**Algoritmy a dátové štruktúry II (IV003)** **SADA PROBLÉMOV 3**

**Autor**: Michal Lukáč, 430614

**Spoluriešiteľ**: -

**Úloha 1**

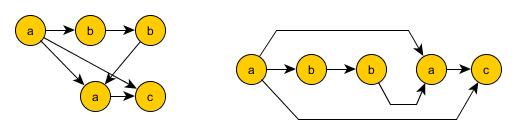
Nech G je orientovaný acyklický graf v ktorom každý vrchol je označený symbolom z konečnej abecedy. Rôzne vrcholy môžu byť označené takým istým symbolom. Každá orientovaná cesta v G má stopu tvorenú zreťazením symbolov označujúcich vrcholy cesty. Sekvenciou grafu G je podpostĺpnosť stopy niektorej orientovanej cesty v G. Navrhnite efektivný algoritmus pre výpočet najdlhšej spoločnej sekvenciou dvoch daných orientovaných acyklyckých grafov.

Riešenie úlohy je skryté priamo v obrázku zadania úlohy. Využijeme algoritmu dfs a dynamického programovania.

Budeme postupovať následovne najprv si vytvoríme topologické usporiadanie daných grafov. Topologické usporiadanie grafu G je lineárne usporiadanie uzlov takých, že pokiaľ G obsahuje hranu (u,v) potom u je v usporiadaní před v. Na topologické usporiadanie sa môžeme pozerať ako na usporiadanie uzlov na horizontálnej čiare, na ktorej hrany všetkých uzlov smerujú zľava doprava. V tomto topologickom usporiadaní voláme práve DFS, čo je spôsob prehľadávanie grafu do hĺbky. Postupne pridávame dokončené uzly v prehľadávaní do lineárneho zoznamu. Pseudokód pre algoritmus nájdenia topologického usporiadania:

1. Topological-Sort(G):
2. DFS(G)
3. Postupne pre každý uzol dokončený dfs pridávaj do zoznamu
4. Vráť zoznam

Zložitosť DSF je O(V +E) a O(1) zložitosť pre pridania položky do zoznamu. Zložitosť celková je teda O(V+E) kde V sú uzly a E sú hrany grafu. Existuje viacero topologických usporiadaní grafu, v našom prípade berieme topologické usporiadanie podľa dokončenia prehľadávania uzlov pomocou DFS. V graf G obsahuje hranu z uzlu u do v potom v.f < u.f teda u je topologickom usporiadaní pred v, keďže časová známka že prehľadávanie bolo dokončené je väčšia pre u. Preto pre každú hranu u do v orientovanom acyklickom grafe je v.f < u.f..



Pokiaľ chceme nájsť najdlhšiu spoločnú sekvenciu dvoch daných orientovaných acyklických grafov najprv potrebujeme dostať pomocou Topological-Sort usporiadanie g1 a g2. Následne využijeme dynamického progamovanie pre nájdenie najdlhšej spoločnej sekvencie. Tento problem môže byť vyriešený v čase O(|x||y|) kde x,y sú reťazce využítím následného vzťahu rekurentného zápisu:

Kde Cij udáva dĺžku najdlhšej spoločnej subsekvencie. Výsledný pseudokód pre nájdenie spoločnej sekvencie dvoch grafov:

1. ***SequenceGraph****(A,B,C):*
2. *g1top = Topological-Sort(G1)*
3. *g2top = Topological-Sort(G2)*
4. *n = počet vrcholov G1, m = počet vrcholov G2*
5. *nech C je je matica = n \* m*
6. *dynamicProgrammingSequence(ref C, g1top, g2top)*
7. *return max z C*

Avšak samotná rekurentná rovnica uvedená vyššie postačuje pre jednoduché reťazce a nie pre topologické usporiadanie grafu reprezentujúci nejaký reťazec. Preto si musíme odvodiť nový rekurentný vzťah pre dynamícké programovanie. Nech v1,i reprezentuje i-tý uzol v topologickom usporiadaní grafu G1. Nech L1(v1,i) reprezentuje funkciu ktorá priraďuje každému uzlu nejaké písmeno z abecedy ktorá je použitá v grafu G1. Nech Cij označuje dĺžku najdlhšej sekvencie L1(P(v1,i). Potom Cij môžeme vypočítať ako:

Rozoberme si najprv prvú rovnicu pokiaľ teda pokiaľ máme dva uzly, ktoré obsahujú rovnaké písmeno, môžu nastať dva prípady. Prvým prípadom je keď do daného uzlu v1,i alebo v2,j nevedú žiadne hrany. Potom vieme, že najdlhšia sekvencia takéhoto problému je 1, preto je v danej rovnici zjednotenie s 0. Inak vyberieme maximálnu hodnotu predchádzajúcej sekvencie ktorá vedie do nášho uzlu.

V druhej rovnici počítáme s tým kedy , teda kedy znaky v dvoch uzloch nie sú rovnaké. V tomto prípade pokiaľ do nich nevedie žiadny uzol tak vieme, že veľkosť sekvencie je 0.

