FAKULTA
APLIKOVANÝCH VĚD
ZÁPADOČESKÉ
UNIVERZITY
V PLZNI

Katedra informatiky / PC

ŘEŠENÍ KOLIZÍ FREKVENCÍ SÍTĚ VYSÍLAČŮ

(Semestrální práce)

student: Milan Hajžman
studijní číslo: A13B0303P
email: qwerty2@students.zcu.cz
datum: 10. ledna 2016

Obsah

1 Zadání

Naprogramujte v ANSI C přenositelnou **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.

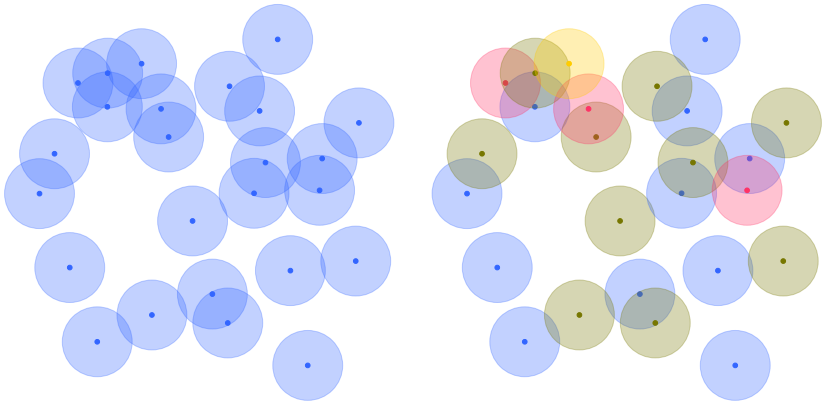
Program se bude spouštět příkazem `freq.exe` *<soubor-s-vysílači>*. Symbol *<soubor-s-vysílači>* zastupuje jméno textového souboru, který obsahuje informaci o rozmístění vysílačů na mapě a o dostupných vysílacích frekvencích, které jim je možné přidělit. Váš program tedy může být během testování spuštěn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



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).

