VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

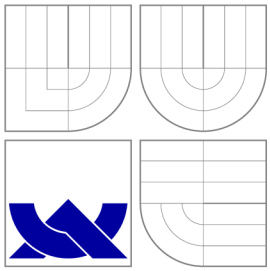
VÝPOČET ROZVRŽENÍ VRCHOLŮ GRAFU A VIZUALIZACE GRAFŮ VE DVOUROZMĚRNÉ ROVINĚ

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

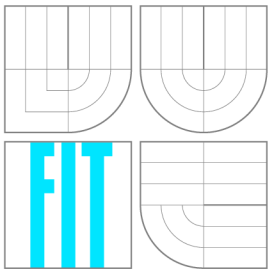
AUTOR PRÁCE
AUTHOR

VÁCLAV ŠUŠLÍK

BRNO 2009



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

VÝPOČET ROZVRŽENÍ VRCHOLŮ GRAFU A VIZUALIZACE GRAFŮ VE DVOUROZMĚRNÉ ROVINĚ

COMPUTATION OF GRAPH VERTICES LAYOUT AND VISUALIZATION OF GRAPHS IN TWO-DIMENSIONAL PLANE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — VÁCLAV ŠUŠLÍK
AUTHOR

VEDOUCÍ PRÁCE — Ing. JIŘÍ ZUZAŇÁK
SUPERVISOR

BRNO 2009

Abstrakt

Graf je struktura sloužící k zobrazení vztáhů mezi entitami. Je využívána v mnoha rozlišných oborech lidského vědění. Velký význam má pak grafická reprezentace grafu – diagram. Tato práce je zaměřena na transformaci grafu na diagram, tj. na jeho vykreslení. V práci je detailně popsán hierarchický přístup kreslení, pro většinu jeho kroků je popsáno několik metod, které jsou níže v práci porovnávy a zhodnoceny. Součástí práce je pak knihovna a testovací aplikace, které tento přístup implementují.

Abstract

The graph is structure used to display relations between entities. It is used in many fields of human knowledge. Great importance is the graphic representation of graph - diagram. This text is dealing with transformation graph to diagram - graph drawing. The text is described in detail the hierarchical approach of drawing. For most steps, there are some methods, which are then compared and evaluated. Part of this work is the library and test application, which implemented this aproach.

Klíčová slova

Rozvržení vrcholů grafu, Kreslení grafů, Vrstevné kreslení

Keywords

Computation of graph vertices layout, Graph drawing, Layered drawing

Citace

Václav Šušlík: Výpočet rozvržení vrcholů grafu a vizualizace grafů ve dvourozměrné rovině, bakalářská práce, Brno, FIT VUT v Brně, 2009

Výpočet rozvržení vrcholů grafu a vizualizace grafů ve dvourozměrné rovině

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Jiřího Zuzaňáka.

.....
Václav Šušlík
19. května 2009

Poděkování

Tímto bych chtěl poděkovat vedoucímu své bakalářské práce, panu Ing. Jiřímu Zuzaňákovi, za pomoc a veškerý čas, který mi věnoval.

Obsah

Kapitola 1

Úvod

Graf je struktura, která slouží k zobrazení vztáhů mezi entitami. Je využívána v mnoha rozlišných oborech lidského vědění. Velký význam má pak grafická reprezentace grafu – diagram. Proces transformace grafu na diagram můžeme označit jako vykreslování grafů, které je úkolem této práce. Vykreslování grafů je ovšem velmi obsáhlá disciplína teorie grafů, tudíž je práce dále omezena pouze na hierarchický přístup kreslení (někdy též označovaný jako vrstevné kreslení). Tento přístup transformuje graf na diagram v několika krocích. Součástí práce je dynamická knihovna pro vykreslování grafů „GraphDrw“ a testovací aplikace „TestApp“. Knihovna i aplikace jsou implementovány v jazyce C++ za použití multiplatformních knihoven. Měly by být bez větších problémů přenositelné na jiné architektury.

Práce je rozdělena do několika kapitol. V kapitole 2 jsou vysvětleny základní pojmy teorie grafů, používané v dalších částech práce. Kapitola 3 pak popisuje základní myšlěnky vizualice grafů, její konvence a estetická kritéria. Kapitola 4 již popisuje hlavní část této práce – hierarchickou metodiku kreslení grafů. V každé části této kapitoly je specifikován jeden krok tohoto přístupu. Pro většinu kroků je uvedeno několik algoritmů spolu s doprovodným pseudokódem, případně připomínkou zjištěnou v době implementace. Kapitola 5 je zaměřena na testování vybraných algoritmů, zkoumána je zejména doba výpočtu. Závěrečná kapitola 6 seznamuje čtenáře s výslednou knihovnou a testovací aplikací. Dále popisuje vhodně navrhnutou strukturu reprezentující graf, pro výpočty rozlišných algoritmů hierarchického kreslení.

Kapitola 2

Základní pojmy

Graf je obecně definován jako dvojice (V, E). V je pak neprázdná množina prvků, které označujeme vrcholy grafu. E definujeme jako množinu obsahující dvojice prvků $(u, v) \in V$, tyto dvojice označujeme jako hrany. Pokud existuje hrana z vrcholu v do vrcholu u můžeme říci, že vrchol v sousedí s vrcholem u. Relaci z množiny V do E označujeme v teorii grafů jako incidenční relaci a za předpokladu, že vrchol v je v relaci s hranou e, můžeme říci, že v je s touto hranou incidenční.

Hrany lze obecně rozdělit na několik typů:

- **Orientované hrany**
  Prvky množiny E jsou v tomto případě uspořádány, při průchodu striktně záleží na jejich uspořádání. Hranou jde v grafu projít pouze v jejím směru. Obvykle jsou tyto hrany kresleny jako šipky.
- **Neorientované hrany**
  Prvky množiny E nejsou uspořádány. Směr průchodu hranou je libovolný.
- **Násobné hrany**
  Prvky množiny (dvojice vrcholů (u, v)) E nejsou v jejím rámci unikátní.
- **Smyčky**
  Hrany z vrcholu v do vrcholu v. Obecněji lze smyčku ožnačit jako hranu vedoucí z vrcholu do téhož vrcholu.

Stupeň vrcholu

Stupeň vrcholu v teorii grafů chápeme jako počet hran, které do vrcholu zasahují, nebo počet hran jež jsou s daným vrcholem incidentní. Bližší definice je spojena s typem grafu. Pro neorientované grafy jej označujeme jako $deg(u)$, kde $u \in V$. V případě smyčky z vrcholu v do vrcholu v je pak tato hrana započítána 2krát (koncové body tvoří tentýž vrchol). Pro grafy orientované je stupeň vrcholu rozdělen na stupeň vstupní a stupeň výstupní. Vstupní stupeň značíme jako $deg^+(u)$, výstupní pak $deg^-(u)$. Vstupující a vystupující hrany se v tomto případě počítají zvlášť. Celkový stupeň vrcholu orientovaného grafu získáme součtem vstupního a výstupního stupně.

