Slovenská technická univerzita v Bratislave
Fakulta Informatiky a Informačných Technológií

Datové štruktúry a algoritmy

Zadanie 3 – Binárne rozhodovacie diagramy

Emma Macháčová

Meno cvičiaceho : Ing. Dominika Dolhá
Čas cvičení : pondelok 9:00
Dátum vytvorenia : 13. apríl 2021

Emma Macháčová
ID: 103037

Obsah

Emma Macháčová
ID: 103037

Cieľ práce

Cieľom projektu v rámci tohto zadania je vytvoriť program, ktorý bude vedieť vytvoriť, redukovať a použiť dátovú štruktúru BDD (Binárny Rozhodovací Diagram) so zameraním na využitie pre reprezentáciu Booleovských funkcií.

Konkrétne je cieľom implementovať **tieto funkcie**:

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

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. Návratovou hodnotou funkcie BDD_create je ukazovateľ na zostavený binárny rozhodovací diagram, ktorý je reprezentovaný vlastnou štruktúrou 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. 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).

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 konkrétnu kombináciu vstupných premenných Booleovskej funkcie.

Emma Macháčová
ID: 103037

Opis riešenia

V projekte pracujem s troma **vlastnými definovanými štruktúrami**: binaryFunction, binaryDecisionDiagram a diagramNode.

- Štruktúra **binaryFunction** reprezentuje binárnu funkciu. Obsahuje pole, v ktorom sa uchováva vektor (sekvenciu 0/1), ktorým je daná.
- Štruktúra **diagramNode** predstavuje jeden uzol binárneho rozhodovacieho diagramu. Jeho zložkami sú binárna funkcia, a ukazovatele na nasledujúce uzly (ľavý a pravý).

```
13  // funkcia
14  typedef struct binaryFunction {
15      char vektor[200000]; // vektor
16  } BF;
17
18  // diagram
19  typedef struct binaryDecisionDiagram {
20      int pocetPremennych; // pocet premennych (A B C D) diagramu
21      int pocetUzlov; // pocet uzlov
22      struct diagramNode* koren; // root, prvy uzol
23      struct  diagramNode* listTrue; // po redukcii jediny list s hodnotou 1
24      struct diagramNode* listFalse; //    ----- || -----       s hodnotou 0
25  } BDD;
26
27  // uzol
28  typedef struct diagramNode {
29      BF funkcia;
30      int mark; // udava, ci je to jeden z dvoch samostatnych listov (redukcia)
31      struct diagramNode* false; // lavy child
32      struct  diagramNode* true; // pravy child
33  } node;
```

- Štruktúra **binaryDecisionDiagram** reprezentuje už zostavený binárny rozhodovací diagram. Obsahuje počet premenných, počet uzlov a ukazovateľ na koreň (prvý uzol).

Okrem štruktúr a funkcií zo zadania využívam aj vlastné **pomocné funkcie**:

```
40  // pomocne funkcie
41  int getPocetPremennych (BF bfunkcia); // ziska pocet premennych diagramu
42  int jeSymetricky (char* vektor);
43  int jeZbytocny (char* vektor); // posudi ci sa ma uzol zredukovat
44  char* leftHalf (char* vektor); // ziska lavu polku vektora
45  int malyTest (); // test na ukazku
46  void najdiPrebytocne (node* uzol, BDD *bdd); // pomocna funkcia pre reduce, pouziva jeZbytocny
47  void najdiSymetricke (node* uzol, BDD *bdd);
48  void generujVsetkyMoznosti (int n, char arr[], int i);
49  void printOut (node *uzol); // vypis korenov (listov)
50  char* rightHalf (char* vektor); // ziska pravu polku vektora
51  void rozdel(node *uzol, BDD *diagram); // vetvenie diagramu
52  node* trasa (node* uzol, char *vstupy, int index); // pomocna funkcia pre use
53  void spojKonce (node* uzol, BDD *bdd, node* true, node* false); // pomocna funkia pre reduce
54  void uvolni (node* node); // uvolnovanie uzlov
55  int velkyTest (); // velky test, pouziva generujVsetkyMoznosti
56  BF *vytvorFunkciu (int premenne); // vytvara nahodnu funkciu
57  char* vytvorVstup (BDD *diagram); // vytvara nahodne vstupy
```

Diagram je vytváraný pomocou vstupnej funkcie (vektora) z hora nadol, postupným delením (vetvením) funkcie.

