ZADÁNÍ SEMESTRÁLNÍ PRÁCE
ŘEŠENÍ KOLIZÍ FREKVENCÍ SÍTĚ VYSÍLAČŮ

Zadání

Naprogramujte v ANSI C přenositelnou[1] **konzolovou aplikaci**, která jako vstup načte z parametru příkazové řádky název textového souboru obsahující informaci o pozici vysílačů na mapě a na jeho základě přidělí každému vysílači frekvenci tak, aby jeho signál nekolidoval s vysílači v blízkém okolí. Úloha je znázorněna obrázku 1.

Program se bude spouštět příkazem `freq.exe` <*soubor-s-vysílači*>. Symbol <*soubor-s-vysílači*> zastupuje jméno textového souboru, který obsahuje informaci o rozmístění vysílačů na mapě a o dostupných vysílacích frekvencích, které jim je možné přidělit.

Váš program tedy může být během testování spuštěn například takto:

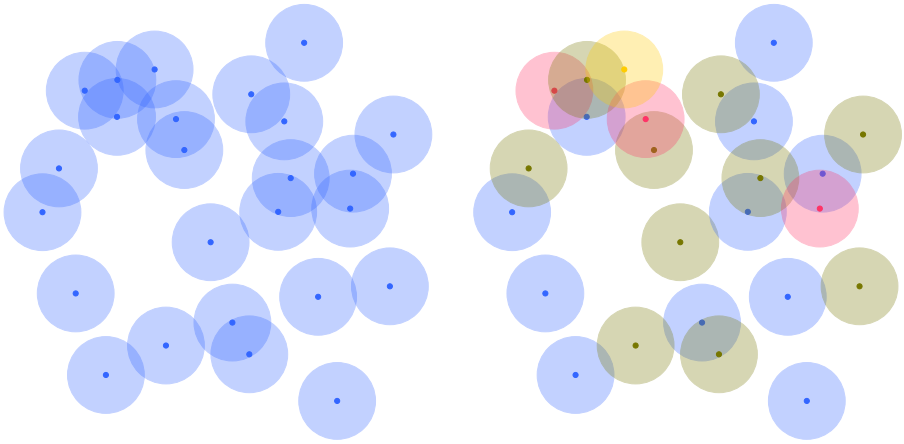
```
freq.exe vysilace-25.txt
```

Výsledkem práce programu bude výpis do konzole se seznamem přidělených frekvencí každému vysílači ze vstupního souboru (viz Specifikace výstupu programu). V případě chyby nebo neřešitelné situaci skončí program výpisem příslušné chybové hlášky.

Pokud nebude na příkazové řádce uveden právě jeden argument, vypište chybové hlášení a stručný návod k použití programu v angličtině podle běžných zvyklostí (viz např. ukázková semestrální práce na webu předmětu Programování v jazyce C). **Vstupem programu je pouze argument na příkazové řádce – interakce s uživatelem pomocí klávesnice či myši v průběhu práce programu se neočekává.**

Hotovou práci odevzdejte v jediném archivu typu ZIP prostřednictvím automatického odevzdávacího a validačního systému. Postupujte podle instrukcí uvedených na webu předmětu. Archiv nechť obsahuje všechny zdrojové soubory potřebné k přeložení programu, **makefile** pro Windows i Linux (pro překlad v Linuxu připravte soubor pojmenovaný `makefile` a pro Windows `makefile.win`) a dokumentaci ve formátu PDF vytvořenou v typografickém systému TEX, resp. LATEX. Bude-li některá z částí chybět, kontrolní skript Vaši práci odmítne.

[1] Je třeba, aby bylo možné Váš program přeložit a spustit na PC s operačním prostředím Win32/64 (tj. operační systémy Microsoft Windows NT/2000/XP/Vista/7/8/10) a s běžnými distribucemi Linuxu (např. Ubuntu, Mint, OpenSUSE, Debian, atp.). Server, na který budete Vaši práci odevzdávat a který ji otestuje, má nainstalovaný operační systém Debian GNU/Linux 7.8 Wheezy s jádrem verze 3.2.51-1 x86_64 a s překladačem gcc 4.7.2.



