Martin Hric
ID:111696

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE

FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ

# DÁTOVÉ ŠTRUKTÚRY A ALGORITMY

# Binárne rozhodovacie diagramy

Vypracoval: Martin Hric

2020/2021

Martin Hric
ID:111696

# ÚVOD

Zadaním bolo vytvoriť program, ktorý bude vedieť vytvoriť, redukovať apoužiť dátovú štruktúru BDD (Binárny Rozhodovací Diagram) so zameraním na využitie pre reprezentáciu Booleovských funkcií.

Konkrétne implementujte tieto funkcie:

- BDD *BDD_create(BF *bfunkcia);
- int BDD_reduce(BDD *bdd);
- char BDD_use(BDD *bdd, char*vstupy);

Konkrétne, ja som funkciu BDDreduce neimplementoval, implementoval som len create a use.

Martin Hric
ID:111696

# CREATE:

Funkcia BDD_create má slúžiť na zostavenie úplného (t.j. nie redukovaného) binárneho rozhodovacieho diagramu, ktorý má reprezentovať/opisovaťzadanú Booleovskú funkciu (vlastná štruktúrasnázvom BF), na ktorú ukazuje ukazovateľbfunkcia, ktorý je zadaný ako argument funkcie BDD_create.

Štruktúru BF si definujete sami –podstatné je, aby nejakým (vami vymysleným/zvoleným spôsobom) bolo možné použiť štruktúru BF na opis Booleovskej funkcie. Napríklad, BF môže opisovať Booleovskú funkciu ako pravdivostnú tabuľku, vektor, alebo výraz. Návratovou hodnotou funkcie BDD_create je ukazovateľ na zostavený binárny rozhodovací diagram, ktorý je reprezentovaný vlastnou štruktúrou BDD.

Štruktúra BDD musí obsahovať minimálne tieto zložky: počet premenných, veľkosť BDD (počet uzlov) a ukazovateľ na koreň (prvý uzol) BDD. Samozrejme potrebujete aj vlastnú štruktúru, ktorá bude reprezentovať jeden uzol BDD.

```
typedef struct node{
    struct node *left;
    struct node *right;
    long long dlzka;
    char *list;
}NODE;

typedef struct BDD{
    unsigned int premenne;
    unsigned int uzly;
    NODE *root;
}BDD;
```

Martin Hric
ID:111696

Create robím metódou zhora nadol, vstupom je vektor.

V create alokujem miesto pre štruktúru BDD, zavolám funkciu na alokovanie a nastavenie 1 uzla, na začiatku vytvorím takto len koreň, a nakonci volám  rekurzívnu funkciu, ktorá ma za úlohu vytvoriť celý diagram.

```
BDD *bdd_create(char *bfunkcia){  //alokujem si miesto 1 BDD

    BDD *bdd=(BDD*)calloc(1,sizeof(BDD));
    NODE *new= create_node(bfunkcia);
    bdd->root=new;
    int size=(int) strlen(bfunkcia);
    bdd->premenne=(int) log2(size);
    if(bdd->premenne != ceil(bdd->premenne)){
        printf("zly vstup\n");
        return NULL;
    }
    bdd->uzly++;
    recursive_create(new,bdd);
    return bdd;
}
```

Martin Hric
ID:111696

## USE:

Funkcia BDD_use má slúžiť na použitie BDD pre zadanú (konkrétnu) kombináciu vstupných premenných Booleovskej funkcie azistenie výsledkuBooleovskej funkcie pre túto kombináciu vstupných premenných. Vrámci tejto funkcie „prejdete“ BDD stromom smerom od koreňa po list takou cestou, ktorú určuje práve zadaná kombinácia vstupných premenných. Argumentami funkcie BDD_usesú ukazovateľ snázvom bddukazujúci na BDD (ktorý sa má použiť) aukazovateľ snázvom vstupyukazujúci na začiatok poľa charov (bajtov). Práve toto pole charov/bajtov reprezentuje nejakým (vami zvoleným) spôsobom konkrétnu kombináciu vstupných premenných Booleovskej funkcie. Napríklad, index poľa reprezentuje nejakú premennú ahodnota na tomto indexe reprezentuje hodnotu tejto premennej (t.j. pre premenné A, B, C aD, kedy AaC sú jednotky aB aD sú nuly, môže ísť napríklad o “1010“), môžete si však zvoliť iný spôsob.
Návratovou hodnotou funkcie BDD_use je char, ktorý reprezentuje výsledok Booleovskej funkcie –je to buď ‘1’ alebo ‘0’.

```
char bdd_use(BDD *bdd, char *vstupy){ //prejde
    NODE *temp=bdd->root;
    int i=0;
    while(vstupy[i] != '\0'){
        if(vstupy[i]== '1') {
            temp=temp->right;
        }
        else if(vstupy[i]=='0'){
            temp=temp->left;
        }
        i++;
    }
    if(temp->list[0]== '0') return '0';
    else return '1';
}
```