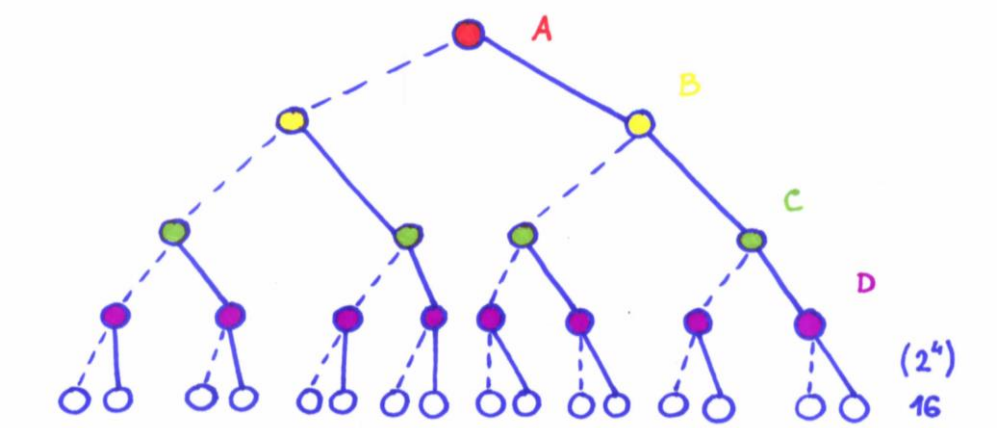
Emma Macháčová
ID: 103037

Funkcia BDD_create

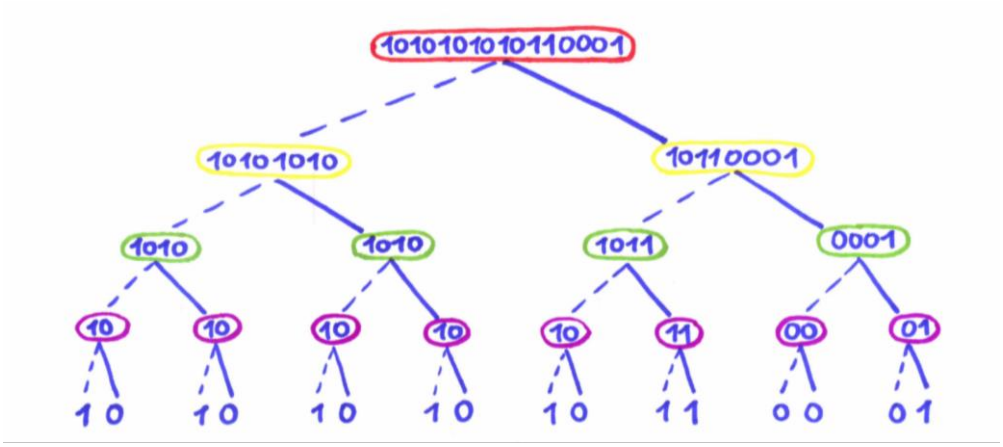
Funkcia má slúžiť na zostavenie úplného binárneho rozhodovacieho diagramu, ktorý má reprezentovať/opisovať zadanú Booleovskú funkciu. Návratovou hodnotou funkcie BDD_create je ukazovateľ na zostavený binárny rozhodovací diagram.

Funkcia najskôr alokuje koreň (prvý uzol), štruktúru diagramu, a vstupný parameter binárnej funkcie vloží do (teraz alokovaného) prvého uzla.

