ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Programování v jazyce C

Semestrální práce

Č.3: Nástroj pro řešení úloh lineárního programování

Antonín Hobl A23B0306P

3. ledna 2025

Obsah

Kapitola 1

Zadání

ZADÁNÍ SEMESTRÁLNÍ PRÁCE
Nástroj pro řešení úloh lineárního programování

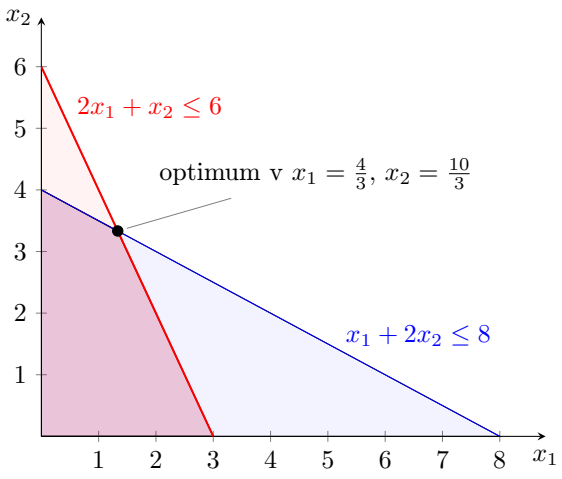
Na moment si představte, že jste vedoucí v podniku, který generuje dva produkty: A a B. Ty je třeba vyrobit a zabalit. Výroba produktu A trvá 1 hodinu a balení další 2 hodiny, zatímco výroba produktu B trvá 2 hodiny a balení 1 hodinu. Pracovní doba výrobního oddělení je 8 hodin. Pracovníci balírny si ovšem se svým vedoucím kvůli náročnému úklidu domluvili 6hodinovou pracovní dobu. Zisk za jeden kus produktu A jsou 3 jednotky, za produkt B inkasuje podnik 2 jednotky. Vaším úkolem je zjistit, kolik kusů produktu A a B by podnik měl vyrobit a zabalit, aby maximalizoval zisk. Grafické znázornění úlohy je ukázáno na obrázku 1.

V této úloze lineárního programování tedy maximalizujeme účelovou funkci

$$z = 3x_1 + 2x_2,$$

kde x_1, resp. x_2 jsou počty kusů vytvořených produktů A, resp. B. Funkci optimalizujeme s ohledem na omezující podmínky:

$$\begin{aligned} x_1 + 2x_2 &\leq 8 \text{ (výrobní čas)}, \\ 2x_1 + x_2 &\leq 6 \text{ (čas balení)}, \\ x_1 &\geq 0, \\ \text{a } x_2 &\geq 0. \end{aligned}$$



Obrázek 1: Grafické znázornění úlohy lineárního programování.

Pokud bychom v této úloze nepřipouštěli rozdělané výrobky na konci směny, tj. hledané řešení musí být celočíselné, jednalo by se o úlohu celočíselného lineárního programování, která je ovšem NP-těžká. V této semestrální práci se budeme držet při zemi: budeme řešit úlohy lineární optimalizace s reálným řešením a pouze jednou účelovou funkcí.

Pomocí lineárního programování je možné řešit takřka nekonečné množství reálných problémů. Od optimalizace procesů, dopravních sítí, přidělování vysílacích frekvencí, logistiky, plánování zdrojů, genomiky a mnoho dalšího.

Zadání

Naprogramujte v jazyce ANSI C přenositelnou[1] **konzolovou aplikaci**, která bude řešit úlohy lineárního programování zadané ve zjednodušeném formátu LP.

Program bude spouštěn příkazem `lp.exe`[2] s kombinací následujících argumentů – výrazy v lomených závorkách (`<>`), resp. hranatých závorkách (`[]`) označují povinné, resp. nepovinné argumenty (příklad spuštění programu je uveden v ukázce konzolového výstupu 1):

`<input-file>`	Soubor s popisem úlohy ve formátu LP. V případě, že uživatel zadá neexistující soubor, program vypíše chybové hlášení `"Input file not found!\n"` a vrátí hodnotu 1.
`-o <path>`	Výstupní soubor s řešením úlohy. Pokud umístění neexistuje, bude vypsáno hlášení `"Invalid output destination!\n"` a program skončí s návratovou hodnotou 2. V případě, že uživatel tento přepínač nezadá, bude výsledek optimalizace vypsán na obrazovku. Do tohoto souboru neuvádějte chybová hlášení.
`--output <path>`	Stejné jako v případě přepínače `-o`. Použití obou přepínačů `-o` a `--output` není chybou, program pak bude akceptovat poslední zadanou hodnotu.

V případě nalezení konečného optimálního řešení úlohy program vrátí hodnotu `EXIT_SUCCESS`. Chybové stavy týkající se zpracování vstupních souborů nebo samotného algoritmu optimalizace jsou popsány v dalších sekcích.

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 (LATEX). Bude-li některá z částí chybět, kontrolní skript Vaši práci odmítne.

