KSI 2012/2013
Úloha 3-5: Vysokohorský problém

Jan Horáček
Gymnázium, Brno, Vídeňská 47; jan.horacek@seznam.cz

29. prosince 2012

1 Úvod

Opět typická grafová úloha. Stačilo nakreslit si pár grafů a řešení bylo na světě.

2 Řešení

V tomto řešení je využíváno pseudokódu. Tento pseudokód je založen na jazyce C.

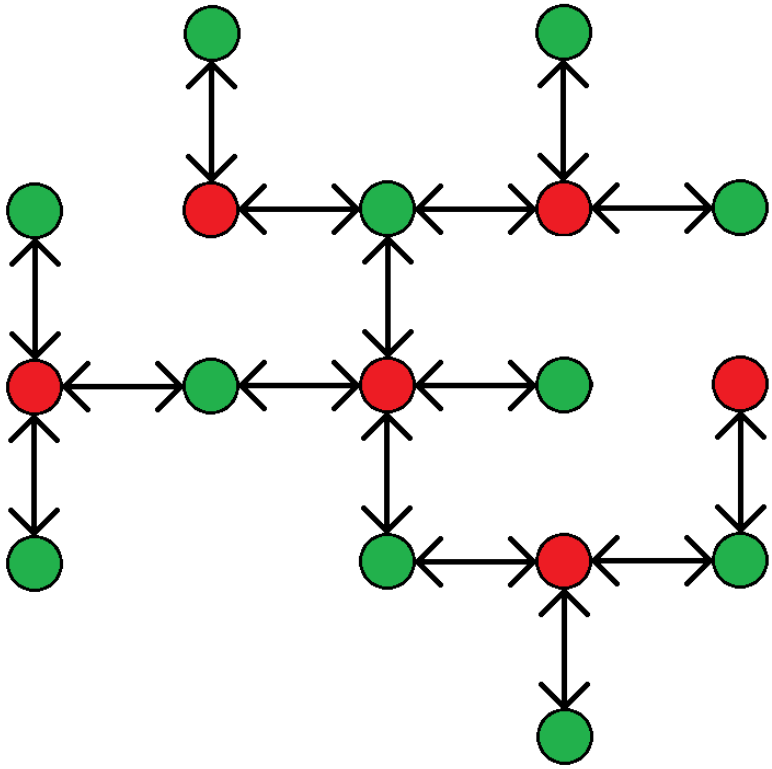
2.1 Teoretické řešení problému

Jde nám o to, aby na žádných dvou sousedních vrcholech nevznikly hotely. Proto definujme pro každý vrchol proměnnou *bool a*. Bohužel je tato proměnná značně abstraktní a tak jsem pro ni nevymyslel lepší název.

Jde nám o to, abychom vrchol v označili jako $true$ a všechny sousední vrcholy s vrcholem v (označme je w) jako $false$. Analogicky, všechny sousední vrcholy s vrcholem w označíme jako $true$. Tedy pro každý sousední vrchol invertujeme tuto hodnotu proměnné a. Ve výsledném algoritmu je jedno, jestli začneme hodnotou $true$, nebo $false$. Pro ilustraci přikládám obrázek 1.

Z toho plyne, že existují pouze 2 možnosti, jak rozestavět hotely. Buď hotely postavíme na těch vrcholech, kde proměnná a nabývá hodnoty $true$, nebo tam, kde proměnná a nabývá hodnoty $false$.

Tedy spočítáme, jestli je více $true$, nebo $false$ vrcholů a podle toho, kterých je více, tak na takovýcho vrcholech necháme vystavět hotely. Ve výsledném algoritmu bude tento krok propojen s deklarací stavu proměnné a pro každý vrchol a výsledná časová náročnost tedy poměrně malá.



Obrázek 1: stav proměnné a u každého vrcholu

2.2 Praktické řešení problému

V praxi budeme potřebovat strom a u každého vrcholu proměnnou a a proměnnou $visited$.

Tento strom projdeme mírně upraveným prohledáváním do šířky, které zajistí definici stavu proměnné a u každého vrcholu:

Pozn.: předpokládejme, že pole lze indexovat proměnnou typu *bool* tak, že $false = 0$ a $true = 1$.

```
int cnt[2] = {0, 0};

void Sirka(vrchol v)
{
    bool now = true;

    Queue.PUSH(v);
    while (Queue.count > 0)
    {
        v = Queue.POP();
        v.visited = true;
        v.a = now;
        cnt[now]++;
        for (int i = 0;i<v.childs_cnt;i++)
            if (!v.childs[i].visited)
                Queue.PUSH(v.childs[i]);
        now != now;
    }//while
}
```

Listing 1: Upravené prohledávání do šířky

Pak provedeme porovnání proměnných $cnt[false]$ a $cnt[true]$. Podle toto, která má větší hodnotu, postavíme hotely na daných vrcholech:

1. cnt[false] $>$ cnt[true] $\rightarrow$ postavit hotely na $a = false$
2. cnt[false] $<$ cnt[true] $\rightarrow$ postavit hotely na $a = true$
3. cnt[false] $=$ cnt[true] $\rightarrow$ je jedno, kde postavíme hotely

Firma tedy ví, kde postavit hotely a naše úloha je vyřešena.

3 Závěr

Zpočátku jsem nečekal, že existují pouze 2 možnosti, jak rozestavět hotely podle zadání bez ohledu na jejich počet.

[1] Časová složitost algoritmu je $O(|V|+|E|)$, kde V je množina vrcholů a E je množina hran grafu.

Reference

[1] *Prohledávání do šířky*
http://en.wikipedia.org/wiki/Breadth-first_search