Protokół: Znajdowanie najmniejszego i najwiekszego elementu w tablicy haszujacej

1 Wstep

Celem projektu było zbadanie efektywności algorytmu znajdowania najmniejszego i najwiekszego elementu w tablicy haszujacej. Przeprowadzono pomiary czasu wykonywania operacji dla różnych liczności danych oraz różnych rozmiarów tablicy haszujacej. Algorytm wykorzystuje iteracje po łańcuchach kolizji, co pozwala na dokładne określenie minimalnej i maksymalnej wartości.

2 Opis wykorzystanych algorytmów i struktur danych

W projekcie wykorzystano tablice haszujaca z łańcuchami kolizji. Każda komórka tablicy zawiera liste (łańcuch), do której trafiaja elementy o tym samym indeksie wynikajacym z funkcji haszujacej.

2.1 Funkcja haszujaca oparta na metodzie mnożenia

W celu zminimalizowania liczby kolizji w tablicy haszujacej zastosowano **funkcje haszujaca oparta na metodzie mnożenia**. W tej metodzie klucz jest przekształcany do indeksu za pomoca operacji matematycznych, co pozwala na bardziej równomierne rozłożenie elementów w tablicy.

Funkcja wykorzystuje stała $A = 0.6180339887$, która jest bliska odwrotności złotego podziału ϕ. Wybór tej wartości pozwala osiagnać korzystne właściwości rozkładu indeksów, dzieki czemu zmniejsza sie prawdopodobieństwo kolizji.

2.1.1 Algorytm działania

Działanie funkcji można opisać w nastepujacych krokach:

1. Mnożymy klucz k przez stała A.
2. Pobieramy cześć ułamkowa wyniku, co daje wartość z zakresu $[0, 1)$.
3. Mnożymy cześć ułamkowa przez rozmiar tablicy m, a nastepnie bierzemy cześć całkowita, aby otrzymać indeks w tablicy.

2.1.2 Wzór matematyczny

Funkcja haszujaca definiowana jest jako:

$$\text{hash}(k) = \lfloor m \cdot ((k \cdot A) \bmod 1) \rfloor$$

gdzie:

- k — klucz,
- A — stała złotego podziału ($A \approx 0.618$),
- m — rozmiar tablicy haszujacej.

2.1.3 Właściwości funkcji

- **Równomierność**: Wykorzystanie stałej A zmniejsza ryzyko zgrupowania kluczy w jednej cześci tablicy (przy założeniu, że klucze sa równomiernie rozłożone).
- **Unikalność**: Funkcja daży do równomiernego rozkładu indeksów dla różnych kluczy, co minimalizuje kolizje.
- **Efektywność**: Obliczenia sa szybkie, ponieważ wykorzystuja jedynie podstawowe operacje arytmetyczne.

2.2 Algorytm wstawiania

1. Oblicz indeks za pomoca funkcji haszujacej
2. Dodaj element (klucz, wartość) do listy w komórce o indeksie index.

2.3 Algorytm znajdowania mininimalnej i maksymalnej wartości

1. Ustaw min_value = $+\infty$, max_value = $-\infty$, min_key = None, max_key = None.
2. Dla każdej komórki w tablicy:
   (a) Dla każdego elementu (klucz, wartość) w łańcuchu:
   - Jeśli wartość < min_value: zaktualizuj min_value i min_key.
   - Jeśli wartość > max_value: zaktualizuj max_value i max_key.
3. Zwróć (min_key, min_value) oraz (max_key, max_value).

2.4 Złożoność

- **Wstawianie**: $O(1)$ w optymistycznym przypadku (brak kolizji), $O(n)$ w pesymistycznym (wszystkie klucze w jednym łańcuchu).
- **Znajdowanie min/max**: $O(n)$, gdzie n to liczba elementów w tablicy (algorytm musi przejrzeć wszystkie elementy).

3 Opis działania programu

Program generuje losowe dane w postaci par (klucz, wartość), gdzie klucz reprezentuje identyfikator firmy, a wartość to wynik finansowy. Nastepnie dane te sa wstawiane do tablicy haszujacej, a algorytm wyszukuje elementy o najmniejszym i najwiekszym wyniku finansowym. Działanie programu obejmuje:

- Wstawianie elementów do tablicy haszujacej.
- Znajdowanie elementów minimalnych i maksymalnych.
- Pomiar czasu wykonywania powyższych operacji.

