FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ

Slovenská technická univerzita

Ilkovičova 2, 842 16, Bratislava 4

2021/2022

Dátové štruktúry a algoritmy

Zadanie č.2

Cvičiaci: Mgr. Peter Lehoczký

Čas cvičenia: Štvrtok 13:00-14:40

Vypracoval Dávid Varinský

AIS ID : 116323

Dávid Varinský DSA 2. zadanie BDD

Uvod:

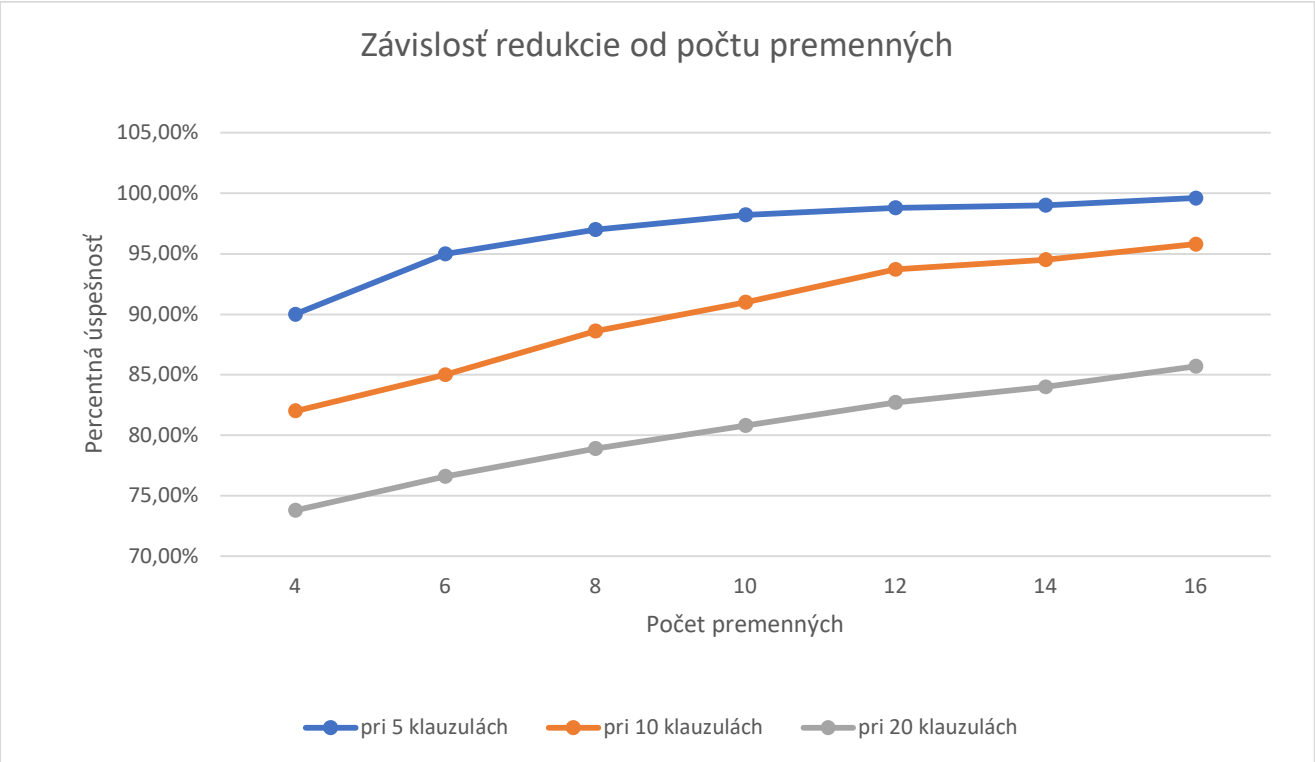
V tomto zadaní vytvárame a testujeme časovú náročnosť a redukciu BDD ktorý sa rozhodoval podľa nami zadanej boolovskej funkcie alebo funkcie ktorá bude náhodne generovaná. Následne po vytvorení stromu si po dosadení hodnôt pre príslušné premenné vieme zistiť či daný výraz pre tieto hodnoty platí alebo nie.

