AULA 07 – COMPONENTES FORTEMENTE CONEXOS

Prof. Daniel Kikuti

Universidade Estadual de Maringá

8 de abril de 2015

Sumário

- Resolução exercício
- Introdução
- Componentes fortemente conexos
- Exercícios

Exercício de aula anterior

[Cormen 22.3-12] Mostre que uma busca em profundidade de um grafo não orientado G pode ser usada para identificar os componentes conexos de G, e que a floresta da busca em profundidade contém tantas árvores quantos componentes conexos existem em G. Mais precisamente, mostre como modificar a busca em profundidade de modo que cada vértice v receba a atribuição de uma etiqueta inteira $v.cc$ entre 1 e k, onde k é o número de componentes conexos de G, de tal forma que $u.cc = v.cc$ se e somente se u e v estiverem no mesmo componente conexo.

Resolução

```
Componentes-Conexos(G)
para cada vértice u em G.V
  u.cor = branco
  u.pred = nil
cc = 0;
para cada vértice u em G.V
  se u.cor == branco
    cc = cc + 1
    dfs-visit(u)

dfs-visit(u)
u.cor = cinza
u.cc = cc
para cada vértice v em u.adj
  se v.cor == branco
     v.pred = u
     dfs-visit(v)
u.cor = preto
```

Resolução - Correção do algoritmo

Lema do exercício

Para quaisquer dois vértices s e t em um grafo G, seus componentes conexos são iguais ou disjuntos.

Resolução - Correção do algoritmo

Lema do exercício

Para quaisquer dois vértices s e t em um grafo G, seus componentes conexos são iguais ou disjuntos.

Demonstração

- Se existe caminho de s a t, então $s.cc = t.cc$ (mesmo componente conexo), pois para qualquer vértice v com $v.cc = s.cc$, o vértice v também deve ser alcançável de t por um caminho ($t \rightsquigarrow s \rightsquigarrow v$). O mesmo raciocínio vale para s e t invertidos, e portanto, v está no componente conexo de um, se e somente se está no componente conexo do outro.
- Se não existe caminho de s a t, então $s.cc \neq t.cc$, pois não existe um vértice v que esteja em ambos componentes conexos. Se existisse tal vértice, então poderíamos fazer o percurso $s \rightsquigarrow v \rightsquigarrow t$, construindo um caminho de s a t. Portanto, se não existe caminho de s a t, então seus componentes conexos são disjuntos.

Introdução - Aplicação

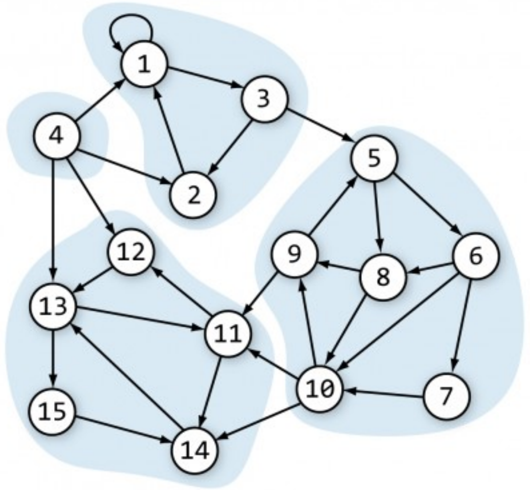


Figura copiada de: http://www.scribegriff.com/studios/index.php?post/2013/03/26/Strongly-Connected-Components-and-the-2-SAT-Problem-in-Dart

Componente fortemente conexo

Definição de SCC

Um **componente fortemente conexo (SCC)** de um grafo orientado $G = (V, E)$ é um conjunto máximo de vértices $C \subseteq V$, tal que para todo par de vértices u e v, existe um caminho de u para v e de v para u.

Componente fortemente conexo

Definição de SCC

Um **componente fortemente conexo (SCC)** de um grafo orientado $G = (V, E)$ é um conjunto máximo de vértices $C \subseteq V$, tal que para todo par de vértices u e v, existe um caminho de u para v e de v para u.

Grafo transposto

O grafo transposto G^T de G possui os mesmos SCC's de G.

Componente fortemente conexo

Definição de SCC

Um **componente fortemente conexo (SCC)** de um grafo orientado $G = (V, E)$ é um conjunto máximo de vértices $C \subseteq V$, tal que para todo par de vértices u e v, existe um caminho de u para v e de v para u.

