Diskrétna simulácia

# Zadanie práce

Vedúci vývojového tímu je zodpovedný za vedenie a riadenie zákazníckeho projektu, ktorý sa skladáz niekoľkých navzájom závislých činností. Aktivity projektu a ich vzájomný vzťah je možné zapísať prostredníctvom nasledovného sieťového grafu.

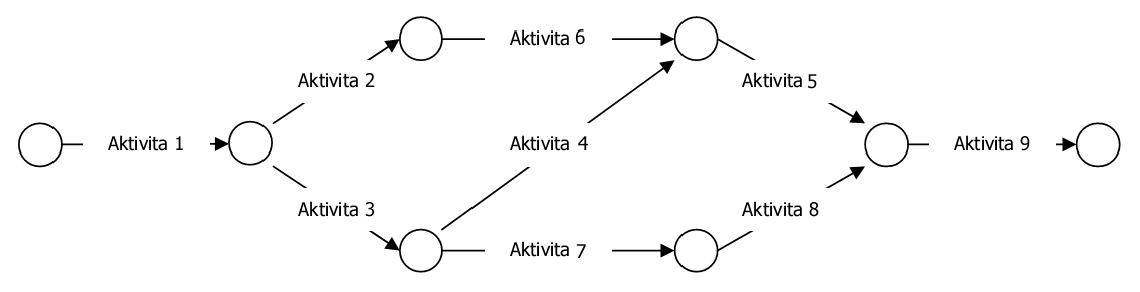
Vo vrchole do ktorého vchádza hrana s aktivitou 3 dôjde k rozhodnutiu, či sa bude v projekte realizovať aktivita 4. Pravdepodobnosť, že sa aktivita 4 nebude vykonávať je 32%. V prípade, že sa aktivita 4 z projektu vypustí predĺži sa čas trvania aktivity 7 o 15%.

Pre potreby efektívneho nasadenia zdrojov, tvorby cien a informovania zákazníka je potrebné zodpovedať nasledovné otázky:

**1. Aký je čas trvania projektu, ak požadujeme *TP=80%* pravdepodobnosť splnenia tohto termínu?**

**2. Aká je pravdepodobnosť, že sa projekt podarí ukončiť za *TF=140* dní?**

Vaše výsledky je pre manažment firmy nutné dokumentovať prostredníctvom grafov v programe (vytvorte histogram pre trvanie projektu a graf vývoja pravdepodobnosti dokončenia projektu v závislosti na jeho trvaní).

Obrázok 1 – Graf aktivít projektu

|  |  |  |
| --- | --- | --- |
| Aktivita | Typ rozdelenia | Trvanie v dňoch |
| 1 | Diskrétne rovnomerné | Tmin = 4, Tmax = 15 |
| 2 | Diskrétne empirické | Tmin = 10, Tmax = 29; p = 0.2  Tmin = 30, Tmax = 48; p = 0.4  Tmin = 49, Tmax = 65; p = 0.4 |
| 3 | Diskrétne rovnomerné | Tmin = 48, Tmax = 92 |
| 4 | Diskrétne empirické | Tmin = 19, Tmax = 27; p = 0.2  Tmin = 28, Tmax = 44; p = 0.8 |
| 5 | Diskrétne empirické | Tmin = 5, Tmax = 19; p = 0.2  Tmin = 20, Tmax = 39; p = 0.5  Tmin = 40, Tmax = 55; p = 0.3 |
| 6 | Spojité rovnomerné | T = <10, 16) |
| 7 | Spojité rovnomerné | T = <20, 29) |
| 8 | Spojité rovnomerné | T = <12, 17) |
| 9 | Spojité rovnomerné | T = <13, 27) |