Specifikace vstupních souborů (formát LP)

Pro zachycení optimalizačního modelu bude program používat redukovanou a zobecněnou verzi formátu LP, který je popsán v [1]. Vstupní soubory mohou obsahovat následující sekce:

`Maximize/Minimize`	Výraz uvozující řádek se zápisem optimalizované účelové funkce. Program musí být schopen zpracovat standardní operátory `+`, `-`, `*`, `=` nebo závorky `()`, `[]` a `{}`. Oproti originální verzi ovšem nevyžadujeme, aby jednotlivé operandy a operátory byly v matematických výrazech striktně odděleny mezerou (to platí i v ostatních sekcích souboru). Názvy proměnných tedy dříve uvedené operátory a závorky obsahovat nesmí. Při násobení není nutné použít operátor `*`, například `"2.5z"` značí *2,5 krát* z, zatímco `"z2"` je pouze název proměnné.
`Subject To`	Sekce obsahující seznam podmínek ve formátu `"<název>: <výraz>"`. Navíc oproti účelové funkci mohou podmínky obsahovat porovnávací operátory `<`, `>`, `<=` a `>=`.

[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/7/8/10/11) a s běžnými distribucemi Linuxu (např. Ubuntu, Debian, Red Hat, atp.). Server, na který budete vaši práci odevzdávat a který ji otestuje, má nainstalovaný operační systém Debian GNU/Linux 11 (bullseye) s jádrem verze 5.10.0-28-amd64 a s překladačem gcc 10.2.1.

[2] Přípona `.exe` je povinná i při sestavení pro Linux, zejména při automatické kontrole validačním systémem.

`Bounds`	Omezení hodnot rozhodovacích proměnných. V této sekci jsou povoleny pouze porovnávací operátory uvedené výše.
`Generals`	Obsahuje seznam použitých rozhodovacích proměnných oddělených znakem mezery. Pokud je v souboru nalezena proměnná, která v této sekci není uvedena, program skončí s chybovou hláškou `"Unknown variable '<j>'!\n"`, kde `<j>` je neznámá proměnná, a návratovou hodnotou 10. Pokud sekce obsahuje nepoužitou rozhodovací proměnnou `<n>`, program vypíše pouze varování `"Warning: unused variable '<n>'!\n"`.
`End`	Uvozuje konec souboru, tzn. že se vyskytuje vždy jako poslední a sekce uvedené za ním jsou syntaktickou chybou.

Až na návěští `End` není pořadí jednotlivých sekcí fixní. Na výskyt neplatných operátorů, neznámých sekcí a jiných problémů program reaguje vypsáním chybového hlášení `"Syntax error!\n"` a skončí s návratovou hodnotou 11. Komentáře v souboru jsou uvozeny znakem `"\"`. Ukázku vstupního souboru si můžete prohlédnout v konzolovém rozhraní 1 na straně 4.

Optimalizační algoritmus

Při analýze úlohy jistě narazíte na problémy degenerovaných úloh a jiné, které budeme pro jednoduchost ignorovat. Algoritmus hledání optimálního řešení úlohy lineárního programování může tedy teoreticky skončit následovně.

1. **Nalezení konečného optimálního řešení**
   V takovém případě program vypíše optimální hodnoty rozhodovacích proměnných a skončí s návratovou hodnotou `EXIT_SUCCESS` (viz ukázka konzolového rozhraní 1). Optimálních řešení může být více, s čímž validační skript počítá.

2. **Úloha je neomezená**
   Účelová funkce může nabývat libovolně velkých hodnot, aniž by porušila některou z omezujících podmínek, tj. optimum je v nekonečnu. Program na tuto skutečnost upozorní chybovým hlášením `"Objective function is unbounded.\n"` a skončí s návratovou hodnotou 20.

3. **Neexistence přípustného řešení**
   Soustava omezení nemá žádnou společnou přípustnou oblast, tj. neexistuje žádný bod, který by vyhovoval všem omezením současně – úloha je nesplnitelná. V takovém případě program vypíše chybovou hláškou `"No feasible solution exists.\n"` a vrátí hodnotu 21.

Užitečné techniky a odkazy

[1] **Specifikace formátu LP**
`https://www.gurobi.com/documentation/current/refman/lp_format.html`

[2] **Algoritmus Shunting yard**
`https://en.wikipedia.org/wiki/Shunting_yard_algorithm`

[3] **Lineární programování** (2. kapitola, str. 18)
`http://najada.fav.zcu.cz/~ryjacek/students/ps/TGD2.pdf`

[4] **Simplexový algoritmus**
`https://en.wikipedia.org/wiki/Simplex_algorithm`

Řešení úlohy je zcela ve vaší kompetenci – uvedené dokumenty je možné využít při řešení úlohy, ale můžete zvolit libovolné algoritmy a techniky, které podle vás nejlépe povedou k cíli.