Cesta v grafu

Cesta je obecně definována jako posloupnost vrcholů a hran $(v_1, e1, v_2, e_2, \ldots, v_n)$, kde hrany $e_1..e_n \in E$ a vrcholy $v_1..v_n \in V$ jsou navzájem různé. S cestou grafu je svázán pojem souvislost grafu. Řekněme, že graf je souvislý, pokud pro každou libovolnou dvojici vrcholů (u, v) existuje v grafu cesta z u do v.

Cyklus v grafu

Cyklus lze chápat jako posloupnost vrcholů a hran $(v_1, e1, v_2, e_2, \ldots, v_n)$, kde $v_n = v_1$, hrany $e_1..e_n \in E$ a vrcholy $v_1..v_n \in V$ jsou navzájem různé. Jedná se vlastně o speciální případ cesty.

Rozdělení grafů

Podle výše definovaných pojmů můžeme grafy rozdělit do základních skupin.

- **Podle četnosti násobných hran**
  - **Prosté grafy**
    Neobsahují násobné hrany.
  - **Multigrafy**
    Obsahují $2..n$ násobných hran.
  - **Řídké grafy**
    Zajímá nás poměr počtu hran ($|V|$) ku počtu vrcholů ($|E|$). Pokud počet vrcholů výrazně převyšuje počet hran, označíme graf jako řídký.
  - **Husté grafy**
    Jedná se o opačný případ grafu řídkého. Počet vrcholů je výrazně menší než počet hran.
- **Podle orientace hran**
  - **Orientované grafy**
    Hrany grafu jsou orientovány.
  - **Nerientované grafy**
    Hrany grafu nejsou orientovány.
- **Podle existence cyklů v grafu**
  - **Cyklické grafy**
    Grafy obsahující $1..n$ cyklů.
  - **Acyklické grafy**
    Grafy neobsahující žádný cyklus.
- **Podle planarity**
  - **Planární grafy**
    Lze je nakreslit do roviny bez křížení hran.

  - **Neplanární grafy**
    Nelze je do roviny nakreslit bez křížení hran.

- **Podle ohodnocení**
  - **Neohodnocené**
    V neohodnoceném grafu není hranám přiřazeno žádné ohodnocení.
  - **Ohodnocené**
    Hrany nesou doplňující informace. Klasickým případem ohodnocení je mapa, vrcholy grafu reprezentují města, hrany pak cesty mezi těmito městy. Každá hrana je doplněna o reálnou vzdálenost jejich uzlů (měst). Následně můžeme řešit úlohy např. s hledáním maximální/minimální cesty z města A do města B.

Další důležité typy grafů

- **Úplný graf**
  V tomto grafu jsou všechny vrcholy množiny V spojeny hranou.

- **Bipartitní graf**
  Je graf, jehož množinu vrcholů V můžeme rozdělit na 2 podmnožiny V_1, V_2 a to tak, že vrcholy podmnožiny V_1 jsou sousední pouze s vrcholy podmnožiny V_2. Pro vrcholy vrstvy V_2 platí totéž v opačném znění. Pokud jsou navíc všechny vrcholy zmíněných podmnožin spojeny hranou, mluvíme o úplném bipartitním grafu.

Matematická reprezentace

Nejčastějším matematickým zápisem grafu je zápis pomocí matice sousednosti. Tj. čtvercovou maticí A typu $\{0,1\}^{n*n}$, kde n je celkový počet vrcholů v grafu. Pořadí vrcholů je zvoleno libovolně. Hodnoty prvků $A_{i,j}$ matice A jsou dány předpisem

$$\begin{cases} 1, \text{pro}(A_i, A_j) \in E \\ 0, \text{jinak} \end{cases}$$

. Tedy pro případ, kdy existuje z vrcholu na pozici A_i hrana do vrcholu A_j je hodnota $A_{ij} = 1$, v opačném případě 0. Specifickým příkladem je úplný graf, jehož matice sousednosti má pro všechny A_{ij} hodnotu 1 vyjma její hlavní diagonály, která má hodnoty nulové. Mezi další možnosti zápisu patří např. Laplaceova matice, matice vzdálenosti, matice incidence.

Diagram

Lze chápat jako grafické či schématické znázornění nějakého jevu, množiny vztahů, matematických údajů atd. V této práci budeme diagram chápat jako rovinné zobrazení grafu s tím, že se budeme snažit o co jeho nejvyšší čitelnost.

Kapitola 3

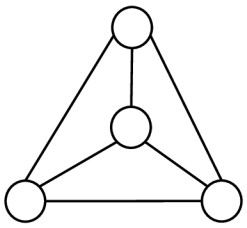
Vizualizace grafů

Vizualizace grafu [3] je problém nalezení jeho vhodné grafické reprezentace. Jedná se vlastně o transformaci grafu G na diagram. Při této transformaci je množina nesoucí vrcholy V a množina E nesoucí hrany grafu doplněna o další údaje. V případě vrcholů zejména o souřadnice, v případě hran např. o řídící body křivky. Při vizualici je potřeba zvolit vhodnou konvenci a dodržovat jistá estetická kritéria zmíněná níže. Jednotlivá dění v oboru vizualizace grafů včetně popisu metod lze sledovat na „Journal of Graph Algorithms and Applications“ [6].

3.1 Konvence kreslení

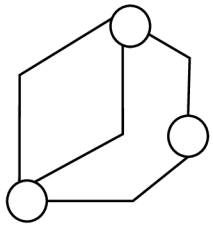
Vrcholy mohou být reprezentovány několika způsoby, např. bodem, obdélníkem nesoucí vepsaný text, kruhem atd. Taktéž hrany, které obecně při vykreslování chápeme jako křivky, můžeme reprezentovat mnoha způsoby. Při volbě vhodné reprezentace je směrodatná zejména oblast použití grafu, ať už informatika (DFD, ERD), přírodověda (Vývojové diagramy), matematika atd. Druhotným faktorem je jaké informace má daný graf obsahovat. Vizualizaci grafu můžeme rozdělit do několika tříd:

- **Planární kreslení**
  Při tomto kreslení jsou hrany reprezentovány přímými čarami, a to tak, že se žádná dvojice hran $(e_1, e_2) \in E$ nekříží. Tímto způsobem jdou ovšem vykreslit pouze některé grafy. Velmi lehce můžeme nakreslit graf, kde se hrany kříží vždy.



