FAKULTA
APLIKOVANÝCH VĚD
ZÁPADOČESKÉ
UNIVERZITY
V PLZNI

ŘEŠENÍ KOLIZÍ FREKVENCÍ SÍTÍ VYSÍLAČŮ

Martin Hamet
os.č. A14B0254P

10. ledna 2016

Obsah

Kapitola 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.

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 frekvencí, 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 situace skončí program výpisem příslušné chybové hlášky.

Pokud nebude na příkazové řádce uveden právě jeden argument, vypíše 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 odevzdávejte 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 script Vaši práci odmítne.

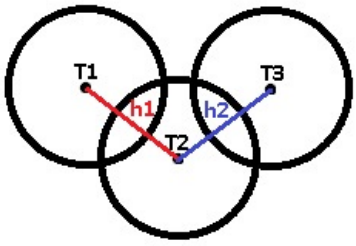
Podrobnější informace o vstupu a výstupu programu jsou uvedeny v souboru **zadani.pdf**.

Kapitola 2

Analýza úlohy

Vstupem programu budou dostupné frekvence a rozmístění vysílačů na dvourozměrné mapě. Úkolem je přiřadit vysílačům frekvence, tak aby nedocházelo ke kolizi signálů. Každý vysílač pokryje oblast danou dosahem svého signálu a k rušení signálu dochází v případě že dva, nebo více různých vysílačů se stejnou frekvencí pokrývá stejnou část území. Taková situace nesmí nastat. Volných frekvencí máme ovšem omezený počet. Budeme se tedy snažit využít co nejmenší počet z těchto dostupných frekvencí.

Nejprve je nutné zjistit, které vysílače navzájem kolidují. Vysílačům bez kolizí snadno přiřadíme první dostupnou frekvenci. Pro hledání kolizních vysílačů bude nejsnazší zjistit vzdálenost každých dvou vysílačů a pokud jejich vzdálenost bude menší než součet dosahů jejich signálů označíme je za kolizní.



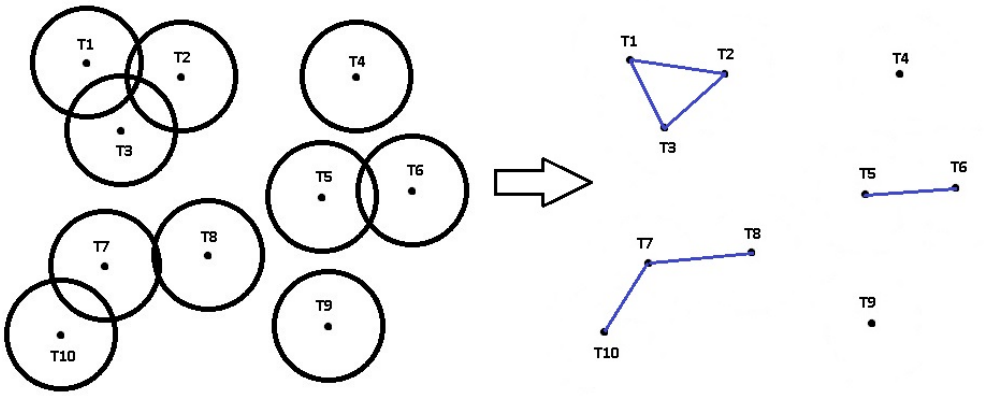
Obrázek 2.1: Kolize třech vysílačů.

Kolizním vysílačům by bylo možné přiřazovat stále vyšší frekvence, ale tím bychom použili příliš mnoho dostupných frekvencí. Například podle obr. 2.1. Vysílač T1 koliduje pouze s T2 a proto T3 může mít stejnou frekvenci jako T1. Pro usnadnění práce si vytvoříme graf kolizí, kde vrcholy grafu jsou jednotlivé vysílače a hrany mezi vysílači představují kolizi (hrany h1 a h2 v obr. 2.1).

