证明强连通算法的正确性

Kasaraju, Tarjan, Gabow

郑奘巍

zzw@smail.nju.edu.cn

October 28, 2018

Definition

Strongly connected components:

a maximal set of vertices C $\in$ V such that for every pair of vertices u and v in C, we have path from u to v and path from v to u.

Definition

Component Graph:

$G^{SCC} = (V^{SCC}, E^{SCC})$. v_i stands for a strongly connected component c_i in V. Edge (v_i, v_j) exists for there exists edge between a vertex in c_i and c_j.

Lemma (1)

Let C and C' be distinct strongly connected components in directed graph G = (V,E), let u,v ∈ C, let u', v' ∈ C', and suppose that G contains a path from u to u' then G cannot also contain a path from v' to v.

Lemma (2)

Let C and C' be distinct strongly connected components in directed graph G = (V,E). If there is an edge (u,v) ∈ E and u ∈ C, v ∈ C'. After running DFS on G, we get f(C) > F(C').

Proof.

Case 1: d(C)<d(C') -> construct the path -> white-path theorem -> Nesting of descendants' intervals

Case 2: d(C)>d(C') -> all of C' is visited -> lemma 1 □

Above are only proof lines.

Corollary (1)

Let C and C' be distinct strongly connected components in directed graph G = (V,E). Suppose that there is an edge (u,v) $\in E^T$, where u$\in$C and v$\in$ C' then f(C)<f(C').

Algorithm 1 Kasaraju's algorithm

1: **function** STRONGLY-CONNECTED-COMPONENTS(G)
2: call DFS(G) to compute finishing times u.f for each vertex u
3: compute G^T
4: call DFS(G^T), but in the main loop of DFS, consider the vertices in order of decreasing u.f
5: output the vertices of each tree in the depth-first forest formed in line 2 as a separate strongly connected component.

Proof.

Induction to prove: the first kth tree formed by line 5 are all strongly connected components.

k = 0, obvious.

Supposing it holds when k = n-1, when k = n, since all unvisited nodes in strongly connected components C have u.f = f(C) > f(C') for any visited components C'. By Corollary, u cannot reach any other connected components. Thus we get the (k+1)th tree. □

Problem (22.5-3)

Can we DFS(G) in ascending order in line 5? Why?

Problem (22.5-3)

Can we DFS(G) in ascending order in line 5? Why?

Due to lemma 2, f(C) $>$ f(C') (where f(U)=min$\{$u.d$\}$), but cases are there exists v in C and u in C' that v.f $<$ u.f

Algorithm 2 Tarjan's algorithm

```
1: function TARJAN(G)
2:     for all v in V do
3:         if not visited then
4:             DFS'(v)
```

Algorithm 3 Tarjan's algorithm

```
1:  function DFS'(v)
2:      Push(v)
3:      for all edge(v,u) do
4:          if not visited u then
5:              DFS'(u)
6:              low[v] = min(low[v],low[u])
7:          else
8:              if u is in stack then
9:                  low[v] = min(low[v],v.d)
10:     if v.d == low[v] then
11:         Pop all nodes above v in stack to form a strongly connected components
```

Lemma (1)

For all v, there exists w reachable from v such that w.d <= low[v] <= v.d

Lemma (2)

When determining whether v is a root, we have for all w reachable from v that low[v] <= w.d

Proof.

we will prove by conduction that when we meet a node with low[u] == u.d at the kth time, then u and the nodes above u form the kth strongly connected components.

When k = 0 it is obvoius.

Suppose when k<n it is correct. When k=n:

Consider four kinds of nodes: 1. not visited 2. nodes below u in stack 3. nodes above u in stack 4.nodes visited but not in stack.

□

Proof.

Case 1: since the DFS process do not visit v, it cannot be reached from u.

Case 2: Since u.d == low[u], by lemma 2, u cannot reach any node below it.

Case 3: Since it is above u, it can be visited by u. Suppose u is the vertex with smallest u.d cannot be visited by v, by lemma 1, there exists the w reachable from v such that w.d <= low[v] <= v.d. If w.d $<$ u.d, since u can reach v, then u can visit w but w.d $<$ u.d $=$ low[u], which contracts lemma 2. Otherwise, w is above u and since w can reach u, v can reach u.

Case 4: By induction, they belong to other strongly connected components. □

For Gabow's algorithm, it is similar to Tarjan's algorithm instead that it uses one extra stack to substitude u.d and low[u]'s work. You can prove it by comparing with Tarjan.
For more information about Gabow's algorithm, visit Wiki Gabow


Thank
You!