SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE

FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ

Zadanie 3

BINÁRNE ROZHODOVACIE DIAGRAMY

Anna Yuová

Predmet: Dátové štruktúry a algoritmy

Akademický rok: 2020/2021

Semester: letný

1. ÚVOD

Pri vypracovaní môjho zadania som urobila funkcie BDD_create, BDD_use a pokúšala som sa aj o BDD_reduce.

Program som implementovala v jazyku C a v programe DEV-C++ (verzia 5.11). Kódy sú spustiteľné.

2. OPIS FUNKCIE BDD_CREATE

Na vypracovanie funkcie BDD_create som si najprv musela vytvoriť 3 štruktúry – štruktúra BDD, ktorá obsahuje počet premenných, veľkosť BDD a ukazovateľ na koreň. Druhá štruktúra sa volá Node a obsahuje hodnotu a ukazovatele na potomkov, ktorí pôjdu v strome buď doľava (zero) alebo doprava (one). Posledná štruktúra sa volá BD a obsahuje moju vlastnú štruktúru - tabuľku a počet premenných. Na vypracovanie môjho zadania som si zvolila ako vlastnú štruktúru pre Booleovskú funkciu pravdivostnú tabuľku.

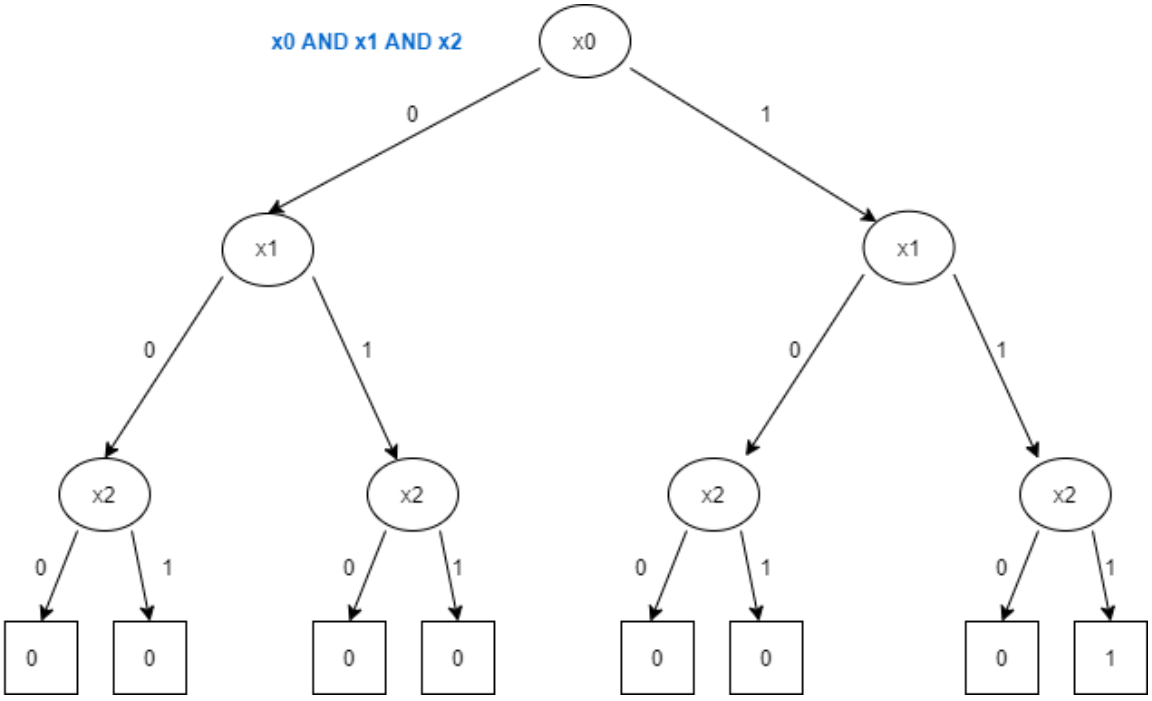
Pri vytváraní stromu si najskôr spočítam počet prvkov tabuľky a vytvorím si ukazovateľ na začiatok (koreň) stromu. Do vysledok->koren si uložím novo vytvorený node, ktorý si vrátim z funkcie vytvor_node(). Alokujem si tam priestor pre môj node, nastavím mu potomkov zero a one na null, hodnotu dám na -1 a vrátim ukazovateľ na tento node. S týmto ukazovateľom potom pracujem ako s aktuálnym uzlom a všetko prechádza cez neho. Vo for cykle prechádzam toľkokrát, koľko je prvkov mojej pravdivostnej tabuľky a podľa 3 podmienok sa rozhodujem, čo idem robiť – ak je prvok z tabuľky znak nového riadku alebo to je číslo.. Ak je to číslo 0, tak sa posuniem v strome doľava. Ak som ešte na začiatku nikdy nešla doľava, musím si tento node vytvoriť a znovu zavolám funkciu vytvor_node(). Ak už tento node mám vytvorený len tade prejdem (aktualny = aktualny->zero). To isté urobím aj v prípade ak je to 1, akurát idem doprava. Ak je to znak nového riadku, tak nastavím hodnotu na predošlú hodnotu ako bol koniec znaku a posuniem sa znova do koreňa.

```
if (split_tabulka[i] == '0')
{
    if (aktualny->zero == NULL)
    {
        aktualny->zero = vytvor_node();
        aktualny->zero= aktualny;
    }
    aktualny = aktualny->zero;

}

if (split_tabulka[i] == '1')
{
    if (aktualny->one == NULL)
    {
        aktualny->one = vytvor_node();
        aktualny->one->previous = aktualny;
    }
    aktualny = aktualny->one;
}
```