Převedením problému do grafu získáme souvislé komponenty grafu, kde dochází ke kolizím. Stačí tedy vyřešit přidělování frekvencí v každé komponentě zvlášť.

2.1 Graf kolizí

Neorientovaný graf kolizí se skládá z vrcholů které reprezentují vysílače. Hranou jsou propojené pouze vrcholy, které jsou natolik blízko u sebe že může docházet ke kolizi signálů. Protože mají všechny vysílače stejný dosah signálu zjišťujeme pouze zda jsou vrcholy od sebe vzdálené méně než dvojnásobek dosahu signálu. Vytvoření grafu kolizí z mapy vysílačů je znázorněno na obr. 2.2.

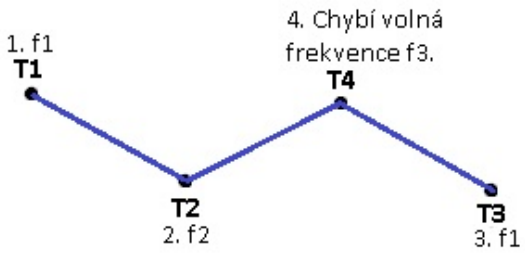


Obrázek 2.2: Vysílače a dosah jejich signálů (vlevo) a výsledný kolizní graf (vpravo).

2.2 Přiřazování frekvencí

1. Vytvoříme graf kolizí (viz výše).
2. Označíme frekvence všech vrcholů za nepřiřazené.
3. Pokud ještě existuje nějaký vrchol s nepřiřazenou frekvencí.Procházíme všechny vrcholy grafu a vybereme vrchol s nepřiřazenou frekvencí. Pokud takový vrchol neexistuje ukončíme algoritmus, protože bylo nalezeno řešení.
4. Vybranému vrcholu postupně přiřazujeme dostupné frekvence od nejnižší dokud nenalezneme takovou, že z vrcholu nevede hrana k vrcholu se stejnou frekvencí. Pokud nemáme dostatečný počet dostupných frekvencí ukončíme algoritmus, protože se nepodařilo nalézt řešení.
5. Pokračujeme od bodu 3.

Tento algoritmus funguje ovšem v některých situacích nevhodné pořadí vrcholů při jejich průchodu snadno ukončí algoritmus s nenalezením řešení. Podle obr. 2.3 kde máme pouze dvě dostupné frekvence a pořadí ve kterém procházíme vrcholy grafu je T1, T2, T3, T4.

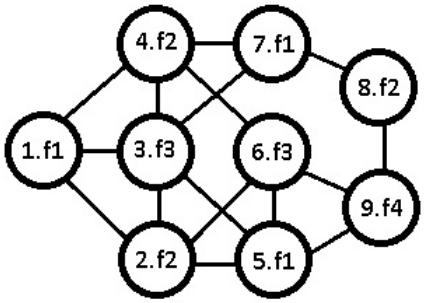


Obrázek 2.3: Přiřazování frekvencí.

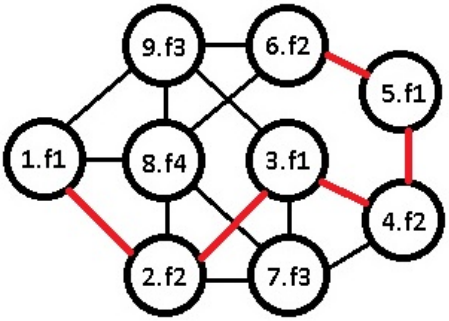
Jak je vidět na obrázku vrcholu T4 byla přiřazována frekvence jako poslední. Tohoto problému s pořadím v vrcholů, pokud jsou v jedné souvislé komponentě grafu, se můžeme částečně zbavit budeme-li graf procházet nejprve po nalezených sousedech až poté budeme hledat další vrchol s nepřiřazenou frekvencí v seznamu všech vrcholů. Takový průchod můžeme zajistit např. frontou, nebo zásobníkem do kterého budeme ukládat nalezené vrcholy. Budeme muset tedy upravit bod 4. kde je potřeba ještě projít všechny sousedy vybraného vrcholu a pokud nemají přiřazenou frekvenci přidat je do zásobníku pokud v něm nejsou.