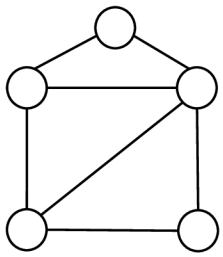
Obrázek 3.1: Planární graf

- **Kreslení lomených čár**
  Pro kreslení hran jsou použity lomené čáry, tj. sled na sebe navazujících úseček.



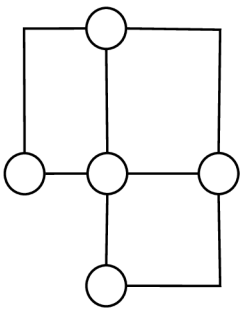
Obrázek 3.2: Kreslení lomených čar

- **Kreslení přímých čar**
  Pro kreslení hran je použito jednoduchých úseček.



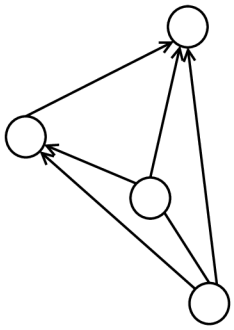
Obrázek 3.3: Kreslení přímých čar

- **Ortogonální kreslení**
  Pro kreslení hran je použito pouze vertikálních a horizontálních úseček.



Obrázek 3.4: Ortogonální kreslení

- **Vzestupné kreslení**
  Při tomto kreslení musí být směr všech hran buď sestupný nebo vzestupný.



Obrázek 3.5: Vzestupné kreslení

3.2 Estetická kritéria

Estetická kritéria byla stanovena pro možnost ohodnocení čitelnosti výsledného grafu. Výše zmíněné konvence nám totiž nedávají žádná pravidla, jak by měl výsledný graf vypadat. Pro dosažení co možná největší čitelnosti je nutné dodržet následující pravidla:

- **Symetričnost**
  Pokud graf obsahuje jisté symetrie, je vhodné tyto symetrie přenést do diagramu.
- **Počet křížení hran**
  V ideálním případě je počet křížení hran v grafu 0. Nicméně toho lze dosáhnout pouze za předpokladu planarity grafu.
- **Délka hran**
  Snažíme se minimalizovat celkový součet délek všech hran na minimum.
- **Rozloha**
  Zajímá nás zejména poměr výšky / šířky a celkový obsah zabrané plochy.

Graf nemusí splňovat všechna tato pravidla, dokonce některá pravidla se mohou částečně vylučovat. Nejlepším řešením je zvolit vhodný kompromis a zaměřit se na dodržení pravidel, která jsou pro nás směrodatná.

Kapitola 4

Hierarchické metody kreslení grafů a jejich implementace

Tato kapitola popisuje jeden z přístupů ke kreslení grafů, který je hlavním předmětem této práce. Jednotlivé podkapitoly popisují kroky, které je nutno provést až po výsledné vykreslení.

4.1 Odstranění cyklů

Algoritmy popsané v této kapitole počítají na vstupu s acyklickým grafem. To je poměrně velké omezení, které se dá obejít např.: pomocí převodu cyklického grafu na graf acyklický. Tento převod probíhá pomocí „rozpojení“ cyklů a to změnou orientace jedné či více hran. Úloha se pak rozpadá na problém nalezení vhodné kombinace hran (změnou orientace jedné hrany pro odstranění detekovaného cyklu můžeme do grafu zanést n cyklů dalších). Hrany, kterým bylo změněno pořadí, označíme speciálním příznakem. Před zobrazením výsledného grafu označené hrany otočíme nazpět, tím zaručíme jeho korektnost. Tuto problematiku spolu s popisem dalších algoritmů je možné nález v [1].

Před touto fází je potřeba ostranit z grafu smyčky. Dále je vhodné zredukovat duplicitní hrany, ať už stejné či opačné orientace.

4.1.1 Greedy-cycle removal

Patří mezi algoritmy odstraňující cykly v orientovaném gafu na základě hledání vhodného uspořádní vrholů. Toto uspořádní nalezne s lineární časovou složitostí. Rozhodujícím faktorem pro určení pořadí vrcholů je vstupní / výstupní stupeň daného vrcholu. Na začátku algoritmu máme 2 prázdné množiny $S_l \leftarrow \varnothing$ a $S_r \leftarrow \varnothing$ a množinu V nesoucí všechny vcholy grafu. V algoritmu postupně vybíráme prvky z množiny V a vkládáme je do množin S_r nebo S_l podle pravidel popsaných pseudokódem, ve kterém jsou uvedeny 2 pojmy – příjemce a zdroj. Vrchol v označíme jako příjemce pokud platí $deg^+(v) > 0$, zdrojem pro případ kdy $deg^-(v) > 0$. Je zřejmé, že vrchol v může být současně zdrojem i příjemcem.

```
S_r ← ∅
S_l ← ∅
while graf není prázdný
  while graf obsahuje přijemce
    Vyber příjemce p
    Odstraň příjemce p z grafu
    Přidej p na začátek S_r
  end while
  while graf obsahuje zdroj
    Vyber zroj z
    Odstraň zdroj z z grafu
    Přidej z na konec S_l
  end while
  if graf není prázdný
    Vyber vrchol v s max rozdílem deg(v)^+ a deg(v)^-
    Odstraň zdroj v z graf
    Přidej v na konec S_l
  end if
end while
```

Zde již máme určenou prioritu jednotlivých vrcholů, kterou použijeme k dočasné změně orientace hran grafu. Orientaci změníme u všech hran jejichž počáteční bod má vyšší prioritu než bod koncový.

4.2 Zasazení do vrstev

Zasazení do vrstev je transformace acyklického grafu na graf hladinový. Hladinový graf je graf, ve kterém jsou všechny vrcholy množiny V rozloženy do podmnožin $L_1 - L_n$ a platí: pokud $(u, v) \in E$, pak $u \in L_i, v \in L_j$, kde $j > i$. Výška hladinového grafu je pak počet vrstev (podmnožin L). Šířka grafu je dána jako maximální velikost vrstvy (podmnožiny L_k) $w = \max(|Li|)$, pro $0 < i \leqq k$, kde k je počet vrstev.