Grafo transposto

O grafo transposto G^T de G possui os mesmos SCC's de G.

Grafo de componentes

- $G^{SCC} = (V^{SCC}, E^{SCC})$
- V^{SCC} tem um vértice para cada SCC em G
- E^{SCC} contém uma aresta se existe uma aresta correspondente entre os SCC's de G.

Propriedade

Lema

Sejam C e C' componentes fortemente conexos distintos no grafo orientado $G = (V, E)$ e, $u, v \in C$ e $u', v' \in C'$. Se G contém um caminho $u \rightsquigarrow u'$, então G não pode conter também um caminho $v' \rightsquigarrow v$.

Propriedade

Lema

Sejam C e C' componentes fortemente conexos distintos no grafo orientado $G = (V, E)$ e, $u, v \in C$ e $u', v' \in C'$. Se G contém um caminho $u \rightsquigarrow u'$, então G não pode conter também um caminho $v' \rightsquigarrow v$.

Demonstração

Se G contém um caminho de $v' \rightsquigarrow v$, então ele contém um caminho $u \rightsquigarrow u' \rightsquigarrow v'$ e $v' \rightsquigarrow v \rightsquigarrow u$. Portanto u e v' são alcançáveis um do outro, contradizendo o fato de C e C' serem SCC's distintos.

Algoritmo para Componentes Fortemente Conexos

```
Strongly-Connected-Components(G)
1 chamar DFS(G) para calcular o tempo de término u.f
  para cada vértice u;
2 calcular G^T;
3 chamar DFS(G^T) mas, no laço principal de DFS,
  considerar os vértices em ordem decrescente de u.f;
4 os vértices de cada árvore na floresta de busca em
  profundidade formada na linha 3 pertencem a um
  componente fortemente conexo distinto.
```

Um Exemplo

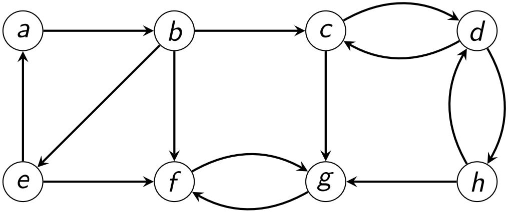


Figura copiada de: http://en.wikipedia.org/wiki/File:Scc.png

Análise do algoritmo

Consumo de tempo

- *DFS* nas linhas 1 e 3 consome $\Theta(V + E)$.
- Conforme os vértices são finalizados na chamada do *DFS* da linha 1, os vértices são inseridos na frente de uma lista ligada ($O(1)$), como cada vértice é inserido apenas uma vez, o tempo total de operações de inserções é $\Theta(V)$.
- O tempo para calcular o grafo transposto na linha 2 é $\Theta(V + E)$.

Análise do algoritmo

Consumo de tempo

- *DFS* nas linhas 1 e 3 consome $\Theta(V + E)$.
- Conforme os vértices são finalizados na chamada do *DFS* da linha 1, os vértices são inseridos na frente de uma lista ligada ($O(1)$), como cada vértice é inserido apenas uma vez, o tempo total de operações de inserções é $\Theta(V)$.
- O tempo para calcular o grafo transposto na linha 2 é $\Theta(V + E)$.

Conclusão

A complexidade do algoritmo `Strongly-Connected-Components(G)` é $\Theta(V + E)$.

Análise do algoritmo - Correção

Para um conjunto $U \subseteq V$, definimos os tempos em função do conjunto (considerando a primeira execução do DFS):

- $d(U) = \min_{u \in U}\{u.d\}$ (menor tempo de descoberta) e,
- $f(U) = \max_{u \in U}\{u.f\}$ (maior tempo de término).

Lema principal

Sejam C e C' SCC distintos em $G = (V, E)$. Suponha que exista uma aresta $(u, v) \in E$, tal que $u \in C$ e $v \in C'$. Então $f(C) > f(C')$.

Análise do algoritmo - Correção

Para um conjunto $U \subseteq V$, definimos os tempos em função do conjunto (considerando a primeira execução do DFS):

- $d(U) = \min_{u \in U}\{u.d\}$ (menor tempo de descoberta) e,
- $f(U) = \max_{u \in U}\{u.f\}$ (maior tempo de término).

Lema principal

Sejam C e C' SCC distintos em $G = (V, E)$. Suponha que exista uma aresta $(u, v) \in E$, tal que $u \in C$ e $v \in C'$. Então $f(C) > f(C')$.