Fronta by zajistila průchod z vybraného vrcholu do šířky (tzv. BFS). Jak je naznačeno na obr. 2.4. Počáteční vrchol byl vrchol 1. a byla mu přidělena první frekvence f1. Průchod do šířky nejprve projde všechny sousedy počátečního vrcholu, dále sousedy sousedů atd. Oproti zásobníku, který zajistí průchod do hloubky (tzv. DFS). Naznačeno na obr. 2.5.

Průchod do hloubky se nejprve snaží co nejvíce zanořit do grafu, přes ještě neobjevené vrcholy. Vznikne tedy nejprve kostra spojité komponenty, jak je naznačeno červenou čarou na obr. 2.5, na které se střídají první dvě dostupné frekvence. V obou případech BFS i DFS je výsledek závislý na pořadí uložených sousedů u vrcholu a celkově se nedá říci, který ze zmíněných způsobů bude výhodnější. V každém případě oba dokážou vždy vyřešit komponenty, nebo jejich částí, řetězového typu, jako byla komponenta na obr. 2.3. Do řešení jsem ve výsledku zvolil způsob se zásobníkem kvůli nepatrně snazší implementaci, protože oba způsoby vypadají ekvivalentní co se týče nalezení řešení.



Obrázek 2.4: BFS



Obrázek 2.5: DFS

Tento způsob by řešil problém po souvislých komponentách jak bylo řečeno výše. Ovšem není zajištěno že algoritmus vždy nalezne řešení. Při nevhodném pořadí přiřazení sousedů a nedostatku frekvencí tento způsob nemusí fungovat, i když řešení existuje. Tento problém se nedá s jistotou vyřešit. S vyšším počtem dostupných frekvencí a s menším počtem kolizních vysílačů roste šance úspěšného nalezení řešení.

2.3 Datové struktury

2.3.1 Dostupné frekvence

Dostupné frekvence se budou načítat ze vstupního souboru (viz kapitola 4.2) bez předem známého počtu. Každá frekvence má svoje id a vlastní hodnotu frekvence v [Hz]. Tím se nabízí dvě možnosti. Vytvoření klíčovací tabulky pro id a ve vlastním programu pracovat pouze s id dané frekvence a pro výsledek rozklíčovat jednotlivé id podle tabulky, nebo vytvoření struktury pro každou frekvenci, která by obsahovala svoje id a frekvenci. Z pohledu paměťové náročnosti jsou obě možnosti ekvivalentní, proto jsem zvolil možnost se strukturou. V případě nutnosti úprav programu nebo dodatečného počítání s vlastní hodnotou bude snazší program rozšířit.

Vzhledem k tomu že se snažíme použít co nejmenší počet dostupných frekvencí budeme seznam frekvencí procházet vždy od začátku. Nepředpokládají se žádné náhodné přístupy doprostřed tohoto seznamu a také dopředu neznáme jejich počet ve vstupním souboru, proto se přikláním k vytvoření spojového seznamu frekvencí místo uložení do pole.

2.3.2 Vysílače

Vysílače načítáme ze vstupního souboru podobně jako dostupné frekvence, každý vysílač bude mít vlastní identifikační číslo a souřadnice (x,y) určující umístění na dvourozměrné mapě. Tentokrát bude potřeba jednotlivým vysílačům přiřazovat frekvence a jejich sousedy v případě grafu kolizí. Struktura je tedy výhodnou volbou.