4.2.1 Podle nejdelší cesty

Pro určení výsledné vrstvy daného vrcholu je použito hledání maximální cesty. V počáteční fázi jsou vrcholy s nulovým výstupním stupněm $deg^-(v)$ umístěny do nulté vrstvy. Po počátečním kroku je pro zbylé vrcholy hledána délka d maximální cesty k nějakému z vrcholů nulté vrstvy. Vrchol je pak zasazen do vrstvy L_{1+d}. Nevýhodou tohoto algoritmu je špatný poměr výsledné šířky / výšky grafu (neuspokojivá je zejména výška). Dosáhnutí optimálního poměru je NP-úplný problém, nicméně pokud zavedeme jistá omezení, je částečně řešitelný pomocí Coffman-Graham algoritmu.

```
for each vrchol v in G
  if deg⁻(v) == 0
    vlož v do nulté vrstvy
    odstraň v vrchol z G
  end if
end each
for each vrchol v in G
  najdi délku d max cesty k vrcholu nulté vrstvy
  vlož v do vrstvy L_{1+d}
  odstraň vrchol v z G
end each
```

4.2.2 Coffman-Graham

Jak již bylo zmíněno v popisu předchozího algoritmu, dosáhnutí optimálního poměru šírky a výšky je částečně řešitelné zavedením určitých omezení. Coffman-Graham algoritmus (dále jen CFA) zavádí maximální šířku vrstvy W – snaží se udržet minimální výšku grafu při zachování maximální šířky vrstev. Pokud je tato šířka překročena, vrchol je přidán do následující vrstvy L_{i+1}.
CFA probíhá ve 2 krocích, v prvním kroku dochází k očíslování a seřazení vrcholů, v druhém jsou pak tyto vrcholy zařazeny do vrstev s dodržením výše zmíněné hraniční šířky. Při řazení algoritmus pracuje s množinami kladných celých čísel A, B, nad kterými zavádí relaci „$>$“, kde pokud $A > B$ platí jeden ze vztahů:

$$A \neq \varnothing \wedge B = \varnothing \tag{4.1}$$

$$A \neq \varnothing \wedge B \neq \varnothing \wedge \max(A) > \max(B) \tag{4.2}$$

$$A \neq \varnothing \wedge B \neq \varnothing \wedge \max(A) = \max(B) \wedge |A| > |B| \tag{4.3}$$

Algoritmus lze pak zapsat následovně:

for $i = 1$ **to** $|V|$
  Vyber minimální neoznačený vrchol v podle
  výše zmíněné relace „>“
  Označ vybraný vrchol v hodnotou i
end for

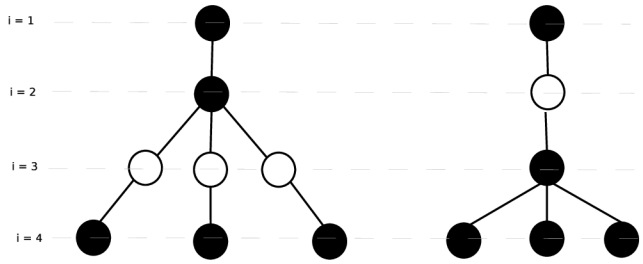
$U \leftarrow \varnothing$
$k = 1$
while $U \neq V$
  $R = V - U$
  Vyber vrchol v z R, takový, že všichni jeho následníci
  patří již do nějaké vrstvy $L_1, L_2 ... L_k - 1$ a jehož
  označení má v rámci množiny R maximální hodnotu.

  if $|L_k| < W$
    $L_k = L_k + v$
  else
    $k = k + 1$
    $L_k = v$
  end if

  $U = U + v$
end while

4.3 Přidání prázdných vrcholů

Pro minimalizaci křížení je hladinový graf potřeba předpřipravit za pomocí prázdných vrcholů. Prázdný vrchol je vrchol, který můžeme z hlediska vizualizace grafu chápat jako zalomení hrany / řídící bod křivky. Každou hranu e, jejíž vrcholy $(u, v) \in E$ leží ve vrstvě $u \in L_i$, $v \in L_j$ a platí $|j - i| > 0$ je nutno rozdělit pomocí prázdných vrcholů. A to tak, aby pro každý vrchol platilo $|i - j| = 1$. Počet prázdných vrcholů by z důvodů značného nárůstu složitosti dalších algoritmů měl být co nejmenší, což dává tomuto bodu velký význam pro optimalizaci.



Obrázek 4.1: Možná redukce snížení počtu prázdných vrcholů

4.4 Minimalizace křížení hran

Pro snížení počtu hranových průsečíků hladinového grafu je možné použít několik přístupů. Tato práce je zaměřena na metodu zametání vrstvami. Je dokázáno, že výsledný počet křížení hran je dán pouze pořadím vrcholů ve vrstvě. Úloha tedy spočívá v nalezení co možná nejlepší permutace vrcholů, při které je tento počet minimální – optimálně nulový. Je vybrána jedna vrstva např. L_i, kde $i = 1$ (obvykle první), u níž je pořadí vrcholů neměnné. Poté vždy pro vrstvu L_{i+1} hledáme vhodné pořadí vrcholů. Po nalezení je tato vrstva uzamčena a algoritmus pokračuje až do vrstvy L_{k-1}. Existuje mnoho modifikací, můžeme např. označit jako výchozí vrstu $L_{k/2}$ a následně hledat pořadí vrcholů pro vrstvu $L_{(k/2)+1}$ a $L_{(k/2)-1}$. Je taktéž zřejmé, že pro nalezení co nejlepší kombinace vrcholů všech vrstev většinou nestačí 1 iterace, obecně, čím víc iterací provedeme, tím bude výsledek přijatelnější. V následujících kapitolách jsou popsány metody pro nalezení vhodného pořadí vrcholů.

4.4.1 Výměna sousedů

Jedná se o obdobu známého třídícího algoritmu bubble-sort. Rozdílná je hlavně porovnávací funkce. Algoritmus zkoumá dvojice sousedů, pokud dojde jejich záměnou ke snížení počtu hranových průsečíků, zamění je. Asymptotická složitost algoritmu je $O(N^2)$

```
for i = 1 to |V|
  for j = 1 to |V| − i
    if dojde ke snížení počtu křížení
      V_i ↔ V_j
    end if
  end for
end for
```

4.4.2 Split

Opět se jedná o obdobu známého třídícího algoritmu, tentokrát Quick-sort. Nejhorší časová složitost může být $O(N^2)$, obecně je ale jeho časová složitost $O(Nlog_2N)$. Základem algoritmu je rozdělení množiny V nesoucí vrcholy na 2 podmnožiny a následné rekurzivní volání.

```
Vyber dělící bod množiny V
V_l ← ∅
V_r ← ∅
for i = 1 to |V|
  if Přesunutím vrcholu V_i před dělící
    bod dojde ke snížení počtu křížení
    V_l = V_l + {V_i}
  else
    V_r = V_r + {V_i}
  end if
end for
Rekurzivně aplikuj algoritmus pro množiny V_l, V_r.
Výstup získej zřetězením těchto množin.
```

4.4.3 Baricentric

Hlavní předností této metody spolu s metodou mediánovou je lineární časová složitost. Je postavena na aritmetrickém průměru a myšlence, že vrcholy mají být co nejblíže svým sousedním vrcholům. Další výhodou je snadná imlementace. Pozice každého vrcholu je počítána jako aritmetický průměr pozic jeho vrcholů sousedních – ležících v předchozí vrstvě. Pokud je na této pozici nějaký vrchol, je vybrána libovolná volná pozice, např $\max(Pozice) + 1$.

```
for i = 1 to |V|
   Spočítej aritmetický průměr ar pozic sousedních
   vrcholů v předchozí vrstvě

   if Pozice ar není obsazena
      P_{V_i} = ar
   else
      P_{V_i} = max(P)
   end if
end for
```

Kde P označuje pozici vrcholu ve vrstvě.

