Metodický list: Prohledávání grafu do hloubky (DFS)

Pro studenty učitelství informatiky

1 Pseudokódy algoritmu

Cílem algoritmu DFS (depth first search, prohledávání do hloubky) je v tomto textu spočítat počet vrcholů v komponentě souvislosti, která obsahuje počáteční vrchol. Uvádíme dvě varianty algoritmu DFS.

1.1 Varianta A: Rekurzivní řešení

Rekurzivní algoritmus volá sám sebe na podúlohy. Řešení pro úlohu pak sestaví z výsledků pro podúlohy.

Algoritmus 1 DFS (Rekurzivní varianta)

Vstup: Graf G, Počáteční vrchol v, Množina $Navstivene$
Výstup: Počet vrcholů v komponentě

```
1: funkce DFS_Rekurzivne(v, Navstivene)
2:     Přidej v do Navstivene
3:     velikost ← 1                                  ▷ Zapocitame aktualni vrchol
4:     pro každé soused s vrcholu v v grafu G dělej
5:         pokud s ∉ Navstivene pak
6:             velikost ← velikost+ DFS_Rekurzivne(s, Navstivene)
7:         konec pokud
8:     konec pro
9:     vrať velikost
10: konec funkce
```

1.2 Varianta B: Použití zásobníku (LIFO – last in first out)

Tato varianta explicitně ukazuje práci s pamětí (LIFO struktura) a odlišuje DFS od BFS (které používá frontu).

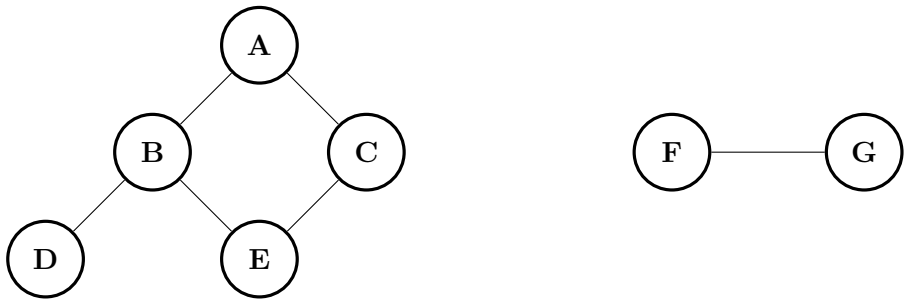
Algoritmus 2 DFS (Varianta se zásobníkem)

Vstup: Graf G, Počáteční vrchol v
Výstup: Počet vrcholů v komponentě

```
1:  funkce DFS_LIFO(v, G)
2:      Vytvoř prázdný Zasobnik
3:      Vlož v do Zasobnik
4:      Vytvoř prázdný Navstivene
5:      Vlož v do Navstivene
6:      pocet_vrcholu ← 0
7:      dokud Zasobnik není prázdný dělej
8:          aktualni ← VyberVrchol(Zasobnik)
9:          pocet_vrcholu ← pocet_vrcholu + 1
10:         pro každé soused s vrcholu aktualni v grafu G dělej
11:             pokud s ∉ Navstivene pak
12:                 Přidej s do Navstivene
13:                 Vlož s do Zasobnik
14:             konec pokud
15:         konec pro
16:     konec dokud
17:     vrať pocet_vrcholu
18: konec funkce
```

2 Příklad pro krokování

Následující graf obsahuje cyklus (čtyřúhelník A-B-E-C) i izolovanou komponentu (F-G). Slouží k demonstraci toho, že algoritmus se nezacyklí a nenavštíví nesouvislé části.



3 Tabulka krokování (Varianta se zásobníkem)

Předpoklad: Do zásobníku vkládáme sousedy tak, aby abecedně dřívější soused byl na vrcholu zásobníku (byl zpracován jako první).
Počáteční vrchol: Vrchol **A**.

Krok	**Zásobník (po akci)**	**Vybráno**	**Akce / Poznámka**	**Velikost** ($pocet\_vrcholu$)
Start	$[A]$	–	Inicializace, $Navstivene = \{A\}$	0
1.	$[C, B]$	**A**	Sousedé B, C (noví). Vloženo B, C.	1
2.	$[C, E, D]$	**B**	Sousedé A (starý), D, E (noví).	2
3.	$[C, E]$	**D**	Soused B (starý). Zásobník se zmenšil.	3
4.	$[C]$	**E**	Sousedé B, C (oba v $Navstivene$).	4
5.	[]	**C**	Zpracován vrchol čekající od kroku 1.	5
6.	[]	–	Zásobník prázdný. Konec.	**5**

4 Otázky k zamyšlení pro studenty

1. **Pořadí vkládání:** Jak by se změnilo pořadí prohledávaných vrcholů, kdybychom v kroku 1 vložili do zásobníku nejprve B a až poté C (takže C by bylo na vrchu)? Ovlivnilo by to výslednou velikost komponenty?

2. **Význam množiny** $Navstiveni$**:** Co by se stalo, kdybychom vynechali v algoritmu řádek 12?

3. **Izolované vrcholy:** Proč se v tabulce nikdy neobjevily vrcholy F a G? Jak bychom museli upravit algoritmus, abychom prohledali *celý* graf (všechny komponenty)?