Budeme se snažit udržet vyřešení problému ve dvou průchodech všech vysílačů, po načtení vstupních dat. Prvním průchod bude nutný pro vytvoření grafu kolizí (viz 2.1). Ve druhém průchodu budeme už přiřazovat frekvence jednotlivým vysílačům (viz 2.2).

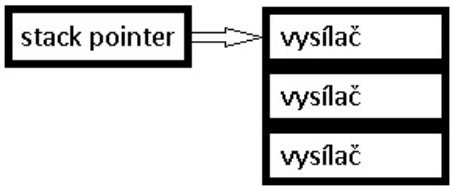
Sousedé každého vysílače by měli být reprezentováni jednosměrným spojo-

vým seznamem vzhledem k tomu že nevíme jak hustý graf kolizí vznikne a tím kolik bude mít každý vysílač sousedů. Je zde další možnost použití id jednotlivých vysílačů a v seznamu sousedů mít pouze id daných vysílačů. Tuto možnost jsem (ze stejných důvodů jako u frekvencí) zavrhnul a sousedé budou tvořeni ukazateli přímo na struktury daných vysílačů. Díky této volbě by neměl být nutný náhodný přístup doprostřed seznamu vysílačů a můžeme také použít jednosměrný spojový seznam. Tím se opět zjednodušuje načítání neznámého počtu vysílačů ze vstupního souboru vůči použití pole.

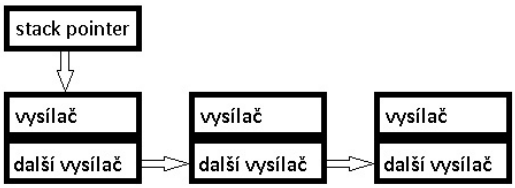
2.3.3 Zásobník

U zásobníku se nabízí reprezentace polem. Po vytvoření grafu kolizí by jsme měli vědět kolik by měla být maximální velikost zásobníku. V zásobníku by se mohli najednou vyskytovat všechny kolizní vysílače najednou. Taková možnost je sice nepravděpodobná, ale možná např. pro malý počet vstupních vysílačů blízko sebe. Bylo by nutné vytvořit dynamicky se zvětšující zásobník.

Z pohledu paměťové náročnosti a za předpokladu že by jsme měli dostatečnou velikost zásobníku, je výhodnější reprezentace polem, která pro ukládání vysílačů (bez vlastní režie zásobníku) potřebuje uložit jeden ukazatel na jeden vysílač (viz obr. 2.6). Naproti tomu zásobník reprezentovaný spojovým seznamem potřebuje ukazatele dva (jeden pro vlastní vysílač a druhý pro další položku spojového seznamu) jak se znázorněno na obr. 2.7. Tedy pro velmi hustý graf by se vyplatil zásobník polem, ovšem by se muselo ošetřit dynamické zvětšování pro krajní případy a ne všechna alokovaná paměť by se nutně využívala.



Obrázek 2.6: Zásobník polem.



Obrázek 2.7: Zásobník spojovým seznamem.

Výhoda použití spojového seznamu spočívá v jeho dynamické velikosti z podstaty spojového seznamu. S předpokladem že graf bude spíše řídký a případ, že každý vysílač koliduje s každým nebude běžně nastávat, budu realizovat zásobník spojovým seznamem. V našem problému ovšem nebudeme potřebovat zásobník pro jiné prvky než vysílače. Použijeme tedy struktur, které budou potřeba pro spojový seznam všech vysílačů. Zásobník zůstane použitelný pouze pro vysílače a implementace bude jednodušší.

Kapitola 3

Popis implementace

3.0.1 Popis datových struktur

Dostupné frekvence

Struktura `frequency` — Představuje dostupnou frekvenci. Obsahuje id a vlastní hodnotu frekvence v [Hz] a v programu se pracuje s frekvencí výhradně uložené v této struktuře.