BDD_create pre pravdivostnú tabuľku (000,001,010,011,100,101,110,111)



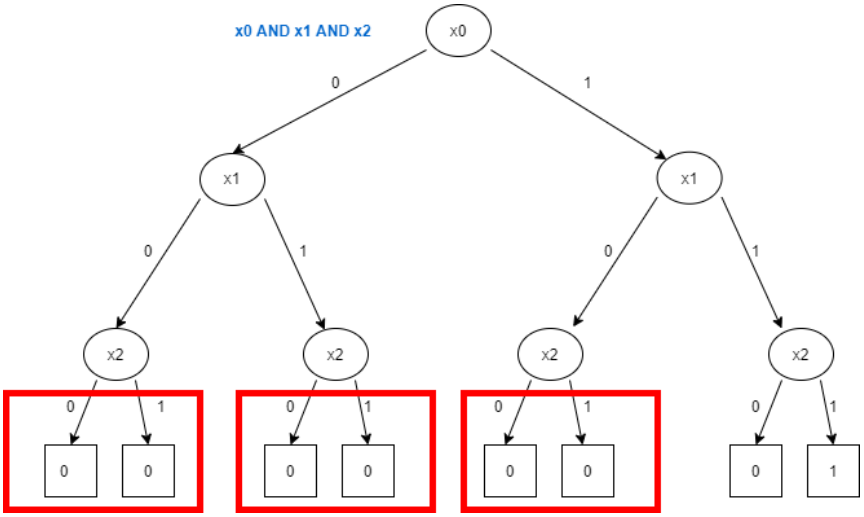
Obrázok č.1 – vytváranie binárneho rozhodovacieho diagramu a prideľovanie hodnôt

3. OPIS FUNKCIE BDD_USE

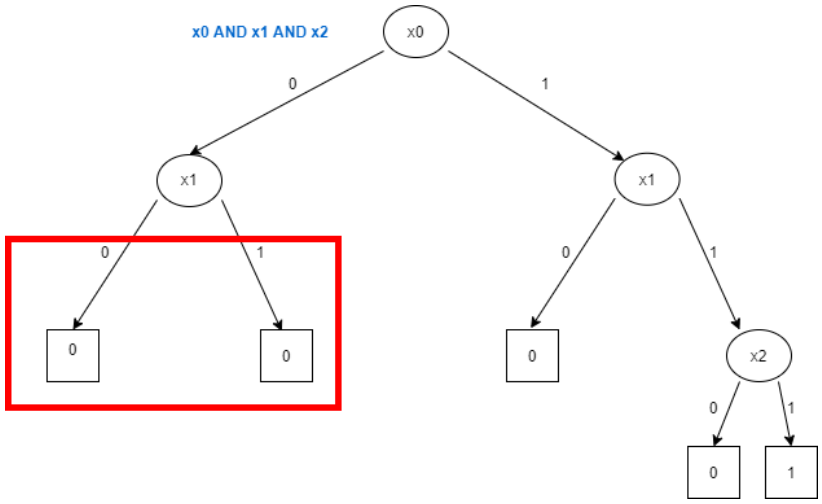
Pri BDD_use postupujem na začiatku rovnako ako pri BDD_create. V argumente si pošlem hodnoty jednotlivých premenných ako ich idem prechádzať – napr. 111 znamená, že A = 1, B = 1 a C = 1. Vytvorím si pomocnú premennú, do ktorej so vložím môj koreň a odtiaľ prechádzam v strome – ak mi v argumente prišlo napr. 111, tak z koreňa pôjdem: koreň -> doprava -> doprava a na konci sa pozriem akú hodnotu má node uloženú a tú vrátim (ako char).

