Laboratorium AiSD

Lista 7

Wyszukiwanie, kolejki priorytetowe, tablice mieszające

Proszę pamiętać, że **część rozwiązania** zadania stanowi również **zestaw testów** zaimplementowanych algorytmów i/lub struktur danych. Dodatkowo, proszę zwracać uwagę na **powtarzające się fragmenty** kodu i wydzielać je do osobnych funkcji/klas.

1. Wykorzystując dostarczoną paczkę kodu zaimplementuj i przetestuj następujące warianty **tablic mieszających**:
   a. z adresowaniem otwartym ***OpenAddressingHashTable<T>***:
      i. z **próbkowaniem przyrostowym**,
      ii. z **próbkowaniem kwadratowym**,
   b. z tablicą list dowiązaniowych ***SeparateChainingHashTable<T>***,

   dziedziczących po klasie ***HashTable<T>*** z dostarczonej paczki kodu i implementujących metody:

   - ***int capacity()*** – zwracającą rozmiar tablicy wartości lub tablicy list,
   - ***int size()*** – zwracającą liczbę aktualnie wstawionych elementów,
   - ***void insert(T object)*** – wstawiającą obiekt do tablicy,
   - ***boolean lookUp(T object)*** – sprawdzającą, czy obiekt występuje w tablicy,
   - ***int collisions()*** – zwracającą liczbę kolizji ***C*** przy wstawianiu wartości,
   - ***int insertComparisons()*** - zwracającą liczbę porównań wartości wykonaną podczas wstawiania elementów C_i,
   - ***int lookUpComparisons()*** – zwracającą liczbę porównań wartości wykonaną podczas wyszukiwania elementów C_l,
   - ***int hashFunctionEvaluations()*** – zwracającą liczbę wyliczeń wartości funkcji mieszającej ***E*** podczas wykonywania operacji.

   Dodatkowo zaimplementuj funkcję ***String toString()*** zwracającą **reprezentację wewnętrzną** tablicy:

   - [el1, null, el2, el3, null, null] dla **adresowania otwartego**,
   - [el1, el2 | | el3 | el4| | el5] dla **tablicy list dowiązaniowych**.

   Przyjmij, że wartość null jest niepoprawna!

Podczas implementacji tablic, jeśli przed wykonaniem operacji wstawiania **stopień wypełnienia *α*** (*ang. load factor*) przekroczy przyjęte maksimum ($\alpha \geq \alpha_{max}$), rozmiar tablicy należy **podwoić**, a elementy oryginalnie przechowywane – wstawić ponownie. Podczas tej operacji również należy **zliczać wszystkie** powyższe parametry!

Podczas zliczania **kolizji** dla implementacji z **tablicą list** przyjąć, że kolizja ma miejsce tylko wtedy, gdy wskazana lista **nie jest** pusta.

Funkcje mieszające przekazywać do konstruktora w postaci obiektów implementujących interfejs ***HashFunction<T>***. Ponadto, **funkcję przyrostową** dla adresowania otwartego przekazywać jako obiekty implementujące interfejs ***IncrementalFunction<T>***.

2. Porównać efektywność implementacji poprzez wstawianie losowych obiektów typu ***Integer*** (wykorzystać klasę ***java.util.Random***) i przygotowywując następujące wykresy:
   a. Stopnia wypełnienia ***α*** w zależności od liczby wstawianych wartości ***N***,
   b. Liczby kolizji ***C*** w zależności od liczby wstawianych wartości ***N***,
   c. Liczby wyliczeń wartości funkcji mieszającej ***E*** w zależności od liczby wstawianych wartości ***N***,
   d. Liczby porównań przy wstawianiu C_i w zależności od liczby wstawianych wartości ***N***,

   Sprawdzić dla **rozmiaru tablicy 10** oraz α_{max} **= 0,1; 0,2; 0,5; 0,9.**