JOI春季セミナー 上級コース Day1－2

11章 データ構造(4)：Union-Find 2024.03.21

by **電気通信大学4年 後藤 照佳**

11章 データ構造(4)：Union-Find

あの人はどのグループ？を判別してみよう

- 担当者：後藤 照佳
- 目標： **Union-Find** とばれるアルゴリズムの理解をしよう！

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．(テキストp181-191)

1. Union-Findってなに？
2. データ構造とは？
3. Union-Findの実装の話
4. Union-Findを用いた連結判定
5. Union-Findを使って解ける問題を解いてみよう！

1. Union-Findって何？

Union-Findはグループ分けを管理するデータ構造．素集合データ構造とも

- **Union** ：2つの集合を1つにまとめる
- **Find** ：ある要素がどの集合に属しているかを求める
  - これができると2つの要素が同じ集合にあるかどうかがわかるね

この2つができる **データ構造** のことを指します．

上記の操作を高速に行うアルゴリズムそのものを指して呼ぶこともあると思います．(要出典)

データ構造？？？結局何ができるの？？？？

2. データ構造とは

データの持ち方 のこと(テキストp122)を指します.

キューやスタックもその一部！配列とかもそうだね.
8章や9章で扱ってる内容がまさにデータ構造の一部だね

イメージは掴めたかな？

スタック

最後に積み上げた本を
先に取り出す

キュー

最初に並んだ人を
先にサービスする

画像引用元：https://qiita.com/drken/items/6a95b57d2e374a3d3292

2. データ構造とは (図解用白紙)

1. Union-Findって何？

本題に戻ろう．UnionFindはデータ構造でした．

つまり，データを工夫して持つと

要素同士が同じ集合に属しているかどうか が高速にわかるよ！

ということです．ここからできることを考えるとこんな感じになります

> **UnionFindを使ってできること(一例)**

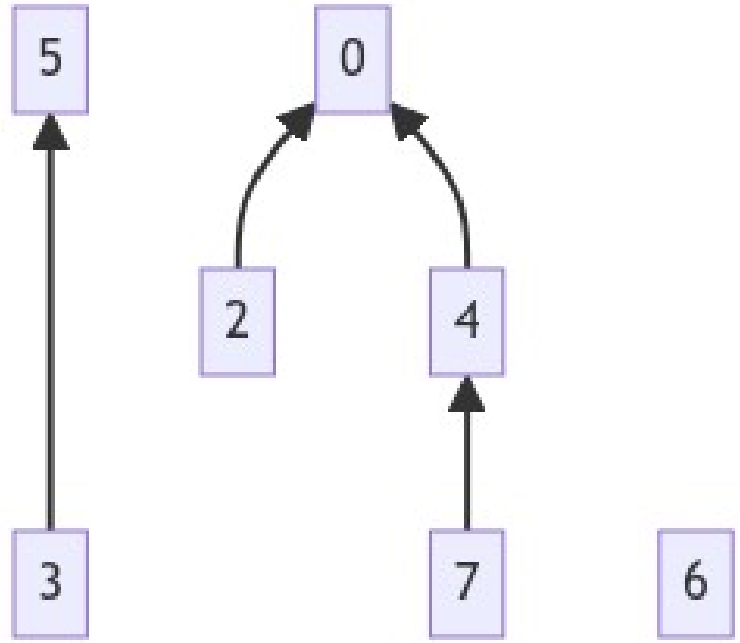
- 同じグループかどうかの判別
- 知り合いの知り合いの...とやってたどり着けるかどうか
- グラフの連結判定
    - 繋がってるかどうかという話

3. Union-Findの実装の話

動作のイメージをつかんでもらうために先にuniteから紹介する.

unite(2,3)を行う

1. **root(2),root(3)を求める** (それぞれ 0,5)
2. すでに同じグループなら何もしない(root(2)==root(3))
3. そうでない時，root(3)の親をroot(2)にする

