ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta aplikovaných věd

Fyzikální praktikum 2

Ohyb dráhy elektronu v magnetickém poli

Autor: **Antonín NEUMANN**
Akademický rok: **2012/2013**

Zadání

Jakožto velký expert na všechny druhy algoritmů a řešení obtížných problémů (firma Ferda Mravenec, práce všeho druhu, s.r.o.) jste byli pověřeni firmou vlastnící počítačovou síť kontrolou optimality informačních toků po síti. Počítače jsou již zapojeny do sítě, je známa cena přenosu v každém komunikačním kanálu (může jít o kapacitu, délku, cenu spojení...). Síť zahrnuje až 1000 uzlů, ceny jsou kladná čísla. Propojení v nejhorším případě může tvořit úplný graf. Zajímá nás minimalizace ceny při posílání informace – musíte najít, které kanály z existujících se mají používat a které naopak doporučujete nepoužívat.

Váš úkol založený na tomto příběhu je následující: Implementujte algoritmus založený na greedy strategii a na tom, co víte o řešení problému MST, určete, jestli jde o přesné nebo přibližné řešení, určete jeho složitost. Není požadována žádná vizualizace sítě, stačí prosté načtení vstupu z text. souboru ve tvaru: nV nE (tj. 1. řádek udává počet uzlů a počet hran) ei ej wij .... (tj. n řádek se zadáním hran a jejich ohodnocením – předpokládá se, že ohodnocení je stejné v obou směrech) Výstup na obrazovku nebo opět do souboru (výstupem je počet hran doporučených k používání a jejich výčet v nějakém přijatelném formátu). Získáte 6 bodů. Pokud vaše řešení bude nejvýše O(E log E), kde E je počet kanálů, získáte další 4 body.

Nemusíte se vázat na MST a greedy, můžete vymyslet řešení zcela jiné. Opět je třeba uvést a zdůvodnit, jestli jde o přesné nebo přibližné řešení. Ostatní pokyny a hodnocení je stejné jako ve variantě 1.

Řešení

Greedy algoritmus, který je na tento příklad vhodné použít je Kruskalův algoritmus. Tento algoritmus prochází postupně všechny hrany, tyto hrany jsou seřazeny vzestupně podle jejich ceny. Zjistí jestli vložením hrany nevznikne v kostře kružnice, pokud ne hranu vloží, pokud by vznikla hranu zahodí a jde na další v pořadí.

Tento algoritmus potřebuje ke svému chodu strukturu disjoint-set, která je založená na metodě Union & Find.

Tento problém je velmi dobře znám a existuje spoustu algoritmů, které jej řeší. A většina z nich je již implementována do většiny používaných programovacích jazyků. Takže po chvilce hledání je možné nalézt řešení, které s drobnými úpravami bude fungovat přesně pro náš problém.

Časová složitost

Časová složitost tohoto řešení je O(N*logN)

Popis Union & Find algoritmu

Máme pole vrcholů (Node). Každý vrchol v sobě nese informace o svém ID a ID svého předka (rodiče). V této struktuře udržujeme informace o tom, do které komponenty daný uzel patří.

Metoda union

Provede sjednocení podgrafů (komponent) jejichž jsou vrcholy A a B součástí.

Metoda find

Nalezne a vrátí reprezentanta (kořen) pro zadaný prvek. Pokud je reprezentant prvku A vzdálen o hloubku více než jedna, provede opravu (kompresi) této cesty aby příští hledání bylo rychlejší.

Metoda repair

Opraví cestu z uzlu A tak, že všechny uzly na cestě z A ke kořenu, přidá přímo pod kořen čímž výrazně urychlí další proběhnutí metody union.

Popis programu

Program je dodán jako JAR soubor. Při jeho spouštění je možné jako parametr uvést jméno vstupního souboru (jeden vstupní soubor je též součástí odevzdání). Pokud nebude jméno vstupního souboru uvedeno jako parametr, program si jeho název sám vyžádá. Výstup probíhá jak na obrazovku, tak i do souboru s názvem vystup.txt (tento název je neměnný).

Zdroje

http://www.algoritmy.net/article/1417/Kruskaluv-algoritmus
http://www.fi.muni.cz/~hlineny/Vyuka/GT/Grafy-lect--5.pdf
http://teorie-grafu.cz/vybrane-problemy/minimalni-kostra.php
http://www.algoritmy.net/article/1422/Disjoint-set
http://cs.wikipedia.org/wiki/Kruskal%C5%AFv_algoritmus
http://en.wikipedia.org/wiki/Disjoint-set_data_structure