Algoritmus sa teda skladá z dvoch zavolaní pre nájdenie topologického usporiadania a následným vyriešením problem najdlhšej sekvencie pomocou dynamického programovania. Nájdenie topologického usporiadania je O(|E|+|V|). A keďže zoradenie uzlov je v lineárnom čase a riešenie sekvencie je O(|E1||E2|) celková časová náročnosť je O(|E1||E2|), pričom E1 reprezentujú hrany grafu G1 a E2 grafu G2. Priestorová náročnosť je daná tabulkou pre zapamatanie si výsledkov medzi jednotlivými uzlami o veľkosti |V1||V2|, pričom V reprezentujú vrcholy grafov.

**Algoritmy a dátové štruktúry II (IV003)** **SADA PROBLÉMOV 3**

**Autor**: Michal Lukáč, 430614

**Spoluriešiteľ**: -

**Úloha 2**

Pre ľubovolný reťazec x a prirodzené číslo k, nech označuje reťazec vzniklý zreťazením k kopií reťazca x. Repticí reťazca x je ľubovolný prefix Spojením dvoch reťazcov x a y je reťazec vzniklý preložením repetíciou reťazca x a y. Navrhnite efektivný algoritmus, ktorý na vstup dostane tri reťazce x, y, z a ktorý rozhodne zda z je spojením x a y.

Úlohu môžeme vyriešiť zadefinovaním si dátovej štruktúry, ktorá bude obsahovať znaky ktoré môžeme použiť pri aktuálnej kontrole. Takouto dátovou štruktúrou môže byť pole prípadne zoznam.

Vezmime si napríklad x = slovo a y =word. Pri analýze reťazca SLWOOVOSRDLWORO postupujeme tak že do našej dátovej štruktúry si uložíme prvé písmena reťazca x a y teda [s, w] pretože reťazec môže začať jedine na tieto písmená. Skontroluje sa teda či prvé písmeno je jedno z dvoch písmen a pokiaľ áno dané písmeno sa nahradí v našej štruktúre ďaľším z reťazca x a to l a dostáváme [l, w]. Ďruhým znakom je ‘l‘, pričom ‚l‘ sa nachádza v [l, w], takže zmažeme uložené ‚l‘ a pridáme znak ‘o‘ keďže to je tretie písmeno v reťazci x. V ďaľšom znaku vyberieme prvé písmeno z reťazca y a to ‚w‘. V našej štruktúre teraz bude [o,o]. V ďaľšom kroku vyberiem iba jedno o a pridáme ďaľší znak z oboch reťazcov, keďže sme sa mohli zanoriť do jedného z nich a dostáváme [o’,v,r]. Ďaľším znakom je opäť o ale keďže sme si zvyšné o označili ako o’ tak toto písmenko z nášho zoznamu iba vyškrtnem a pokračujem v prechádzaní reťazca.

Ak by bol však ďaľším znakom namiesto o napríklad ‚v‘, tak potom vieme, že predchádzajúci znak o pochádza z reťazca x a pridáme teda písmeno ‚o‘ z reťazca x. Naopak ak by bol ďaľším znakom r tak potom vieme že predchádzajúce o pochádza z y a ‘o’ ktoré nám ostalo v datovej štruktúre je z reťazca x.

Pseudokód je nasledujúci:

1. ***IsSpojenim****(X,Y,Z):*
2. *Inicializuj zoznam s prvými znakmi z X, Y.*
3. *Pokiaľ neprejdem celý reťazec:*
4. *Kontroluj znaky v reťazci Z podľa nášho zoznamu:*
5. *Pokiaľ znak nesedí podľa postupnosti x,y vráť neúspech*
6. *Inak odober znak podľa X alebo Y a pridaj ďaľší znak*

Aby algoritmus fungoval korektne musíme zaistit, že pokiaľ prečítame posledné písmeno z reťazca x alebo y tak potom pridávame do zoznamu znaky opäť odzačiatku. Pokiaľ kontrolujeme znak ktorý môže byť v danej chvíli buď X alebo Y nastáva špeciálny prípad, ktorý sa musí ošetriť tak ako to bolo ozrejmené vyššie.

Časová zložitosť je lineárna voči veľkosti reťazca Z.

**Algoritmy a dátové štruktúry II (IV003)** **SADA PROBLÉMOV 3**

**Autor**: Michal Lukáč, 430614

**Spoluriešiteľ**: -

**Úloha 3**

Nech G = (V, E) je orientovaný graf s celočíselným ohodnocením hran. Predpokladajme, že graf neobsahuje cyklus zápornej dĺžky. Úlohou je navrhnúť algoritmus, ktorý vypočíta dĺžku najkratšej cesty medzi každou dvojicou vrcholov. Výstupom algoritmu bude dvojrozmerné pole dist[1…V,1….V], kde dist[i,j] je dĺžka najkratšej cesty z i do j.