Struktura `frequencies` — obsahuje ukazatel na strukturu `frequency` a ukazatel na další `frequencies`. Tato struktura slouží pro vytváření spojového seznamu z dostupných frekvencí. Pokud je to poslední frekvence v seznamu je její ukazatel na další `frequencies` nastaven na `NULL`.

Vysílače

Struktura `coordinates` — Obsahuje pouze informace o souřadnicích. Přiřazuje se danému vysílači jako reprezentace jeho pozice na mapě. Struktura

`transmitter` — Představuje vysílač. Obsahuje id vysílače, informaci o jeho pozici strukturou `coordinates`, ukazatel na přiřazenou frekvenci `frequency` a odkaz na první prvek spojového seznamu sousedů `neighbours`. Pokud vysílač nemá zatím přiřazenou žádnou frekvenci je ukazatel nastaven na `NULL`, to samé platí pro sousedy.

Struktura `neighbours` — Slouží k vytvoření spojového seznamu vysílačů stejně jako `transmitters`, v tomto případě představuje seznam sousedů který se přiřadí jednomu vysílači. Obsahuje odkaz na vysílač `transmitter` (který je sousedem) a odkaz na dalšího souseda `neighbours`. Ukazatel na dalšího souseda je nastaven na `NULL` pokud je to poslední soused.

Struktura `transmitters` — Slouží k vytvoření spojového seznamu vysílačů. Obsahuje ukazatel na vysílač `transmitter` a odkaz na další prvek seznamu `transmitters`. Pokud je to poslední prvek je odkaz na další nastaven na `NULL`.

3.0.2 Moduly

`freq.c`

Jedná se o hlavní modul, který se stará o načítání vstupního souboru, volání funkčního modulu `function.c` a uvolnění všech alokovaných prostředků (i v případě nenalezení řešení), které byly potřeba pro ostatní moduly.

Ukazatel `freqs_head` je ukazatel na hlavičku seznamu všech dostupných frekvencí.

Ukazatel `trans_head` je ukazatel na hlavičku seznamu všech vysílačů.

Ukazatel `m_stack` je ukazatel na vrchol zásobníku pro vysílače.

Hodnota `input_buffer_size` je maximální délka jednoho řádku ve vstupním souboru.

Funkce `load_available_frequencies` zistí zda se na začátku souboru nachází korektně pojmenovaná sekce dostupných frekvencí. Pokud ne vypíše do konzole chybu a vrátí hodnotu `1`. Pokud byla rozpoznána sekce dostupných frekvencí načítá frekvence pomocí funkce `read_frequency`. Dokud tato funkce úspěšně načítá řádky frekvence sou přidávány do seznamu dostupných frekvencí funkcí `frequencies_add_frequency` z modulu `function.c`. Po úspěšném načtení celého bloku frekvencí funkce vrátí hodnotu `0`.

Funkce `load_radius` zistí zda se na dalším řádku souboru nachází korektně pojmenovaná sekce dosahu signálu vysílačů. Pokud ne vypíše do konzole chybu a vrátí hodnotu `2`. Pokud byla rozpoznána sekce dosahu signálu vysílačů načte hodnotu a pomocí funkce `transmitters_set_radius` z modulu `transmitters.c` ji uloží. Po úspěšném načtení hodnoty přejde k dalšímu řádku souboru a vrátí hodnotu `0`.

Funkce `load_transmitters` zjistí zda se na začátku souboru nachází korektně pojmenovaná sekce vysílačů. Pokud ne vypíše do konzole chybu a vrátí hodnotu `3`. Pokud byla rozpoznána sekce vysílačů načítá frekvence pomocí funkce `read_transmitter`. Dokud tato funkce úspěšně načítá řádky vysílačů sou přidávány do seznamu všech vysílačů funkcí
`transmitters_add_transmitter` z modulu `transmitters.c`. Po úspěšném načtení celého bloku vysílačů funkce vrátí hodnotu `0`.

Funkce `load_file` se postará o načtení vstupního souboru pomocí funkcí:
`load_available_frequencies`
`load_radius`
`load_transmitters`.