Martin Hric
ID:111696

# TESTOVANIE:

V rámci testovanie je potrebné, aby ste náhodným spôsobom generovali Booleovské funkcie, podľa ktorých budete vytvárať BDD pomocou funkcie BDD_create. Vytvorené BDD následne zredukujete funkciou BDD_reduce anakoniec overíte 100%funkčnosť zredukovaných BDD opakovaným (iteratívnym) volaním funkcie BDD_use tak, že použijete postupne všetky možné kombinácie vstupných premenných. Počet premenných v rámci testovania BDD by mal byť minimálne 13. Počet Booleovských funkcií / BDD diagramov by mal byť minimálne 2000.

**Testovanie som robil na 13-17 premenných, prvé na 2000 krát ale vzhľadom na čas som pri väčších premenných postupne zmenšoval opakovania.**

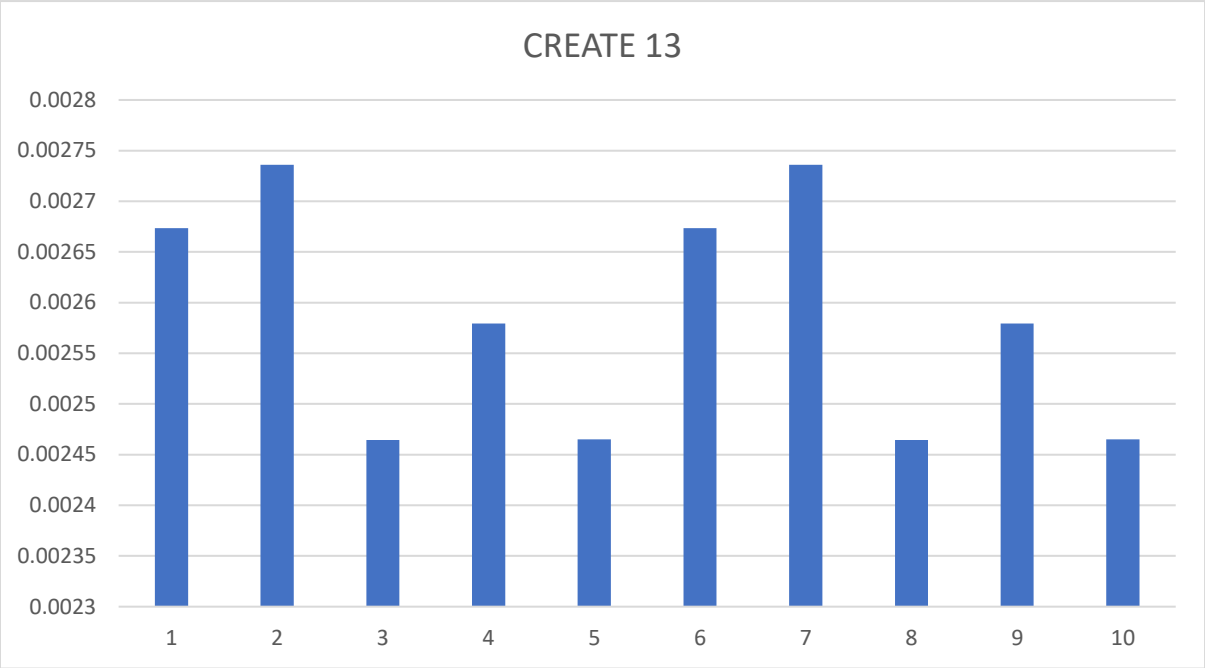
| POCET PREMENNE | PRIEMER CREATE | CELKOVY CREATE | PRIEMER USE | CELKOVO USE | POCET UZLOV | POCET VSTUPOV | SPOLU |
|---|---|---|---|---|---|---|---|
| 13 | 0.00267351s | 5.34701s | 0.00199149s | 3.98299s | 16383 | 8192 | 9.33s |
| 13 | 0.002736s | 5.47201s | 0.0020595s | 4.119s | 16383 | 8192 | 9.59101s |
| 13 | 0.0024645s | 4.929s | 0.0018145s | 3.629s | 16383 | 8192 | 8.558s |
| 13 | 0.0025795s | 5.159s | 0.0019445s | 3.889s | 16383 | 8192 | 9.048s |
| 13 | 0.002465s | 4.93s | 0.00185s | 3.7s | 16383 | 8192 | 8.63s |
| | | | | | | | |
| 14 | 0.004985s | 9.97s | 0.0037115s | 7.423s | 32767 | 16384 | 17.393s |
| 14 | 0.00495298s | 9.90596s | 0.00388901s | 7.77802s | 32767 | 16384 | 17.684s |
| 14 | 0.004961s | 9.922s | 0.0038485s | 7.697s | 32767 | 16384 | 17.619s |
| 14 | 0.00508551s | 10.171s | 0.00387749s | 7.75497s | 32767 | 16384 | 17.926s |
| 14 | 0.00495599s | 9.91198s | 0.00374751s | 7.49501s | 32767 | 16384 | 17.407s |
| | | | | | | | |
| 15 | 0.009949s | 19.898s | 0.00793548s | 15.871s | 65535 | 32768 | 35.769s |
| 15 | 0.00985999s | 19.72s | 0.00815151s | 16.303s | 65535 | 32768 | 36.023s |
| 15 | 0.010044s | 20.088s | 0.00801349s | 16.027s | 65535 | 32768 | 36.115s |
| 15 | 0.010255s | 20.51s | 0.00796001s | 15.92s | 65535 | 32768 | 36.43s |
| 15 | 0.010091s | 20.182s | 0.00823049s | 16.461s | 65535 | 32768 | 36.643s |

