Algorytmy i Struktury Danych

Krzysztof Wyszyński

Laboratorium 4

Tablice mieszające – adresowanie otwarte

Decyzje implementacyjne:

1. W celu umożliwienia usuwania elementów tablicy stworzyłem klasę DeletedElem, implementującą interfejs Comparable. Dzięki temu podczas usuwania elementu umieszczam pod danym adresem hashId nową instancję klasy DeletedElem. Podczas wyszukiwania elementu algorytm sprawdza, czy nie jest on instancją klasy DeletedElem – jeśli jest, przechodzi dalej. Podobnie w przypadku umieszczania nowego elementu w tablicy, jeśli algorytm napotka instancję DeletedElem, umieszcza pod danym adresem hashId nowy element.

   Początkowo zaimplementowałem usuwanie posługując się listą przechowującą usunięte indeksy. Jest to jednak nie tylko dodatkowe obciążenie pamięci, lecz również potencjalne spowolnienie działania algorytmu w przypadku, gdy wiele elementów zostanie usuniętych – podczas każdego umieszczania oraz pobierania elementu z tablicy, lista byłaby przeszukiwana w celu sprawdzenia, czy dany indeks nie został uznany za usunięty.

2. W przypadku umieszczania nowego elementu do tablicy, może zaistnieć sytuacja, gdy jego klucz wygenerowany za pomocą metody *hashCode()* będzie ujemny. Nie do końca jest to pożądany efekt – w przypadku adresowania liniowego skutkuje to przemieszczaniem się algorytmu „do tyłu" tablicy, odbiciem od indeksu 0 oraz ponownym przemierzeniem drogi po tych samych indeksach, aż do punktu wyjścia. W algorytmie mieszania podwójnego efektem ubocznym może być bardzo prosty przykład – próba umieszczenia elementu o kluczu -1 do tablicy 10-elementowej, gdy znajduje się tam już inny element. Pomijając fakt, że w metodzie tej wiele zależy od doboru prawidłowych funkcji mieszających, w przykładzie realizowanym w ramach laboratorium wynikiem obliczeń zawsze będzie wartość 1, niezależnie od iteratora – wystąpi nieskończona pętla. Aby rozwiązać powyższe problemy postanowiłem wprowadzać do funkcji mieszających wartości bezwzględne kluczy.

3. Dobór funkcji w algorytmie mieszania podwójnego skutkuje ryzykiem nieskończonego zapętlenia – podczas kolizji algorytm przemieszcza się każdorazowo o tę samą liczbę indeksów, w poszukiwaniu wolnego miejsca w tablicy. Aby zapobiec temu zjawisku, można w przypadku detekcji zapętlenia zwiększyć rozmiar tablicy mieszającej.

Otrzymane wyniki:

1. Algorytm wykorzystujący adresowanie podwójne wykazał się najkrótszym średnim czasem umieszczania elementów w tablicy. Wynika to z faktu, że rozwiązanie to zapobiega grupowaniu w przypadku kolizji. Przy początkowym rozmiarze tablicy <= 16384 algorytm wpadł w nieskończoną pętlę – koniecznym okazało się zwiększenie jej rozmiarów. Wynika to z faktu, że dla pewnej wartości klucza wszystkie dostępne indeksy były już zajęte.
2. Algorytm wykorzystujący adresowanie kwadratowe uzyskał porównywalne czasy dla każdego wariantu a i b. W tym przypadku umieszczanie nowych elementów do tablicy było znacznie

mniej czasochłonne niż w adresowaniu liniowym. W związku z tym można stwierdzić, że nie doszło w tym przypadku do grupowania wtórnego – być może wynikło to ze szczęśliwego doboru parametrów a i b.

3. Algorytm wykorzystujący adresowanie liniowe wykazał się najdłuższym średnim czasem umieszczania elementów w tablicy. Widać wyraźną różnicę między rozmiarem 262144, a 131072 – dominującym efektem w tym przypadku niewątpliwie jest grupowanie elementów. Współczynnik wypełnienia tablicy o rozmiarze 262144 w momencie, gdy umieszczone zostanie w niej 100 000 słów wynosi ok. 38% - zapełniona jest mniej niż połowa dostępnych miejsc. W tym przypadku problem grupowania nie będzie miał dużego wpływu na wydajność algorytmu. Jednak jeśli zmniejszymy rozmiar tablicy o połowę, współczynnik wypełnienia w pewnym momencie osiągnie wartość graniczną (75%) i konieczne będzie powiększenie tablicy.
4. Proces zwiększania rozmiaru tablicy w celu zredukowania współczynnika wypełnienia nie jest znacząco czasochłonny – ponowne umieszczenie elementów w tablicy o niewielkim współczynniku wypełnienia wykonuje się szybko. Z tego powodu różnice wydajnościowe zacierają się wraz ze zmniejszającym się początkowym rozmiarem tablicy.
5. Wyniki uzyskane w metodzie adresowania podwójnego dowodzą, że nie występuje w tym przypadku grupowanie – czas wstawiania elementów do tablicy o rozmiarze 131072 jest ok. dwa razy większy niż w przypadku tablicy o rozmiarze 262144, co wynika z konieczności ponownego wstawienia elementów do nowej tablicy.