Funkce `main` zajistí inicializaci hlaviček, zkontroluje počet argumentů a pokusí se načíst vstupní soubor funkcí `load_file`. Při úspěšném načtení souboru zajistí vytvoření kolizního grafu voláním funkce
`function_find_collisions` a následně přiřazení frekvencí vysílačům funkcí `function_assing_frequencies`. Při úspěšném přiřazení vypíše všechny vysílače do konzole a uvolní alokovanou paměť funkcí `clear`. Pokud program nedokázal najít řešení nebo nastala jiná chyba, vypíše příslušnou chybovou hlášku do konzole a uvolní paměť voláním `clear`.

frequencies.c

Modul obsahuje funkce pro snadné vytvoření a smazání spojového seznamu dostupných frekvencí.

Funkce `frequencies_add_frequency` vytvoří novou frekvenci `frequency` podle předaných parametrů a zařadí ji do seznamu dostupných frekvencí předaného v podobě ukazatele na poslední přidaný prvek. Po zařazní nové frekvence vrátí ukazatel na poslední zařazený prvek. Pokud se jedná o první prvek seznamu stačí místo posledního prvku do funkce předat ukazatel `NULL`.
Dále modul obsahuje funkce pro výpis frekvence, nebo frekvencí do konzole. Na funkci celého programu nemají vliv. Jsou určeny k testování.

transmitters.c

Modul obsahuje funkce pro vytvoření seznamu vysílačů, přidávání jejich sousedů, mazání spojového seznamu vysílačů a výpis seznamu do konzole. Tento modul je závislý na modulu `frequencies.c`, protože jeden z atributů vysílače je jeho frekvence.

Funkce `transmitters_add_transmitter` vytvoří nový vysílač podle předaných parametrů a zařadí ho do seznamu vysílačů předaného pomocí ukazatele na poslední přidaný prvek. Po zařazení nového vysílače vrátí ukazatel na poslední zařazený prvek. Pokud se jedná o první prvek seznamu stačí místo posledního prvku do funkce předat ukazatel `NULL`.

Funkce `transmitters_print_all` vypíše do konzole předaný prvek spojového seznamu a všechny následující. Tato funkce je použita jako standardní výstup programu a je volána až po úspěšném přiřazení frekvencí všem vysílačům.
Dále modul obsahuje funkce pro výpis jednoho vysílače nebo sousedů konkrétního vysílače do konzole. Jsou určeny k testování.

transmitters_stack.c

Modul slouží pro vytvoření a obsluhu zásobníku. Do zásobníku lze vkládat pouze vysílače `transmitter`. Modul je tedy závislý na modulu `transmitters.c`.

Funkce `t_stack_pop` vyjme prvek ze zásobníku, předaného jako ukazatel na ukazatele vrcholu zásobníku, a ukazatel na něj vloží do předaného ukazatele na ukazatele `transmitter`. Dále uvolní alokovanou paměť kterou zásobník potřeboval. Pokud tedy odebereme všechny prvky ze zásobníku není nutné volat funkci `t_stack_free_stack`.

function.c

Modul je závislý na všech předchozích, protože poskytuje funkce pro nalezení kolizí vysílačů, přiřazování frekvencí atp.

Hodnota `power_radius` je umocněný dosah signálu vysílačů. Mocnina je zde kvůli snažšímu porovnávání vzdáleností dvou vysílačů, aby nebylo nutné pokaždé vzdálenost odmocňovat.

Ukazatel `f_encountered` je ukazatel na ukazatel na `NULL`. Používá se pouze jako označení vysílače za nalezený bez přiřazené frekvence, který se přidává do zásobníku. Tímto označením snadno zabráníme zbytečnému přidání některého vysílače do zásobníku dvakrát. Tento ukazatel se vytváří při volání funkce `function_assing_frequencies`.

