

アルゴリズムクイック リファレンス

Algorithms in a Nutshell

第6章：グラフアルゴリズム
お気持ちスライド

@rian_tkb

2018/05/31

6章：グラフアルゴリズム

▶ この章の内容

▶ DFS、BFS、最短経路、最小全域木

▶ この回もそこそこ重いとは思うのですが、
言いたいこともそこそこあるので難しいにゃあ.....



色指定
仕上検査

藤田 舞

仕上

WTish

奥井恵美子

千葉陽子

T2 studio

イーグルネスト

泉 貴明

畠野玲美

伊藤淳帆

山瀬仁美

田中照佳

▶ Dijkstra 法ってよくわからない.....

- ▶ なんとなくやろうとしてることはわからなくもないけど、実装できる気がしない.....



Dijkstra

- ▶ そんなことはない、Dijkstra は幅優先探索の **Queue** を **Priority Queue** に変えただけ！



幅優先探索

```
vector<int> bfs(int s, vector<vector<int>> edges) {
    int n = edges.size();
    vector<int> dist(n, INF);
    queue<int> q;
    dist[s] = 0;
    q.push(s);
    while (!q.empty()) {
        int e = q.front();
        q.pop();
        for (int ne : edges[e]) {
            if (dist[ne] > dist[e] + 1) {
                dist[ne] = dist[e] + 1;
                q.push(ne);
            }
        }
    }
    return dist;
}
```

▶ これが普通の
幅優先探索のコード

幅優先探索

```

vector<int> bfs(int s, vector<vector<edge>> edges) {
    int n = edges.size();
    vector<int> dist(n, INF);
    queue<edge> q;
    dist[s] = 0;
    q.push(edge(0, s));
    while (!q.empty()) {
        edge e = q.front();
        q.pop();
        for (edge ne : edges[e.to]) {
            if (dist[ne.to] > dist[e.to] + ne.cost) {
                dist[ne.to] = dist[e.to] + ne.cost;
                q.push(edge(dist[ne.to], ne.to));
            }
        }
    }
    return dist;
}

```

- ▶ とりあえず Queue のまま辺の重さに対応する
- ▶ いっぱい変わってるよう見えるかもだけど
本質的な部分は同じ

幅優先探索

- ▶ ちなみに edge はこんな感じ

```
struct edge {  
    int cost, to;  
    edge(int c, int t) : cost(c), to(t) {}  
    bool operator<(const edge& e) const {  
        return cost > e.cost;  
    }  
};
```

Dijkstra

```

vector<int> dijkstra(int s, vector<vector<edge>> edges) {
    int n = edges.size();
    vector<int> dist(n, INF);
    priority_queue<edge> q;
    dist[s] = 0;
    q.push(edge(0, s));
    while (!q.empty()) {
        edge e = q.top();
        q.pop();
        for (edge ne : edges[e.to]) {
            if (dist[ne.to] > dist[e.to] + ne.cost) {
                dist[ne.to] = dist[e.to] + ne.cost;
                q.push(edge(dist[ne.to], ne.to));
            }
        }
    }
    return dist;
}

```

- ▶ Queue を Priority Queue に変えただけで Dijkstra になった！

Dijkstra

▶ 深さ優先・幅優先の例題

- ▶ https://beta.atcoder.jp/contests/atc001/tasks/dfs_a
- ▶ https://beta.atcoder.jp/contests/abc007/tasks/abc007_3
- ▶ https://beta.atcoder.jp/contests/abc020/tasks/abc020_c
- ▶ https://beta.atcoder.jp/contests/abc021/tasks/abc021_c

▶ Dijkstra の例題

- ▶ https://beta.atcoder.jp/contests/abc035/tasks/abc035_d
- ▶ https://beta.atcoder.jp/contests/abc022/tasks/abc022_c