4.4.4 Median

Lineární složitostí odpovídá metodě baricentrické. Nejdříve je každému vrcholu přiřazen medián pozic sousedních vrcholů ve vrstvě. Poté jsou vrcholy ve vrstvě podle přiřazeného mediánu setřízeny. Pokud mají některé z vrcholů u, v ve vrstvě L_i stejnou hodnotu mediánu, je v případě stejné parity vstupního a výstupního stupně vrcholů určena pozice libovolně, jinak je vrchol s lichou paritou vstupního stupně vložen vlevo, druhý z vrcholů pak vpravo.

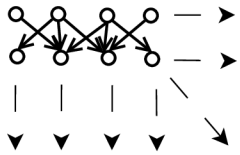
4.5 Určení horizontálních souřadnic

V tomto bodu již máme hladinový graf s takovým uspořádním, aby byl počet křížení co nejmenší. Dále graf obsahuje prázdné vrcholy, které budou těsně před vykreslením nahrazeny za zalomení hran. Vertikální souřadnice vrcholů je již jednoznačně určena vrstvou, ve které se vrchol nachází. Horizontální souřadnici můžeme ponechat jako výchozí pozici ve vrstvě, která vznikla aplikováním popsaných algoritmů. Další možností je jednotlivé vrcholy rozmístit ve vrstvě tak, aby celkový graf působil co nejsymetričtěji. Pokud se vrhneme do takové optimalizace, je hlavním požadavkem zanechání pořadí jednotlivých vrcholů. Většina metod pro takovéto rozmístění je postavena na půlení intervalu. V prvním kroku je nalezena délka maximální vrstvy. V dalším kroku je pro každou vrstvu volán rekurzivní algoritmus rozmísťující v případě lichého počtu prvků vrstvy její středový vrchol a následně rekurzivního volání pro levou a pravou polovinu této vrstvy. Délka maximální vrstvy je pak použita pro určení optimální pozice středového vrcholu.

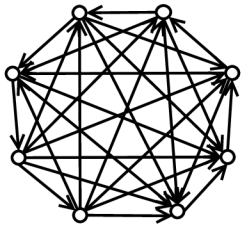
4.6 Testování algoritmů

Již během vývoje výsledné knihovny a aplikace bylo zjištěno, že největší časový interval celkového procesu vykreslení zabírá minimalizace křížení vrcholů. Tomu je v této kapitole

věnována velká část. Pro testování byly s různou složitostí použity tyto grafy.

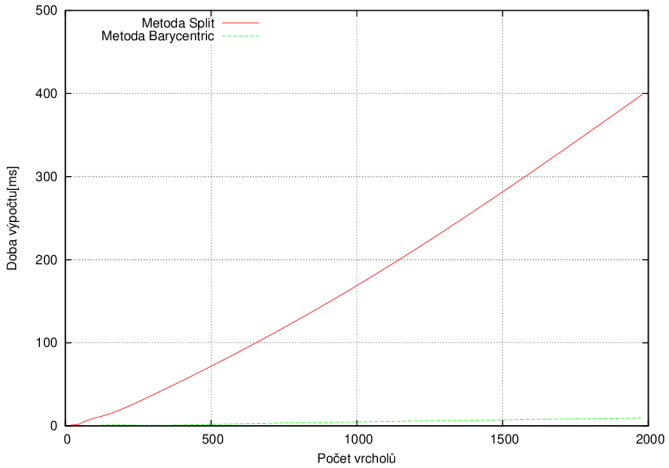


Obrázek 4.2: První testovaný dynamicky vytvářený graf o velikosti n vrcholů. Šipky znázorňují směr rozšiřování grafu související s počtem vrholů.



Obrázek 4.3: Úplný graf k_8, pro test bylo použito několik variant k_n, kde $n > 0$

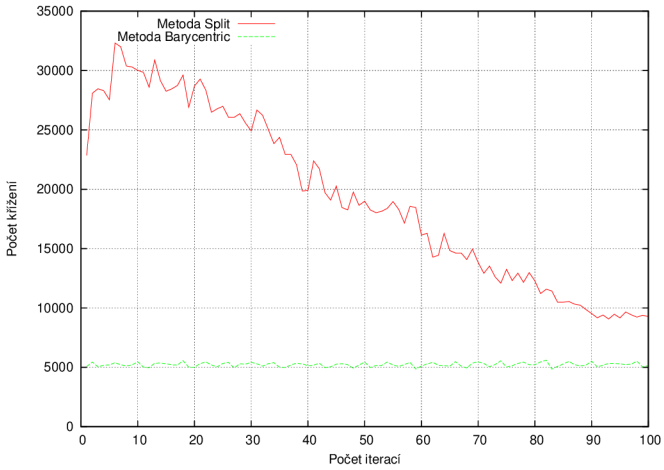
V první fázi testování byla zkoumána závislost počtu vrcholů na celkovou dobu výpočtu dvojicí metod split, barycentric. Tato dvojice byla vybrána v případě metody split jako zastánce rychlých třídících metod, v případě metody barycentric, jako zastánce využívající zajímavou heuristiku.



Obrázek 4.4: Závislost doby výpočtu na počtu vrcholů.

Z grafu 4.4 je zejména pro velký počet vrcholů patrná nevýhoda třídíci metody Split. Při takovémto počtu vrcholů jsou metody založené na heuristice značně efektivnější.

