Dokumentace k zápočtovému programu – hledání maximálního toku

## Anotace

Úkolem tohoto zápočtového programu je naprogramovat 2 algoritmy na hledání maximálního toku v grafu, tento program popsat a udělat pro něj dokumentaci

Obsah

[Anotace 1](#_Toc95171743)

[Vysvětlení pojmů 2](#_Toc95171744)

[Popis Edmonds-Karpova algoritmu 2](#_Toc95171745)

[Ford-Fulkersonův algoritmus 2](#_Toc95171746)

[Edmonds-Karpův algoritmus 2](#_Toc95171747)

[Popis Goldbergova algoritmu 2](#_Toc95171748)

[Implementace uživatelského rozhraní 3](#_Toc95171749)

[Implementace Edmonds-Karpova algoritmu 3](#_Toc95171750)

[Vstup 3](#_Toc95171751)

[Výstup 3](#_Toc95171752)

[Řídící funkce 3](#_Toc95171753)

[BFS 4](#_Toc95171754)

[Implementace Goldbergova algoritmu 4](#_Toc95171755)

[Vstup 4](#_Toc95171756)

[Výstup 4](#_Toc95171757)

[Řídící funkce 4](#_Toc95171758)

[Funkce na odstranění přebytku 4](#_Toc95171759)

[Funkce na zvedání vrcholu 4](#_Toc95171760)

[Pro uživatele – vstupní data 4](#_Toc95171761)

[Pro uživatele – výstupní data 5](#_Toc95171762)

[Co nebylo doděláno 5](#_Toc95171763)

## Vysvětlení pojmů

Zdroj = vrchol v grafu, ze kterého bude vše proudit. Můžeme si představit jako začátek potrubí, zdroj umí vyslat neomezeně toku.

Stok = vrchol v grafu, do kterého chceme dostat co nejvíce toku přes celý graf, stok tedy může dostávat tok z více hran k němu připojeným.

Kapacita hrany = způsob, jak určit, kolik je každá hrana schopna pojmout toku. Nejčastěji se udává jako číslo.

Zbývající kapacita hrany = určuje kolik toku lze přes danou hranu ještě pustit. Řekněme, že hranou s kapacitou 25 prochází 10 toku, zbývající tok je tedy 15.

## Popis Edmonds-Karpova algoritmu

K pochopení, jak funguje Edmonds-Karpův algoritmus je potřeba nejdříve vědět, jak funguje Ford-Fulkersonův algoritmus. Oba algoritmy jsou skoro identické, avšak Edmonds-Karpův algoritmus je časově efektivnější, protože používá rychlejší hledání cest mezi 2 vrcholy.

### Ford-Fulkersonův algoritmus

Algoritmus začíná s nulovým tokem. Základní myšlenkou je postupně přidávat další zlepšující cesty ze zdroje do stoku pomocí kterých lze při aktuálním toku protlačit grafem další nenulový tok a tím zvýšit celkový tok co dorazí do stoku. Každou takovou cestu v grafu od zdroje do stoku, co najdeme, využijeme na její maximum, tedy na nejnižší zbývající kapacitu hrany, která je na této cestě. Cesty ze zdroje do stoku se hledají pomocí prohledávání do hloubky.

Běh algoritmu končí, když už neexistuje žádná cesta ze zdroje do stoku, co by zlepšila tok.

Asymptotická časová složitost tohoto algoritmu je , kde je počet vrcholů, je počet hran a je maximum z kapacit hran ve vstupním grafu.

Úplný popis algoritmu je v Průvodci labyrintem algoritmů ([zde](https://knihy.nic.cz/files/edice/pruvodce_labyrintem_algoritmu.pdf)) na straně 321.

### Edmonds-Karpův algoritmus

Cílem E-K algoritmu je zmenšit počet vyhledávaných zmenšujících cest z F-F algoritmu. Hlavní myšlenka je přidání libovolné *nejkratší* cesty v grafu. Algoritmus používá prohledávání do šířky k nalezení těchto cest.