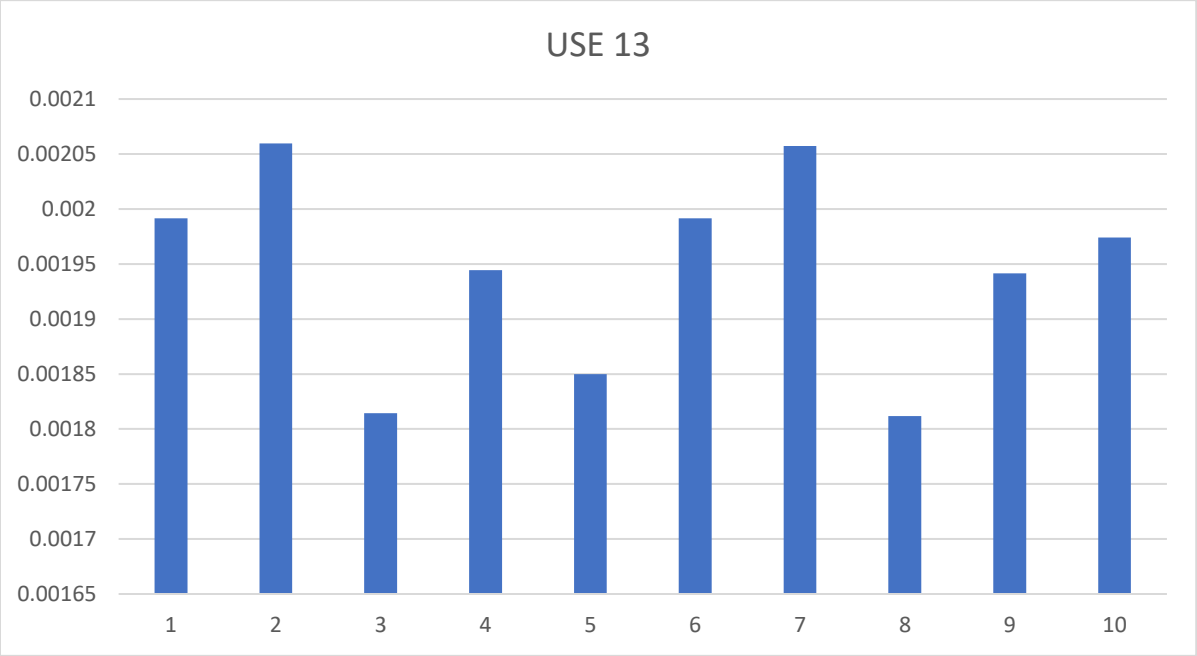
Martin Hric
ID:111696

| | PRIEMER CREATE | CELKOVO CREATE | PRIEMER USE | CELKOVO USE | UZLY | VSTUPY | SPOLU |
|---|---|---|---|---|---|---|---|
| 16 | 0.020288s | 40.576s | 0.017898s | 35.79604s | 131071 | 65536 | 76.327s |
| 16 | 0.019902s | 39.804s | 0.01635s | 32.7s | 131071 | 65536 | 72.504s |
| 16 | 0.019014s | 38.02804s | 0.016026s | 32.052s | 131071 | 65536 | 70.08s |
| 16 | 0.01914s | 38.28s | 0.016616s | 33.23196s | 131071 | 65536 | 71.512s |
| 16 | 0.01991s | 39.82004s | 0.016286s | 32.57196s | 131071 | 65536 | 72.392s |
| | | | | | | | |
