Temat: Operacje słownikowe realizowane w tablicy z haszowaniem

1. Tablice z haszowaniem

Idea tablic z haszowaniem polega na ustaleniu tzw. funkcji haszującej

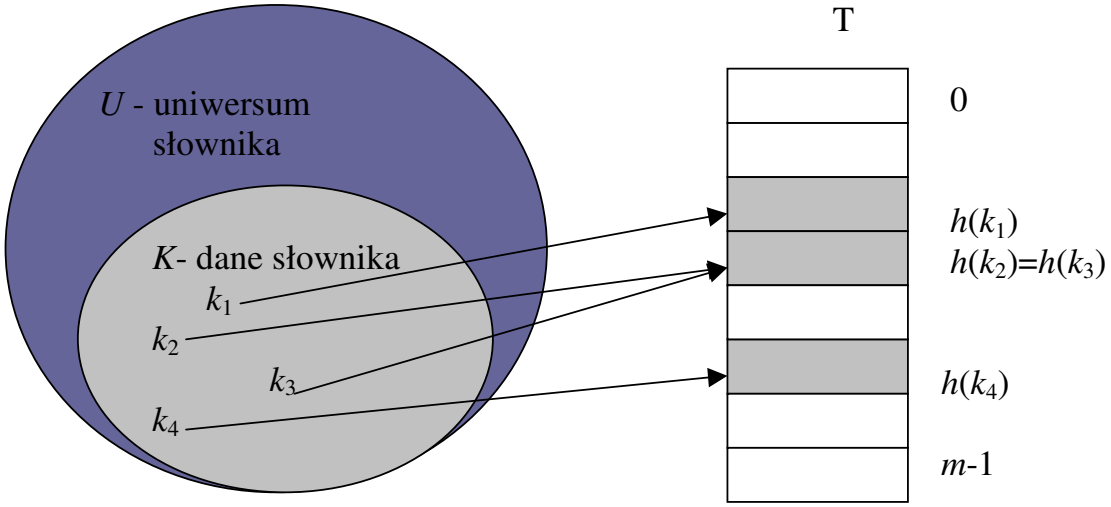
$$h : U \rightarrow \{0, 1, 2, ..., m-1\}$$

gdzie U jest zbiorem możliwych danych słownika (uniwersum słownika), a m ustaloną liczbą naturalną, określającą rozmiar tablicy haszującej.

Zbiór $\{0, 1, 2, ..., m-1\}$ jest zbiorem indeksów tablicy haszującej, którą oznaczymy przez T .

Wartość $h(e)$ jednoznacznie określa miejsce elementu e w słowniku. Realizacja operacji *insert*(e, T) w tablicy haszującej polega na:

- obliczeniu wartości $h(e)$
- wstawieniu elementu e w indeks $h(e)$



Kluczowym problemem haszowania jest rozwiązanie problemu kolizji. Kolizją jest sytuacja, w której dwa różne elementy uniwersum mają tę samą wartość funkcji haszującej.

Sposoby rozwiązywania problemu kolizji są podstawą podziału metod haszowania na dwa zasadnicze rodzaje:

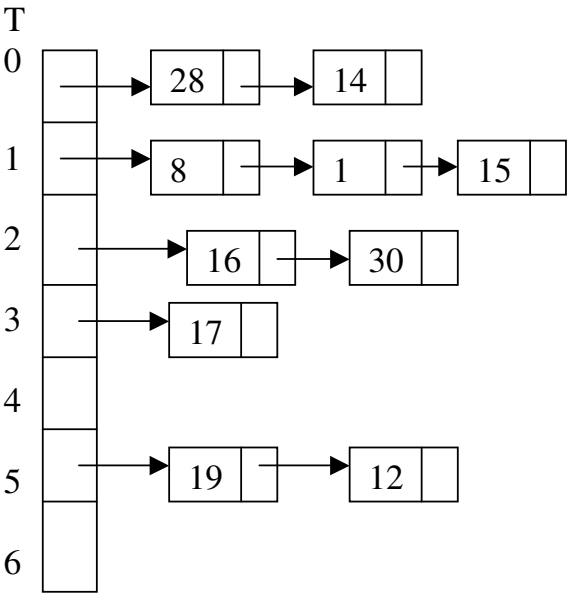
- metoda łańcuchowa (inaczej adresowanie zamknięte),
- adresowanie otwarte