Funkce `function_find_collisions` očekává předání hlavičky spojového seznamu vysílačů. Prochází všechny dvojice vysílačů a zjišťuje zda může nastat kolize signálu pomocí funkce `calculate_collision`. Pokud je nalezena kolize dvou vysílačů, je mezi nimi vytvořena neorientovaná hrana pomocí seznamů sousedů. Díky tomu že vytváříme neorientované hrany, hlavní cyklus prochází seznam vysílačů pouze jednou a sekundární cyklus prochází pouze zbývající vysílače. Nekontroluje se tedy nikdy dvakrát stejná dvojice, což znamená urychlení v případě velkého počtu vysílačů. Tato funkce se volá z hlavního modulu `freq.c` po načtení vstupních dat.

Funkce `is_possible_to_assign` očekává předání vysílače, kterému byla dočasně přiřazena frekvence a zjišťuje zda vysílač má nějaké sousedy, pokud nemá funkce úspěšně skončí a vysílači zůstane frekvence přiřazená. V případě že má sousedy postupně je prochází a zjišťuje zda frekvence předaného vysílače nekoliduje. Všechny jeho sousedy, kteří nemají přiřazenou frekvenci a ještě nejsou v zásobníku označí `f_encountered` za nalezené a vloží je do zásobníku. Pokud nenastala žádná kolize se sousedem funkce úspěšně skončí a vrací hodnotu `0`. Pokud dojde ke kolizi vrací hodnotu `1`. Volá se pouze vnitřně z tohoto modulu.

Funkce `assign_frequency` očekává předání vysílače a hlavičky seznamu dostupných frekvencí. Funkce přiřazuje vysílači frekvence ze seznamu a testuje úspěšné přiřazení funkcí `is_possible_to_assign`. Pokud přiřazení neproběhne a vyčerpá se seznam dostupných frekvencí funkce vrátí `1`. Při úspěšném přiřazení vrací `0`. Volá se pouze vnitřně z tohoto modulu.

Funkce `function_assing_frequencies` očekává ukazatel na hlavičku seznamu všech vysílačů, ukazatel na hlavičku všech dostupných frekvencí a ukazatel na ukazatel prázdného zásobníku. Funkce prochází seznam všech vysílačů. Vysílač který nemá přiřazenou frekvenci vloží do zásobníku a čeká na konec vedlejšího cyklu. Tento cylus vybírá vysílače ze zásobníku a pokouší se jim přiřazovat frekvence pomocí funkce `assign_frequency`. Tato funkce přidává do zásobníku další vysílače (viz 3.0.2 Funkce `assign_frequency`). Vedlejší cyklus tedy končí v případě, že byly úspěšně přiřazeny frekvence celé souvislé komponentě ve které se daný vysílač nacházel. Pokud se přiřazení některému z vysílačů nepodařilo funkce končí a vrací hodnotu `1`. Při úspěšném přiřazení všem vysílačům vrací `0`. Tato funkce se volá z hlavního modulu `freq.c` po načtení vstupních dat a vytvoření sousedů pomocí funkce `function_find_collisions`.

Kapitola 4

Uživatelská příručka

4.1 Program

4.1.1 Přeložení programu

UNIX/Linux

Správnou kompilaci programu lze zajistit použitím příkazu `make` v adresáři se zdrojovými soubory a se souborem makefile.

Win32/64

Správnou kompilaci programu lze zajistit použitím příkazu: `mingw32-make` v adresáři se zdrojovými soubory a se souborem `makefile`. Pro tento způsob je nutné mít nainstalovaný balík `mingw32` a překladač `gcc` a oba přidané v sytémové proměnné `PATH`.

