3. domácí úkol | Vilém Zouhar

1

1.1 Popis

Pro každý list platí, že je vždy výhodnější vložit policajta do jeho rodiče než do samotného listu (rodič tak pokryje předchozí hranu + nějaké navíc). Tedy (skoro) nechceme dávat policajta do listů. Ale když ho tam nedáme, tak už nutně musíme dát policajta do rodiče. Tedy vezmeme graf G, odstraníme z něj všechny listy, vznikne graf G', ve kterém do listů už policajty musíme dát. Utrhneme všechny listy a vytvoříme graf G'', ve kterém jsou všechny listy hlídané, tedy rovnou jdeme k G'''', kde opět do listů musíme vložit policajty, atd..

Problém nastává, že nemáme v pěkné složitosti po ruce seznam aktuálních listů. Tedy skoro. Můžeme použít topologické očíslování grafu (algoritmus rozebraný na přednášce), ze kterého dostaneme takový pěkný seznam listů. Tento seznam projdeme a rozmístíme policajty dle popisu.

1.2 Korektnost

Rozebráno v popisu. Jdeme od konce, kde jsme si jistí správností umístění.

1.3 Pseudokód

```
f = fronta z topologickeho ocislovani (zacina listy)
while f.not_empty():
        p = f.pop()
        // duplikatni
        if p.policeman or p.is_leaf or (for all c in p.children: not c.policeman):
                p.policeman = true
        else:
                p.parent.policeman = true
```

Jednoduše jdeme hladově, ale ve správném pořadí.

1.4 Složitost

Topologické očíslování grafu umíme v lineárním čase, pak jen projdeme podle počtu vrcholů, tedy *O(n+m).*

1.5 Rozšíření

Co kdyby vrcholy byly oceněné? Tedy na některé by bylo dražší umístit policajta. Na to by už nefungoval náš přístup, ale museli bychom se uchýlit k rekurzi. U kořene k bychom zavolali *k.policeman = k.price + minimalize(k.left, true) + minimalize(k.right, true) > minimalize(k.left, false) + minimalize(k.right, false);* Museli bychom si však ohlídat, abychom neutekli do exponenciály, tedy za pomocí dynamického programování.

2

2.1 Popis

Cesta mezi dvěma nejvzdálenějšími vrchol musí nutně putovat přes kořen (viz. korektnost). Pustíme tedy BFS na kořen, tím nalezneme jeden krajní bod (nejvzdálenější od kořene), který označme A. Z A pak pustíme BFS, čímž najdeme druhý krajní vrchol B. AB jsou pak nejvzdálenější vrcholy, tedy |AB| je průměr.

2.2 Korektnost

Co to je vlastně kořen? To by mohl být jakýkoliv vrchol a orientace nás v tomto případě stejně nezajímá. Opravdu si stačí vybrat jakýkoliv vrchol a z něj pustit BFS. Pokud máme jeden krajní bod a pustíme z něj BFS, pak samozřejmě najdeme dva nejvzdálenější body, ale proč BFS z libovolného vrcholu najde jeden krajní bod? Náznak důkazu:

Víme, že dva vrcholy od sebe nejvíce vzdálené jsou oba listy (pokud ne, můžeme prodloužit, spor). Tedy cesta musí jít z A do nejvyššího společného předka A a B a z něj pak do B. Pro všechny možné dvojice A a B platí, že A nebo B je v nejnižší hladině nějak zakořeněného stromu (protože se jedná o průměr). Tedy pokud si ten strom nějak zakořeníme a provedeme BFS z kořene, dostaneme jeden krajní bod.

2.3 Pseudokód

```
BFS(G.random_node, G)
A = node discovered last in BFS
BFS(A, G)
B = node disovered last in BFS
output(|AB|) // taky posledni hladina v poslednim BFS
```

2.4 Složitost

Děláme pouze dva BFS, tedy potřebujeme *O(n+m)* času.