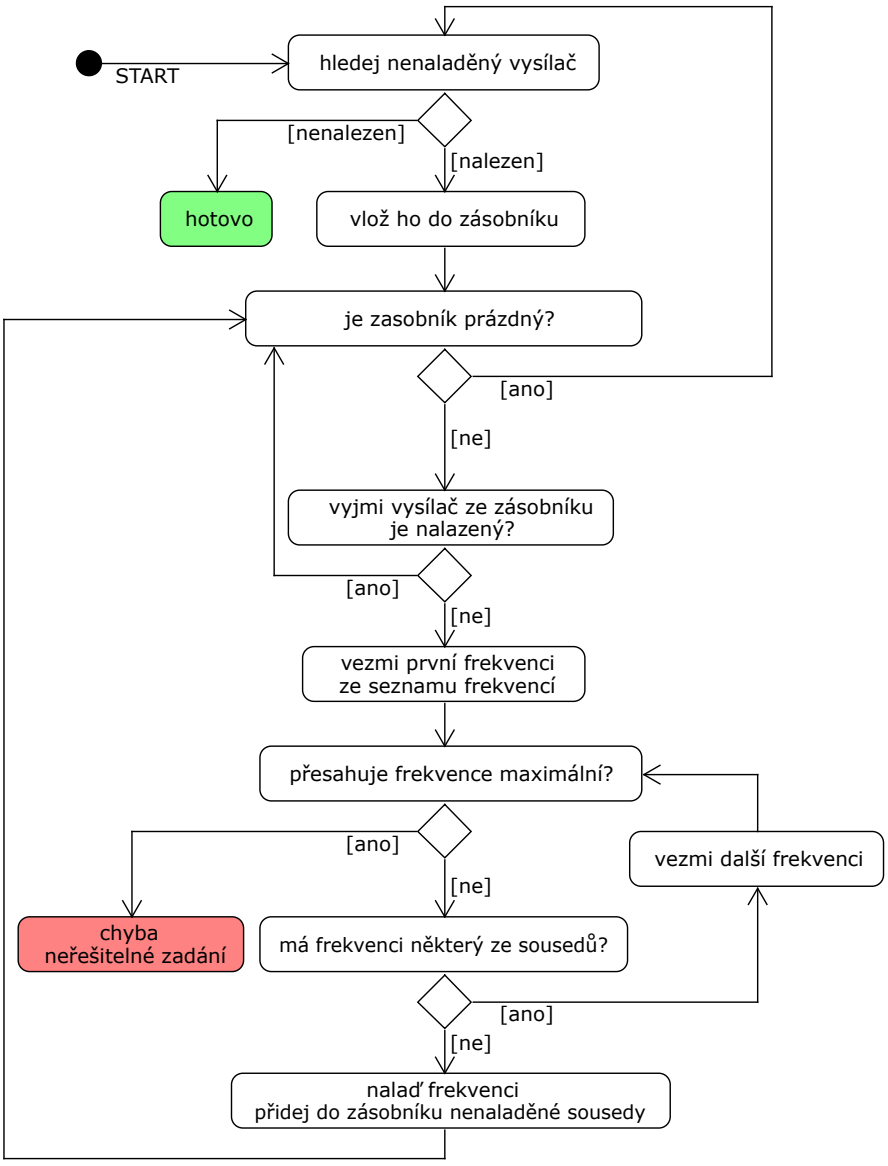
2 Analýza úlohy

K urychlení procházení sousedních vysílačů je zapotřebí vytvořit graf sousednosti. V tomto grafu každá hrana představuje, že se oblasti dvou vysílačů překrývají. Taková situace nastane, pokud je vzdálenost mezi dvěma vysílači menší než dvojnásobek jejich dosahu. Při počtu vysílačů je možno provést kontrolu stylem „každý s každým“. Po kontrole vzdáleností může grafů vzniknout více, stejně tak se mohou vyskytovat osamocené vysílače, proto je zapotřebí vícenásobně projít seznam vrcholů a zkontrolovat zda nezvbývá nenaladěný vysílač.

Počet vysílačů může sahat do tisíců, proto je vhodné graf nedefinovat maticí, ale spojovým seznamem nebo polem obsahujícím hrany grafu. Pole je však nevhodné, protože dopředu neznáme počet hran, které do nich budeme vkládat. Zároveň jeho výhodu jít kdykoli na kterýkoli prvek pole nepotřebujeme, protože budeme vždy procházet všechny sousední vrcholy.

Při procházení grafu budu potřebovat zásobník, do kterého budu ukládat nenaladěné sousední vysílače. Implementuji ho pomocí pole, které v případě potřeby realokuji na větší.

Algoritmus hledání frekvencí je podrobněji popsán UML diagramem na obrázku 2.



Obrázek 2: UML diagram algoritmu pro nalezení nejnižších frekvencí v grafu sousednosti.

3 Popis implementace

3.1 Datové struktury

3.1.1 Overlap

Abych nezapomněl nějakou část paměti uvolnit, rozhodl jsem se udržovat vše v jedné struktuře `Overlap`, která bude pointery ukazovat na další struktury, ty pak na dalsí atd. Pak lze kaskádově uvolnit všechna naalokovaná místa v paměti jedinou funkcí. `Overlap` obsahuje počet frekvencí a vysílačů, dosah vysílačů, pole frekvencí a pole s odkazy na struktury `Transmitter`.

```
typedef struct {

    /* pole s frekvencemi */
    int *freqlist;
    /* dvojity pointer ve smyslu pole ukazatelu na vysilac */
    Transmitter **translist;
    /* delky poli */
    int freqc,translistc;
    /* maximalni vzdalenost mezi dvema vysilaci
     * pri niz dochazi ke kolizi signalu */
    int radius;

} Overlap;
```

3.1.2 Transmitter

Struktura `Transmitter` představuje jeden vysílač, obsahuje jeho souřadnice, naladěnou frekvenci a první položku spojového seznamu sousedních vrcholů grafu. Tyto položky jsou strukturového typu `Node`. Nenaladěný vysílač má nastavenou frekvenci rovnající se konstantě `UNTUNED`. Pokud nemá vysílač žádného souseda, pak se odkaz na seznam rovná `NULL`.

```
#define UNTUNED -1

typedef struct {

    /* spojovy seznam sousedu */
```

```
    Node *node;
    /* pozice vysilace */
    double x,y;
    /* naladena frekvence, ve smyslu indexu v poli frekvenci */
    int freq;

} Transmitter;
```