| 17 | 0.03923s | 78.46s | 0.033085s | 66.17s | 262143 | 131072 | 144.63s |
| 17 | 0.038205s | 76.4101s | 0.03453s | 69.06s | 262143 | 131072 | 145.47s |
| 17 | 0.03881s | 77.62s | 0.033395s | 66.79s | 262143 | 131072 | 144.41s |
| 17 | 0.038935s | 77.87s | 0.03372s | 67.44s | 262143 | 131072 | 145.31s |
| 17 | 0.0388499s | 77.6998s | 0.03492s | 69.8399s | 262143 | 131072 | 147.54s |

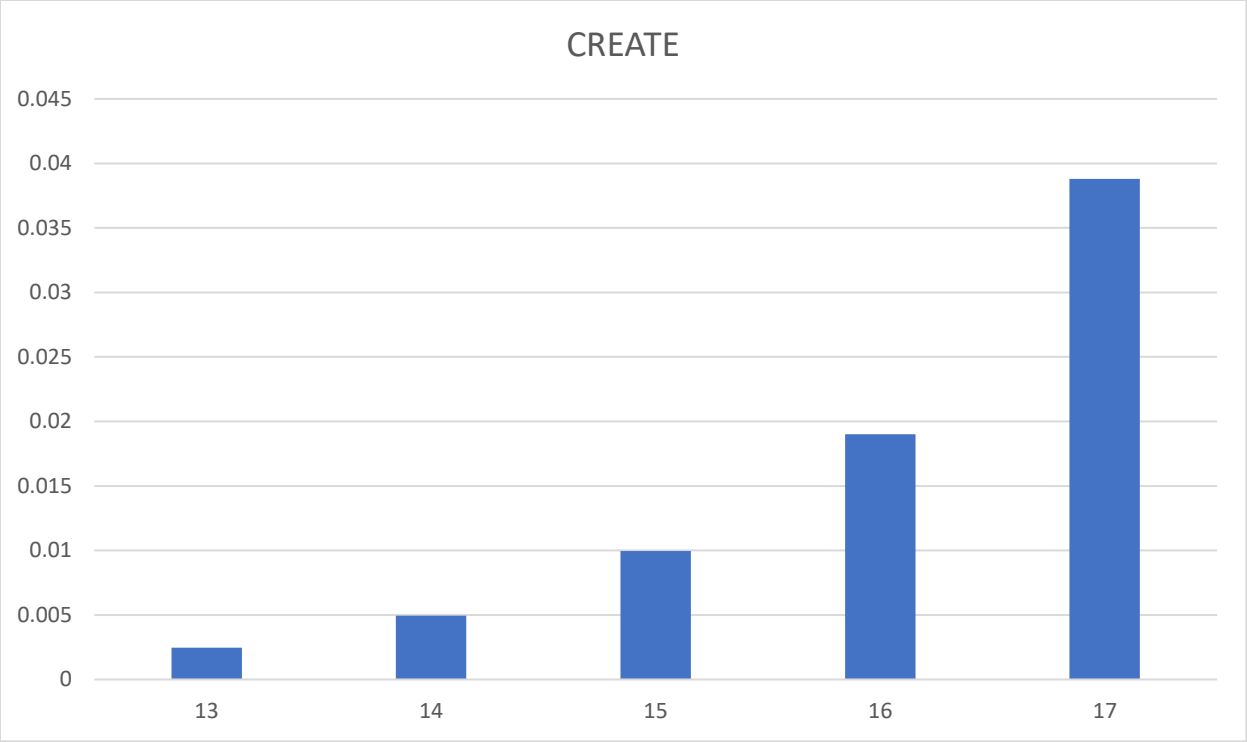
***Pre 16 premenných v cykle 500
*** Pre 17 premenných v cykle 200
Pre celkové časy som to vynásobil, aby sa to rovnalo počtu opakovaniam pre všetky premenné.