2. Rozwiązywanie kolizji metodą łańcuchową

Wszystkie elementy, którym odpowiada ta sama wartość w tablicy, zostają umieszczone na jednej liście. Tablica haszująca jest w tym przypadku tablicą wskaźników na listy tych elementów, które mają taką samą wartość funkcji haszującej.

Przykład 1

Załóżmy, że dane w słowniku to liczby typu Word, a funkcja haszująca h jest następująca: $h(e) = e \ \% \ 7$ ($m = 7$). Do początkowo pustego słownika zostały wstawione elementy: 30, 14, 15, 12, 17, 1, 8, 28, 16, 19



Realizacja operacji słownikowych

search(*e*, T): Wyszukaj element *e* na liście T[*h*(*e*)].

insert(*e*, T): Jeżeli *search*(*e*, T) = =1, to wstaw element *e* na listę T[*h*(*e*)].

delete(*e*, T): Jeżeli *search*(*e*, T) = =1, to usuń element *e* z listy T[*h*(*e*)].

Analiza kosztów haszowania metodą łańcuchową

Niech α oznacza **współczynnik zapełnienia tablicy haszującej:** $\alpha = \frac{n}{m}$**,** gdzie n to aktualna liczba danych w słowniku, a m to rozmiar tablicy haszującej. Wartość współczynnika α to średnia długość listy (łańcucha).

W przypadku pesymistycznym wszystkie operacje słownikowe mają złożoność $\Theta(n)$, zakładając, że koszt obliczania wartości funkcji haszującej dla danego elementu jest $\Theta(1)$.

Przy określaniu złożoności średniej przyjmuje się, że losowo wybrany element z jednakowym prawdopodobieństwem trafia na każdą z m pozycji, niezależnie od tego gdzie trafiają inne elementy. Jeżeli spełnione jest to założenie, to mówimy o prostym równomiernym haszowaniu.

Można pokazać, że:

Twierdzenie
W tablicy z haszowaniem wykorzystującym łańcuchową metodę rozwiązywania problemu kolizji, przy założeniu, o prostym równomiernym haszowaniu, średni czas działania operacji *search* wynosi $\Theta(1+\alpha)$.

Wniosek:
Średni czas realizacji operacji *insert* i *delete* wynosi $\Theta(1+\alpha)$.

3. Funkcje haszujące

Dobra funkcja haszująca powinna spełniać (w przybliżeniu) założenie prostego równomiernego haszowania.

Załóżmy, że znamy rozkład prawdopodobieństwa $P(e)$ określający prawdopodobieństwo wyboru losowego elementu e z uniwersum U. Warunek prostego równomiernego haszowania oznacza wtedy, że:

$$\sum_{e:h(e)=j} P(e) = \frac{1}{m} \quad \text{dla} \quad j = 0, 1, ..., m-1$$

Funkcja haszująca jest najczęściej postaci:

$$h(e) = e\%m$$

Dobrymi wartościami m są liczby pierwsze niezbyt bliskie potęgom dwójki.

Przykład 2

Załóżmy, że $n = 2000$. Podejmujemy przy tym decyzję, że dopuszczamy przeglądanie list o średniej długości 3 ($\alpha = 3$). Liczba $m = 701$ jest w tym przypadku dobra, ponieważ jest liczbą pierwszą leżącą blisko $2000/\alpha = 2000/3$, a zarazem nie leży blisko żadnej potęgi 2. Zatem: $h(e) = e\%701$.

4. Adresowanie otwarte

Wszystkie elementy słownika są przechowywane wprost w tablicy. Wyszukiwanie elementu polega na systematycznym sprawdzaniu pozycji w tablicy, aż zostanie znaleziony szukany element albo wiadomo już, że na pewno nie ma go w tablicy. Nie ma żadnych dodatkowych list. Przy stosowaniu haszowania otwartego współczynnik zapełnienia α nie może większy od jedynki, aby nie nastąpiło przepełnienie tablicy.