Program ďalej pomocnou funkciou **getPocetPremennych** vypočíta to, koľko premenných má daná funkcia. Tento počet je logaritmom so základom dva z počtu outputov (listov) funkcie (ak má vstupný vektor dĺžku 16, počet listov diagramu je rovnako 16, a logaritmus so základom 2 vráti 4 – čo je zodpovedajúce počtu premenných funkcie).



Prvý uzol sa nastaví ako koreň diagramu, a počet uzlov diagramu sa nastaví na 1. Následne sa na rozvetvenie diagramu volá rekurzívna funkcia **rozdel.** Táto funkcia od koreňu k listom postupne každému uzlu vytvorí dvoch potomkov, a pomocou funkcií **leftHalf** a **rightHalf** rozdelí pôvodný vektor na pravú a ľavú polovicu, ktoré im pridelí ako ich funkciu. Rekurzia pokračuje dovtedy, kým je ešte možné vektor deliť na dve polovice.



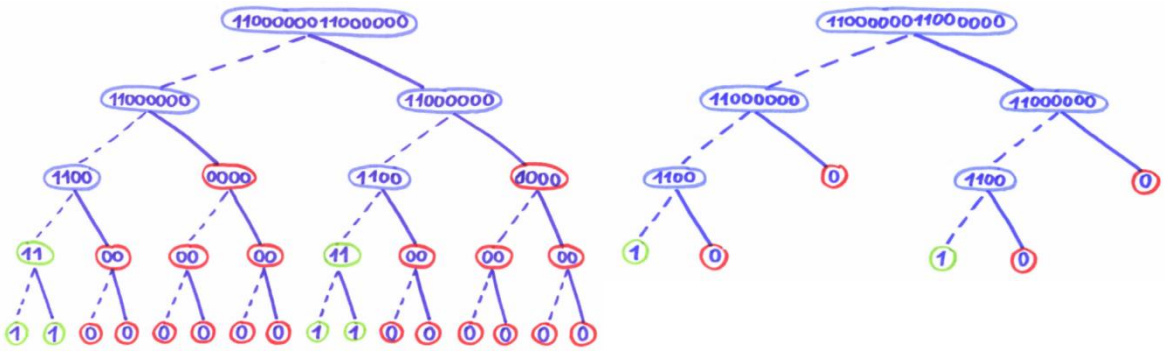
Keď sa ukončí delenie, funkcia vráti ukazovateľ na vytvorený binárny rozhodovací diagram.

Emma Macháčová
ID: 103037

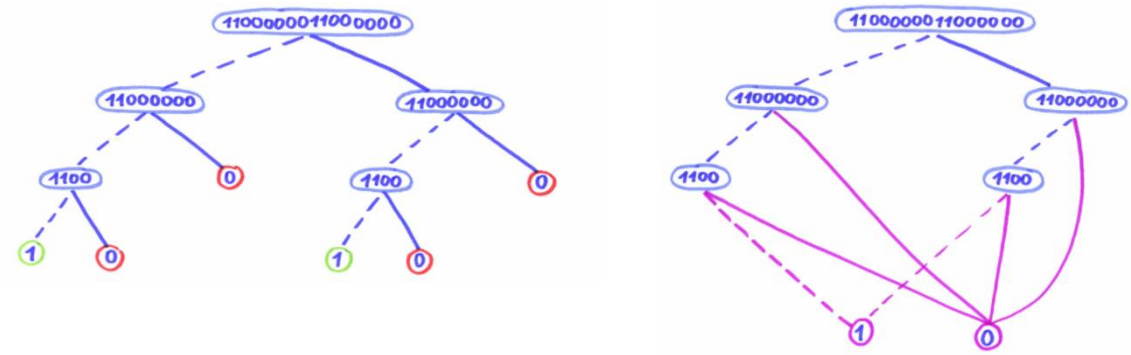
Funkcia BDD_reduce

Funkcia má slúžiť na redukciu existujúceho binárneho rozhodovacieho diagramu. Cieľom redukcie je zmenšiť BDD odstránením nepotrebných 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. Návratovou hodnotou funkcie BDD_reduce je číslo typu int (integer), ktoré vyjadruje počet odstránených uzlov.