Obrázek 1: Znázornění úlohy. Na mapě je dána množina vysílačů, jejichž pozice je znázorněna tečkou. Vysílač dokáže vyslat svůj signál pouze v okruhu do předem definované vzdálenosti. Ve skutečnosti spolu ale signály sousedních vysílačů mohou kolidovat (situace vlevo). Každému vysílači je proto potřeba přiřadit jinou frekvenci (barvu) tak, aby signál žádné dvojice vysílačů nekolidoval (vpravo).

Specifikace vstupu programu

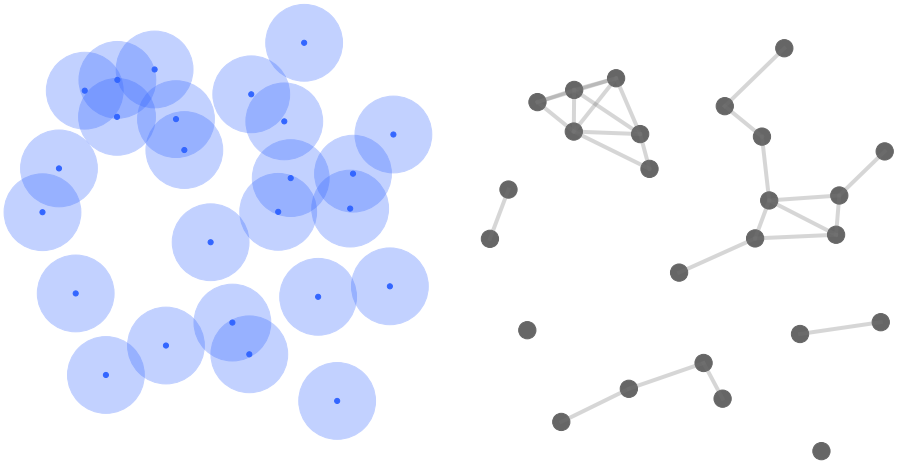
Vstupem programu je **pouze** parametr na příkazové řádce: soubor s informacemi o vysílačích a frekvencích, které má program přidělovat. Tento soubor je textový a má následující formát:

`Available frequencies:`
ID_{f_0} f_0
ID_{f_1} f_1
ID_{f_2} f_2
...
$ID_{f_{N_f-1}}$ f_{N_f-1}
`Transimission radius:`
R
`Transmitters:`
ID_{t_0} X_{t_0} Y_{t_0}
ID_{t_1} X_{t_1} Y_{t_1}
ID_{t_2} X_{t_2} Y_{t_2}
...
ID_{t_N-1} X_{t_N-1} Y_{t_N-1}

Kde ID_{f_i} je číselný identifikátor i-té vysílací frekvence a f_i je příslušná vysílací frekvence v Hertzích. N_f je celkový počet dostupných vysílacích frekvencí a N je celkový počet vysílačů. Konstanta R udává poloměr kruhu v kilometrech, v němž má vysílač pokrytí signálu a kde může docházet ke kolizím.

V sekci `Transmitters` se nachází identifikátory a pozice jednotlivých vysílačů. ID_{t_i} je identifikační číslo i-tého vysílače a X_{t_i} a Y_{t_i} jsou jeho příslušné souřadnice $[X, Y]$ na mapě. Tyto souřadnice jsou stejně jako R udávány v kilometrech.

Poznámka: Předpokládejte, že veškerá identifikační čísla jsou vždy číslována vzestupně od nuly a jsou ve vstupním souboru seřazena.



Obrázek 2: Převod informací o vysílačích na graf kolizí. Pokud se signály (kruhy) překrývají, může dojít ke kolizi, a proto jsou tyto vysílače (vrcholy) v grafu spojeny hranou.

Graf kolizí

K vyřešení úlohy je třeba sestrojit graf kolizí, který bude mít tyto vlastnosti:

- Každý vysílač je v grafu reprezentován unikátním vrcholem.
- Vrcholy (vysílače) propojuje hrana pouze tehdy, pokud mezi nimi může docházet ke kolizi signálu. (Jejich vysílací oblasti se překrývají.)
- Každý vrchol nese informaci o přiřazené frekvenci. Tato informace je vyjádřena jako celé číslo, které se v teorii grafů označuje jako *barva* vrcholu. Nechť barva vrcholu odpovídá identifikačním číslům frekvencí ID_{f_i} ve vstupním souboru.

