Politechnika Śląska w Gliwicach Wydział Informatyki, Elektroniki i Informatyki

Podstawy Programowania Komputerów

Czerwono Czarni

---

autor	Daniel Wilk
prowadzący	dr inż, Wojciech Łabaj
rok akademicki	2023/2024
kierunek	Teleinformatyka
rodzaj studiów	SSI
semestr	1
termin laboratorium/ćwiczeń	Środa, 15:45 - 17:15
grupa	2
sekcja	4
termin oddania sprawozdania	2024-02-20
data oddania sprawozdania	2024-02-19

---

1. Treść zadania

Napisać program sortujący liczby rzeczywiste w pewnym zbiorze. Liczby podawane są w dość specyficzny sposób. Liczba może być dodana lub usunięta ze zbioru. Dodanie liczb jest realizowane przez komendę add, po której może wystąpić jedna lub więcej liczb (rozdzielonych białymi znakami). Komenda remove usuwa podaną po niej liczbę (lub liczby rozdzielone białymi znakami) ze zbioru. Komenda print powoduje wypisanie liczb zawartych w zbiorze w porządku rosnącym. Po komendzie tej można podać nazwę pliku, wtedy wartości zostaną zapisane do tegoż pliku zamiast na standardowe wyjście. Komenda graph wypisuje drzewo w postaci graficznej – głębsze poziomy drzewa są wypisywane z coraz większym wcięciem. Dodatkowo wartości w węzłach czarnych są wypisywane w nawiasach kwadratowych, np. [13], w węzłach czerwonych – w nawiasach okrągłych, np. (13). Podobnie jak w przypadku komendy print po komendzie graph można podać nazwę pliku do zapisu. Jeżeli komendy print i graph są użyte do zapisu do pliku, możliwe jest poprzedzenie nazwy pliku znakiem +. Wtedy plik nie zostanie nadpisany, ale nowa treść zostanie dopisana na końcu pliku, nie niszcząc dotychczasowej jego zawartości. Znak % rozpoczyna komentarz do końca linii. Każda komenda jest zapisana w osobnej linii. Jeżeli zostanie podana niepoprawna komenda, program ignoruje ją. Przykładowy plik wejściowy: % przykladowy plik wejsciowy add -3.14 43.9 4 graph % wypisane z wcieciami i z oznaczeniami kolorow wezlow remove 4 print % wypisane na ekran add 3.45 -0.32 print test-1.txt % wypisanie do pliku add 490 32.3 print % na ekran print + test-1.txt % dopisanie do pliku Po komendzie graph zostanie wypisane drzewo: (-3.14) [4] (43.9) Program uruchamiany jest z linii poleceń z potrzebnymi przełącznikami, natomiast uruchomienie programu bez parametrów powoduje wypisanie krótkiej instrukcji.

2. Analiza zadania

Zagadnienie przedstawia problem samoorganizującego się drzewa binarnego.

2.1. Struktury danych

W programie wykorzystano drzewo czerwono-czarne. Użyta została struktura danych zawierająca informacje o konkretnym węźle oraz algorytmy wykonujące operacje wstawiania i usuwania węzłów.

3. Specyfikacja programu

3.1. Komendy

add <liczby> - wykonuje operacje wstawiania węzła. Można podać jedną lub więcej liczb rozdzielonych spacją.

remove <liczby> - wykonuje operacje usuwania węzła. Można podać jedną lub więcej liczb rozdzielonych spacją.

print - wykonuje operacje wypisania wszystkich liczb zawartych w węzłach w porządku rosnącym.

print <nazwa pliku> - wykonuje to samo co komenda print ale zapisuje wynik do pliku.

print +<nazwa pliku> - wykonuje to samo co komenda print ale zapisuje wynik do pliku bez usuwania zawartych już danych w tym pliku.

graph - wykonuje wypisania drzewa w postaci grafu.

graph <nazwa pliku> - wykonuje to samo co komenda graph ale zapisuje wynik do pliku

graph +<nazwa pliku> - wykonuje to samo co komenda graph ale zapisuje wynik do pliku bez usuwania zawartych już danych w tym pliku.

3.2. Struktury

```
struct Node {
  double val;
  Node *parent;
  Node *left;
  Node *right;
  Color color;
};
```

Typ służy do przechowywania informacji na temat poszczególnych węzłów

---

```
enum Color {
    BLACK,
    RED
};
```

Typ służy do identyfikacji węzłów w drzewie czerwono czarnym

3.3. Algorytmy

```
void tree_insert(double key) {
    // Inicjalizacja nowego węzła
    while (x != nullptr && x != dummy_node) {
        y = x;
        if (node->val < x->val) {
            x = x->left;
        }
        else {
            x = x->right;
        }
    }

    // Poszukiwanie odpowiedniego miejsca
    // dla nowego węzła

    tree_organize_after_insert(node);
    // Uruchomienie funkcji organizującej drzewo
}
```

```
void tree_delete_node(double key) {
    // Szukanie węzła o danym kluczu
    while (node != dummy_node) {
        if (node->val == key) {
            z = node;
        }

        if (node->val <= key) {
            node = node->right;
        }
        else {
            node = node->left;
        }
    }

    if (z == dummy_node) {
        cout << "Key not found in the tree" << endl;
        return;
    }

// Po wykonaniu operacji usuwania zmień kolor sąsiednich węzłów
// Zorganizuj drzewo
    if (y_original_color == BLACK) {
        tree_organize_after_delete(x);
    }
    delete z;
}
```

4. Wnioski

Drzewo czerwono czarne jest efektywnym narzędziem do organizacji danych. Operacje wykonywane na tej strukturze danych są całkiem szybkie.
Ich złożoność czasowa to log(n). Strukturalna stabilność drzewa czerwono czarnego sprawiają, że jest ona mniej podatna na gorsze przypadki.

5. Literatura

[1] Robert Sedgewick, Algorithms in C, Parts 1-4: Fundamentals, Data Structures, Sorting, Searching