Funkcja haszująca jest tym razem dwuargumentowa:

$$h : U \times \{0, 1, ..., m-1\} \to \{0, 1, ..., m-1\}$$

o tej własności, że dla każdego elementu *e* ciąg pozycji (**ciąg kontrolny elementu** *e*)

$$\langle h(e,0), h(e,1), ..., h(e, m-1) \rangle$$

jest pewną permutacją ciągu $\langle 0, 1, ..., m-1 \rangle$.

Jeżeli tablica będzie prawie zupełnie wypełniona, to przy próbie wstawienia nowego elementu zostaną sprawdzone wszystkie bez wyjątku pozycje w tablicy.

Realizacja operacji słownikowych

insert(*e*, T)

```
i=0; jest_miejsce=0;
  do
   {
     j=h(e, i);
     if  (T[j]=="wolne")
          {
            T[j]=e; jest_miejsce=1;break;
          }
     else i++;
    }while (i != m);
if  (!jest_miejsce) „przepełnienie tablicy”;
```

search(*e*, T)

```
 i=0;
  do
     {
        j=h(e, i);
        if (T[j] ==e) return 1;
        else i++;
      } while (i != m && T[j]!= "wolne");
   return 0;
```

Operacja *delete* realizowana metodą haszowania otwartego jest dość skomplikowana i kosztowna czasowo (koszt nie zależy od współczynnika zapełnienia tablicy).

a) Adresowanie liniowe

Stosujemy funkcję postaci: $h(e,i)=(h'(e)+i)\ \%\ m$,
gdzie $h': U \to \{0,1,...,m-1\}$ jest zwykłą jednoargumentową pomocniczą funkcją haszującą.

Dla danej *e* jej ciąg kontrolny zaczyna się od pozycji $T[h'(e)]$. Następną pozycją w tym ciągu jest $T[h'(e)+1]$ i tak dalej, aż do pozycji $T[m-1]$. Dalej występują pozycje $T[0], T[1], ..., T[h'(e)-1]$.

Przykład 3

$h(e,i)=(h'(e)+i)\ \%\ 13$ gdzie $h'(e)=e\ \%\ 13$
Do początkowo pustego słownika wstawiamy liczby: 4, 17, 26, 39, 52

$h(4,0)=4\ \%\ 13=4$

$h(17,0)=4\ \%\ 13=4,\ h(17,1)=5\ \%\ 13=5$

$h(26,0)=0\ \%\ 13=0$

$h(39,0)=0\ \%\ 13=0,\ h(39,1)=1\ \%\ 13=1$

$h(52,0)=0\ \%\ 13=0,\ h(52,1)=1\ \%\ 13=1,\ h(52,2)=2\ \%\ 13=2$

T:

26	39	52		4	17							
0	1	2	3	4	5	6	7	8	9	10	11	12

Poważną wadą adresowania liniowego jest tendencja do grupowania się zajętych pozycji. Długie spójnie wypełnione ciągi zajętych pozycji szybko się powiększają, co znacznie spowalnia operację wyszukiwania.

b) Adresowanie kwadratowe

Stosujemy tu funkcję postaci: $h(e,i)=\left(h'(e)+c_1 i+c_2 i^2\right)\%\, m$
gdzie $h': U \to \{0,1,...,m-1\}$ jest zwykłą jednoargumentową funkcją haszującą, c_1 i $c_2 \neq 0$ są stałymi, a $i=0,1,...,m-1$.
Dla danej *e* jej ciąg kontrolny zaczyna się od pozycji $T[h'(e)]$. Następna pozycja w tym ciągu jest oddalona od pierwszej o wielkość zależną od kwadratu numeru pozycji *i* w ciągu kontrolnym. Dobór stałych c_1 i c_2 w decydujący sposób może wpłynąć na efektywność operacji *insert* i *search* realizowanych metodą adresowania kwadratowego.