4.1.2 Další funkce makefile

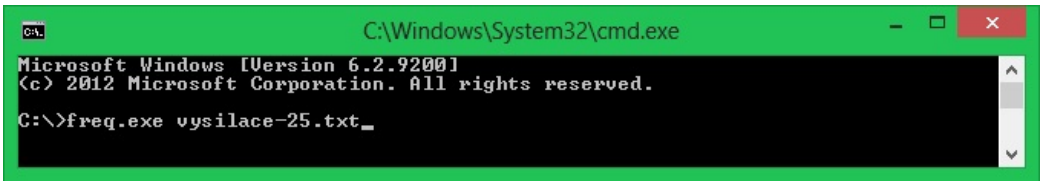
Vytvořené soubory `makefile` a `makefile.win` obsahují další funkčnost pro usnadnění práce se zdrojovými kódy přidáním klíčového slova za příkaz `make`.

Klíčová slova:
`clean` — smaže všechny objektové soubory v adresáři včetně `freq.exe`.

`rebuild` — smaže všechny objektové soubory v adresáři včetně `freq.exe` a provede znovu kompletní kompilaci.

4.1.3 Spuštění programu

Program se spouští z příkazové řádky příkazem:
`freq.exe` <*soubor-s-vysílači.txt*>
Jediný a povinný parametr je název vstupního textového souboru. Který se nachází ve stejném adresáři jako `freq.exe`. Příkaz na spuštění programu by mohl například vypadat jako na obr. 4.1.



Obrázek 4.1: Spuštění programu.

Při úspěšném nalezení šeření program vypíše do konzole všechny vysílače reprezentované jejich vlastním `ID` a přiřazenou hodnotou frekvence v Hertzích.

4.2 Formát vstupního souboru

Vstupní textový soubor začíná řetězcem `Available frequencies:` za kterým následují dostupné frekvence. Na každém řádku je pouze jedna frekvence

ve formátu: `ID-frekvence hodnota`
kde `ID-frekvence` je identifikační (celé kladné) číslo a frekvence jsou v souboru seřazeny vzestupně podle jejich ID. Hodnota frekvence `hodnota` je udaná v Hertzích.

Dále následuje na samostatném řádku řetězec `Transmission radius:` za kterým je udán dosah signálu všech vysílačů udaný jako celé kladné číslo.

Na dalším řádku následuje řetězec `Transmitters:` za kterým následují vysílače. Na každém řádku je pouze jeden vysílač ve formátu:
`ID-vysílače souřadnice-x souřadnice-y`
kde `ID-vysílače` je identifikační (celé kladné) číslo a `souřadnice-x` a `souřadnice-y` udávají pozici vysílače na mapě a mohou to být desetinná čísla.

Ukázkový vstupní soubor by mohl vypadat např. takto:

```
Available frequencies:
0 93400
1 104600
2 139700
Transmission radius:
15
Transmitters:
0 115.698096 3.112792
1 95.047235 112.320582
2 74.776052 33.719497
3 29.709430 114.079607
```

Kapitola 5

Závěr

Program byl původně navržen bez zásobníku, ale problémy s využíváním příliš velkého množství frekvencí se značně projevili především na malých testovacích datech. Po úpravách program úspěšně přiřazuje zadaným vysílačům dostupné frekvence bez kolizí. Nelze ovšem zajistit nalezení existujícího řešení ve všech případech z podstaty daného problému. Program byl upraven tak aby zajišťoval co největší úspěšnost při hledání řešení bez složitých algoritmů.

Pro nalezení řešení by mohl poskytovat různé možnosti procházení sousedů v grafu, což by mohlo při opakovaném spouštění zajistit různé výsledky a případné nalezení řešení, které předtím nebylo možné. Program také nedokázal najít řešení pro testovací data `vysilace-25.txt`, kde řešení existuje, ale kvůli závislosti na pořadí průchodu sousedů v grafu ho nebylo možné v tomto případě najít. Průměrný běh programu pro testovací data `vysilace-1000.txt` byl 0,1 sekundy což se zdá jako rychlý průběh po zkušenostech s rychlostí podobných aplikací implementovaných v Javě.