Dokumentace k semestrální práci
KIV/PT 2020/2021

JAN RÁDL, BARBORA VOBOŘILOVÁ

Obsah

Zadání semestrální práce

Zadání je určeno pro **dva** studenty. Práce zahrnuje dvě dílčí části — vytvoření funkčního programu diskrétní simulace a napsání strukturované dokumentace.

Zadání:

Společnost „Dřevák a syn“ vlastní po celém světě síť D dřevozpracujících továren. Snaha společnosti je minimalizovat cenu přepravy výrobků z továren do supermarketů. Starý Dřevák nikdy nešel příliš s dobou, co se kam má přepravit (samozřejmě aby to vyšlo nejlevněji), vždy počítal na papír. Nyní ale převzal společnost mladý Luboš Dřevák, který si na to chce napsat software a místo celovečerního počítání raději sledovat TokTik nebo hrát GallFuys přes Diskord s přáteli.

Všechny dřevozpracující továrny dokáží produkovat Z druhů různého zboží, od skříní přes dekorace až po záchodová prkénka. Své výrobky rozváží do S supermarketů, nejčastěji do supermarketů BOBI a PATu, ale ta prkénka chtějí i v CUUPu. Vedení každého supermarketu s je však náročné, zvláště pak v BOBI na Borech (který mají interně zařazený jako pátý v pořadí, tedy $s = 5$, kde to vypadá, jako by management generoval poptávku náhodně) a každý den t hodlá daný supermarket prodat jiný počet zboží z (hodnotu poptávaného počtu zboží značíme $r_{s,z,t}$) svým zákazníkům. S továrnami to také snadné není, protože každá továrna d dokáže vyprodukovat pouze omezený počet produktů daného typu $p_{d,z,t}$ v závislosti na tom, kolik jim dodavatelé daný den zrovna (například kvůli koronaviru) dodají materiálu a také jak se zrovna chce zaměstnancům do práce. V továrně $d = 2$ v Mostu se navíc poslední dobou často stávkuje. Cena převozu zboží z továrny do supermarketu je dána pouze továrnou a supermarketem (jeden kus libovolného zboží lze převézt mezi dvěma uzly s konstantní cenou $c_{s,d}$, tedy například převoz jednoho záchodového prkénka, dvou matrjošek a dvou skříní z továrny v Mostu do supermarketu BOBI na Borech stojí $5c_{5,2}$).

Mladík však převzal společnost po otci náhle (otec mu řekl, že to nedokáže lépe, protože už to dělá léta, at’ se tedy ukáže), a v supermarketech tedy zbyly zásoby zboží $q_{s,z}$. Pomozme Lubošovi naplánovat přepravu zboží pro T dní dopředu. Luboš zná:

D – počet továren,

S – počet supermarketů,

Z – počet druhů zboží,

T – počet dní,

$c_{s,d}$ – cenu převozu jednoho kusu libovolného zboží z továrny d do supermarketu s,

$q_{z,s}$ – počáteční zásoby výrobku z v supermarketu s.

$p_{d,z,t}$ – produkci továrnou d druhu zboží z v den t,

$r_{s,z,t}$ – poptávku zákazníků po zboží z v supermarketu s dne t,

a chce znát $k_{s,d,z,t}$ pro $\forall s \in \{1,\dots,S\}, d \in \{1,\dots,D\}, z \in \{1,\dots,Z\}$ a $t \in \{1,\dots,T\}$, tedy počty kusů všech druhů zboží z rozvezených ze všech továren d do všech supermarketů s každý den t. Pokud by se stalo, že továrny nedokáží uzásobit supermarkety, Luboš potřebuje znát, kdy to bude (den t), aby doobjednal zboží z Číny.

Supermarkety mají povoleno prediktivní plánování s neomezenými zásobami (někteří tomu mohou říkat „křečkování“) – lze tedy předzásobit supermarket dle dat poptávky z „budoucna“. Jestli toho využijete či nikoliv je na Vás, je ale třeba toto uvést do dokumentace (viz dále).