Demonstração

Duas situações possíveis:

- $d(C) < d(C')$
- $d(C) > d(C')$

Análise do algoritmo - Correção

$d(C) < d(C')$

Seja x o primeiro vértice descoberto em C. No tempo $x.d$ todos os vértices de C e C' estão brancos. Como existe uma aresta $(u, v) \in E$, então todos os vértices de C' também são alcançáveis de x (todos os vértices de C e C' são descendentes de x). Portanto, x possui o maior tempo de término, o que implica que $f(C) > f(C')$.

Análise do algoritmo - Correção

$d(C) < d(C')$

Seja x o primeiro vértice descoberto em C. No tempo $x.d$ todos os vértices de C e C' estão brancos. Como existe uma aresta $(u, v) \in E$, então todos os vértices de C' também são alcançáveis de x (todos os vértices de C e C' são descendentes de x). Portanto, x possui o maior tempo de término, o que implica que $f(C) > f(C')$.

$d(C) > d(C')$

Seja y o primeiro vértice descoberto em C'. Todos os vértices de C' são descendentes de y, e portanto, $y.f = f(C')$. No tempo $y.d$, todos os vértices de C estão brancos. Como existe uma aresta (u, v), sabemos pelo lema anterior que não pode haver uma aresta (v, u) e, consequentemente, todos os vértices em C continuarão brancos no tempo $y.f$. Então, para qualquer vértice $w \in C$ temos que $w.f > y.f$ e, portanto, $f(C) > f(C')$.

Análise do algoritmo - Correção

Corolário

Sejam C e C' SCC's distintos em $G = (V, E)$. Suponha que existe uma aresta $(u, v) \in E^T$, tal que $u \in C$ e $v \in C'$, então $f(C) < f(C')$.

Demonstração

$(u, v) \in E^T \Rightarrow (v, u) \in E$. Como os componentes conexos de G e G^T são os mesmos, $f(C') > f(C)$.

Algoritmo - Correção

Teorema

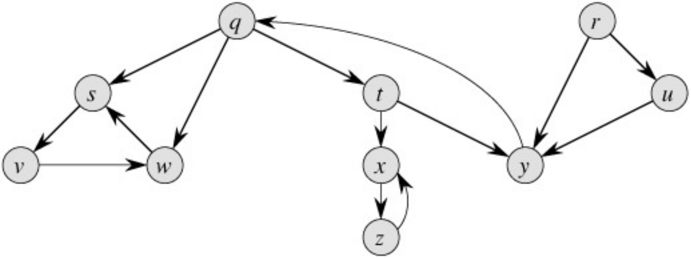
`Strongly-Connected-Components(G)` encontra corretamente os SCC's de um grafo orientado G.

Prova formal por indução – ver livro texto

1. A segunda DFS, começa com um SCC C tal que $f(C)$ é máximo.
2. Seja $x \in C$ o vértice inicial, a segunda DFS visita todos (e apenas) os vértices de C. [Pelo corolário, $f(C) > f(C')$ para todo $C \neq C' \Rightarrow$ não existe aresta de C para C' em G^T].
3. A próxima raiz da segunda DFS está em um SCC C' tal que $f(C')$ é máximo em relação a todos os outros SCC (sem considerar C). DFS visita todos os vértices de C', e as únicas arestas saindo de C' vão para C, cujo os vértices já foram visitados.
4. O processo continua até que todos os vértices sejam visitados.
5. Cada vez que uma raiz é escolhida na segunda DFS, ela só alcança:
   - vértices no SCC dele (através de arestas da árvore);
   - vértices que já foram visitados na segunda DFS.

Exercício 1

[Cormen 22.5-2] Mostre como o procedimento `Strongly-Connected-Components` funciona sobre o grafo da figura abaixo. Especificamente, mostre os tempos de término calculados na linha 1 e a floresta produzida na linha 3. Suponha que o laço das linhas de 5 a 7 de DFS considere os vértices em ordem alfabética e que as listas de adjacências estejam em ordem alfabética.



Exercício 2

[Cormen 22.5-3] O professor Bacon afirma que o algoritmo para componentes fortemente conexos pode ser simplificado pelo uso do grafo original (em lugar do transposto) na segunda chamada do DFS e pela varredura dos vértices na ordem crescente dos tempos de término. Este algoritmo sempre produz resultados corretos?