Přílohy

Konzolové rozhraní 1: Ukázka sestavení a činnosti programu `lp.exe`.

```
user@machine:~$ cd pc && ls
doc src CMakeLists.txt dokumentace.pdf Makefile Makefile.win
user@machine:~/pc$ make &>/dev/null && ls
build doc src CMakeLists.txt dokumentace.pdf Makefile Makefile.win lp.exe
user@machine:~/pc$ ./lp.exe
Input file not found!
user@machine:~/pc$ echo $?
1
user@machine:~/pc$ cat ../vyroba.lp
\ uloha z uvodu zadani
Subject To \ poradi neni fixni
    vyroba: x_1 + 2x_2 <= 8
    baleni: 2 *x_1  + 1 * x_2    <= 6
Maximize
    3x_1+   2 * x_2  \ ucelova funkce
Generals
    x_2 x_1 x_3
Bounds
    0 <= x_1
    0 <= x_2
End
user@machine:~/pc$ ./lp.exe ../vyroba.lp
Warning: unused variable 'x_3'!
x_1 = 1.3333
x_2 = 3.3333
user@machine:~/pc$ echo $?
0
user@machine:~/pc$ ./lp.exe --output vystup_1.txt ../vyroba.lp -o vystup_2.txt
Warning: unused variable 'x_3'!
user@machine:~/pc$ echo $?
0
user@machine:~/pc$ ls
build doc src CMakeLists.txt dokumentace.pdf Makefile Makefile.win lp.exe
vystup_2.txt
user@machine:~/pc$ cat vystup_2.txt
x_1 = 1.3333
x_2 = 3.3333
user@machine:~/pc$ .lp.exe -o ./out_dir/vystup_3.txt ../vyroba.lp
Invalid output destination!
user@machine:~/pc$ echo $?
2
user@machine:~/pc$ sed -i '1,5d' ../vyroba.lp  # odstraneni prvnich 5 radek
user@machine:~/pc$ ./lp.exe --output vystup_1.txt ../vyroba.lp
Syntax error!
user@machine:~/pc$ echo $?
11
user@machine:~/pc$ ls  # soubor "vystup_1.txt" nevznikl
build doc src CMakeLists.txt dokumentace.pdf Makefile Makefile.win lp.exe
vystup_2.txt
user@machine:~/pc$
```

Kapitola 2

Analýza úlohy

V této kapitole bych chtěl popsat problémy, které přináší mnou vybrané zadání, nastínit možnosti jejich řešení a ukázat, proč jsem si vybral to jedno konkrétní.

2.1 Zpracování souboru

Jako první problém nastalo samotné zpracování souboru. Problém byl, jak ho vlastně zpracovat, v jakém pořadí zpracovávat sekce, nebo jakým způsobem jeho obsah vhodně uložit. Na stole jsem měl následnující možnosti.

2.1.1 Zpracovávání souboru rovnou

Jako první možnost bylo zpracovat soubor rovnou, to znamená postupně číst jeho řádky a z jednotlivých sekcí rovnou parsovat důležitá data a ukládat je do struktur.

Tahle možnost se zdá jako nejjednodušší, ale přináší s sebou problémy. Hlavní problém je, že nemáme zaručeno pořadí jednotlivých sekcí, to znamená, že pokud bychom se spoléhaly, že například sekce Generals bude jako první, tak by námi zvolený způsob nefungoval. Přesně tohle je hlavní problém tohoto řešení, protože na parsování dalších sekcí je potřeba základní seznam proměnných, tedy sekce Generals. My ale nemáme zaručeno, že tato sekce bude v souboru jako první. Tento fakt velmi ztěžuje tuto možnost zpracování souboru a proto jsem se rozhodl ji nepoužít.

2.1.2 Zpracovat nejdříve generals

Druhá možnost byla zpracovat nejdříve sekci Generals a poté zbytek souboru. To znamená přečíst soubor jednou a vyparsovat z něj pouze sekci Generals,

kvůli seznamu proměnných, a uložit jej do struktury. Následně přečíst soubor znovu a zpracovat zbylé sekce souboru, kdy už bychom měli k dispozici seznam proměnných.

Tahle možnost je lepší, neboť při zpracování zbylých sekcí budeme mít k dispozici nutný seznam proměnných. Rozhodl jsem se ale tuto možnost zamítnout a to proto, že bych byl nucen číst soubor dvakrát a to mi přijde zbytečné.

2.1.3 Rozdělit soubor podle sekcí

Poslední možnost byla rozdělit soubor podle sekcí do struktury. To znamená číst soubor po řádkách a postupně, podle toho, na jaké sekce program narazí ukládat řádky jednotlivých sekcí do struktury, kde už bylo to bylo obsahově seřazeno podle sekcí. Následně by bylo možné jednotlivé sekce zpracovat v libovolném pořadí a to bez nutnosti procházet soubor vícekrát. Tímhle máme zaručeno, že sekci Generals můžeme zpracovat jako první a při zpracování následujících sekcí budeme mít k dispozici seznam proměnných.

Tento způsob je sice složitější, ale nabízí velkou výhodu, a to zpracování sekcí v libovolném pořadí. Z toho důvodu jsem tento způsob zvolil ve své implementaci.

2.2 Zpracování aritmetického výrazu

2.3 Optimalitační algoritmus

Kapitola 3

Popis implementace

Kapitola 4

Uživatelská příručka

Kapitola 5

Závěr