4. OPIS FUNKCIE BDD_REDUCE

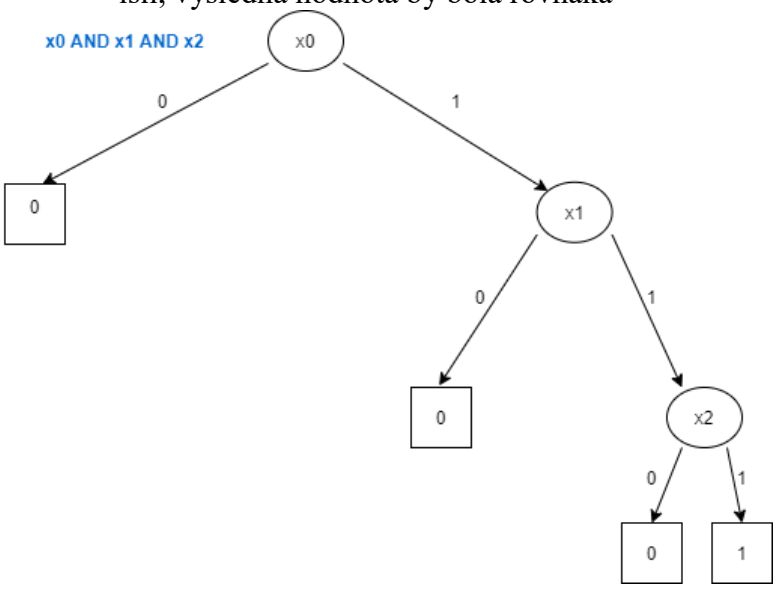
Pri funkcii BDD_reduce sa zas presuniem do koreňa a zavolám pomocnú funkciu BDD_reduce_node, do ktorej si pošlem koreň a počet odstránených uzlov (na začiatku 0). Vo funkcii BDD_reduce_node volám rekurentne túto funkciu, podľa toho v akom uzle som. Ak koreň->doprava /doľava nie je null, tak sa len posuniem na daný node a počet odstránených uzlov nemením (stále 0). Ak prídem na podmienku, že dané uzly nie sú null a ich hodnoty oboch uzlov (doľava aj doprava) sú rovnaké, tak ich môžem vymazať – uvoľním uzol, ktorý je naľavo (zero) a aj predošlý.
Z x2 je jedno kam by išiel, aj tak by bola výsledná hodnota 0. Zredukuje ->zero a ostane iba jeden node.



Obrázok č.2: Ak sú na rovnakej úrovni a majú rovnaké hodnoty aj zľava aj zprava, tak sa dajú zredukovať



Obrázok č.3: Zredukované uzly ukazujú rovno na hodnotu, nezáleží na tom do akého nodu by išli, výsledná hodnota by bola rovnaká



Obrázok č.4: Úplne zredukovaný binárny rozhodovací diagram

5. TESTOVANIE

Testovala som zavolanie funkcie BDD_create pre rôzne tabuľky a následne som volala viackrát po sebe BDD_use a kontrolovala som či mi vracia tie hodnoty, ktoré som čakala.

```
272     tabulka.tabulka = NULL  ;
273     tabulka.tabulka = tabulka_hodnoty2;
274     bdd = BDD_create(&tabulka);
275     printf("BDD je vytvoreny pre tabulku:\n");
276     printf(tabulka_hodnoty2);
277
278     use = BDD_use(bdd, "11");
279     printf("\nTestovanie pre BDD_use: 11\n");
280     if (use == '1')
281     {
282         printf("Pre A = 1, B = 1 ocakava hodnotu: 1\nA vratil hodnotu: %c\n", use);
283         printf("BDD_use funguje SPRAVNE\n");
284     }
285     else
286     {
287         printf("ERROR\nBDD_use funguje NESPRAVNE\n");
288         printf("Pre A = 1, B = 1 ocakava hodnotu: 1\nA vratil hodnotu: %c\n", use);
289     }
290
```