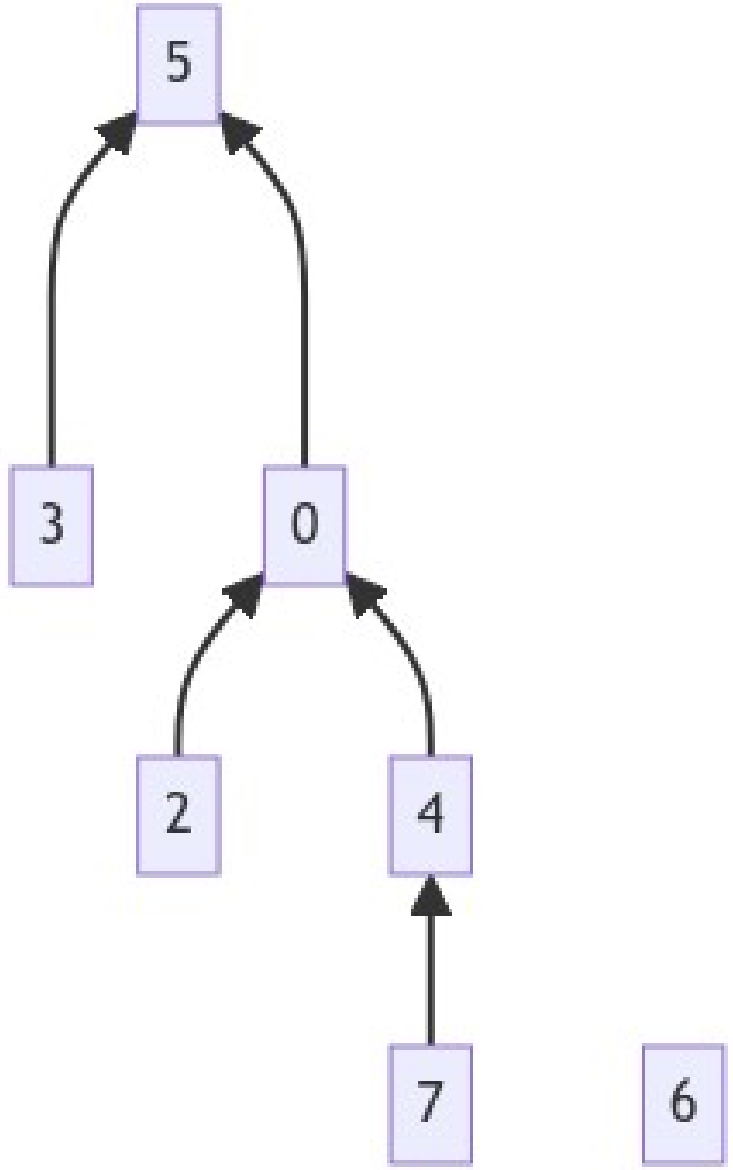


3. Union-Findの実装の話

動作のイメージをつかんでもらうために先にuniteから紹介する.

unite(2,3)を行う

1. root(2),root(3)を求める
2. すでに同じグループなら何もしない(root(2)==root(3))
3. **そうでない時，root(3)の親をroot(2)にする** (0-->5を付けた)



3. Union-Findの実装の話

先に出てきたrootはどうやって実装するの？

根を取得する方法(code 11.1)

```
int root(int x) {
    if (par[x] == -1) return x; // x が根の場合は x を直接返す
    else return root(par[x]); // x が根でないなら再帰的に親へと進む
}
```

`par[x]=root(par[x])` とすることで，経路圧縮を行うことができる．
ただ，これだと **効率が悪くなる場合** が存在する．どんなパターンだろうか?

3. Union-Findの実装の話

経路圧縮について (code 11.2)

先の実装だと，3の親を探すのときにどうなるか考える

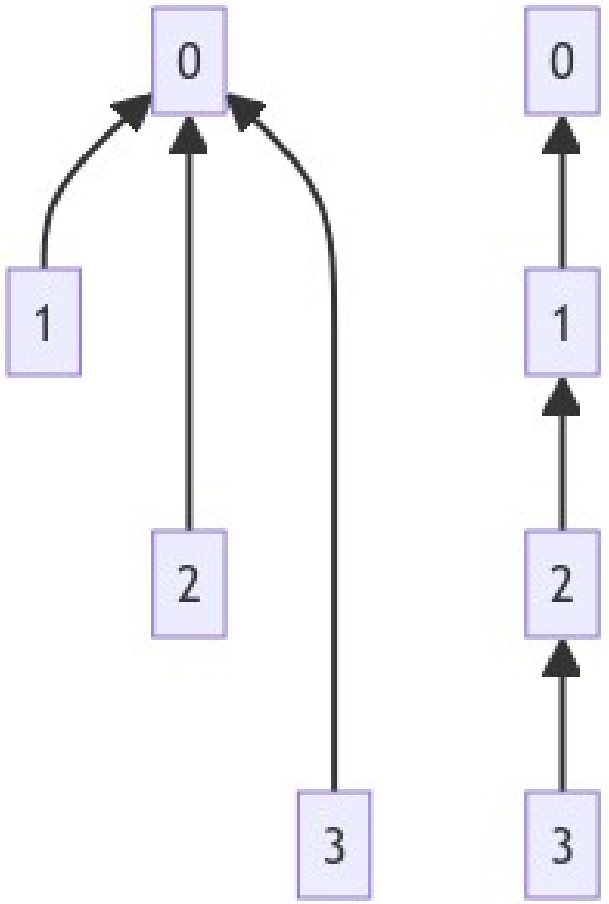
`root(3)→root(2)→root(1)→root(0)` と進み，0が帰ってくる．