Testovanie

Testovanie vytvárania:

Redukciu sme testovali tak že pri vytváraní nových node sme inkrementovali celkový počet node o 1 pri redukcii typu S sme neprirátavali nič a pri redukcii typu I sme dekrementovali o 1 následne sa nám výsledok vypísal do konzoly v percentách pri 10 klauzulách a 100 opakovaniach.

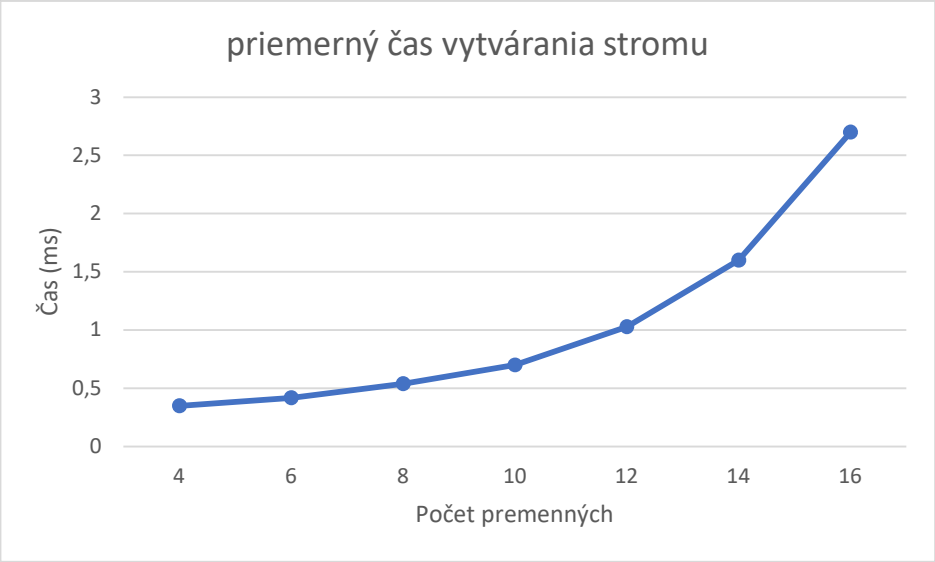
pocet premenných	4	6	8	10	12	14	16
pri 5 klauzulách	90%	95%	97%	98,20%	98,80%	99%	99,60%
pri 10 klauzulách	82,0%	85,0%	88,6%	91,0%	93,7%	94,5%	95,8%
pri 20 klauzulách	73,80%	76,60%	78,90%	80,80%	82,70%	85%	85,70%



Graf 1

V grafe si môžeme všimnúť že čím menej premenných si do grafu dosadíme tým menej dokážeme daný diagram redukovať. Zároveň si ale môžeme všimnúť že pri 5 klauzulách je strom viac zredukovaný ako pri 20 čo môžeme odôvodniť tým že vzniká menej rovnakých klauzúl ktoré môžu mať rovnakú hodnotu

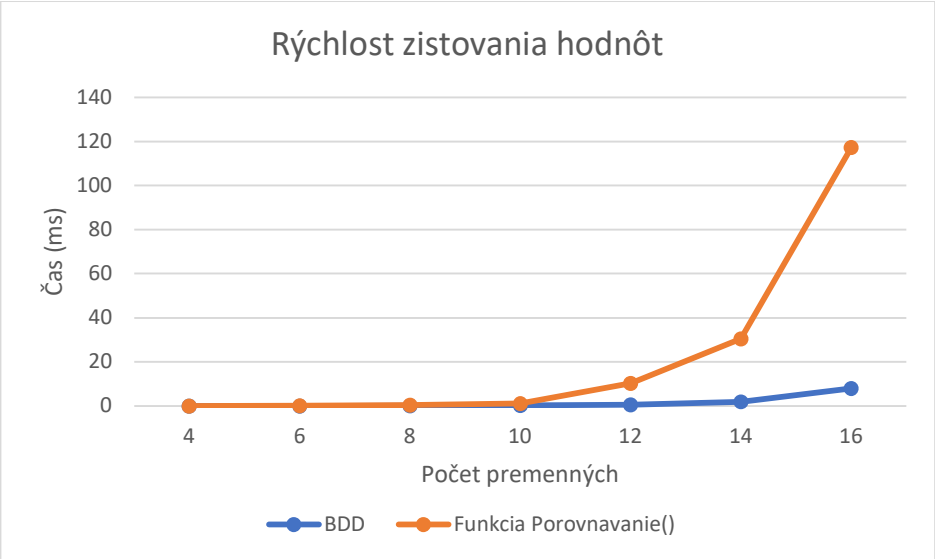
Počet premenných	4	6	8	10	12	14	16
Priemerný čas (ms)	0,35	0,42	0,54	0,7	1,03	1,6	2,7



