Wizualizacja i edycja drzewa BST w C++ z użyciem Qt

Mikołaj Bieniek

Marzec 2025

Spis treści

1 Wstęp

Projekt stanowi aplikację graficzną pozwalającą użytkownikowi na interakcję z drzewem binarnym poszukiwań (BST). Umożliwia dodawanie, usuwanie oraz przeszukiwanie elementów, przy czym wszystkie zaimplementowane operacje są natychmiast wizualizowane w interfejsie graficznym zbudowanym na frameworku Qt.

2 Opis struktury projektu

Projekt składa się z następujących plików:

- `bst.cpp` / `bst.h` — implementacja struktury danych BST w C++.
- `mainwindow.cpp` / `mainwindow.h` — logika interfejsu użytkownika oraz połączenie GUI z logiką drzewa.
- `mainwindow.ui` — plik opisujący układ interfejsu graficznego (stworzony w Qt Designer).
- `bsttree.pro` — plik konfiguracyjny Qt (projektowy).

3 Opis implementacji drzewa BST

3.1 Wstawianie elementu

Dodanie nowego elementu odbywa się iteracyjnie, zgodnie z klasycznymi zasadami BST:

- Elementy mniejsze trafiają do lewego poddrzewa.
- Elementy większe do prawego.

Po każdej operacji aktualizowana jest liczba elementów (`size`).

3.2 Usuwanie elementu

Usuwanie realizuje trzy przypadki:

- Węzeł nie ma dzieci — usuwany bezpośrednio.
- Węzeł ma jedno dziecko — dziecko przejmuje jego miejsce.
- Węzeł ma dwoje dzieci — następuje zamiana z następnikiem (successor) i usunięcie go.

3.3 Wyszukiwanie elementu

Funkcja `search(int key)` zwraca wskaźnik na węzeł, jeśli znajdzie go w drzewie, lub `nullptr`, jeśli go nie ma.

3.4 Obliczanie głębokości drzewa

Zastosowano funkcję rekurencyjną `calculateDepth(Node*)`, która dynamicznie oblicza wysokość drzewa. Wcześniejsze podejście z przechowywaną zmienną `depth_` zostało usunięte na rzecz dokładniejszego i bezbłędnego obliczania.

3.5 Inne metody pomocnicze

- `minimum()` — zwraca najmniejszy klucz.
- `maximum()` — zwraca największy klucz.
- `inorder()` — zwraca listę elementów w porządku inorder.

4 Wizualizacja drzewa

Wizualizacja odbywa się z użyciem `QGraphicsScene`. Węzły reprezentowane są jako białe koła z numerem wewnątrz. Dodatkowo białe koła zwiększają się, jeśli numer wewnątrz staje się zbyt duży. Krawędzie między rodzicami i dziećmi rysowane są jako linie. Po każdej operacji (`insert`, `delete`, `delete whole tree`) scena jest aktualizowana.

5 Interfejs użytkownika (Qt GUI)

Komponenty użyte w interfejsie graficznym:

- `QLineEdit` — pole wprowadzania liczby.
- `QPushButton` — przyciski do wstawiania, usuwania, szukania, czyszczenia drzewa.
- `QGraphicsView` — wizualizacja drzewa.
- `QLabel` — dynamiczne informacje o drzewie: liczba elementów, głębokość, minimum, maksimum, traversale.

6 Podsumowanie

Projekt ten ukazuje praktyczne zastosowanie struktury danych BST wraz z graficzną prezentacją jej działania. Obliczanie głębokości w czasie rzeczywistym, aktualizacja sceny i intuicyjny oraz prosty interfejs użytkownika sprawiają, że aplikacja nadaje się do praktycznego testowania oraz dydaktyki.

7 Źródła

- Cormen T. H., Leiserson C. E., Rivest R. L., Stein C., *Wprowadzenie do algorytmów*, PWN.
- `https://www.geeksforgeeks.org/binary-search-tree-data-structure/`
- `https://doc.qt.io/`
- `https://www.programiz.com/dsa/binary-search-tree`