直接根に繋げたほうが効率が良さそう！となります．

```
int root(int x) {
    if (par[x] == -1) return x; // x が根の場合は x を直接返す
    else return par[x]=root(par[x]); // x が根でないなら再帰的に親へと進む
}
```

左図だと **rootを1回** 呼び出せば親にたどり着く

右図だと **rootを頂点数** だけ呼び出す必要がある．(最悪ケース)



3. Union-Findの実装の話

sameとsize，uniteの実装例 (code 11.3)

```
// x と y が同じグループに属するかどうか (根が一致するかどうか)
bool issame(int x, int y) { return root(x) == root(y); }
// x を含むグループと y を含むグループとを併合する
bool unite(int x, int y) {
    // x, y をそれぞれ根まで移動する
    x = root(x);
    y = root(y);
    // すでに同じグループのときは何もしない
    if (x == y) return false;
    // union by size (y 側のサイズが小さくなるようにする)
    if (siz[x] < siz[y]) swap(x, y);
    // y を x の子とする
    par[y] = x;
    siz[x] += siz[y];
    return true;
}
// x を含むグループのサイズ
int size(int x) { return siz[root(x)]; }
```

3. Union-Findの実装の話

union findにともなう計算量の話

木の高さの分だけ根に向かって遡ることで親が求まる
→ 高さがどれだけになるかで，計算回数が求まる！

木の高さを抑える工夫

→ 頂点数が少ない方をくっつける．実装は以下

```
// union by size (y 側のサイズが小さくなるようにする)
if (siz[x] < siz[y]) swap(x, y);
```

3. Union-Findの実装の話

まとめるときに頂点はどれくらい増えるの？

s(xが含まれている)とs'をまとめることを考える

1. $s \leq s'$ のとき，xの深さは１だけ増える
2. $s > s'$ のとき，xを含む根付き木の深さは変化しない

と考えると，１増えるごとに頂点数は2倍以上になる...！

$$2^h \leq N \iff h \leq \log N$$

となり，高さhは$\log N$以下だと言える！計算量は$O(\log N)$となりますね
rootの経路圧縮もするともっと早くなって嬉しいね！となります

4. Union-Findを用いた連結判定

1. 辺の両端点をuniteする
2. sameで連結判定可能！

この性質を用いて，様々な問題を解いていくことになります！

5. UnionFind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. ACL Practice Contest A - Disjoint Set Union
2. AtCoder ABC 075 C - Bridge
3. ABC177 D - Friends
4. 競プロ典型 90 問 012 - Red Painting
5. ARC106 B - Values
6. ABC120 D - Decayed Bridges
7. ABC097 D - Equals
8. ABC049 D – 連結
9. ABC264 E - Blackout 2
10. ABC256 E - Takahashi's Anguish

オマケ

Ryo Suzuki さん作 Union-Find可視化

- https://siv3d.jp/web/sample/unionfind/unionfind.html

Union-Find 実装例

- C++
  - https://github.com/drken1215/book_algorithm_solution/blob/master/codes/chap11/code_11_4.cpp
- Python
  - AC-Libraryを参考に(こちらではDSUと呼ばれている)
  - https://github.com/not522/ac-library-python/blob/master/atcoder/dsu.py
    - 使い方：https://atcoder.github.io/ac-library/master/document_ja/dsu.html

とりあえず貼って使ってみてね

初心者環境構築問題

皆さんは大丈夫ですよね？

今回はWeb上のEditorを使って解決

- Simple C++ Editor
  - https://tumoiyorozu.github.io/SimpleCppEditor/
- PyTry
  - https://pro-ktmr.github.io/pytry/

この後の進め方

1. テキストのサンプルコードを写します(自力で書けちゃう天才は写さなくてもOK)
    i. Python Onlyの方へ → ごめん．ググって....
2. 質問があれば聞いてください
    i. これは講義内容でも問題についてでもOK
3. **詰まったらすぐ声かけてください**
    i. わからないところを一緒に探すとかもやってあげたいから是非ね
    ii. **マジで声かけてください**．わからないまま黙ってると辛いからね

いや，でもよくわからん...

わかる．自分が中学生，高校生のときなら多分わかってない．

結局 **実践あるのみ** なのよね．やってきましょう．

もしわからないことが言語化できている(＝質問ができる)のなら積極的に聞いてください！

ちなみにだけど当時の僕より君達できてるから自信持っていいよ！

サンプルコード

注意書き

- C++ は少し僕のテンプレートが乗ってる実装になってます
- Pythonはなるべくきれいに書きました
- 下のリンク先から言語を選んで見てみてね

実装例

- Python
  - https://atcoder.jp/contests/practice2/submissions/44632039
- C++
  - https://atcoder.jp/contests/practice2/submissions/44632541