Graf 2

V grafe č.2 si môžeme všimnúť že graf rastie exponenciálne s počtom premenných čo nám vyplýva aj zo vzorca $O(2^{n+1}-1)$

Počet premenných	4	6	8	10	12	14	16
BDD	0	0	0,05	0,24	0,49	1,98	8
Funkcia Porovnavanie()	0,04	0,12	0,35	1,14	10,31	30,5	117,25



Testovanie správnosti:

Správnosť sme testovali pomocou funkcie porovnávanie() do ktorej sme vložili booleanové hodnoty pre daný výraz, samotnú booleanovú funkciu a postupnosť znakov. Následne sme znaky nahradili booleanoskými hodnotami a výstup nám vypísal či je pre dané booleanosvké hodnoty výsledok true /false tieto výsledky sme porovnali s výsledkami ktoré nám vyšli v danom strome ak bolo riešenie nesprávne vypísalo booleanovské hodnoty pre ktoré sa daný BDD líšil od funkcie porovnávanie

Funkcia porovnávanie()

```
for (int i = 0 ; i<poradie.length();i++){
    function = function.replace(poradie.charAt(i),numbers.charAt(i));
    if (numbers.charAt(i)=='1'){
        function = function.replace(Character.toUpperCase(poradie.charAt(i)), newChar: '0');
    }
    if (numbers.charAt(i)=='0'){
        function = function.replace(Character.toUpperCase(poradie.charAt(i)), newChar: '1');
    }
}
String[] temp = function.split( regex: "-",  limit: 0);
for (int j = 0; j < temp.length; j++){
    if (!(temp[j].contains("0"))){
        return true;
    }
}
return false;
```

Porovnávanie výsledkov:

```
if (useOutput != porovnanie(s[i], inputLogic, order)) {
    System.out.println(s[i] + "incorrect\n");
    System.out.println("--" + use(s[i], root));
    System.out.println(porovnanie(s[i], inputLogic, order));
}
```

Vďaka tejto funkcii sme si overili že náš BDD use a BDD create sú 100% korektné.

Generovanie funkcie:

Na generovanie funkcie používame funkciu generate() do ktorej vkladáme postupnosť znakov vďaka ktorej vie aké písmená môže dosadiť. Funguje na jednoduchom princípe náhodného generovania čísel od 0 do 1000 následným modulom dĺžky postupnosti znakov dostaneme aký znak sa má dosadiť a následne zistíme zvyšok po delení 2 a ak je 0 dosadíme malé písmeno ak 1 dosadíme veľké písmeno

```
for (int i = 0; i<100;i++){
    random = (int)(Math.random()*100);
    random = random % poradie.length();
    random+= 65;
    if (random%2==0){
        output = output + Character.toLowerCase((char)random);
    }else output = output + (char)random;
    if (i%20==19){
        output+="+";
    }
```

Funkcia firstcorection():

Táto funkcia sa nám postará o to že v danej klauzule sa bude každý znak vyskytovať iba 1 krát

```
s = s.replace( oldChar: '+',  newChar: '-');
String[] returned = s.split( regex: "-",  limit: 0);
s="";
for (int i = 0 ; i< returned.length;i++) {
    for (int j = 0; j < poradie.length(); j++) {
        if (returned[i].contains(poradie.substring(j,j+1))){
            returned[i] = returned[i].replaceAll(poradie.substring(j,j+1), replacement: "");
            returned[i]=  returned[i]+=poradie.substring(j,j+1);
        }
    }
}
for (int i = 0 ;i <returned.length; i++){
    s = s+ "+" +returned[i];
}
s=s.substring(1);

return s;
```

Funkcia formation:

Funkcia formation sa nám stará o upravovanie výrazu, odstránenie nasledujúceho písmena v poradí z celej booleanovej funkcie a aj o zákon o absorbcii (ab+a = a).