Vytvoření funkčního programu:

- Seznamte se se strukturou vstupních dat (počty, ceny cest, objem poptávek...) a načtěte je do svého programu. Formát souborů je popsán přímo v záhlaví vstupních souborů **(5b.)**.
- Navrhněte vhodné datové struktury pro reprezentaci vstupních dat, zvažujte časovou a paměťovou náročnost algoritmů pracujících s danými strukturami **(10b.)**.
- Proveďte základní simulaci pro první den, vypište celkovou cenu přepravy prvého dne pomocí výrazu $\sum_{s=1}^{S}\sum_{d=1}^{D}(c_{s,d}\sum_{z=1}^{Z}k_{s,d,z,1})$ **(10b.)**.

Výše popsaná část bude váš minimální výstup při kontrolním cvičení cca v polovině semestru.

- Vytvořte prostředí pro snadnou obsluhu programu (menu, ošetření vstupů) – nemusí být grafické, umožněte manuální zadání požadavku **(5b.)**,
- umožněte sledování (za běhu simulace) aktuálního stavu přepravy různých druhů zboží a dopravního prostředku – (dopravní prostředky jsou libovolné, např. nákladní vůz, vlak, loď, lamy, saně apod.) **(5b.)**,
- proveďte simulaci pro zadaný počet dnů a vygenerujte do souborů následující statistiky (uložte je do vhodných souborů **(10b.)**:
  - přehled jednotlivých továren – rozpis, co kam a kdy produkovaly, kolik zboží vyprodukovaly zbytečně
  - přehled jednotlivých supermarketů – každodenní rozpis skladových zásob
  - celková cena přepravy $\sum_{s=1}^{S}\sum_{d=1}^{D}(c_{s,d}\sum_{z=1}^{Z}k_{s,d,z,1})$
  - celková doba běhu simulace
  - nemá-li úloha řešení, vypište, který den již nebylo možné supermarkety uzásobit a kolik zboží kde chybělo.
- Vytvořte generátor vlastních dat. Generátor bude generovat vstupní data pomocí rovnoměrného, normálního nebo extremálního rozdělení (zvolte jedno z uvedených rozdělení, podrobnosti poskytnou cvičící na cvičeních) s vhodnými parametry. Data budou generována do souboru (nebudou přímo použita programem) o stejném formátu jako již dodané vstupní soubory. Při odevzdání přiložte jeden dataset s řešitelnou úlohou a jeden dataset, kdy se nepodaří supermarkety uzásobit **(5b.)**.
- Vytvořte dokumentační komentáře ve zdrojovém textu programu a vygenerujte programovou dokumentaci (Javadoc) **(10b.)**,
- vytvořte kvalitní dále rozšiřitelný kód – pro kontrolu použijte softwarový nástroj PMD (více na http://www.kiv.zcu.cz/~herout/pruzkumy/pmd/pmd.html), soubor s pravidly pmdrules.xml najdete na portálu v podmenu Samostatná práce **(10b.)**
  - mínus 1 bod za vážnější chybu, při 6 a více chybách nutno opravit,
  - mínus 2 body za 10 a více drobných chyb.

V rámci dokumentace:

- připojte zadání **(1b.)**,
- popište analýzu problému **(6b.)**,
- popište návrh programu (např. jednoduchý UML diagram) **(6b.)**,
- vytvořte uživatelskou dokumentaci **(5b.)**,
- zhodnoťte celou práci, vytvořte závěr **(2b.)**.

Analýza problému

Dle zadání supermarkety zpracovávají pouze zboží z vlastních zásob a když tyto zásoby nemají tak požadují zboží po továrnách. Jednotlivé požadavky spolu nesouvisejí, cesty se po čas simulace nemění a vždy je cesta mezi kteroukoliv továrnou a kterýmkoliv supermarketem, proto není potřeba použít strukturu grafu. Pro tento případ stačí každému supermarketu pouze pole cest do továren seřazené podle jejich cenové náročnosti. Současně díky tomu můžeme využít statická pole.

Výrobu zboží v továrnách lze jednoduše realizovat maticí vyrobených kusů v dané dny. A pouze pro vybraný den překopírovat hodnoty do dočasného pole určeného k expedici.

Objednávky jsou rozděleny po dnech a uloženy ve spojovém seznamu, který je sice více náročný na paměť, ale velice výhodný pro lineární zpracování dat. Pro zpracovaní jsme se rozhodli využít algoritmus z rodiny online greedy algoritmů. Pro jeho jednoduchost implementace a intuitivního průběhu programu, a také proto, že není vyžadováno, aby simulace běžela co nejdéle, nemusíme vymýšlet složité rozhodování kam s nespotřebovaným zbožím.

