Algorytmy i struktury danych

Lista nr 4

Waga listy: 3

(termin oddania: 5 laboratorium - ostatnie własne zajęcia przed 24 maja)

Zadanie 1 Napisz program, który symuluje działanie wybranych struktur danych przechowujących ciągi znaków (przyjmujemy porządek leksykograficzny). Program powinien przyjmować jako parametr wejściowy typ struktury:

(1pkt) --type bst drzewo BST,

(1pkt) --type rbt drzewo czerwono–czarne,

(1pkt) --type splay samoorganizujące drzewo binarne,

Każda ze struktur powinna udostępniać przynajmniej poniższe funkcjonalności podawane na standardowym wejściu

- insert s - wstaw do struktury ciąg s (jeśli na początku lub końcu ciągu znajduje się znak spoza klasy [a-zA-Z] to znak ten jest usuwany)
- delete s - jeśli struktura nie jest pusta i dana wartość s istnieje, to usuń element s
- search s - sprawdź czy w strukturze przechowywana jest wartość s (jeśli tak to wypisz 1, w p. p. wypisz 0)
- load f - dla każdego, oddzielonego białym lub interpunkcyjnym znakiem, wyrazu z pliku f wykonaj operację insert, lub zwróć informację o nieistniejącym pliku
- inorder - wypisz elementy drzewa w posortowanej kolejności (od elementu najmniejszego do największego) lub, dla struktur pustych pustą linię.

Wynik powinien być wypisywany na standardowe wyjście, a na standardowym wyjściu błędów powinny być wypisywane w kolejności: czas działania całego programu, liczba operacji każdego typu, maksymalna liczba elementów (maksymalne zapełnienie struktury w czasie działania programu), końcowa liczba elementów w strukturze, oraz całkowitą liczbę porównań i modyfikowanych elementów drzewa (zmiana pól).

Przeprowadź eksperymenty pozwalające oszacować średni czas działania każdej z operacji.

Wejście: Wejście składa się z $n + 1$ linii. W pierwszej, znajduje się liczba n określająca liczbę wykonywanych operacji, w liniach 2–$(n + 1)$ znajdują się kolejne operacje zgodnie z ich specyfikacją. Program może wykorzystywać więcej niż jeden wątek, jednak operacje muszą być wykonane w zadanej kolejności. Długość pojedynczego ciągu znaków nie przekracza 100, natomiast $n + 1$ nie przekracza zakresu Integera.

Wyjście: Wyjście składa się z $k \leqslant n$ linii, będących wynikami kolejnych operacji podanych na wejściu.

Przykład: Przykładowe wywołanie

```
./main --type rbt <./input >out.res
```

Zadanie 2 (1pkt) Wykonaj eksperymenty polegające na wstawieniu każdego słowa z pliku, spytania się o każde słowo z pliku, i usunięcia każdego słowa z pliku. Zaprezentuj sprawozdanie, które pozwoli określić, które z zaimplementowanych w zadaniu 1 struktur są najbardziej efektywne (średni koszt poszczególnych operacji – liczba porównań i liczba modyfikowanych węzłów). Testy wykonaj na liście unikatowych ciągów posortowanych (np. `aspell_wordlist.txt`), ciągów losowych (wykonaj losowe permutacje słów w plikach), oraz takiej, gdzie możliwe są powtórzenia (np. `lotr.txt`, `KJB.txt`).

Zadanie 3 (1pkt) Uzupełnij sprawozdanie z zadania 2 o przykłady danych (odpowiednio dużych), które pozwalają każdej z tych struktur być najlepszą. Uzasadnij, dlaczego takie dane powodują ten efekt – scharakteryzuj słabości i zalety każdej z trzech struktur.