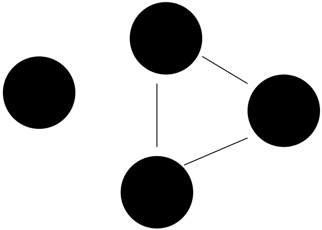
Graph Theory: DFS Applications

David Jacobo
jguillen@cimat.mx

September 18, 2015

Definition

$$G = (V, E)$$

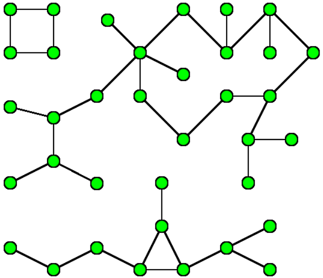


Outline

1. Flood fill
   - Intuition
   - Algorithm
2. Bi-coloring
   - Intuition
   - Algorithm
3. Cycle detection
   - Intuition
   - Algorithm

Flood fill

Using DFS and starting from a vertex **u**, find all the reachable vertices either directly or indirectly.



Usage:

- Component labeling
- Reachability test

Algorithm

Algorithm 1 Flood fill

$\forall\, u \in V, u_{traversed} \leftarrow \text{false}$
$component \leftarrow 0$
for $u \in V$ **do**
  if $u_{traversed} = \text{false}$ **then**
    $\text{dfs}\,(u, component)$
    $component \leftarrow component + 1$
  end if
end for

Algorithm

Algorithm 2 dfs$(u, component)$

1: $u_{traversed} \leftarrow$ true
2: $u_{component} \leftarrow component$
3: **for** $v \in u_{neighbours}$ **do**
4:     **if** $v_{traversed} =$ false **then**
5:         dfs$(v, component)$
6:     **end if**
7: **end for**

Outline

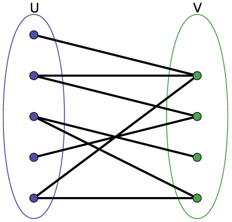
1. Flood fill
   - Intuition
   - Algorithm
2. Bi-coloring
   - Intuition
   - Algorithm
3. Cycle detection
   - Intuition
   - Algorithm

Bi-coloring

Can a graph be labeled with 2 'colors'?

Usage:

- Bipartite matching is based on this algorithm as pre-processing in the case the 2 groups are not given in beforehand.



Algorithm

Algorithm 3 Bi-coloring

1: $\forall\ u \in V, u_{color} \leftarrow$ -1
2: **for** $u \in V$ **do**
3:     **if** u_{color} = -1 **then**
4:         $u_{color} \leftarrow 1$
5:         dfs-color (u)
6:     **end if**
7: **end for**

Algorithm

Algorithm 4 dfs-color(u)

```
for v ∈ u_neighbours do
    if v_color = -1 then
        v_color ← 1 − u_color
        dfs-color(v)
    else if v_color = u_color then
        Fail, not bipartite!
    end if
end for
```

Outline

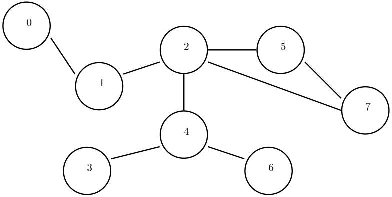
1. Flood fill
   - Intuition
   - Algorithm
2. Bi-coloring
   - Intuition
   - Algorithm
3. Cycle detection
   - Intuition
   - Algorithm

Cycle detection

A cycle could be described as: $v_0, v_1, v_2 ... v_n, v_0$. Could you detect a cycle on the graph, if present?

Usage:

- Topological sorting
- Tree detection
- Strong connected components (on a directed graph)
- Bridges detection
- Articulation point detection



Algorithm

Algorithm 5 Cycle detection

$\forall\ u \in V, u_{color} \leftarrow \text{BLACK}, u_{parent} \leftarrow -1$
for $u \in V$ **do**
    if $u_{color} = \text{BLACK}$ **then**
        dfs-cycle (u)
    end if
end for

Algorithm

Algorithm 6 dfs-cycle(u)

```
u_color ← GRAY
for v ∈ u_neighbours do
    if v_color = BLACK then
        v_parent ← u
        dfs-cycle(v)
    else if v_color = GRAY AND u_parent ≠ v then
        We found a cycle!
    else if v_color = WHITE then
        We don't care about this case for cycles
    end if
end for
u_color ← WHITE
```

Q & A

References

- Competitive Programming site
- Algorists' repository