Zauważmy, że jeżeli dwa elementy mają w takie same początkowe pozycje, to i całe ich ciągi kontrolne są równe, ponieważ z $h(e_1,0)=h(e_2,0)$ wynika, że $h(e_1,i)=h(e_2,i)$. Prowadzi to do tzw. grupowania wtórnego. Początkowa pozycja określa jednoznacznie, podobnie jak w adresowaniu liniowym, cały ciąg kontrolny.

Przykład 4

$h(e,i)=\left(h'(e)+i^2\right)\%\, 13$ gdzie $h'(e)=e\ \%\ 13$

Do początkowo pustego słownika wstawiamy liczby: 4, 17, 26, 39, 52, 65

$h(4,0)=4\ \%\ 13=4$

$h(17,0)=4\ \%\ 13=4,\ h(17,1)=5\ \%\ 13=5$

$h(26,0)=0\ \%\ 13=0$

$h(39,0)=0\ \%13=0,\ h(39,1)=1\ \%\ 13=1$

$h(52,0)=0\ \%\ 13=0,\ h(52,1)=1\ \%\ 13=1,\ h(52,2)=4\ \%\ 13=4$

$h(52,3)=9\ \%\ 13=9$

$h(65,0)=0\ \%\ 13=0,\ h(65,1)=1\ \%\ 13=1,\ h(65,2)=4\ \%\ 13=13$

$h(65,3)=9\ \%\ 13=9,\ h(65,4)=16\ \%\ 13=3$

T:

26	39		65	4	17				52			
0	1	2	3	4	5	6	7	8	9	10	11	12

c) Haszowanie dwukrotne

Stosujemy tu funkcję postaci: $h(e,i)=(h_1(e)+ih_2(e))\ \%\ m$
gdzie $h_1, h_2 : U \to \{0,1,...,m-1\}$ są pomocniczymi jednoargumentowymi funkcjami haszującymi. Pierwszą pozycją w ciągu kontrolnym dla danej *e* jest $T[h_1(e)]$. Kolejne pozycje są oddalone od początkowej o $h_2(e)\ \%\ m$. Pozycja początkowa nie określa jednoznacznie ciągu kontrolnego.

Przykład 5

Funkcja haszująca:
$h(e,i)=(h_1(e)+ih_2(e))\ \%\ 13$, gdzie $h_1(e)=e\ \%\ 13$, $h_2(e)=1+(e\ \%\ 11)$.
Do słownika zawierającego już liczby: 79, 69, 98, 72, 50 wstawiamy liczbę 14.

$h_1(14)=14\ \%\ 13=1,$

$h(14,0)=1\ \%\ 13=1,$

$h_2(14)=1+(14\ \%\ 11)=4,$

$h(14,1)=(1+1\cdot 4)\ \%\ 13=5,$

$h(14,2)=(1+2\cdot 4)\ \%\ 13=9$

T:

	79			69	98		72		**14**		50	
0	1	2	3	4	5	6	7	8	9	10	11	12

Haszowanie dwukrotne dopuszcza użycie $\Theta(m^2)$ różnych ciągów kontrolnych, a nie tylko $\Theta(m)$ ciągów jak w haszowaniu liniowym i kwadratowym. Każda para $(h_1(e), h_2(e))$ wyznacza inny ciąg kontrolny, a dla różnych elementów wartości $h_1(e)$ i $h_2(e)$ zmieniają się niezależnie.

Dzięki temu haszowanie dwukrotne bardzo dobrze przybliża "idealne" warunki haszowania równomiernego.

Koszt operacji *insert i search* w adresowaniu otwartym

Twierdzenie

Jeżeli współczynnik zapełnienia tablicy z haszowaniem $\alpha = \frac{n}{m} < 1$, to średnia liczba porównań wykonanych w czasie realizacji operacji *search* i *insert* metodą adresowania otwartego jest nie większa niż $\frac{1}{1-\alpha}$, o ile jest spełnione założenie o równomiernym haszowaniu.