2 PREMENNÉ A OR:

```
C:\Users\Anna\Desktop\DSA_zadanie3\Project1.exe
-----------------------------------------------

BDD je vytvoreny pre tabulku:
000
011
101
111

Testovanie pre BDD_use: 11
Pre A = 1, B = 1 ocakava hodnotu: 1
A vratil hodnotu: 1
BDD_use funguje SPRAVNE


Testovanie pre BDD_use: 00
Pre A = 0, B = 0 ocakava hodnotu: 0
A vratil hodnotu: 0
BDD_use funguje SPRAVNE


Testovanie pre BDD_use: 10
Pre A = 1, B = 0 ocakava hodnotu: 1
A vratil hodnotu: 1
BDD_use funguje SPRAVNE
-----------------------------------------------
```

3 PREMENNÉ A AND:

```
C:\Users\Anna\Desktop\DSA_zadanie3\Project1.exe
BDD je vytvoreny pre tabulku:
0000
0010
0100
0110
1000
1100
1111
1010

Testovanie pre BDD_use: 111
Pre A = 1, B = 1, C = 1 ocakava hodnotu: 1
A vratil hodnotu: 1
BDD_use funguje SPRAVNE


Testovanie pre BDD_use: 000
Pre A = 0, B = 0, C = 0 ocakava hodnotu: 0
A vratil hodnotu: 0
BDD_use funguje SPRAVNE


Testovanie pre BDD_use: 010
Pre A = 0, B = 1, C = 0 ocakava hodnotu: 0
A vratil hodnotu: 0
BDD_use funguje SPRAVNE


Testovanie pre BDD_use: 110
Pre A = 1, B = 1, C = 0 ocakava hodnotu: 0
A vratil hodnotu: 0
BDD_use funguje SPRAVNE


Testovanie pre BDD_use: 001
Pre A = 0, B = 0, C = 1 ocakava hodnotu: 0
A vratil hodnotu: 0
BDD_use funguje SPRAVNE
------------------------------------------------
```

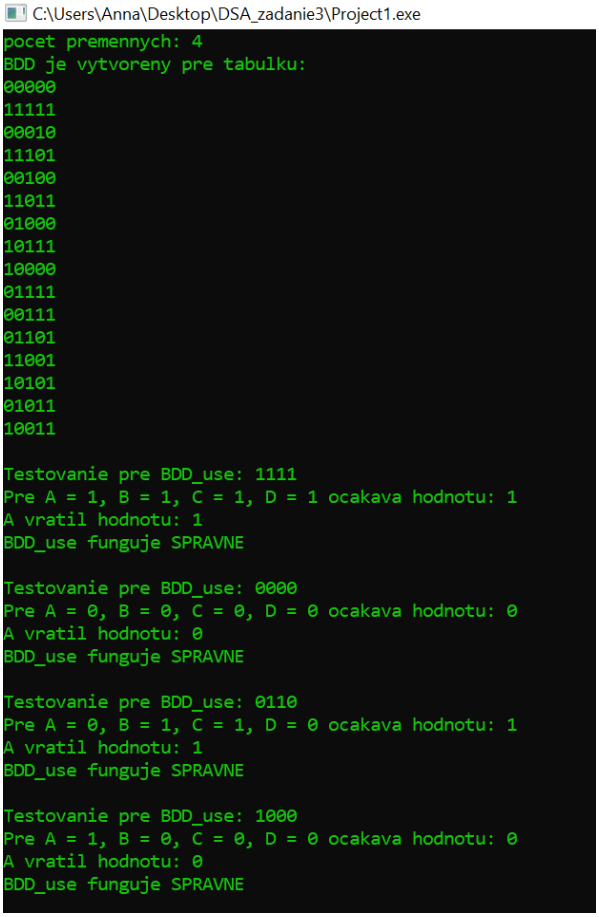
NÁHODNE VYGENEROVANÝ STRING PRE 3 PREMENNÉ:
-1 je ak je a = 1 a b = 1

```
387     snprintf(str, 1000, "%d", vysledok2);
388     printf("\n\nTestovanie pre nahodne vygenerovane cislo pre 3 premenne\n");
389     printf("Nahodne Vygenerovany string: %s\n", str);
390
391     if (a == 1 && b == 1 && c == 0)
392     {
393         ocakavane = '1';
394         cakaneCislo = 1;
395     }
396     else
397     {
398         ocakavane = '0';
399         cakaneCislo = 0;
400     }
401
402     use = BDD_use(bdd, str);
403     printf("Testovanie pre BDD_use: %s\n",str);
404     if (use == ocakavane)
405     {
406         printf("Pre A = %d, B = %d, C = %d ocakava hodnotu: %d\nA vratil hodnotu: %c\n",a, c ,b, cakaneCislo, use);
407         printf("BDD_use funguje SPRAVNE\n");
408     }
409     else
```