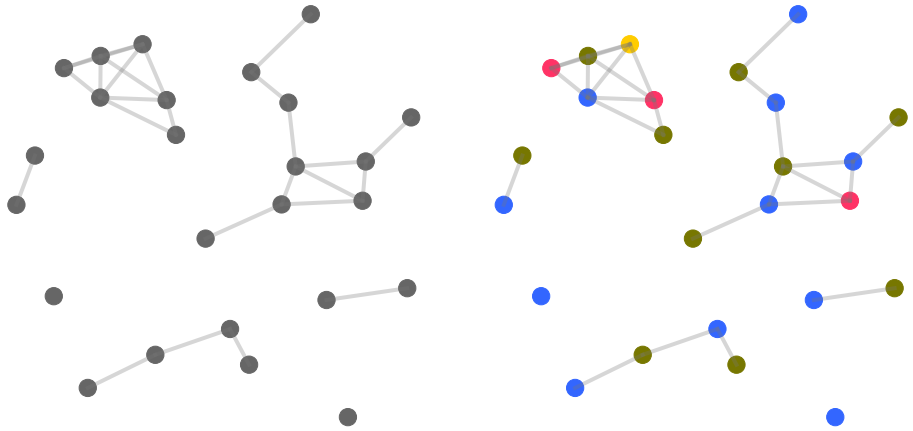
Na obrázku 2 je znázorněn převod informací o vysílačích na graf kolizí dle zmíněných pravidel. Jako podmínku kolize berte situaci, kdy je Euklidovská vzdálenost mezi dvěma vysílači menší než $2R$ (dvojnásobek poloměru kruhu, ve kterém vysílač propaguje signál).

Obarvování grafu

Úlohu přidělování frekvencí vysílačů lze řešit metodou teorie grafů, která se nazvývá *obarvování grafu*. Obarvování grafu je postup, při kterém každému vrcholu přiřadíme celé číslo (barvu) tak, aby každá dvojice vrcholů spojená hranou měla rozdílnou barvu. Zároveň požadujeme, aby k tomuto obarvení bylo použito co nejméně barev (frekvencí vysílačů). V tomto případě bude obarvován graf kolizí. Rozdílná barva v tomto případě znamená rozdílná vysílací frekvence, a je tedy zřejmé, že pokud správně obarvíme kolizní graf, ke kolizi signálů nemůže dojít. Správně obarvený graf je znázorněn na obrázku 3.

Úloha korektního obarvení grafu s použitím nejmenšího počtu barev je ve skutečnosti *NP-úplná*, z čehož vyplývá, že dosud není znám algoritmus, který by problém dokázal optimálně vyřešit s polynomiální časovou složitostí. Existuje ale algoritmus, který problém řeší v mnoha případech optimálně nebo je jeho řešení blízké optimálnímu řešení.

Postup tohoto algoritmu je následující:



Obrázek 3: Neobarvený graf kolizí (vlevo) a jeho správně obarvená varianta (vpravo). V obarveném grafu žádná dvojice vrcholů spojená hranou nemá shodnou barvu (a tedy i příslušné vysílače mají různou frekvenci a nekolidují spolu.)

1. Vytvoř graf kolizí.
2. Nastav všechny vrcholy grafu jako dosud neobarvené.
3. Procházej všechny vrcholy grafu a vlož do zásobníku první neobarvený vrchol.
4. Není-li zásobník prázdný, vyjmi vrchol ze zásobníku.
5. Vybranému vrcholu postupně přiřazuj barvy $0, 1, ..., N_f - 1$, dokud není nalezena barva, která nekoliduje se sousedními vrcholy (těmi, do kterých z aktuálního vrcholu vede hrana). Přiřaď aktuálnímu vrcholu nejnižší možnou barvu, která tuto podmínku splňuje a vlož do zásobníku všechny dosud neobarvené sousední vrcholy aktuálně zpracovaného vrcholu. Pokud je nejnižší nalezená barva větší než dostupný počet barev (frekvencí), algoritmus ukonči a označ úlohu jako neřešitelnou.
   (*Poznámka:* úloha sice řešení mít může, ale tento algoritmus jej nedokázal naleznout.)
6. Opakuj postup od bodu 4. Je-li zásobník prázdný, pokračuj na další neobarvený vrchol v grafu (bod 3).