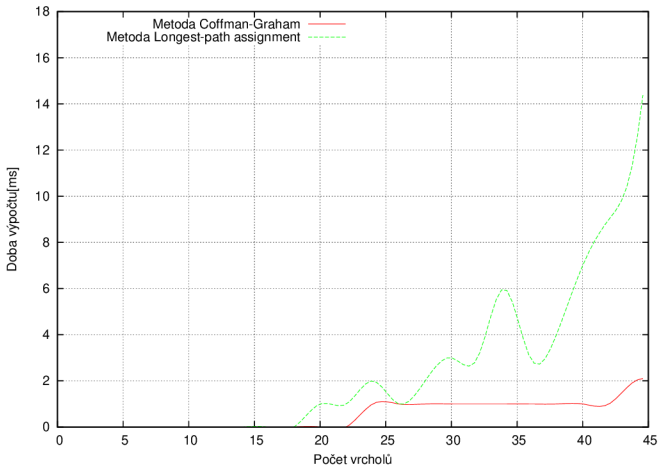
V následující fázi byla zjištěna závislot počtu iterací těchto algoritmů na počtu křížení hran.



Obrázek 4.5: Závislost počtu křížení na počtu iterací.

Pro metodu Barycentric stačilo k dosáhnutí optimálního počtu křížení pouze několik iterací, další iterace již způsobovali kolísání výsledku. Naopak u třídící metody Split bylo potřeba k dosáhnutí tohoto optimální počtu provést značný počet iterací.

Další část testování byla zaměřena na metody určené pro zařazení do vrstev. Testována byla dvojice implementovaných metod Coofman-Graham a Podle nejdelší cesty. Stěžejní byla zejména časová náročnost na počtu vrcholů. Pro metodu Coffman-Graham byla zvolena hraniční šířka $W = 30$.



Obrázek 4.6: Závislost doby výpočtu na počtu vrcholů.

Jak vyplývá z výše uvedeného grafu, i pro malý počet vrcholů je nárůst doby výpočtu vrstev pomocí hledání nejdelší cesty značný. Zejména při velkém počtu hran. Pro zasazení

do vrstev je vždy vhodnější použít algoritmus Coffman-Graham.

Veškeré testy byly prováděny na dvoujádrovém procesoru AMD Turion 64 x2 TL50(1.6GHz, 512KB L2 cache), na systému Debian verze jádra 2.6.18-6-amd64.

Kapitola 5

Možnosti knihovny a testovací aplikace

Výsledná práce byla rozdělena na 2 části, na dynamickou knihovnu pro vykreslování grafů a testovací aplikaci. Obě tyto části byly implementovány v jazyce C++ za použití zásad objektového programování. Knihovna i aplikace je platformě nezávislá, ovšem v OS Windows pouze za předpokladu kompilace zdrojových souborů aplikace společně se soubory knihovny – není možné vytvářet dynamickou knihovnu.

5.0.1 Knihovna GraphDrw

Vlastní knihovnu pro vykreslování grafů jsem nazval GraphDrw, její binární verze je pak podle zvyklostí Unix like systémů nazvána lGraphDrw.so. Knihovna je rozdělena do několika modulů a to tak, aby její použití bylo maximálně intuitivní. Většina těchto modulů obsahuje třídy, jejichž statické metody pracují s hlavní strukturou knihovny popsanou níže.

Moduly knihovny

- **DataStructure**
  Hlavní modul knihovny nesoucí strukturu reprezentující graf a to **Graph**. Z pohledu uživatele knihovny jsou důležité převážně tyto metody
  - **bool LoadFromXml(string Filename, string &Err)** – načtení struktury z *.XML souboru
  - **void SaveToXml(string Filename)** – uložení struktury do *.XML souboru
  - **void AddVertice(Vertice* V)** – přidání vrcholu, existuje i v přetížené verzi s parametrem **int Id**
  - **void AddEdge(Edge* E)** – přidání hrany, existuje v přetížené verzi s parametry **Vertice *From, Vertice *To**
  - **VerticeContr* Layers();** – odkaz na strukturu nesoucí všechny vrstvy, v přetížené verzi s parametrem **int Index** přistupuje k vrstvě na pozici Index
  - **VerticeVector* Vertices();** – odkaz na strukturu nesoucí vrcholy, v přetížené verzi s parametrem **int Index** přistupuje k vrcholu na pozici Index

  Dále struktura obsahuje metody pro zjištění stavu, ve kterém se zrovna nachází, vyčištění struktur atd. Z výše popsaných metod je zřejmé, že k vrcholům lze přistupovat

přímo, nebo po vrstvách. Toho bylo docíleno zéjména z důvodů rozdílnosti jednotlivých algoritmů, které nad strukturou pracují. Zvýšením paměťové náročnosti, která nebyla důležitým kritériem, bylo dosáhnuto vyšší rychlosti většiny algoritmů. Ve skutečnosti je struktura mnohem složitější, zéjmena z důvodů mazání jednotlivých hran, jejich otáčení a nutnosti v posledním kroku vrátit vše do původního stavu.

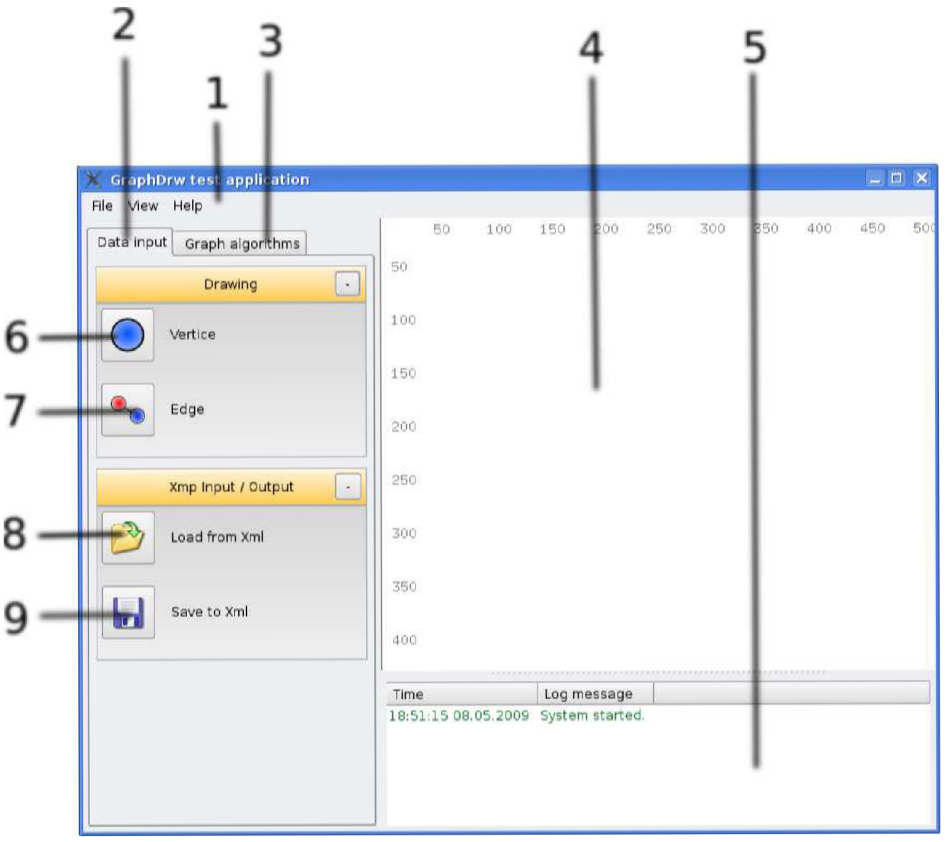
Modul dále obsahuje strukturu **Vertice** reprezentující vrchol. V této struktuře je pro nás zajímavá zejména X-sová a Y-nová souřadnice, což je vlastně výsledek našeho snažení. Reprezentací hrany je v knihovně struktura **Edge**.