最短経路復元

- ▶ 競プロの問題ではそこまで多くないですが、
**最短経路のコストだけでなく最短経路が実際にどのような経路を
辿るか**、という情報が欲しい場合もあり、それも同じ計算量で
実現できます
- ▶ 多分アルゴリズムクイックリファレンスの方に正解は載ってると
思うのでここでは詳しい話は割愛します



最小全域木

- ▶ 最小全域木を求める有名なアルゴリズムには

- ▶ プリム法
- ▶ クラスカル法

の2つがあります



- ▶ どちらも「今追加できる辺のうち最大のものを追加する」という操作を貪欲に繰り返すことにより**最適解**を求めている**貪欲法**です
 - ▶ どうしてそれで最適解が求まるのかを考えてみるとよいです

▶ 例題

- ▶ https://beta.atcoder.jp/contests/arc029/tasks/arc029_3
- ▶ https://beta.atcoder.jp/contests/arc076/tasks/arc076_b
- ▶ https://beta.atcoder.jp/contests/code-festival-2016-qualb/tasks/codefestival_2016_qualB_c



木のはなし

13

- ▶ 本当はここから木の話をするつもりだったのですが、どう考へても時間が足らないしなんなら 90 分かけられそう（なのとスライドが全然完成しておらず）なので、そのうち自分が木の話をする会を入れる説があります



木のはなし

▶ 内容（予定）

- ▶ 木とは？
- ▶ 木のなにがいいのか
- ▶ いろいろな木
 - ▶ 根つき木、二分木、Binary Indexed Tree、Segment Tree、.....
- ▶ 木の上でのアルゴリズム
 - ▶ 木の巡回
 - ▶ pre-order、in-order、post-order、オイラーツアー
 - ▶ 木の直径
 - ▶ ダブリング
 - ▶ 最小共通祖先 (Lowest Common Ancestor)

アルゴリズムクイック リファレンス

Algorithms in a Nutshell

第 ?? 章 : 木
お気持ちスライド

@rian_tkb

2018/MM/DD

?? 章：木

16

▶ この章の内容

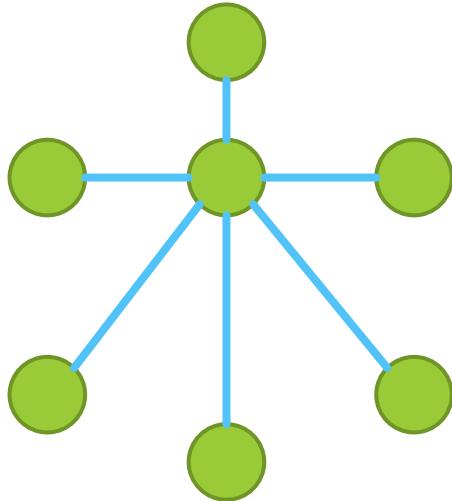
▶ 木

- ▶ 「わたしは木の役を積極的にやりました。
木はいいです。不動のあり方は心があらわれます。」

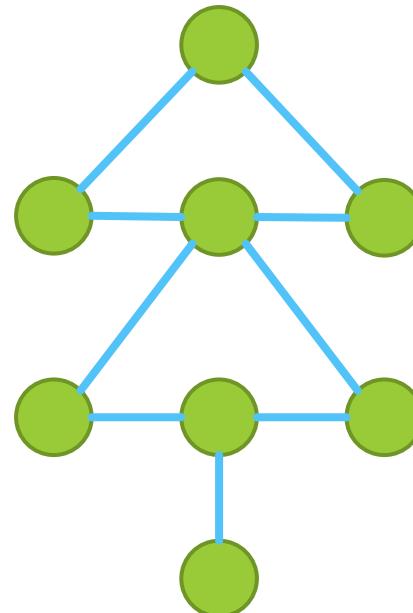


木とは？

- 連結で閉路を持たないグラフ
 - 頂点数を $|V|$ 、辺数を $|E|$ としたとき $|E| = |V| - 1$ が成り立つ



これは木



これは木ではない