3.1.3 Node

Struktura `Node` obsahuje odkaz na další `Node` a index vysílače v poli vysílačů ve struktuře `Overlap`. Poslední prvek spojového seznamu ukazuje na `NULL`.

```
typedef struct node {

    /* dalsi polozka */
    struct node *next;
    /* soused - index v poli vysilacu */
    int transmitter;

} Node;
```

3.1.4 Intstack

Struktura `Intstack` představuje zásobník čísel, který využívám při procházení grafu, k upřednostění souseda při dalším ladění frekvencí. Zásobník začne s zadanou velikostí při jeho vytvoření. V případě, že hrozí pčtečení, tak se zavolá funkce `realloc()`, která ho zvětší na dvojnásobnou velikost.

```
typedef struct {

    /*velikost pole*/
    int stacksize;
    /*pole polozek*/
    int *stack;
    /*vrchol zasobniku*/
    int current_item;

} Intstack;
```

3.2 Zdrojové soubory

`main.c`: Hlavní zdrojový soubor, obsahuje funkci `main()` s během programu a funkci `Overlap *load_file(char *filename)`, která načítá vstupní data ze souboru do struktury `Overlap`.

`overlap.h`: Obsahuje definici struktury `Overlap` a prototypy jejích obslužných funkcí.

`overlap.c`: Obsahuje výkonný kód funkcí ze souboru `overlap.h`.

`transmitter.h`: Obsahuje definici struktury `Transmitter` a prototypy jejích obslužných funkcí.

`transmitter.c`: Obsahuje výkonný kód funkcí ze souboru `transmitter.h`.

`node.h`: Obsahuje definici struktury `Node` a prototypy jejích obslužných funkcí.

`node.c`: Obsahuje výkonný kód funkcí ze souboru `node.h`.

`intstack.h`: Obsahuje definici struktury `Intstack` a prototypy jejích obslužných funkcí.

`intstack.c`: Obsahuje výkonný kód funkcí ze souboru `intstack.h`.

`errors.h`: Obsahuje čísla chybových stavů a jejich hlášky.

`errors.c`: Obsahuje funkci pro předčasné ukončení programu v případě, že nastane chyba programu. Tato funkce vypisuje chybové hlášky ze souboru `errors.h`.

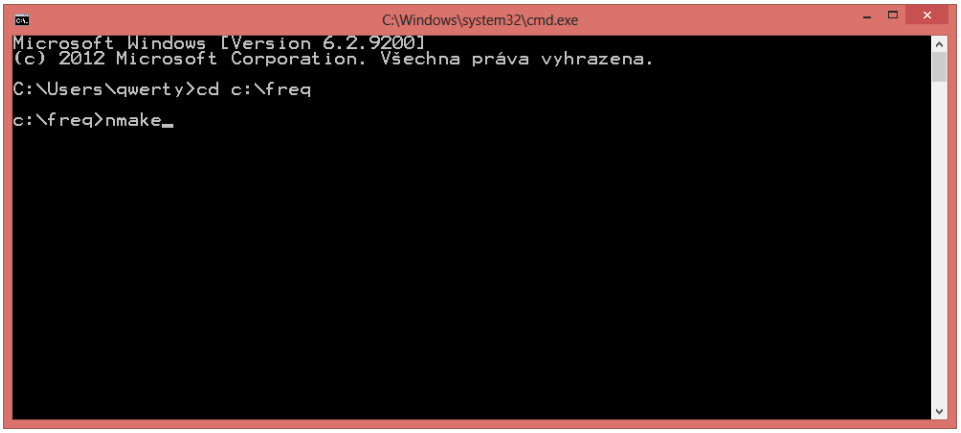
4 Uživatelská příručka

Program je dodáván v archivu ZIP. Pro použití je zapotřebí archiv rozbalit. Archiv obsahuje zdrojové soubory včetně souborů `Makefile` pro překlad na systému Linux pomocí GNU GCC a `Makefile.win` pro překlad na systému Windows pomocí Microsoft Visual Studio. Pro kompilaci je zapotřebí mít nainstalován potřebný nástroj a složku s překladačem mít v systémové proměnné *PATH*.