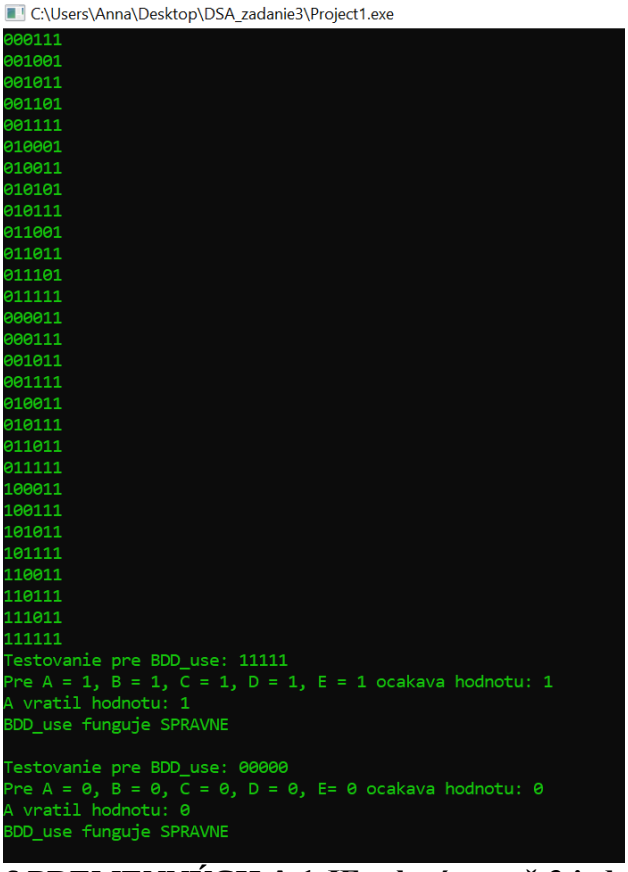
```
------------------------------------------------


Testovanie pre nahodne vygenerovane cislo pre 3 premenne
Nahodne Vygenerovany string: 101
Testovanie pre BDD_use: 101
Pre A = 1, B = 1, C = 0 ocakava hodnotu: 1
A vratil hodnotu: 1
BDD_use funguje SPRAVNE
------------------------------------------------
```