**Algoritmy a dátové štruktúry II (IV003)** **SADA PROBLÉMOV 3**

**Autor**: Michal Lukáč, 430614

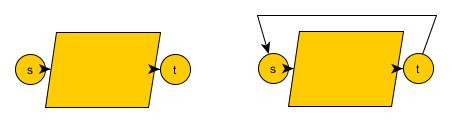
**Spoluriešiteľ**: -

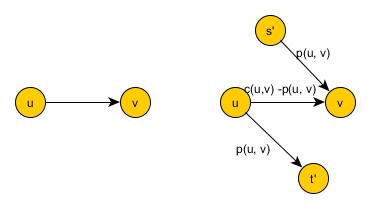
**Úloha 4**

Majme sieť v ktorej každá hana (u,v) má nezápornú poptávku p(u, v). Tok f v sieti s poptávkami je prípustný práve keď, pre každú hranu platí f(u, v) >= p(u, v). Problémom je nájsť prípustný tok s minimálnou hodnotou.

Predpokladám, že kdeže v úlohe je zadaný algoritmus MaxFlow tak pre každú hranu je definovaná okrem poptávky aj maximálna kapacita hrany.

1. Navrhnite efektívny algoritmus, ktorý nájde nejaký prípustný tok.

Môžeme transformovať probem nájdenia prípustneho toku na nájdenie prípustnej cirkulácie. Tú můžeme vyriešiť nájdením maximálního toku v sieti s novým zdrojom a cieľom. Avšak samotné nájdenie maximálního toku nestačí respektive negarantuje, že tok splní potrebnú poptávku každej hrany. Preto vytvoríme graf G‘ v ktorom každá hrana má kapacitu c(u,v)-p(u,v). Pridáme nový zdroj s’ a nový cieľ t’. Pre každú hranu (u,v) tak že vytvoříme hrany medzi (u,t’) a (s’,v) s kapacitami c(u,t’) = p(u,v), c(s’,v) = p(u,v) a kapacitou medzi u,v ako c(u,v)-p(u,v).



Hrane (t,s) priradíme kapacitu ktorá sa rovná sume všetkých kapacít hrán. Potom potrebujeme nájsť maximálny tok z s’ do t’ například ford-fulkersnovým algoritmom pre nájdenie maximálního toku. Skontrolojume čí sú všetky hrany nasýtené a pokiaľ nie sú tak neexistuje prípustný tok pre graf G’. Pokiaľ existuje taký tok tak potom stačí odstrániť hrany, ktoré sme pridali a pridať poptávku ku každej hrane. Výsledný tok je pre hranu je f(u,v) = f‘(u,v) + p(u,v).

Pre rekapituláciu uvádzam zjednodušený pseudokód:

***PrípustnýTok****:*

1. *Konvertuj graf G na G’ pridaním hrany medzi s,t a pridaním uzlov s’,t’*
2. *Prepoj vnútorné uzly grafu s’,t’ a nastav kapacity hrán*
3. *Nájdi Maximálny tok*
4. *Odstráň pridané hrany a vráť výsledný prípustný tok pokiaľ sú splnené podmienky*

Musíme splniť podmienku kedy f(u,v) >= p(u, v) ale aj podmienku kedy tok ktorý ide do uzla v teda fin(v) = fout(v). Nech f je tok taký v ktorom sú všetky poptávky splnené. Uvažujme graf v ktorom sú všetky hrany invertované takže hrana má kapacitu f(u,v) – p(u,v). Nech f’ je hodnota maximálního toku v tomto grafe. Potom je poptávka daná ako g = f – f’. Pre každú hranu teda musí platiť, že g(u,v) = (f(u,v) – f’(u,v))>= p(u,v). Pre v. Pre všetky uzly okrem počiatočného a koncového platí, že gin(v) = gout(v).

1. Predpokladajme, že máme algoritmus maxflow, ktoré pre danú sieř s kapacitami vypočíta maximálny tok. Navrhnite efektivný algoritmus, ktorý vypočíta prípustný tok s minimálnou hodnotou v sieti a využíva MaxFlow práve jeden kráť.

Využijeme postupu z riešenia a) pre nájdenie prípustného toku. Ďalej potrebujeme upraviť náš prípustný tok tak aby bol minimálny. Nech f je náš prípustný tok. Skonštruujeme opačný graf originálneho grafu. Hrany medzi uzlami u a v nahradíme dvomi hranami z u do v s kapacitou c(u,v) – f(u,v). a z v do u s kapacitou f(u,v)- p(u,v). Tieto hrany skombinujeme a dostáváme kapacitu c(u,v)-f(u,v)+f(u,v)-p(u,v). Následne využijeme algoritmu MaxFlow pre spočítanie maximálního toku g medzi s,t v grafu G’. Výsledný tok pričítame do grafu G a dostávame že tok medzi uzlami u,v je f’(u,v) = f(u,v)+g’(u,v)-h(u,v). Tok f’ ktorý týmto dostáváme je minimálny tok.