4.1 Překlad a spuštění

4.1.1 Windows

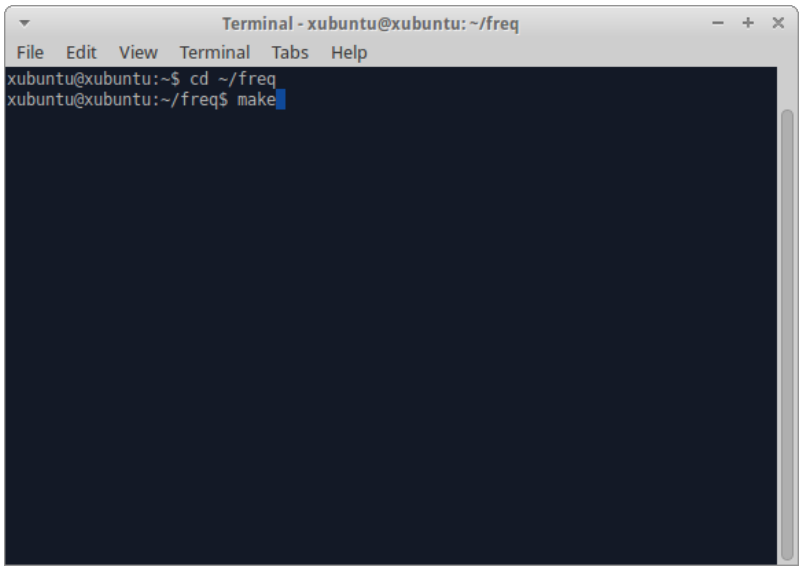
Nejdříve spustíme Příkazový řádek pomocí kláves **WIN + R**, do dialogu napíšeme `cmd` a potvrdíme tlačítkem OK. Jak vypadá příkazový řádek je možné vidět na obrázku 3. V příkazovém řádku se příkazem `cd` *<složka s rozbalenými soubory>* přeneseme do složky, kam jsme si program rozbalili. Překlad provedeme příkazem `nmake`, spuštění pak příkazem `freq.exe` *<soubor-s-vysílači>*.



Obrázek 3: Příkazový řádek v systému Windows 8.

4.1.2 Linux

Nejdříve si spustíme Terminál. Postup se liší podle distribuce Linuxu, ve většině případů lze terminál najít menu s aplikacemi na systémové liště. Jak vypadá terminál je možné vidět na obrázku 4. V Terminálu se příkazem `cd` *<složka s rozbalenými soubory>* přeneseme do složky, kam jsme si program rozbalili. Překlad provedeme příkazem `make`, spuštění pak příkazem `./freq.exe` *<soubor-s-vysílači>*.



Obrázek 4: Terminál v systému Xubuntu 14.04.

5 Závěr

O jazyce C jsem vždy smýšlel jako o atomovém výbuchu v `ASCII-artu` a nikdy jsem se nemohl v těch hvězdičkách, ampersandech a šipkách vyznat. Dříve jsem si napsal program v C využívající knihovnu `SDL`, který ze vstupu sbíral názvy zvukových souborů, které následně přehrával. Zdrojový kód vznikl spíše slepením více *examplů* z internetu, a nějak jsem se divil, proč program během svého běhu zabírá čím dál více prostoru v paměti. Až v průběhu tohoto předmětu mi došlo, jak bláhový jsem byl při psaní onoho kódu.

Během pár hodin předmětu jsem se naučil stokrát víc, než během stovek hodin samostudia, kdy jsem se snažil proniknout do tajů „Céčka“. Myslím, že už konečně rozumím pointerům, které mi dříve dělali velké problémy.

Zároveň si velmi cením toho, že jsem byl donucen seznámit se systémem LaTeX, čehož jistě v budoucnu využiji.