Algorytmy i struktury danych
Laboratorium 4

Termin oddania: 12 maja, 10:34

Zadanie 1 [80%]
Napisz program, który symuluje działanie wybranych struktur danych przechowujących ciągi znaków (przyjmujemy porządek leksykograficzny). Program powinien przyjmować jako parametr wejściowy typ struktury:

(15%) --type bst drzewo BST,

(30%) --type rbt drzewo czerwono–czarne,

(35%) --type hmap tablice hashujące z metodą łańcuchową dla przechowywanych w jednej komórce danych długości mniejszej niż n_t oraz z wykorzystaniem samoorganizujących się drzew binarnych (np. drzew czerwono–czarnych) dla przechowywanych w jednej komórce danych o długości większej niż n_t. Przeprowadź testy mające na celu oszacowanie n_t, dla którego zysk w czasie dostępu do elementu uzasadnia nadkład wykonywanych operacji balansujących. Dobierz liczbę komórek m odpowiednio do wybranej funkcji hashującej.

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)
- load f - dla każdego, oddzielonego białym znakiem, wyrazu z pliku f wykonaj operację insert, lub zwróć informację o nieistniejącym pliku
- delete s - jeśli struktura nie jest pusta i dana wartość s istnieje, to usuń element s
- find s - sprawdź czy w strukturze przechowywana jest wartość s (jeśli tak to wypisz 1, w p. p. wypisz 0)
- min - wypisz najmniejszy element znajdujący się w strukturze lub, dla struktur pustych oraz nie zachowujących porządku (np. hmap), pustą linię
- max - wypisz największy element znajdujący się w strukturze lub, dla struktur pustych oraz nie zachowujących porządku (np. hmap), pustą linię
- successor k - wypisz następnik elementu k lub, jeśli on nie istnieje (np. struktura nie zawiera k, k nie ma następników, struktura nie zachowuje porządku), pustą linię
- inorder - wypisz elementy drzewa w posortowanej kolejności (od elementu najmniejszego do największego) lub, dla struktur pustych oraz nie zachowujących porządku (np. hmap), 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. 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 $k \leq n$ linii, będących wynikami kolejnych operacji podanych na wejściu.

Przykład Przykładowe wywołanie

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

input	out.res
17	
max	a aaa ab b
insert aaa	ab
insert a	1
insert b	1
insert ab	1
inorder	0
delete a	aaa
delete b	
max	
load sample.txt	
find three	
delete three	
find three	
find Three	
delete Three	
find Three	
min	

Zadanie 2 [20%]
Wykonaj i zaprezentuj eksperymenty, które pozwolą postawić tezę na temat dolnego ograniczenia, średniej oraz górnego ograniczenia na liczbę porównań między elementami, wykonywaną przez procedurę find w każdej ze struktur. Testy wykonaj na liście unikatowych ciągów (np. słownik) oraz takiej, gdzie możliwe są powtórzenia (np. txt1, txt2).

Zadanie 3. [30% (dodatkowe)]
Uzupełnij **Zadanie 1** o filtr Blooma (**--type bloom**) – probabilistyczną strukturę danych, która pozwala na wykluczenie istnienia danego ciągu s w strukturze lub stwierdzenia, iż prawdopodobnie zawiera ona szukany ciąg s, w sposób minimalizujący nakład pamięciowy oryginalne sformułowanie. Zaimplementuj metody insert, load, find opisanie w zadaniu 1. Zmodyfikuj interpretację maksymalnego zapełnienia struktury, tak by podana była wartość bitów zajętych przez obiekty w strukturze. Wykonaj testy dla różnych długości filtrów m oraz liczby funkcji hashujących k. Wyciągnij wnioski na temat prawdopodobieństwa błędnej wartości zwracanej przez funkcję find oraz związku m z wartością n_t z zadania 1.