Každá iterace trvá , počet iterací je , časová složitost je tedy , důkaz a detailní vysvětlení dostupné [zde](https://brilliant.org/wiki/edmonds-karp-algorithm/). Můžeme si všimnout, že algoritmus nyní nezáleží na maximální kapacitě hran v grafu, což je výrazné zlepšení.

## Popis Goldbergova algoritmu

Celý algoritmus jsem nalezl popsaný [zde](https://mj.ucw.cz/vyuka/1112/ads2/4-goldberg.pdf).  
Základní myšlenkou je vytvoření tzv. vlny a následně pomocí 2 operací přemístění přebytku a zvedání vrcholu budeme měnit tok na hranách vlny v rámci grafu. Výstupem bude finální tok, což je speciální případ vlny.

Během algoritmu udržujeme pro každou hranu současný tok, její volnou kapacitu a pro každý vrchol jeho výšku a rozdíl přitékajícího a odtékajícího toku. Během přemisťování přebytku se vlna chová stejně jako vlna vody v přírodě, přebytek se přelévá pouze na vrcholy s nižší výškou.

Naplníme všechny hrany vedoucí ze zdroje. Během běhu algoritmu zvyšujeme výšky vrcholů a přebytky se šíří jako vlna mezi vrcholy, dokud nedosáhnou stoku. Pokračujeme dále, zvyšujeme výšku vnitřních vrcholů grafu, dokud se přebytky, které se z jakéhokoliv důvodu nedostaly do stoku nedostanou zpátky do zdroje.

Časová složitost algoritmu je O(|V|\*\*2 \* |E|), důkaz je uvedený v odkazu, který jsem přiložil výše.

Tento algoritmus se dá ještě zrychlit na časovou složitost O(|V| \*\* 3), což jsem původně plánoval, jelikož by byl zásadně rychlejší než Edmonds-Karpův algoritmus, avšak se mi nepodařilo tento vylepšený algoritmus implementovat. Popis této rychlejší implementace je také uvedený v odkazu nahoře.

## Implementace uživatelského rozhraní

Do nekonečna vypisuje „menu,“ tedy možnosti, které celý program má. Nejdříve se volí algoritmus, který chceme použít. Pokud se vloží neplatná hodnota, tak nás program varuje a vyzve opakovat volbu.

Následuje načtení a parsování dat ze vstupního souboru, který se musí jmenovat graf\_toku.txt . Udělá se 2D pole jako reprezentace kapacit hran grafu a slovník se sousedy pro každý vrchol.

Program se nás nyní zeptá, jaký je zdrojový vrchol a jaký vrchol je stok. Opět se kontroluje správnost zadání hodnoty, nemůžeme zadat text nebo číslo vrcholu, které by nemělo smysl – zdroj a stok stejné vrcholy a zdroj nebo stok jako neexistující vrcholy v našem grafu.

Následně se zavolá odpovídající funkce, předají se jí její vstupní hodnoty, ta nám vrátí maximální tok, který my vypíšeme a začínáme znova volbou algoritmu.

## Implementace Edmonds-Karpova algoritmu

### Vstup

Číslo vrcholu zdroje a číslo vrcholu stoku

Graf reprezentovaný 2D polem o velikosti N\*N, kde N je počet vrcholů, kde je na indexu *i*, *j* uloženo, jaká je kapacita hrany z vrcholu *i* do vrcholu *j*. Pokud taková hrana neexistuje, označíme ji 0. Nejsou povoleny hrany s negativní kapacitou.

Slovník, kde klíč je číslo vrcholu a hodnota je pole čísel vrcholů jeho sousedů. Pokud vrchol nemá souseda, pole je prázdné.

### Výstup

Hodnota maximálního toku jako kladné přirozené číslo. Pokud je hodnota 0, mezi zdrojem a stokem neexistuje cesta a nemůže tedy existovat tok.

