Zadanie 3 – Binárne rozhodovacie diagramy

Vytvorte program, ktorý bude vedieť vytvoriť, redukovať a použiť 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*);

Samozrejme môžete implementovať aj ďalšie funkcie, ktoré Vám budú nejakým spôsobom pomáhať v implementácii vyššie spomenutých funkcii, nesmiete však použiť existujúce funkcie na prácu s binárnymi rozhodovacími diagramami.

Funkcia **BDD_create** má slúžiť na zostavenie úplného (t.j. nie redukovaného) binárneho rozhodovacieho diagramu, ktorý má reprezentovať/opisovať zadanú Booleovskú funkciu (vlastná štruktúra s názvom BF), na ktorú ukazuje ukazovateľ *bfunkcia*, 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.

Funkcia **BDD_reduce** má slúžiť na redukciu existujúceho (zostaveného) binárneho rozhodovacieho diagramu. Aplikovaním tejto funkcie sa nesmie zmeniť Booleovská funkcia, ktorú BDD opisuje. Cieľom redukcie je iba zmenšiť BDD odstránením nepotrebných (redundantných) uzlov. Funkcia **BDD_reduce** dostane ako argument ukazovateľ na existujúci BDD (*bdd*), ktorý sa má redukovať. Redukcia BDD sa vykonáva priamo nad BDD, na ktorý ukazuje ukazovateľ *bdd*, a preto nie je potrebné vrátiť zredukovaný BDD návratovou hodnotou (na zredukovaný BDD bude totiž ukazovať pôvodný ukazovateľ *bdd*). Návratovou hodnotou funkcie **BDD_reduce** je číslo typu int (integer), ktoré vyjadruje počet odstránených uzlov. Ak je toto číslo záporné, vyjadruje nejakú chybu (napríklad ak BDD má ukazovateľ na koreň BDD rovný NULL). Samozrejme, funkcia **BDD_reduce** má aktualizovať aj informáciu o počte uzlov v BDD.

Funkcia **BDD_use** má slúžiť na použitie BDD pre zadanú (konkrétnu) kombináciu vstupných premenných Booleovskej funkcie a zistenie výsledku Booleovskej funkcie pre túto kombináciu vstupných premenných. V rámci tejto funkcie „prejdete“ BDD stromom smerom od koreňa po list takou cestou, ktorú určuje práve zadaná kombinácia vstupných premenných. Argumentami funkcie **BDD_use** sú ukazovateľ s názvom *bdd* ukazujúci na BDD (ktorý sa má použiť) a ukazovateľ s názvom *vstupy* ukazujúci 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 a hodnota na tomto indexe reprezentuje hodnotu tejto premennej (t.j. pre premenné A, B, C a D, kedy A a C sú jednotky a B a D 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’. V prípade chyby je táto návratová hodnota záporná, podobne ako vo funkcii **BDD_reduce**.

Okrem implementácie samotných funkcií na prácu s BDD je potrebné vaše riešenie dôkladne otestovať. Vaše riešenie musí byť 100% korektné. 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 a nakoniec 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. V rámci testovania tiež vyhodnocujte percentuálnu mieru zredukovania BDD (t.j. počet odstránených uzlov / pôvodný počet uzlov).

Príklad veľmi jednoduchého testu (len pre pochopenie problematiky):

```
#include <string.h>
int main(){
  BDD* bdd;
  bdd = BDD_create("AB+C"); // alebo vektorom BDD_create("01010111")
  BDD_reduce(bdd);
  if (BDD_use("000") == '1')
    printf("error, for A=0, B=0, C=0 it should be 0.\n");
  if (BDD_use("001") == '0')
    printf("error, for A=0, B=0, C=1 it should be 1.\n");
  if (BDD_use("010") == '1')
    printf("error, for A=0, B=1, C=0 it should be 0.\n");
  if (BDD_use("011") == '0')
    printf("error, for A=0, B=1, C=1 it should be 1.\n");
  if (BDD_use("100") == '1')
    printf("error, for A=1, B=0, C=0 it should be 0.\n");
  if (BDD_use("101") == '0')
    printf("error, for A=1, B=0, C=1 it should be 1.\n");
  if (BDD_use("110") == '0')
    printf("error, for A=1, B=1, C=0 it should be 1.\n");
  if (BDD_use("111") == '0')
    printf("error, for A=1, B=1, C=1 it should be 1.\n");
  return 0;
}
```

Okrem implementácie vášho riešenia a jeho testovania vypracujte aj dokumentáciu, v ktorej opíšete vaše riešenie, jednotlivé funkcie, vlastné štruktúry, spôsob testovania a výsledky testovania, ktoré by mali obsahovať (priemernú) percentuálnu mieru zredukovania BDD a (priemerný) čas vykonania vašich funkcií. Dokumentácia musí obsahovať hlavičku (kto, aké zadanie odovzdáva), stručný opis použitého algoritmu s názornými nákresmi/obrázkami a krátkymi ukážkami zdrojového kódu, vyberajte len kód, na ktorý chcete extra upozorniť. Pri opise sa snažte dbať osobitý dôraz na zdôvodnenie správnosti vášho riešenia – teda dôvody prečo je dobré/správne, spôsob a vyhodnotenie testovania riešenia. Nakoniec musí technická dokumentácia obsahovať odhad výpočtovej (časovej) a priestorovej (pamäťovej) zložitosti vášho algoritmu. Celkovo musí byť cvičiacemu jasné, že viete čo ste spravili, že viete odôvodniť, že to je správne riešenie, a viete aké je to efektívne.

Riešenie zadania sa odovzdáva do miesta odovzdania v AIS do stanoveného termínu (oneskorené odovzdanie je prípustné len vo vážnych prípadoch, ako napr. choroba, o možnosti odovzdať zadanie oneskorene rozhodne cvičiaci, príp. aj o bodovej penalizácii). Odovzdáva sa jeden **zip** archív, ktorý obsahuje zdrojové súbory s implementáciou riešenia a testovaním + jeden súbor s dokumentáciou vo formáte **pdf**. **Vyžaduje sa tiež odovzdanie programu**, ktorý slúži na testovanie a odmeranie efektívnosti týchto implementácií ako jedného samostatného zdrojového súboru (obsahuje funkciu **main**).

Hodnotenie

Môžete získať celkovo 15 bodov, **nutné minimum je 6 bodov**.

Za implementáciu riešenia (3 funkcie) je možné získať celkovo 8 bodov, z toho 2 body sú za funkciu **BDD_create**, 4 body za funkciu **BDD_reduce** a 2 body za funkciu **BDD_use**. Pre úspešné odovzdanie implementácie musíte zrealizovať aspoň funkcie **BDD_create** a **BDD_use**. Za testovanie je možné získať 4 body (treba podrobne uviesť aj v dokumentácii) a za dokumentáciu 3 body (bez funkčnej implementácie 0 bodov). Body sú ovplyvnené aj prezentáciou cvičiacemu (napr. keď neviete reagovať na otázky vzniká podozrenie, že to **nie je vaša práca, a teda je hodnotená 0 bodov)**.