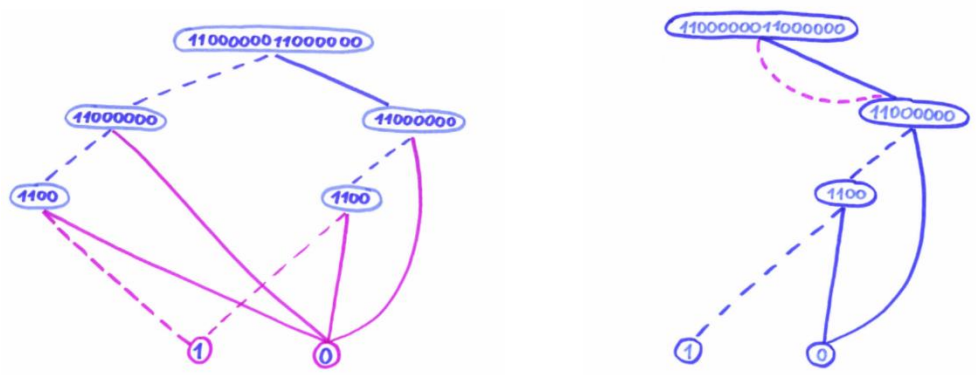
Funkcia najskôr overí to, či bol diagram vytvorený. Následne získa koreň diagramu, a zavolá s ním pomocnú rekurzívnu funkciu najdiPrebytocne. Tá vyhodnotí prebytočné uzly (také, čo obsahujú len 1 alebo len 0), a odstráni ich.



Po odstránení zbytočných uzlov sa volá pomocná rekurzívna funkcia spojKonce, ktorá nahradí všetky listy stromu dvoma špeciálnymi, na ktoré ukazuje aj štruktúra BDD.



Ako posledná sa volá pomocná funkcia najdiSymetricke, ktorá funguje podobne ako funkcia najdiPrebytocne, ale hľadá uzly, ktoré sú symetrické (11001100, 0000110000001100..), a odstráni jedného z ich potomkov.



Emma Macháčová
ID: 103037

Funkcia BDD_use

Funkcia má slúžiť na použitie BDD pre zadanú (konkrétnu) kombináciu vstupných premenných Booleovskej funkcie a zistenie výsledku Booleovskej funkcie pre túto kombináciu vstupných premenných.

Funkcia najskôr kontroluje, či existuje binárny diagram. Pokiaľ nie, vráti „x“.

Ak diagram existuje, zavolá pomocnú rekurzívnu funkciu **trasa**. Tá rekurzívne prejde strom od koreňa po listy. To, do ktorej strany sa bude pohybovať, závisí od premennej lr (left / right), ktorá nadobúda hodnotu pre danú premennú na tej úrovni stromu, podľa poľa vstupu – „string“ 0/1 (čím hlbšie v strome tým vyšší index – čo úroveň to index). Pole vstupu generuje buď funkcia **vytvorVstup** (jeden náhodný vstup pre daný počet premenných diagramu), alebo funkcia **otestujVsetkyMoznosti** (rekurzívna funkcia, vytvorí postupne všetky možné vstupy pre daný počet prvkov diagramu).

Ak bol strom redukovaný a funkcia narazí na list pred prečítaním celého vstupu, vráti tento list (preto že ďalšie čítanie vstupu by bolo zbytočné).

Po ukončení funkcie trasa, funkcia BDD_use overí hodnotu listu, ku ktorému funkcia trasa prišla. Pokiaľ je táto hodnota 0, funkcia vráti 0. Ak je to 1, funkcia vráti 1.

