

SILNĚ SOUVISLÉ KOMPONENTY

Petr Ryšavý

16. září 2025

Katedra počítačů, FEL, ČVUT

PROBLÉM SILNĚ SOUVISLÝCH KOMPONENT

Definice (Silně souvislá komponenta) *Nechť $G = (V, E)$ je orientovaný graf. **Silně souvislé komponenty** grafu G jsou množiny ekvivalence relace $u \sim v$, která je definovaná jako „existuje orientovaná cesta z u do v a z v do u “.*

- Silně souvislé komponenty někdy zkracujeme SCCs (z anglického *strongly connected components*)
- Je-li graf tvořený jednou komponentou souvislosti, pak mluvíme o silně souvislém grafu

KOSARAJIHO ALGORITMUS

Jak najít silně souvislé komponenty?

- DFS nalezne všechny vrcholy dostupné z nějakého vrcholu
- V komponentě, která funguje podobně jako *sink vertex* toto funguje
- Potřebujeme tedy pustit DFS ze „správného“ vrcholu

Kosarajiho dvouprůchodový algoritmus

- Algoritmus spustí dvakrát DFS
- V prvním průchodu si seřadí topologicky representanty silně souvislých komponent
- Tomuto metagrafu tvořenému komponentami se říká *kondenzace grafu*
- V druhém průchodu jsou komponenty jedna po druhé odhalovány pomocí DFS
- Čas běhu v $\mathcal{O}(|V| + |E|)$

function STRONGLY-CONNECTED-COMPONENTS(*graph*) **returns** strongly connected components

of the *graph*

reversed $\leftarrow$ *graph* with all arcs reversed

DFS-LOOP(*reversed*)

DFS-LOOP(*graph*)

return components represented by leader vertex

end function

Require: all nodes of the graph are labelled from 1 to n

function DFS-LOOP(*graph*)

time = 0

start = null

for $i = n$ **to** 1 **do**

if UNVISITED(i) **then**

start $\leftarrow i$

 DEPTH-FIRST-SEARCH(*graph*, i)

end if

end for

end function

function DEPTH-FIRST-SEARCH(*graph*, i)

MARK-VISITED(i)

SET-LEADER(i , *start*)

for all edges (i, j) **do**

if UNVISITED(j) **then**

 DEPTH-FIRST-SEARCH(*graph*, j)

end if

end for

time $\leftarrow$ *time* + 1

$f(i) = \text{time}$

end function

Důkaz

TARJANŮV ALGORITMUS

Lze najít komponenty na jeden průchod?

- Označujeme komponenty podle času DFS návštěvy
- **low-link** je nejmenší index nějakého vrcholu, kam se dá dostat pomocí DFS
- Vrcholy se stejným low-linkem patří do stejné komponenty silné souvislosti

- Low-link nesmíme počítat do topologicky následujících komponent
- Z tohoto důvodu držíme zásobník otevřených uzlů, které využíváme k updatu lowlinku
- Z platných uzlů na zásobníku se pak odebere komponenta (uzly tím uzavřeme)
- Poté, co navštívíme všechny sousedy, je-li low-link stejný jak index vrcholu, pak vrchol začíná na zásobníku komponentu

- Při DFS návštěvě nějakého již navštíveného souseda
- Při návratu z DFS rekurze - jako minimum hodnoty a hodnoty potomka

Tarjanův algoritmus (zdroj: Wikipedia)

```
algorithm tarjan(input: graph G = (V, E)):
    index := 0; S := empty stack
    for each v in V do: if v.index is undefined then: strongconnect(v)

function strongconnect(v)
    v.index := index; v.lowlink := index; index := index + 1
    S.push(v); v.onStack := true

    for each (v, w) in E do
        if w.index is undefined then
            // Successor w has not yet been visited; recurse on it
            strongconnect(w)
            v.lowlink := min(v.lowlink, w.lowlink)
        else if w.onStack then
            // Successor w is in stack S and hence in the current SCC
            // If w is not on stack, then (v, w) points to an SCC already found and must be ignored
            // Note: The next line is correct; w.lowlink is correct too
            v.lowlink := min(v.lowlink, w.index)

    // If v is a root node, pop the stack and generate an SCC
    if v.lowlink = v.index then
        start a new strongly connected component
        repeat
            w := S.pop(); w.onStack := false
            add w to current strongly connected component
        while w != v output the current strongly connected component
```

- heavily inspired by Tim Roughgarden's online courses,
<http://theory.stanford.edu/~tim/videos.html>
- Robert Sedgewick and Kevin Wayne, Algorithms,
<http://algs4.cs.princeton.edu/home/>, namely
<http://algs4.cs.princeton.edu/44sp/>
- Halim, S., Halim, F., Skiena, S. S., & Revilla, M. A. (2013).
Competitive Programming 3. Lulu Independent Publish.
- Tarjanův algoritmus je z wikipedie. https://en.wikipedia.org/wiki/Tarjan%27s_strongly_connected_components_algorithm
- <https://www.youtube.com/watch?v=wUgWX0nc4NY>

DĚKUJI ZA POZORNOST.
ČAS NA OTÁZKY!