- **CyclesRemoval**
  Modul pro odstranění smyček a cyklů z grafu. Výsledkem je předpřipravení pracovních struktur grafu pro další práci. Implementována je metoda Greedy-cycle removal.
- **LayerAssignment**
  Modul nesoucí statické metody implementující zasazení vrcholů do vrstev. Konkrétně Coffman-Graham algoritmus a Podle nejdelší cesty.
- **CrossReduction**
  Modul určený pro minimalizaci křížení, implementovány byly algoritmy Barycentric, Median, Split, Adjancet-Exchange, tj. algoritmy popsané v kapitole 4.
- **XCoordinateAssignment** Modul řešící problém hledání X-sových souřadnic jednotlivích vrcholů v rámci každé vrstvy.
- **SplinePoints** Jak již bylo zmíněno v kapitole 4.3, prázdné vrcholy jsou při finálním kreslení chápany jako zalomení hrany. V případě že bychom chtěli kreslit hrany jako křivky např. jako BSpline křivky, můžeme tyto prázdné vrcholy chápat jako řídící body. SplinePoints je pak modul určený k vytvoření řídících bodů těchto křivek. Jeho výstupem je množina řídících bodů pro každou křivku.

5.0.2 Testovací aplikace

Testovací aplikace je jednoduše nazvána TestApp, jedná se o klasickou GUI aplikaci. Aplikace je postavena na knihovnách wxWidgets. Což je multiplatformní widget toolkit, jehož název vznikl jako zkratka „Windows and X widgets“ – dříve byl známý jako wxWindows. Toolkit je vyvíjen od roku 1992 a v poslední době se dostává do popředí zejména díky své multiplatformosti. Pro kreslení grafů aplikace využívá grafických knihoven OpenGl, u wxWidgetů lze jejich podporu zapnout při jejich kompilaci. Standartní knihovny OpenGl jsou v aplikaci doplněny o GLUT(Open**GL** **U**tility **T**oolkit), který je použit pro kreslení kvadratik.

Při spuštění se nám zobrazí hlavní okno aplikace, které je složeno z následujících částí



Obrázek 5.1: Výchozí okno aplikace

1. Menu aplikace

Hlavní menu aplikace rozdělené do sekcí File, View a Help.

- File
  - **New**
    Nový graf.
  - **Load from XML**
    Načtení grafu z XML.
  - **Save to XML**
    Uložení grafu do XML
  - **Export to *.jpg**
    Export plátna do obrázku formátu jpg.
  - **Save log**
    Uložení logu do textového souboru.
- View

  - **Show log**
    Zobrazení / skrytí záznamu.

- Help
  - **About**
    Zobrazení okna informujícího o aplikaci a jejím autorovi.

2. Stránka kreslení / vstupu / výstupu

Tato stránka obsahuje nástroje pro kreslení vrcholů a hran. Taktéž tlačítka pro rychlou možnost uložení a načtení grafu ze souboru formátu XML.

3. Stránka algoritmů

Stránka obsahující jednoduchého průvodce, který ve správném pořadí aplikuje algoritmy pro vykreslení grafu. Uživateli je vždy nabídnuto zvolit pro aktuální krok jeden z implementovaných algoritmů. V závislosti na kroku, ve kterém se průvodce nachází, mění aplikace způsob vykreslování hran. Taktéž povoluje a zakazuje uživateli jednotlivé akce. Zejména pak zadávání dalších vrcholů a hran ve stavu, kdy průvodce započal svoji činnost a nebyl doposud dokončen. Jedná se o jednoduchý konečný automat.

4. Plátno

Plátno pro kreslení algoritmů, vykreslované pomocí OpenGl. Zde může uživatel nakreslit graf, který bude později zpracován. Plátno lze taktéž podržením levého tlačítka myši posouvat.

5. Záznam

Záznam uchovávající významné události, chyby, a celkové doby výpočtu jednotlivých algoritmů.

6. Nástroj „vrchol“

Nástroj pro kreslení vrcholů. Zrušení nástroje provedeme klávesou Esc, případně vybráním nástroje jiného.

7. Nástroj „hrana“

Nástroj pro kreslení hran.

8. Načtení grafu z XML

Stejná funkcionalita jako u tlačítka v menu.

9. Uložení grafu do XML

Stejná funkcionalita jako u tlačítka v menu.

Kapitola 6

Závěr

Při studiu literatury jsem se seznámil s problematikou vykreslování grafů. Po zvážení mnoha kritérií jsem se zaměřil na vrstevné kreslení, se zaměřením na maximální obecnost. Metody tohoto kreslení očekávaly na vstupu acyklický graf, proto jsem nastudoval i možnosti transformace cyklického grafu na graf acyklický. Kreslení acyklického grafu pak probíha v několika krocích, které lze nejobecněji shrnout jako: zasazení vrcholů do vrstev, minimalizaci křížení hran, určení horizontálních souřadnic.

Pro většinu kroků kreslení jsem implementoval několik metod, zejména pro krok minimalizace křížení počtu hran. Vybrané metody jsem testoval na dynamicky vytvářených grafech. Zaměřil jsem se zejména na celkovou dobu výpočtu dané metody. Pro zasazení do vrstev z testů vyplynula jasná převaha metody Coffman-Graham nad medotou hledání nejdelší cesty. V případě minimalizace křížení vrcholů byla zřejmá převaha metod využívajících heuristiku nad metodami třídícími.

Všechny výše popsané metody v této práci jsem implementoval do dynamické knihovny „GraphDrw“. Pro testování knihovny, zadávání a zobrazovaní grafů jsem implementoval testovací aplikaci postavenou na grafických knihovnách OpenGL.