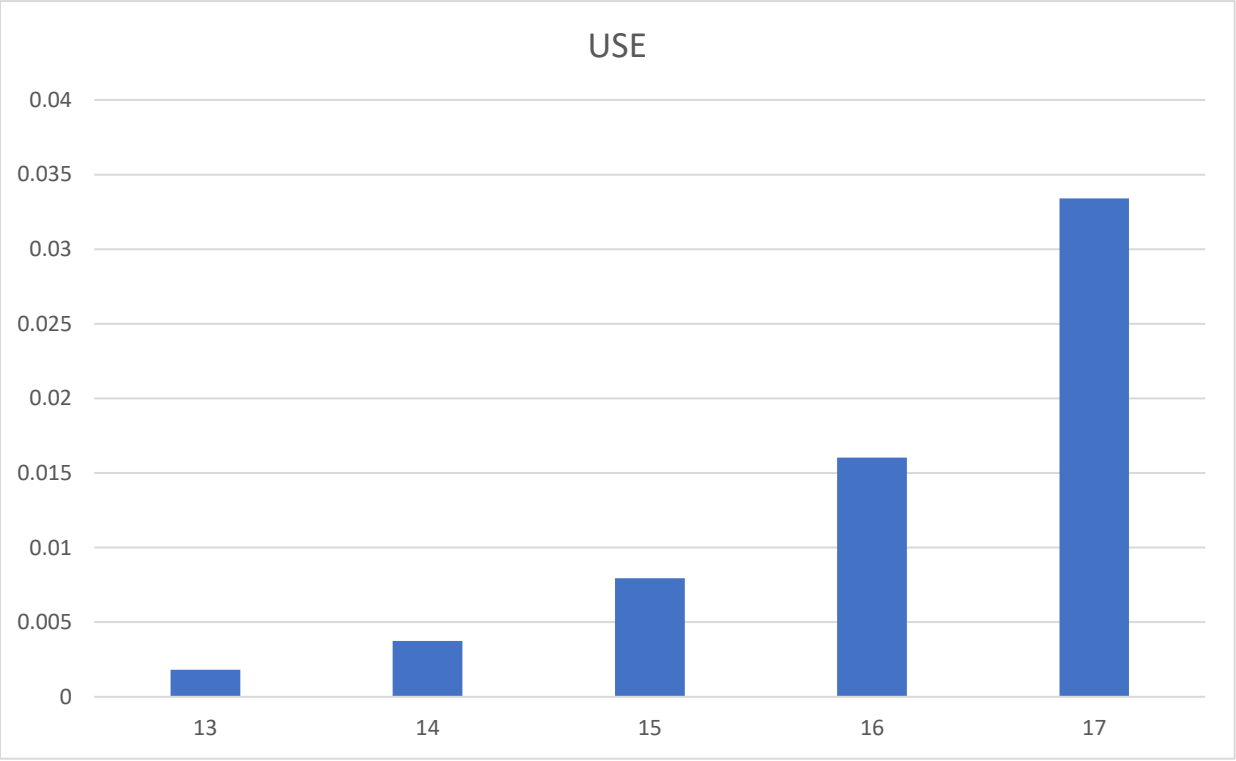
Martin Hric
ID:111696



Martin Hric
ID:111696



Martin Hric
ID:111696

# Zložitosti:

Časová zložitosť pre create: O(2^n) , kde n je počet premenných.

Časová zložitosť pre use: O(n), lebo len prejdem celý strom na základe počtu premenných.

Pamäťová zložitosť: O(2^(n+1) -1