Algorytmy i struktury danych

Laboratorium - lista 4

Termin wysłania: 2025-06-01

Zadanie 1. [20 p.]

Zaimplementuj strukturę BST (Binary Search Tree) (przyjmując, jak na Listach 2. i 3., że kluczami są liczby całkowite) wraz z następującymi operacjami:

- `insert(k)` - wstawianie nowego wystąpienia klucza `k` do drzewa,
- `delete(k)` - usuwanie jednego wystąpienia klucza `k` w drzewie (jeśli istnieje).
- `height` - funkcja zwracająca bieżącą wysokość drzewa.

Zademonstruj poprawność swojej implementacji w prezentacji w następujących dwóch przypadkach, dla *n*=30:

1. wstawianie rosnącego ciągu kluczy $1, 2, \ldots, n$ operacjami `insert`, a następnie usuwanie losowej permutacji tego ciągu operacjami `delete`
2. wstawienie losowej permutacji ciągu kluczy $1, 2, \ldots, n$ operacjami `insert`, a następnie usuwanie losowej permutacji tego ciągu operacjami `delete`

W każdym kroku drukuj wykonywaną operację (np. '`insert 48`' albo '`delete 20`') oraz stan drzewa po tej operacji, w takiej postaci, aby widoczna była jego struktura, na przykład tak jak niżej, gdzie 'lewa-prawa strona' zostały wyświetlone jako kierunki 'góra-dół':

```
         /-[48]
        /-[54]
        | \-[257]
      /-[282]
      | \-[310]
    /-[352]
  /-[370]
  | | /-[370]
  | \-[372]
  |   \-[484]
-[517]
  |     /-[527]
  |     | \-[665]
  |   /-[713]
  | /-[725]
  | | \-[736]
  | |   \-[788]
  \-[924]
    | /-[939]
    \-[978]
```

Na potrzeby wyświetlania drzew w swojej aplikacji możesz przerobić sobie przykładowy program dostępny pod linkiem:
https://gist.github.com/mki1967/a3f79891f4a236f1d55d7f8f554435fe

Zadanie 2. [10 p.]

Przeprowadź eksperymenty badające złożoność dla dużych danych (po 20 testów dla wartości *n* równych: 10 000, 20 000, ..., 100 000), dla takich samych scenariuszy jak w Zadaniu 1. (tj. wstawianie rosnącego ciągu kluczy $1, 2, \ldots, n$ i usuwanie losowej permutacji tego ciągu oraz wstawianie losowej permutacji ciągu kluczy $1, 2, \ldots, n$ i usuwanie losowej permutacji tego ciągu) ale bez wyświetlania wykonywanych operacji i drzew.
Jako miary złożoności każdej operacji zliczaj:

- liczby porównań między kluczami,
- liczby odczytów i podstawień wskaźników łączących elementy struktury drzewa,
- wysokość drzewa po każdej operacji.

Dla każdego *n* zliczaj zarówno średni koszt, jak i maksymalny napotkany koszt pojedynczej operacji.
Przygotuj obrazy z wykresami uzyskanych wyników dla każdej z tych miar.

Zadanie 3. [30 p.]

Podobnie jak w zadaniu 1, zaimplementuj strukturę RB BST (Red-Black Binary Search Tree), oraz demonstracje jej działania dla *n*=30.

Zadanie 4. [10 p.]

Wykonaj odpowiednik zadania 2. dla drzew RB BST.

Zadanie 5. [20 p.]

Podobnie jak w zadaniu 1, zaimplementuj Splay Tree, oraz demonstracje jej działania dla *n*=30.

Zadanie 6. [10 p.]

Wykonaj odpowiednik zadania 2. dla drzew Splay Tree.

Literatura

[1] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. Introduction to Algorithms. The MIT Press, 3rd edition, 2009.
[2] Daniel D. Sleator and Robert E. Tarjan. Self-adjusting binary search trees. J. ACM, 32(3):652–686, July 1985.
[3] Robert E. Tarjan. Data Structures and Network Algorithms. Society for Industrial and Applied Mathematics, USA, 1983.