Testy przeprowadzono dla różnych liczności danych (num_companies) i różnych rozmiarów tablicy (table_size).

4 Wyniki testów

Testy przeprowadzono na sprzecie z procesorem Ryzen 5 3550H i obejmowały zbiory danych o liczności od 100 do 100000 elementów oraz tablice haszujace o rozmiarach od 10 do 10007. Wyniki przedstawiono w tabeli:

Liczba firm	Rozmiar	Czas wstawiania	Czas min/max	Min	Max
100	10	0.000000s	0.000000s	(9105, 1133251)	(5139, 97178313)
100	101	0.000000s	0.000000s	(9948, 516538)	(4503, 99549854)

Liczba firm	Rozmiar	Czas wstawiania	Czas min/max	Min	Max
100	1009	0.000000s	0.000000s	(7849, 919403)	(3409, 98108041)
100	10007	0.000000s	0.000999s	(8537, 3279544)	(4702, 99236037)
1000	10	0.000000s	0.001000s	(8752, 107388)	(3197, 99868107)
1000	101	0.000999s	0.000000s	(3343, 33068)	(8148, 99993767)
1000	1009	0.001001s	0.000000s	(9211, 11772)	(8733, 99901372)
1000	10007	0.000999s	0.000000s	(5323, 348588)	(8186, 99651216)
10000	10	0.010001s	0.000999s	(2823, 9469)	(1184, 99998002)
10000	101	0.009999s	0.002000s	(6168, 8697)	(5513, 99999740)
10000	1009	0.007999s	0.002002s	(1219, 8094)	(3442, 99994357)
10000	10007	0.009998s	0.004003s	(7910, 8043)	(5347, 99998957)
100000	10	0.062003s	0.025002s	(2197, 491)	(4810, 99998021)
100000	101	0.092006s	0.027005s	(2739, 2189)	(8009, 99999670)
100000	1009	0.072006s	0.025002s	(5038, 228)	(6148, 99998849)
100000	10007	0.077007s	0.033001s	(8135, 580)	(4119, 99998677)

Aby obliczyć odchylenie standardowe, używamy wzoru:

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2}$$

gdzie:
- x_i to czas wstawiania lub czas znajdowania min/max dla i-tego pomiaru,
- μ to średnia czasów wstawiania lub czasów znajdowania min/max,
- n to liczba pomiarów.

Obliczenia zostały przeprowadzone dla czasów wstawiania oraz czasów znajdowania najmniejszego i najwiekszego elementu.

1. **Średnia dla czasów wstawiania:**

$$\mu_{\text{wstawianie}} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

2. **Odchylenie standardowe dla czasów wstawiania:**

$$\sigma_{\text{wstawianie}} = 0.0320$$

3. **Odchylenie standardowe dla czasów znajdowania min/max:**

$$\sigma_{\text{find}} = 0.0117$$

5 Wnioski

Algorytm znajdowania najmniejszego i najwiekszego elementu w tablicy haszujacej działa wydajnie, a jego złożoność czasowa w praktyce jest liniowa wzgledem liczby elementów, co odpowiada teorii dla iteracji po wszystkich danych. Rozmiar tablicy haszujacej ma wpływ na równomierność rozkładu danych, co z kolei minimalizuje długość łańcuchów kolizji i poprawia wydajność.

Wyniki pokazuja, że:

- Operacje wstawiania sa szybkie, nawet dla dużych zbiorów danych.
- Operacje wyszukiwania minimalnych i maksymalnych elementów maja stała zależność od liczby elementów w tablicy, ale sa wolniejsze przy dużych zbiorach danych i małych tablicach.
- Wprowadzenie odpowiedniego doboru rozmiaru tablicy oraz poprawna funkcja haszujaca sa fundamentami osiagania wysokiej wydajności w przypadku operacji na dużych zbiorach danych.

Dalsze usprawnienia mogłyby obejmować optymalizacje funkcji haszujacej lub zastosowanie innych struktur danych w celu ograniczenia kolizji.

6 Źródła

- https://slideplayer.pl/slide/434870/
- https://pl.wikipedia.org/wiki/Tablica_mieszaj%C4%85ca