
2 4
2 5
4 5
```

出力

```
N:5 M:5
0[2]: 1 2
1[2]: 3 4
2[0]:
3[1]: 4
4[0]:
```



解答例(例題2)

```
int main(){
    int N,M;
    cin>>N>>M;
    vector<int> G[100];
    for (int i = 0; i < M; i++) {
        int a,b;
        cin>>a>>b;
        G[a-1].push_back(b-1); // 0-indexedへ
        //G[b-1].push_back(a-1); 有向なら
    }
    printf("N:%d M:%d\n",N,M);
    for (int i = 0; i < N; i++) {
        cout<<i;
        cout<<"["<<G[i].size()<<"]:";
        for(auto x:G[i]){
            cout<<" "<<x;
        }
        cout<<endl;
    }
    return 0;
}
```