Pro generovaní statistik a celkově ladění je využit druh logu, kde jsou uloženy všechny informace o běhu simulace.

Návrh programu

Program jako takový je rozdělen do tří částí. První část se stará o reprezentaci dat, druhá o simulaci a třetí se stará o události během simulace.

Reprezentace dat

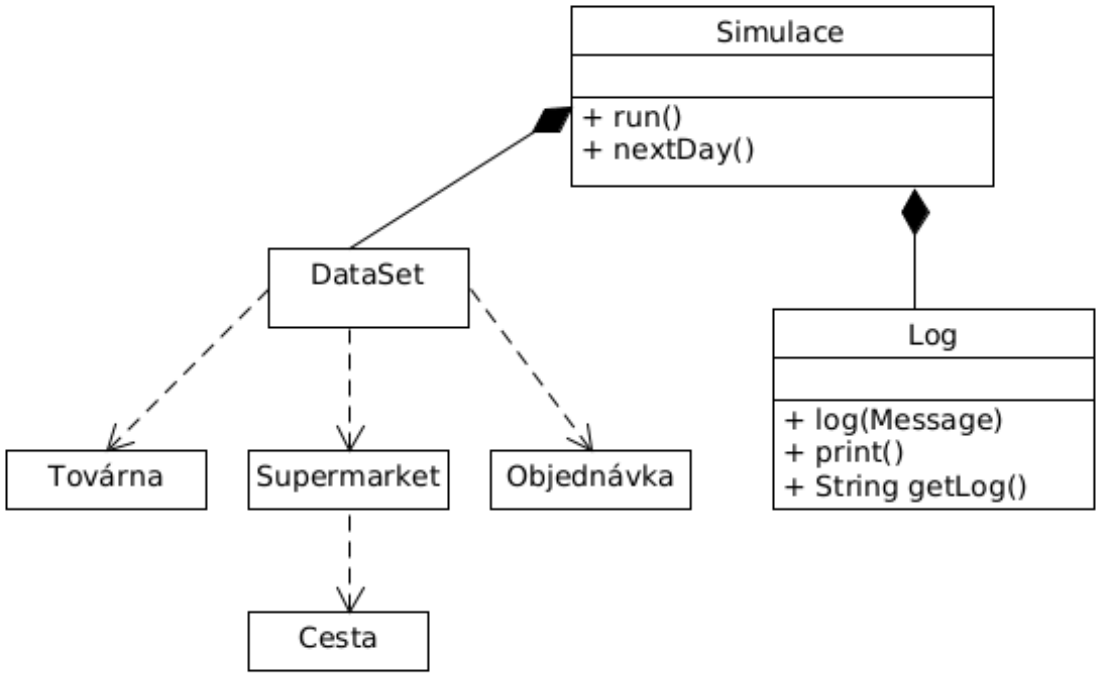
Reprezentace dat je navržená tak, aby co nejvíce odpovídala reálnému světu. Proto máme třídy reprezentující továrny, supermarkety, cesty a objednávky. Instance těchto tříd jsou pak seskupeny v objektu *DataSet*. *DataSet* je navržený tak, aby byl přístup k datům pro simulaci co nejjednodušší a zároveň byl i kód přehledný.

Simulace

Třídy umožňující simulaci jsou už o něco složitější, máme zde totiž dvě možnosti, jak simulaci spustit. Třída *GreedSimulace* spustí automatickou simulaci, která proběhne bez zásahu uživatele a vyhodnotí data uložená v datasetu, zatímco dekorátor *RucniSimulace* umožňuje uživateli upravovat běh simulace a zasahovat do něj. Proto je vytvořený interface *Simulace,* který umožňuje pracovat s oběma simulacemi stejně a mít tak jednotný kód pro spuštění obou běhů.

Události

O ukládání událostí, přístup k nim a následný snadný výpis se stará část pojmenovaná *"*log". Log je vlastně takový seznam zpráv, které můžou mít různou úroveň. Může jít o zprávy informativní, týkající se skladů a zásobování, nebo o upozornění. Během běhu simulace se vytváří mnoho logů, se kterými se pak dále pracuje a je možné z nich vypsat na konci simulace vypsat statistiky.



