

Domácí úkol k cvičení číslo 4

22. března 2024

1 Příklady ke složitosti

1.1 Rozhodněte a zdůvodněte, jestli platí následující tvrzení:

1. Pro funkci $f(n) = 8n^2 + 7n + 6$ platí:

$$f(n) \in \Omega(n^2), \quad (1)$$

$$f(n) \in \Theta(n^2). \quad (2)$$

2. Pro funkci $g(n) = 4n \log(8n) + n + 10$ platí:

$$g(n) \in O(n^2), \quad (3)$$

$$g(n) \in \Theta(n \log n). \quad (4)$$

3. Pro funkci $h(n) = 3n^2 + 10^4n + \pi$ platí:

$$h(n) \in O(n^3), \quad (5)$$

$$h(n) \in \Theta(n^3). \quad (6)$$

Za dostatečné zdůvodnění se považuje výpočet odpovídající limity nebo ukázka toho, že lze nalézt c_1, c_2 a n_0 v definicích Ω, Θ, O , viz přednáška a doporučená literatura. Dostatečné zdůvodnění by mělo obsahovat komentář v přirozeném jazyce, alespoň na úrovni: „Tvrzení platí/neplatí protože:...”

1.2 Určete a zdůvodněte jaká je složitost daného algoritmu

Algorithm 1: What does this do?

```
//Input:  $n \times (n + 1)$  matrix  $A[0 \dots n - 1; 0 \dots n]$  of real numbers
for  $i = 0, \dots, n - 2$  do
    for  $j = i + 1, \dots, n - 1$  do
        for  $k = i, \dots, n$  do
             $A[j, k] = A[j, k] - A[i, k] * A[j, i] / A[i, i]$ 
        end
    end
end
return  $A$ 
```

Předpokládejme, že cena všech operací násobení, sčítání, odečítání a dělení je stejná, a že je nezávislá na velikosti čísel. Dále se nebudeme trápit dělením nulou. Zápis **for** cyklu v algoritmu je myšlený tak, že se začíná v dolní mezi a jde se do horní meze včetně.

Jak by šlo algoritmus snadno zefektivnit?

1.3 Vyřešte rekurentní rovnici

Najděte funkci T takovou, že $T(0) = 1$ pro všechna $n \in \mathbb{N}$ (všechna kladná celá čísla) platí

$$T(n) = 2 + T(n - 1). \quad (7)$$

2 Úloha k naprogramování

Finálním cílem je vyrobit si šikovnou reprezentaci grafu. Budeme používat obvyklou definici grafu jako dvojice množiny vrcholů a hran a budeme chtít mít možnost sestavit jeho incidenceční matici a později i matici sousednosti vrcholů a hran.

2.1 Vstup

Graf dostaneme zadaný textovým souborem, kde každý řádek bude ve formátu
vrchol: seznam sousedních vrcholů oddělený čárkou a mezerou.
Vrcholy jsou označené čísly, ne nutně po sobě jdoucími. Například:

```
1: 5, 6
5: 6
```

je graf o třech vrcholech $\{1, 5, 6\}$ a třech hranách $\{(1, 5), (5, 6), (1, 6)\}$ a lze nakreslit jako trojúhelník.

Pokud bude v textovém souboru nějaká hrana vícekrát, do grafu ji přidáme jen jednou. Tj. následující soubor popisuje stejný graf

```
1: 5, 6
5: 1, 6
6: 1, 5
```

Povolíme si i hrany, které začínají a končí ve stejném vrcholu, tj. $1:1$ je ok. Stejně tak akceptujeme i vrcholy, které nejsou v žádné hraně, tj. $1:$ je ok.

2.2 Implementace grafu

Pro implementaci grafu použijte `std::unordered_map`. Je to lepší přístup než skrze pole/vektor, protože je rychlejší a umožňuje přístup a zápis typu `value_[] = mapa[key]`. Pro graf se hodí `unordered_map<int, vector<int>>`, kde v prvním argumentu budou vrcholy a ve druhém bude seznam/vektor jeho sousedů.

Až budete mít hotové parsování souboru a konstrukci grafu, implementujte funkci, která vytvoří matici sousednosti grafu.

Minimalistická kostra programu je na následující stránce, nenechte se omezovat a rozšiřte podle potřeby jak struktury tak funkce.

```

1 struct Graph {
2     unordered_map<int, vector<int>> adjacency_list;
3     int no_of_vertices;
4     int no_of_edges;
5 };
6
7
8 void printGraph(const Graph& graph) {
9     std::cout << "Graph:\n";
10    for (const auto& [vertex, neighbours] : graph.adjacency_list) {
11        std::cout << vertex << ": ";
12        for( int neighbour : neighbours) {
13            std::cout << neighbour << " ";
14        }
15        std::cout << "\n";
16    }
17    std::cout << std::endl;
18 }
19
20
21 Graph readGraphFromFile(const string& filename) {
22     Graph graph;
23     // parsing goes here
24     return graph;
25 }
26
27
28 vector<vector<int>> createAdjacencyMatrix(const Graph& graph) {
29     int n = graph.no_of_vertices;
30     vector<vector<int>> matrix(n, vector<int>(n, 0));
31     // matrix construction goes here
32     return matrix;
33 }
34
35 void printMatrix(const vector<vector<int>>& matrix) {
36     for (auto& line: matrix) {
37         for(int val: line) {
38             std::cout << val << " ";
39         }
40         std::cout << "\n";
41     }
42 }

```