ćwiczenia (licencjat dzienny): lista zadań nr 6 *20 listopada 2006 r.*

ALGORYTMY I STRUKTURY DANYCH

TABLICE Z HASZOWANIEM

Instytut Informatyki Uniwersytetu Wrocławskiego *Paweł Rzechonek*

1. [∗∗∗] Chcemy zaimplementować słownik za pomocą *adresowania bezpośredniego* w bardzo dużej tablicy (tablica ze wskaźkikami). Początkowo w tablicy tej mogą się znajdować przypadkowe wartości, ale jej zainicjalizowanie nie wchodzi w rachubę ze względu na jej olbrzymi rozmiar. Zaprojektuj metodę reprezentowania dużego słownika w tablicy z adresowaniem bezpośrednim bez konieczności inicjalizowania wszystkich slotów. Każdy przechowywany element powinien zajmować $O(1)$ komórek pamięci. Operacje słownikowe *insert*, *delete* i *search* powinny działać w stałym czasie $O(1)$. Również przygotowanie struktury danych do użytkowania powinno być wykonane w stałym czasie $O(1)$.

   Wskazówka! Użyj dodatkowego stosu o rozmiarze równym liczbie elementów w tablicy, za pomocą którego można będzie określić, czy dana pozycja w tablicy zawiera sensowną wartość.

2. [∗∗] Wykaż, że jeśli $|U| \geq n\,m$, to istnieje podzbiór uniwersum U o rozmiarze n, składający się z kluczy które są odwzorowywane za pomocą funkcji haszującej na tę samą pozycję w m-elementowej tablicy. Jaki jest w tym przypadku pesymistyczny czas wyszukiwania elementu w tablicy z haszowaniem z *łańcuchową metodą rozwiązywania konfliktów*?

3. [∗] Jak długo może trwać wstawienie n kluczy do początkowo pustej tablicy z haszowaniem z *łańcuchową metodą rozwiązywania konfliktów*: *(i)* z nieuporządkowanymi listami, *(ii)* z uporządkowanymi listami?

4. [∗∗] Załóżmy, że klucze są t-bitowymi liczbami całkowitymi. Udowodnij, że dla modularnej funkcji haszującej $h(k) = k \bmod m$, gdzie $m > 2$ jest liczbą pierwszą, prawdziwa jest następująca własność: każde dwa klucze różniące się tylko na jednym bicie mają różne wartości funkcji haszującej.

5. [∗∗] Rozważmy następującą ideę haszowania modularnego dla kluczy całkowitych: $h(k) = (p\,k) \bmod m$, gdzie $p > m$ jest arbitralnie ustaloną liczbą pierwszą. Czy dzięki takiej modyfikacji można używać dowolnego m nie będącego liczbą pierwszą?

6. [∗] Zaimplementuj w pseudokodzie procedurę wstawiającą *insert* i usuwającą *delete* zadaną wartość do/z tablicy z haszowaniem z *adresowaniem otwartym*. Opisz także procedurę szukającą zadanej wartości w tablicy.

7. [$*$] Do 7-elementowej początkowo pustej tablicy z haszowaniem $T[0 \ldots 6]$ wstawiamy kolejno elementy 11, 19, 53, 49, 41, 23 i 61. Powstające konflikty rozwiązujemy metodą *adresowania otwartego liniowego* z funkcją haszującą $h(k, i) = (k + i) \bmod 7$. Ile konfliktów wystąpi przy wstawianiu do niej podanych wartości? Następnie z tablicy usuwamy elementy o wartościach 19, 53 i 41. Zilustruj przeprowadzane na tej tablicy operacje i przedstaw ostateczną zawartość tablicy T.

8. [$***$] Niech dana będzie tablica z haszowaniem o rozmiarze 2^r dla $r \in \mathbf{N}$, w której konflikty są usuwane za pomocą *adresowania otwartego kwadratowego*. Wykaż, że w takim przypadku funkcja $h(k, i) = (k + 2i^2 + i) \bmod 2^r$ trafia we wszystkie pozycje w tablicy dla $i = 0, \ldots, 2^r - 1$. Czy własnośc ta będzie nadal zachowana, gdy współczynnik przy i zmienimy z 1 na dowolną inną liczbę nieparzystą?

9. [$**$] Załóżmy, że zaimplementowałeś tablicę z haszowaniem, w której konflikty są usuwane za pomocą *adresowania otwartego z podwójnym haszowaniem*. Programując funkcje haszujące popełniłeś taki błąd, że albo jedna albo obie funkcje zawsze zwracają tę samą wartość $\neq 0$. Opisz co się zdarzy w sytuacji gdy: *(i)* pierwsza funkcja jest błędna, *(ii)* druga funkcja jest błędna, *(iii)* obie funkcje są błędne.