**BDD Dokumentácia**

Marek Smutný

Opis riešenia (funkcií):

V mojom riešení som použil dve triedy: Node a Tree. Trieda Node reprezentuje jeden prvok v strome a jej parametre sú string, left a right. Tree má iba parametre root a pocet.

Obrázok, na ktorom je text

Automaticky generovaný popis

Na vytváraní stromu sa podieľa veľa funkcií. BDD\_create zavolá funkciu „vytvor“ a po vytvorení stromu zavolá funkciu „count\_nodes“.

Obrázok, na ktorom je text

Automaticky generovaný popis

Vo funkcii vytvor funkcia preorder strom vytvára po leveloch a funkcie redukcia\_I a redukcia\_S ho redukujú.

Obrázok, na ktorom je text

Automaticky generovaný popis

Obrázok, na ktorom je text

Automaticky generovaný popis

Funkcia make dostáva stringy a mení ich na zoznam. Potom zavolá funkciu makeLava na vytvorenie lavej node a makePrava na vytvorenie pravej node.

Obrázok, na ktorom je text

Automaticky generovaný popis

Funkcie makeLava a makePrava sú podobné zrkadlové funkcie na vytváranie pravého a lavého prvku. Na vstupe dostanú príklad v tvare zoznamu a vrátia string pre pravý a ľavý node.

Obrázok, na ktorom je text

Automaticky generovaný popis

Obrázok, na ktorom je text

Automaticky generovaný popis

Ešte predtým ako sa strom vytvorý, funkcia uprav\_vstup upraví používateľov príklad. Napríklad odstráni A.A usporiada premenné podľa toho aký používateľ zvolil postup atd.

Obrázok, na ktorom je text

Automaticky generovaný popis

Nakoniec BDD\_use testuje vytvorený strom spôsobom, že na základe vstupných jednotiek a núl prechádza stromom. Nakoniec vypíše string prvku v ktorom skončí (malo by byť vždy 1 alebo 0).

Obrázok, na ktorom je text

Automaticky generovaný popis

Testovanie korektnosti BDD:

Na testovanie korektnosti výsledkov môjho riešenia som použil funkciu evaluate. Ktorá za samotné premenné príkladu dosadzuje 1 a 0. Počas testovania som zistil, že tento spôsob riešenia booleovských funkcií je veľmi časovo neefektívny.

Obrázok, na ktorom je text

Automaticky generovaný popis

Testovanie pri 4 premenných. (ABCD, 100 náhodne vygenerovaných príkladov)

Obrázok, na ktorom je text

Automaticky generovaný popis

Na obrázku je príklad výpisu. Percentuálna miera redukcie je v tomto prípade 77.4%.

Obrázok, na ktorom je text

Automaticky generovaný popis

Po prehľadaní celej konzoly nenašlo ani jeden nesprávny výpis „NESPRAVNE“.

Podobne som testoval aj pre 8 premenných (QWERTYUI), 12 premenných (LKJHGFDSATRE) a 16 premenných (MNBVCXZASDFGHJKL) ani v jednom z testov nevyšiel nesprávny výsledok.

Percentuálna miera redukcie pri 16 premenných dosahovala v priemere 99.95% .

Testovanie času vytvorenia BDD:

Časovú náročnosť som testoval bez vypisovania korektnosti alebo samotných stromov. Hodnoty v tabuľke zvýraznené zelenou som intuitívne doplnil.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| počet vytvorení/premenné | 4 | 6 | 8 | 10 | 12 | 14 |
| 10000 | 1,405 | 4,203 | 10,7 | 23,777 | 47,903 | 90,703 |
| 20000 | 2,86 | 8,368 | 21,265 | 47,455 | 95,993 | 181 |
| 30000 | 4,214 | 12,679 | 31,668 | 70,653 | 144 | 271 |
| 40000 | 5,651 | 16,911 | 42,951 | 94,739 | 192 | 362 |

Na základe grafu a tabuľky je môj odhad časovej náročnosti O(n\*log n).

Pamäťová náročnosť a miera redukcie:

Na vypočítanie miery redukcie je potrebný počet nodov kompletného neredukovaného diagramu a počet zredukovaných/odstránených nodov. Počet nodov vo vytvorenom, zredukovanom diagrame zisťuje funkcia count\_nodes.

Obrázok, na ktorom je text

Automaticky generovaný popis

Počet nodov v neredukovanom diagrame:



Následne sa miera redukcie vypočíta jednoduchým vzorcom.



Mieru redukcie som testoval vytvorením 100 diagramov pri 13, 14, 15 a 16 premenných. Pri priemernom počte nodov v diagramoch sa vyskytli výnimky, pravdebodobne kvôli funkcii na náhodné vytváranie výrazov, avšak priemernú redukciu to veľmi neovplyvnilo. Priemerná redukcia narastala logaritmicky.

|  |  |  |
| --- | --- | --- |
|  | Priemerný počet nodov | Priemerná redukcia |
| 13 | 40,5 | 99,70% |
| 14 | 41,8 | 99,87% |
| 15 | 48,9 | 99,93% |
| 16 | 44,95 | 99,96% |

Moje testovanie malo tak vysokú mieru redukcie pravdepodobne kvôli tomu, že randomne vygenerovaný výraz sa dal veľmi efektívne upraviť.

Keďže aj pri 16 premenných sa počet nodov pohybuje okolo 45, pamäťová náročnosť je veľmi nízka. A taktiež rastie logaritmicky.