Tento algoritmus rovněž pěkně ilustruje video na adrese:
`http://www.youtube.com/watch?v=o3B-V94VnQg`

Řešitelnost úlohy

Tato úloha **není vždy řešitelná** – existují varianty vstupu, pro které nelze graf kolizí obarvit tak, aby byl použit pouze zadaný počet frekvencí. Dojde-li k tomu, že program nebude schopen úlohu vyřešit, nechť zareaguje chybovým hlášením podle níže uvedené ukázky (s přesným dodržením formátování):

```
ERR#3: Non-existent solution!
```

Upozornění: Graf, který bude Váš program obarvovat může mít i tisíce vrcholů a hran, a je proto nutné zvolit zvolit vhodný způsob implementace vnitřní reprezentace grafu tak, aby výpočet proběhl v rozumném čase.

Specifikace výstupu programu

Výstup programu bude směrován pouze na konzoli. Výstupem (v případě, že nenastala chyba) bude rostoucí seřazená posloupnost identifikačních čísel vysílačů a jim přiřazená vysílací frekvence v Hertzích, například:

```
0 92300
1 105600
2 92300
3 86200
4 86200
```

Každá dvojice těchto hodnot je vypsána na samostatné řádce a ukončena znakem konec řádky '`\n`'. Pokud algoritmus řešení nenašel, nechť program vypíše chybu č. 3 (viz výše a níže).
Pro testovací účely jsou připojeny vstupní soubory s příslušnými vzorovými výstupy. (*Poznámka:* graf lze samozřejmě obarvit mnoha různými způsoby tak, aby byla splněna podmínka korektního obarvení, berte výstup pouze jako příklad správného řešení.)

V souboru `vysilace-25.txt` a `vysilace-25-vysledek.txt` se nachází testovací množina 25-ti vysílačů, resp. odpovídající výstup pro tento vstupní soubor.

Podobně v souborech `vysilace-1000.txt` a `vysilace-1000-vysledek.txt` se nachází vstupní soubor pro 1000 vysílačů a jemu odpovídající vzorové obarvení (tato množina je vhodná pro testování rychlosti vašeho programu).

Chybové stavy

Odhalí-li program chybu, nechť reaguje výpisem jedné z chybových hlášek uvedených v tabulce. Mějte, prosím, na paměti, že z důvodu automatické kontroly odevzdávaného díla je nezbytně nutné **přesně dodržet** formát chybových hlášek.

Chybové hlášení	Význam, popis chybového stavu
`ERR#1: Missing argument!`	Na příkazové řádce nebyl program předán parametr, specifikující vstupní soubor s informacemi o vysílačích.
`ERR#2: Out of memory!`	Není k dispozici dostatek operační paměti.
`ERR#3: Non-existent solution!`	Řešení pro daný počet vysílacích frekvencí nebylo nalezeno.

Hodnoty chyb od 4 výše můžete využít pro vlastní potřeby, ovšem dodržte uvedené formátování, tj. velkými písmeny zkratka `ERR`, následovaná bez mezer znakem '`#`' a číslem chyby – za číslem budiž pak dvojtečka a mezera a pak stručný popis chyby ukončený vykřičníkem ('`!`').

Číselný kód chyby nechť program předá také operačnímu systému prostřednictvím návratové hodnoty z funkce `int main()`. V případě, že program proběhne bez chyby, nechť je návratová hodnota programu 0.

Užitečné techniky a odkazy

Uvedené techniky je možné využít při řešení úlohy. Protože se jedná o postupy víceméně standardní, lze k nim nalézt velké množství dokumentace:

1. teorie grafů `http://cs.wikipedia.org/wiki/Teorie_graf%C5%AF`,
2. barvení grafu `http://en.wikipedia.org/wiki/Graph_coloring`,
3. zásobník.

Řešení úlohy je zcela ve vaší kompetenci – zvolte takové algoritmy a techniky, které podle vás nejlépe povedou k cíli.