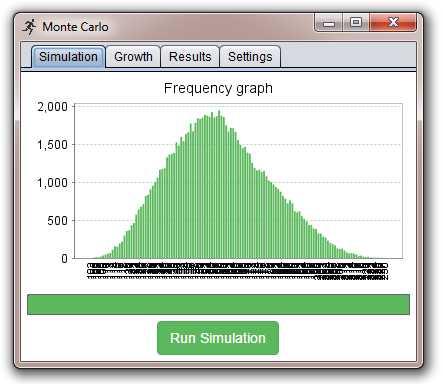
Tabuľka 1 – Prehľad rozdelení pre trvanie jednotlivých aktivít

# Riešenie

Riešenie tejto úlohy ponúka simulácia Monte Carlo, ktorá spočíva v generovaní náhodných čísel z určitého rozdelenia pravdepodobnosti v n replikáciách, ktoré zvyšujú presnosť výpočtov. Implementácia simulácie Monte Carlo abstrahuje problem úlohy od času, je efektívna a zameriava sa na štatistické odhady, podľa ktorých prezentuje výsledné riešenie.

V implementácii Monte Carlo sme sledovali nasledovné výpočty:

* Celkovú priemernú dobu trvania projektu podľa zadaných rozdelení pravdepodobností pre jednotlivé aktivity vrámci jedného simulačného behu
* Parciálne priemerné doby trvania projektu podľa zadaných kritérií pravdepodobností;
* Odchýlky parciálnych dôb trvania projektu zobrazené následne v grafe (Obrázok 3 – Zahrievanie, Simulated annealing)
* Pravdepodobnosť TP < 140 dní
* Pravdepodobnosť pre interval hodnôt pre TP patrí s 80% pravdepodobnosťou



Obrázok 2 – Aplikácia: Frekvencia výskytu TP <102;228>

## Trieda Simulácia

**Deklarácia**

public class Simulation extends SwingWorker<Integer, Integer>

**Vstupy**

**int replications** počet replikácií, ktoré chce používateľ uskutočniť vrámci jedného behu

**long initSeed** vstupná násada pre generator gererujúci násady ostatným generátorom

**Metódy**

**Simulation** konštruktor, inicialuzuje atribúty

**doInBackground** preťažená metóda rozhrania SwingWorker pre eventDispatchThread vlákno

**done** preťažená metóda rozhrania SwingWorker vykonaná po ukončení vlánka

**getNumberOfReplications** getter, vráti počet replikácií

**setNumberOfReplications** setter, nastavuje počet replikácií

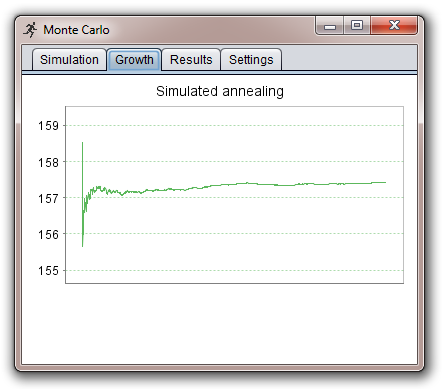
**generateTimes** volaná z metódy doInBackground, generuje časy pre jednotlivé aktivity

**sumTimes** spočítava časy jednotlivých aktivít vrámci jednej replikácie

**getHashMap** vracia HashMap štruktúru obsahujúcu výsledky meraní a ich početnosť

**getGrowthMap** vracia pole uchovávajúce parciálne výsledky jedného behu pre zahrievací graf

**findInterval** vstup su dva parameter, min a max, vrati pocet vysledkov, ktore spadaju do tohto intervalu



Obrázok 3 – Aplikácia: Zahrievanie

## Trieda Aplikácia

**Deklarácia**

public class Application extends javax.swing.JFrame implements PropertyChangeListener

**Metódy**

**Application** konštruktor, inicialuzuje komponenty

**initComponents** inicializuje GUI

**jLabel4MouseClicked** spúšťa nový beh simulácie s novou inštanciou triedy Simulacia

**jLabel5MouseClicked** ukladá používateľské nastavenia aplikácie: počet replikácií, násada

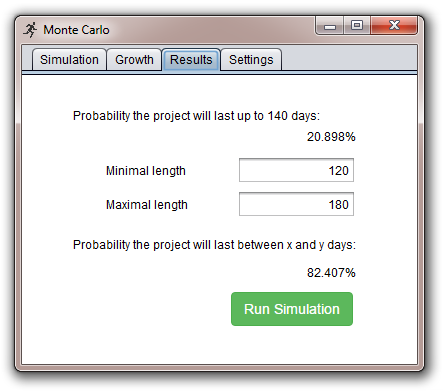
**main**  hlavná metóda aplikácie, spúšťa GUI