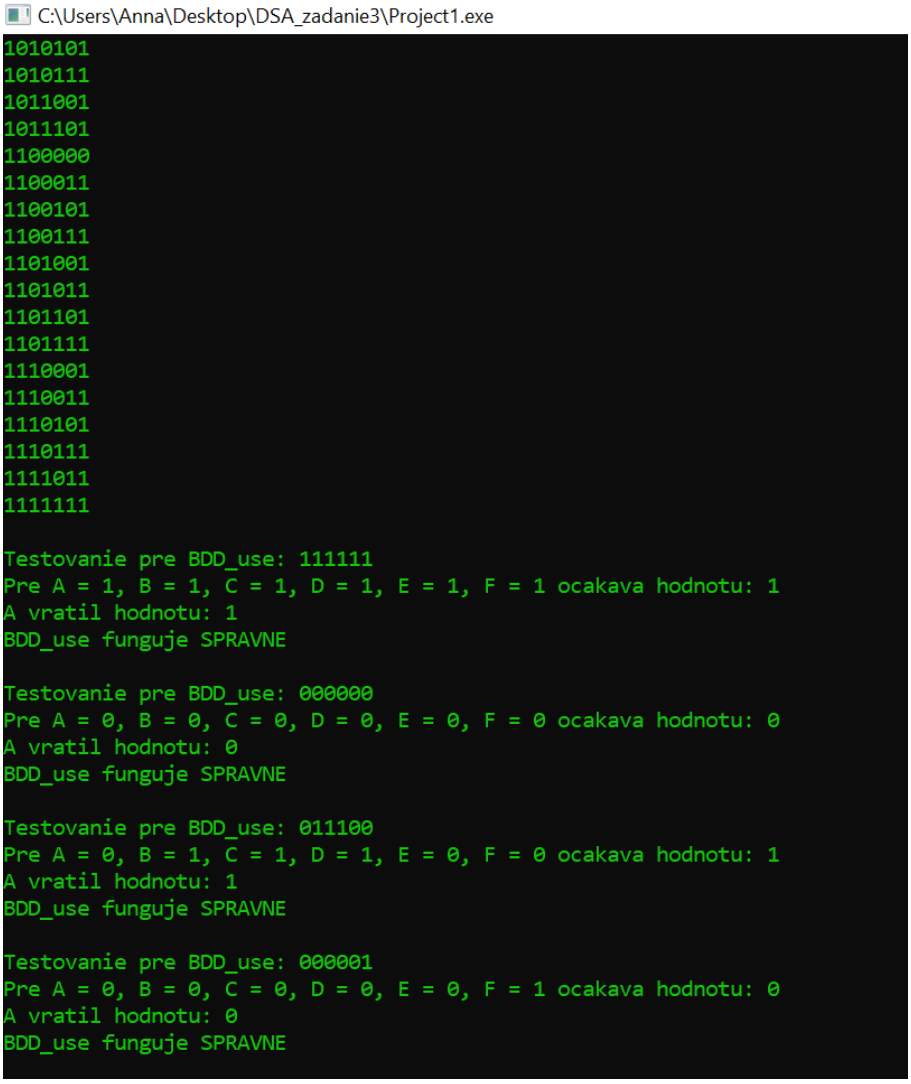
4 PREMENNÉ A 1 je ak sú aspoň 2 jednotky:
-napr. 0001 = 0, 0110 = 1



5 PREMENNÝCH A OR:



6 PREMENNÝCH A 1 JE: ak sú aspoň 3 jednotky:



VÝSLEDNÁ TABUĽKA Z TESTOVANIA

POČET PREMENNÝCH	VYHODNOCOVANIE	TESTOVANÁ VZORKA	VÝSLEDOK	ČAS
2	AND	11	BDD_use vrátilo 1	0.0010s
		00	BDD_use vrátilo 0	0.0020s
		10	BDD_use vrátilo 0	0.0010s
3	AND	111	BDD_use vrátilo 1	0.0010s
		000	BDD_use vrátilo 0	0.0020s
		010	BDD_use vrátilo 0	0.0020s
		110	BDD_use vrátilo 0	0.0010s
		001	BDD_use vrátilo 0	0.0010s
3	1 ak A =1, B =1, C=0	náhodne vygenerované 101	BDD_use vrátilo 1	0.0122s
4	ak sú aspoň 2 jednotky, vráti 1	1111	BDD_use vrátilo 1	0.0070s
		0000	BDD_use vrátilo 0	0.0060s
		0110	BDD_use vrátilo 1	0.0020s
		1000	BDD_use vrátilo 0	0.0060s
5	OR	11111	BDD_use vrátilo 1	0.0060s

		00000	BDD_use vrátilo 0	0.0040s
6	ak sú aspoň 2 jednotky, vráti 1	111111	BDD_use vrátilo 1	0.1010s
		000000	BDD_use vrátilo 0	0.1010s
		011100	BDD_use vrátilo 1	0.0100s
		000001	BDD_use vrátilo 0	0.1060s

6. ZÁVER

Z mojich testovaní vyplýva, že na základe testovania, by mala funkcia BDD_create správne vytvoriť binárny rozhodovací diagram, keďže všetky očakávané hodnoty sa rovnali hodnotám, ktoré vrátila funkcia BDD_use. Ak by sa hodnoty vrátene z funkcie BDD_use nerovnali očakávaným hodnotám, tak by moje riešenie nebolo správne. Funkciu BDD_reduce som sa pokúsila implementovať tiež, ale pri testovaní to niekde padalo a tým pádom nie je úplne dokončená. Pri testovaní som testovala výsledky najviac pre 6 premenných a všetky výsledky vyšli časovo do sekundy a boli hneď aj vypísané.
Časová zložitosť: O(Logn(n))
Priestorová zložitosť: O(n)