Obrázek 1: Přibližný návrh UML

Uživatelská dokumentace

Spuštění programu

Program se spouští třídou *Main* přes příkazovou řádku:

- s parametrem *-f "název souboru" -g*, pro vygenerování dat se zadaným názvem,
- nebo s parametrem *-f "adresa souboru",* pro spuštění simulace.

U spuštění simulace je dále možné použít parametr *-h (-H)* pro spuštění ručně moderované simulace do které je možné zasahovat.

Třída *Main* nejprve pro kontrolu vypíše zadané parametry a poté spustí část programu, kterou si vyžádal uživatel dle parametrů.

Generátor dat

Generátor dat generuje vstupní data pomocí normálního rozdělení, za použití metody *nextGaussian()* třídy *java.util.Random.*

Maximální počet továren a supermarketů je 50, druhů zboží může být maximálně 100 a maximální počet dní, po které simulace běží, je 60. Je zaručeno že tyto hodnoty budou větší než 0.
Cena převozu zboží nabývá hodnot v intervalu $\langle 0; 5 \rangle$. Počáteční skladové zásoby, produkce továren a poptávka zboží jsou vždy v intervalu $\langle 0; 100 \rangle$.

Vygenerovaná data se zapíšou do souboru, který se pojmenuje podle názvu zadaného jako parametr při spuštění programu. V případě, že existuje soubor se stejným jménem, generátor ho přepíše.

Spuštění simulace

Automatická simulace

Pokud uživatel použije pouze parametr *-f* a adresou souboru, spustí se automatická simulace, do které nemusí uživatel nijak zasahovat. Simulace se pouze na konci zeptá, jestli má uživatel zájem o vypsání logu a zápis statistik do souborů.

Ručně řízená simulace

V případě spuštění programu s parametrem *-h* se uživatel dostane do ručně řízené simulace. Pro vypsání nápovědy slouží příkaz *help*. Uživatel simulaci ovládá pomocí příkazů.

- *next [cislo]* – posune simulaci o zadaný počet dní dopředu,
- *exit* – ukončí simulaci,
- *run* – spustí běh simulace až do konce, je možné zadávat příkazy,
- *end* – dokončí běh simulace, bez možnosti do ní zasahovat,
- *objednávka* – umožní zadání nové objednávky mimo vložená data,
- *info [hledaný výraz]* – vyhledá výskyt výrazu v logu událostí,
- *log* – vypíše všechny dosavadní události uložené v logu.

Po doběhnutí simulace je opět možné vypsat log a zapsat statistiky do souborů.

Vyhledávání v logu

Při běhu ručně řízené simulace, je možné vyhledávat v logu událostí příkazem *info [hledaný výraz].*

Vyhledání lze použít speciálních výrazů, jako je například: “den 1“ pro nalezení logu prvního dne, “zbytky den 1“ pro nalezení logu s informací o přebytečné výrobě prvního dne nebo “zásoby den 1“ pro zjištění zásob supermarketů v prvním dni. Stejně lze také použít “továrna 2 vyhozeno“, které zobrazí vyhozené kusy zboží pro každý proběhlý den, kdy továrna něco vyhodila.

Výrazy lze řetězit pomocí znaku & např. “den 1 & továrna 0“ vyhledá informace ze dne 1 týkající se továrny s id 0.

Přidání objednávky

Pro přidání nové objednávky bude program po uživateli chtít zadat číslo objednávaného zboží, množství a id supermarketu, který zboží požaduje.

Pro správné přidání objednávky do systému je nutné zadat i platné číslo dne, které není větší než celková doba trvání. Zároveň není možné zadávat objednávky zpětně do dnů, které už proběhly.

Závěr

Dle našeho mínění výsledná práce splňuje zadání, avšak má i několik nevýhod. Jednou z nich je použití greedy algoritmu, který neumí zásobovat supermarkety nad rámec jejich objednávek nebo ukládat vyrobené zboží v továrnách a přebytečné výrobky zahazuje. Vyhledávání v logu může být občas také trochu problematické, protože může být pomalé, což ale v našem případě není velký problém, jelikož prohledávané zprávy jsou krátké.