Domáca úloha 7 - cvičenie 1 - Internet

Zadanie

Firma má ve městě N budov, které by ráda připojila k internetu. Spojení dvou budov A, B optickým kabelem stojí c(A, B). Najděte algoritmus, který pro dané c navrhne, jak propojit budovy a které připojit k internetu přímo, aby cena připojení byla co nejnižší. Domy propojené optickým kabelem platí cenu připojení jako jeden dům (platí nejnižší cenu připojení za celou komponentu souvislosti). Pokud víte, že:

(a) poskytovatel internetu si účtuje fixní poplatek C za připojení jednoho domu

(b) poskytovatel internetu si účtuje připojení podle lokality, tedy cena připojení budovy A je C(A)

Rozbor

Na to, aby sme vytvorili čo najlacnejšiu sieť nám stačí z grafu, kde vrcholy sú domy spolu s poskytovateľom a hrany sú cena prepojenia medzi budovami, vytvoriť minimálnu kostru grafu. Avšak nemôžeme všetky domy prepojiť a potom ich za cenu *1*C* prepojiť s poskytovateľom. Namiesto toho budeme vytvárať komponenty súvislosti. Vyberieme si ľubovoľný vrchol z množiny domov a budeme k nemu postupne pripájať domy dovtedy, kým nebude cena spojenia domov vyššia, ako keby sme ho pripojili priamo za cenu *C*. Následne postupujeme rovnako s ostatnými vrcholmi bez priradenej komponenty a nakoniec jednotlivé disjunktné komponenty prepojíme s poskytovateľom, čím vytvoríme najmenšiu kostru grafu, vyhodnocovanú podľa daných kritérií.

V druhom prípade bude vytváranie komponent súvislostí o trochu pozmenené, keďže bude cena pripojenia k poskytovateľovi rôzna. Tento problém môžeme riešiť tak, že ako začiatočné body komponent si budeme postupne voliť nepriradené domy od najlacnejších lokalít k najdrahším. Takto zabezpečíme, že všetky voľné domy budú priradené k najlacnejším dostupným komponentám a pokiaľ by c(A, B) stálo viac, ako pripojenie B priamo k poskytovateľovi, tento dom už nepridáme k danej komponente. Na záver, znovu, komponenty pripojíme k poskytovateľovi.

Riešenie

V tomto algoritme využijeme upravený Boruvkov algoritmus, keďže budeme postupne pridávať hrany menšie, ako C a vzniknuté komponenty prepojíme s poskytovateľom.

make a list L of n trees, each a single vertex
while (L has more than one tree and at least 1 edge has c(e) < C):
    for each T in L, find the smallest edge connecting T to G-T
    add all those edges to corresponding trees
for each tree T in L:
    pick a vertex v from T and connect it with vertex Provider

V druhom postupe je myšlienka podobná. To, čo potrebujeme, je spraviť čo najmenej komponent, ktoré budú zároveň minimálnymi kostrami vrámci vrcholov danej komponenty. Poskytovateľ je ďalší vrchol grafu, ktorý má rovnako ohodnotené hrany - podľa lokality domu a hrana vedie od poskytovateľa ku každému domu. Keď budeme vytvárať kostru, tak pri tom, či spojíme dva stromy bude rozhodovať jediná podmienka - či cena hrany c(e) je nižšia, ako keby sme najlacnejší dom komponenty pripojili priamo k poskytovateľovi. To je totižto jediný prípad, kedy sa nám môže stať, že by spájanie komponent nebolo najlacnejšie. Ak by táto hrana stála menej, ako pripojenie komponenty, tak nie je problém - namiesto pripojenia 2 komponent stačí zaplatiť pripojenie jednej. Pre každý takýto strom si nám teda stačí pamätať ktorý je najlacnejší dom a C(A) tohoto domu.

make a list L of n trees, each a single vertex
while (there are 2 trees T1, T2 with c(e) between them such that, **c(e) < C(min(T1)) or c(e) < C(min(T2))**) :
    newTree = connect T1 and T2 with edge e
    min(newTree) := min(min(T1), min(T2))
for each tree T in L:
    pick a min(T) vertex and connect it with vertex Provider

Takto, pomocou pozmeneného Boruvkovho algoritmu si najprv nájdeme komponenty tak, aby mali čo najnižšiu cenu pripojenia komponenty a následne ich za najnižšiu možnú cenu pripojíme k poskytovateľovi.

Časová zložitosť

Keďže používame len zľahka zmenený Boruvkov algoritmus, časová zložitosť bude rovnaká, ako pri normálnej verzii algoritmu - **O(m logn)**.

Priestorová zložitosť

O(n) - v pamäti držíme len údaje o tom, ktorý vrchol patrí do ktorej komponenty, v druhom prípade navyše ešte pre každý strom vrchol, ktorý je najlacnejšie pripojiť k poskytovateľovi, ale $O(2n) \in$ **O(n)**.