Správnosť výsledkov sa overuje porovnávaním ich s hodnotou vektora na príslušnom indexe.

```
// testovanie spravnosti vysledkov
if (VYSLEDKY[Y][Z] != diagram->koren->funkcia.vektor[i]) {
    NEZHOD++;
}
```

Emma Macháčová
ID: 103037

Testy

V rámci testovanie bolo potrebné náhodným spôsobom generovať Booleovské funkcie, podľa ktorých sa vytvárajú BDD pomocou funkcie BDD_create. Vytvorené BDD je potrebné následne zredukovať funkciou BDD_reduce a nakoniec overiť 100% funkčnosť zredukovaných BDD opakovaným (iteratívnym) volaním funkcie BDD_use s postupným použitím všetkých možných kombinácií 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 sa tiež vyhodnocuje percentuálna miera zredukovania BDD (t.j. počet odstránených uzlov / pôvodný počet uzlov).

Výsledky testovania majú obsahovať (priemernú) percentuálnu mieru zredukovania BDD a (priemerný) čas vykonania funkcií. Zhodnotenie má obsahovať odhad výpočtovej (časovej) a priestorovej (pamäťovej) zložitosti algoritmu.

Testovacia funkcia

Program obsahuje dve testovacie funkcie: velkyTest, test a malyTest.

Malý test funguje ako ukážka na strome, ktorý používam ako ilustračný aj v tejto dokumentácií. Je menší a preto na ňom ide jednoducho skontrolovať správnosť programu.

Veľký test funguje podľa zadania – pre 14 premenných, vykoná 2000 testov (čo test to iná funkcia – náhodne generovaná funkciou vytvorFunkciu), a v rámci týchto 2000 testov otestuje všetky možnosti vstupov pre premenné diagramu (2^{14} vstupov).

Tento test najskôr vytvorí funkciu, a podľa nej sa vytvorí diagram. Vypíšu sa kontrolné správy, ktoré je možné aj uložiť do súboru.

Následne sa pre neredukovaný diagram otestujú všetky možné vstupy (2^n), ktoré boli vytvorené dopredu funkciou generujVsetkyMoznosti. Výsledky testov sa uložia do poľa.

Po otestovaní neredukovaného diagramu prebehne redukcia funkciou BDD_reduce, a v redukovanom diagrame sa rovnako otestujú všetky vstupy, a výsledky sa tiež zapíšu do poľa.

Po tom, čo prebehnú testy v oboch diagramoch, porovnajú sa výsledky a počet nezhôd sa zaráta. Diagram sa po dokončení každého cyklu uvoľní.

Toto prebehne toľkokrát, koľko testov užívateľ zadá (alebo 2000 preddefinovaných.. počet testov aj počet premenných diagramu sa dá jednoducho zmeniť).

Funkcia test pracuje tak ako funkcia velkyTest, až na to, že vykoná testy len pre nezredukovaný diagram (spolu s jeho redukciou).

Keď prebehnú všetky testy, program vypíše počet nezhôd, počet testov, počet pôvodných a počet odstránených uzlov, a percentuálnu mieru redukcie uzlov.