Při vykreslování grafů nebyla brána v potaz velikost vrcholů, možným rozšířením by bylo doplnění vykreslování o rozlišnou velikost vrcholů, např. reprezentace vrcholů jako obdélníků nesoucího text a zahrnutí této proměnné délky do algoritmů. Dalším možným rozšířením by mohla být v textu zmíněná optimalizace počtu prázdných vrcholů, která zejména pro třídící metody značně zvyšuje dobu celkového výpočtu. Po provedení těchto optimalizací by se testovací aplikace mohla stát silnějším vizualizačním nástrojem. Pro dosáhnutí kvalit profesionálních vizualizačních nástrojů např. [5] by bylo zapotřebí značného vývoje.

Následuje zhodnocení jednotlivých kroků zadání práce.

- **Seznamte se s problematikou vizualizace grafů v dvourozměrném prostoru.**
  Problematiku jsem popsal v kapitole č. 3. Zaměřil jsem na hierarchický přístup kreslení.
- **Zvolte několik (minimálně tři) metod řešících tuto problematiku a podrobně si je nastudujte.**
  Zvolil jsem několik metod pro každý krok, důkladně jsem tyto metody popsal v kapitole 4.
- **Navrhněte a implementujte program provádějící výpočet rozvržení zadaného grafu a aplikaci zobrazující tento graf.**

Implementoval jsem dynamickou knihovnu řešící vykreslování grafů, dále jsem vytvořil testovací aplikaci, kde jsem tuto knihovnu použil.

- **Navrhněte vhodné rozšíření nastudovaných algoritmů a případně tyto rozšíření také implementujte.**
  V závěru jsem navrl možná neimplementovaná rozšíření, knihovnu jsem dopnil o modul používající ke kreslení hran NURBS křivky.
- **Zhodnoťte dosažené výsledky a porovnejte jednotlivé zvolené metody**
  Vybrané metody jsem otestoval a zhodnotil v kapitole 4.6.

Literatura

[1] P.Eades, K.Sugiama: *How to draw a Directed Graph, Journal of Information Processing, Vol. 13.* Information Processing Society of Japan, 1990, iSSN 0387-6101.

[2] Takao Nishizeki, Md. Saidur Rahman: *Planar graph drawing.* 2004, iSBN 981-256-033-5.
URL http://books.google.com/books?id=upIg1jGpfioC&printsec=frontcover&dq=graph+drawing&hl=cs

[3] WWW stránky: Graph drawing. 2009, [Online; navštíveno 08. 17. 2009].
URL http://en.wikipedia.org/wiki/Graph_drawing

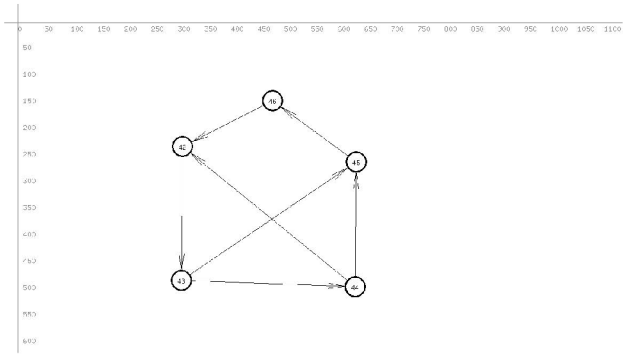
[4] WWW stránky: Graph drawing symposium. 2009, [Online; navštíveno 08. 17. 2009].
URL http://graphdrawing.org/index.html

[5] WWW stránky: Graphviz - Graph Visualization Software. 2009, [Online; navštíveno 08. 17. 2009].
URL http://www.graphviz.org/

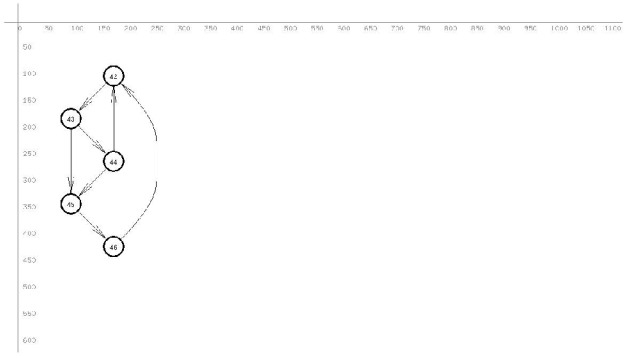
[6] WWW stránky: Journal of Graph Algorithms and Applications. 2009, [Online; navštíveno 08. 17. 2009].
URL http://jgaa.info

Dodatek A

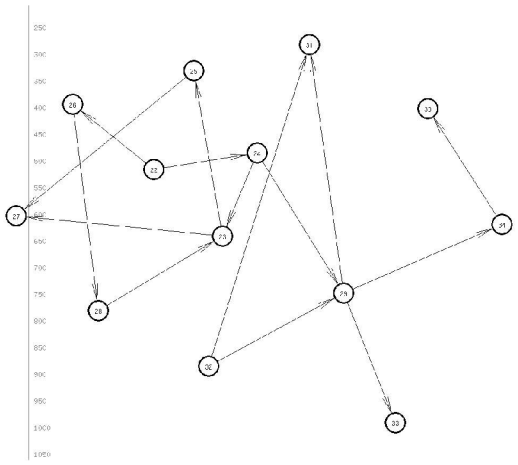
Ukázky vstupu a výstupu testovací aplikace



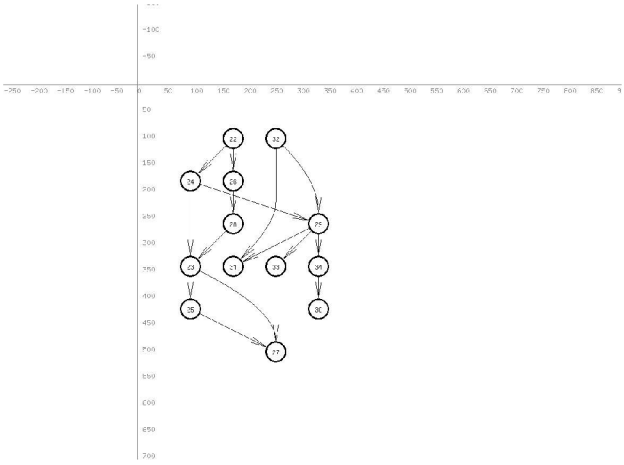
Obrázek A.1: Počáteční vzhled prvního ukázkového grafu



Obrázek A.2: Konečený vzhled prvního ukázkového grafu



Obrázek A.3: Počáteční vzhled druhého ukázkového grafu



Obrázek A.4: Konečený vzhled druhého ukázkového grafu