**propertyChange** preťažená metóda rozhrania PropertyChangeListener, “počúva”, kedy sa zmení hodnota atribútu “progress” v inštancii triedy Simulacia a podľa nej aktualizuje component ProgressBar znázorňujúci beh simulácie

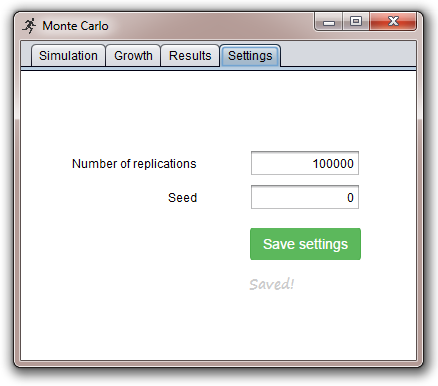
**initProgressBar** nastaví parameter komponentu ProgressBar

**createFrequencyGraph** vytvorí graf frekvencie výsledných hodnôt dĺžky projektu

**createGrowthChart** vytvorí graf z parciálnych priemerných hodnôt dĺžky projektu



Obrázok 4 – Aplikácia: Výsledky meraní



Obrázok 5 – Aplikácia: Nastavenie vstupných parametrov

# Logika

V semestrálnej práci bolo potrebné nasimulovať možnosti trvania deviatich aktivít. Ich hodnoty pochádzali z diskrétneho rovnomerného rozdelenia, diskrétneho empirického rozdelenia a spojitého rovnomerného rozdelenia. Generované boli nasledovne:

* **Spojité rovnomerné rozdelenie:**

Generator.nextDouble() \* (max - min) + min

* **Diskrétne empirické rozdelenie:**

double x = GeneratorA.nextDouble();

if (x < p) { GeneratorB.nextDouble() \* (max1 – min1) + min1 }

else { GeneratorB.nextDouble() \* (max2 – min2) + min2 }

* **Diskrétne rovnomerné rozdelenie:**

Generator.nextInt((max - min) + 1) + min

Cesty jednotlivých aktivít boli počítané nasledovné:

Cesta1: Aktivita2 + Aktivita6

Cesta2: Aktivita3 + Aktivita4

Pre ďalšie pokračovanie sa vyberala cesta max(Cesta1, Cesta2)

Cesta3: Aktivita3 + Aktivita7 + Aktivita8

K celkovej dĺžke projektu sa tak pripočítali cesty nasledovne:

**Trasa = Aktivita1 + max( max(Cesta1, Cesta2) + Aktivita5, Cesta3 ) + Aktivita9**

Takýmto spôsobom je umožnená paralelná práca na viacerých aktivitách, pričom je zohľadnená dĺžka jednotlivých úloch. Je potrebné, aby sa do výslednej dĺžky projektu započítavali vždy dlhšie úseky, pretože podľa grafu nie je možné vykonať niektoré activity skôr, ako iné activity skončia.

Pre počítanie pravdepodobnosti, že dĺžka projektu bude spĺňať zadaný termín, bola použitá upravená metóda pre simuláciu Monte Carlo, ktorá akceptuje vstupné parameter od používateľa a na základe nich počíta pravdepodobnosť ukončenia projektu v tomto časovom intervale (Viď Obrázok 4).

# Záver

Aplikácia využíva knižnicu na tvorbu grafov JFreeChart a aplikačné rozhranie bolo naprogramované v programovacom jazyku Java s využitím Swing komponentov.

Kód je dostupný tu: <https://github.com/Atenna/ds-Monte-Carlo>