Emma Macháčová
ID: 103037

Výsledky testov

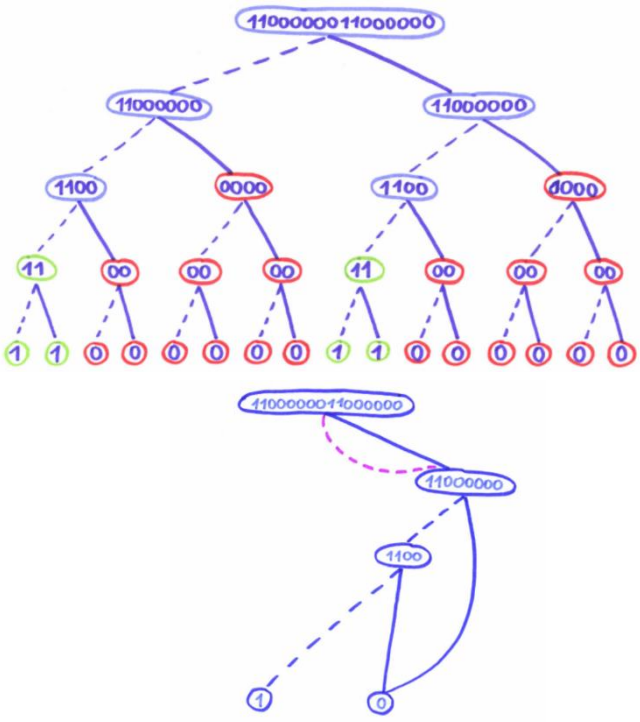
Funkcia **malyTest** (ukážka):

```
------------------ MALE TESTOVANIE ------
Funkcia: 1100000011000000
Pocet outputov funkcie: 16
Pocet premennych pre diagram: 4

Pocet uzlov diagramu: 31
Vstup: 0000
Vysledna boolean hodnota je: 1
Vstup: 0010
Vysledna boolean hodnota je: 0

PO REDUKCII:
Pocet uzlov diagramu: 5
Vstup: 0000
Vysledna boolean hodnota je: 1
Vstup: 0010
Vysledna boolean hodnota je: 0
Percentualna miera zredukovania: 83.87 %

Process finished with exit code 0
```



Funkcia **velkyTest** (nezredukovaný aj zredukovaný diagram, porovnanie výsledkov jednotlivo aj navzájom):

- 2000 funkcií, 13 premenných (5.3 min)

```
PERCENTUALNA MIERA ZREDUKOVANIA BDD: 73.20 %
POCET NEZHODNYCH TESTOV 0
Pocet premennych testovanych funkcii: 13
Pocet povodnych uzlov: 32766000, pocet odstranenych uzlov: 23984652
Pocet testov: 32768000
Cas vykonania 322.437500 s
```

- 2000 funkcií, 14 premenných (11 min)

```
PERCENTUALNA MIERA ZREDUKOVANIA BDD: 73.20 %
POCET NEZHODNYCH TESTOV 0
Pocet premennych testovanych funkcii: 14
Pocet povodnych uzlov: 65534000, pocet odstranenych uzlov: 47970202
Pocet testov: 65536000
Cas vykonania 667.062500 s
```

Emma Macháčová
ID: 103037

Funkcia **test**:

- 2000 funkcií, 13 premenných (5.2 min)

```
TESTOVANIE LEN NEREDUKOVANEHO DIAGRAMU:
PERCENTUALNA MIERA ZREDUKOVANIA BDD: 73.20 %
Pocet premennych testovanych funkcii: 13
Pocet povodnych uzlov: 32766000, pocet odstranenych uzlov: 23984652
Pocet testov: 16384000
Pocet nezhod: 0
Cas vykonania 312.546875 s
```

- 2500 funkcií, 13 premenných (6.9 min)

```
TESTOVANIE LEN NEREDUKOVANEHO DIAGRAMU:
PERCENTUALNA MIERA ZREDUKOVANIA BDD: 73.20 %
Pocet premennych testovanych funkcii: 13
Pocet povodnych uzlov: 40957500, pocet odstranenych uzlov: 29978932
Pocet testov: 20480000
Pocet nezhod: 0
Cas vykonania 419.859375 s
```

- 2000 funkcií, 14 premenných (10.7 min)

```
TESTOVANIE LEN NEREDUKOVANEHO DIAGRAMU:
PERCENTUALNA MIERA ZREDUKOVANIA BDD: 73.20 %
Pocet premennych testovanych funkcii: 14
Pocet povodnych uzlov: 65534000, pocet odstranenych uzlov: 47970202
Pocet testov: 32768000
Pocet nezhod: 0
Cas vykonania 643.468750 s
```

Priemerná komplexnosť je O(log n).

Priestorová zložitosť je priemerne konštantne lineárna.