### Řídící funkce

Funkce, která řídí chod algoritmu. Vytvoří 2D pole, kde si pro indexy *i, j* ukládá využitou kapacitu hrany mezi vrcholem *i* a *j*.

Opakovaně volá funkci BFS na zjištění nejkratší cesty ze zdroje do stoku a pokud nějaká existuje, spočítá maximální tok může přes tuto cestu projít. Zjistí minimum ze zbývající kapacity každé hrany na cestě a každé hraně na této cestě zvedne používanou kapacitu o zjištěný tok.

### BFS

Prohledávání do šířky v grafu na nalezení cesty. Na vstupu jsou čísla vrcholů zdroje a stoku, na výstupu pole dvojic (vrchol, soused) reprezentující samotnou cestu od zdroje přes vrcholy až ke stoku:  
[(zdroj, V1), (V1,V2), … , (Vx, stok)]. Pokud taková cesta neexistuje, vrací None.

Při hledání cesty je důležité, že nemůžu jít po hraně, která má již plnou kapacitu a po hraně, která má na svém konci vrchol, který jsem již navštívil, což ochrana před cyklením.

## Implementace Goldbergova algoritmu

### Vstup

Stejný jako u Edmonds-Karpova algoritmu.

### Výstup

Stejný jako u Edmonds-Karpova algoritmu.

### Řídící funkce

Funkce, která řídí chod algoritmu, vytvoří 2D pole o velikosti N\*N, kde N je počet vrcholů, použité kapacity, pole o velikosti N na uložení výšek jednotlivých vrcholů, pole o velikosti N na uložení přebytku pro každý vrchol a další 2 pomocná pole o velikostech každý N.

Algoritmus pošle počáteční vlnu a začne procházet seznam vrcholů, odstraní přebytek současného vrcholu pomocí funkce na odstranění přebytku. Pokud se výška vrcholu změnila, tak přesune současný vrchol na začátek seznamu vrcholů a začne znovu prohledávat seznam od začátku.

### Funkce na odstranění přebytku

Funkce, která pro vrchol, který má kladný přebytek zkusí najít souseda, ke kterému lze přemístit aspoň část přebytku. Musí najít souseda s nižší výškou a nenulovou volnou kapacitou. Jestli takový soused neexistuje, tak se zavolá funkce na zvednutí tohoto vrcholu a začíná s hledáním souseda na přemístění části přebytku znova.

### Funkce na zvedání vrcholu

Funkce, která na vstupu dostane vrchol a pro každého jeho souseda se pokusí zvednout výšku o minimum z výšek vrcholu a souseda, zvýšenou o 1.

## Pro uživatele – vstupní data

Formát vstupních dat je následující:  
Název souboru musí být graf\_toku.txt. V tomto souboru na řádku *i* jsou kapacity hran z vrcholu do všech ostatních vrcholů od 0 do , kde N je počet vrcholů. Pokud hrana existuje, je uvedena její kapacita. Pokud hrana neexistuje, kapacita je 0. Hrana z vrcholu do vrcholu je také 0.

Zadané kapacity musí být přirozené číslo, hrany jsou orientované a můžou existovat cykly.

Chart

Description automatically generated with medium confidenceTakto vypadá soubor: Takto by se graf nakreslil:

A screenshot of a computer

Description automatically generated with low confidence

Při běhu programu se bude program opakovaně ptát na algoritmus, který chceme pustit – stačí do konzole ve které program běží napsat číslo algoritmu z vypsané nabídky, poté na číslo počátečního a koncového vrcholu. Pokud se někde stane logická chyba, tak to program zjistí a zeptá se na správné hodnoty znovu.

## Pro uživatele – výstupní data

V konzoli se vypíše maximální tok ze zadaného počátečního vrcholu do koncového.

## Co nebylo doděláno

Jak jsem předem avizoval, neimplementoval jsem rychlejší verzi Goldbergova algoritmu. Nezbývalo mi moc času na dodělání programu po konci zkoušek. Možná bych si to do budoucna zkusil implementovat.