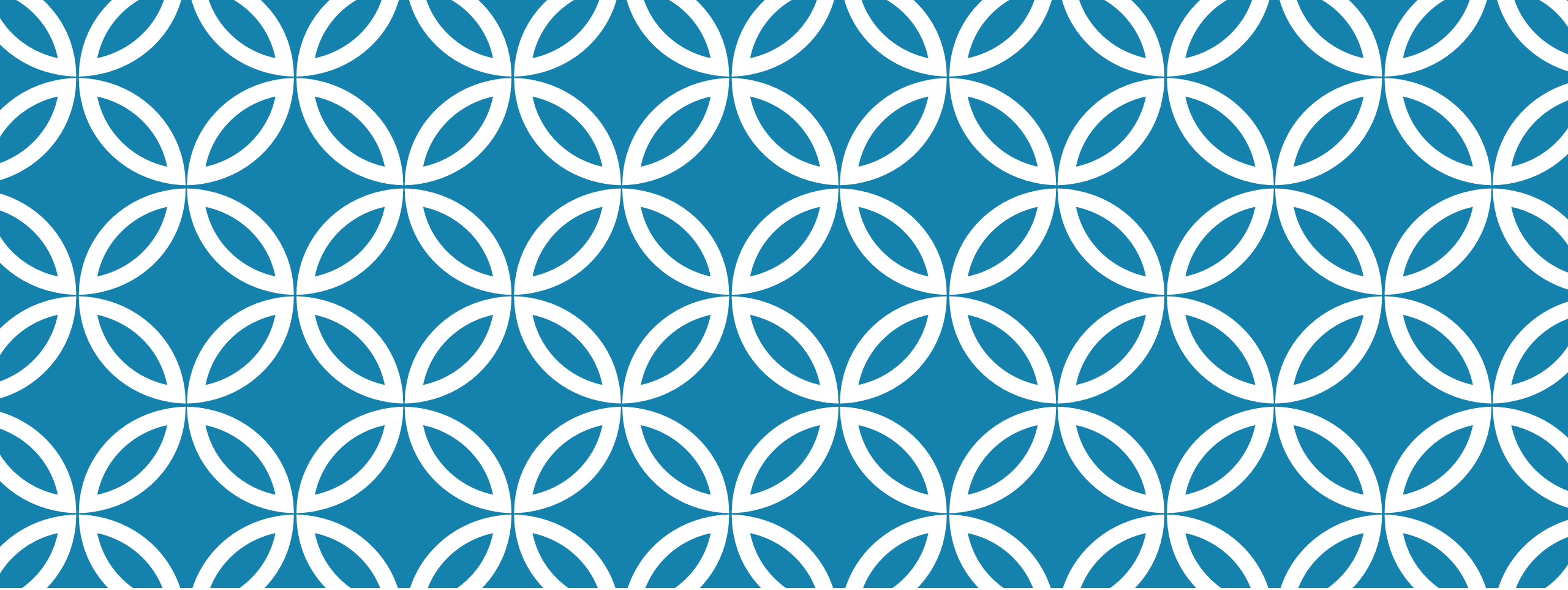


ALGORYTMY I STRUKTURY DANYCH II
CZ 1. HASHTABLE

Dr Paweł Dąbrowski
pawel.dabrowski@uni.lodz.pl
Pokój 530 B (126 B)

CZYM BĘDZIEMY SIĘ ZAJMOWAĆ?

Rozwiążemy pięć problemów:

- **Hashtable – tablica z haszowaniem**
- Heapsort – sortowanie stogowe
- MST – minimalne drzewo rozpinające
- KR – algorytm Karpa Rabina
- SJF

Zaliczenie zadania polega na przesłaniu gotowego zadania w terminie do dwóch tygodni od zakończenia ćwiczeń z danego problemu!

ZASADY ZALICZENIA

Punktacja:

- 10 punktów za poprawnie rozwiązane zadanie – w sumie 50 punktów
- 2 punktów za każde zajęcia – w sumie 14 punktów

OCENY:

60% - 80% **3**
80% - 90% **3.5**
90% - 100% **4**

Na **5** należy dodatkowo rozwiązać zadania konkursowe podawane na wykładzie.

W przypadku braku liczby punktów do zaliczenia jest możliwe uzyskanie dodatkowych 15 punktów rozwiązując dodatkowe zadanie ale konieczny będzie opis działania algorytmu.
Zadanie zostanie podane na zakończenie ćwiczeń – zaliczenie w sesji poprawkowej.

FORMAT PRZESŁANEGO ZADANIA

1. Program stanowiący rozwiązanie zadania musi zawierać się w jednym pliku o nazwie zgodnej ze standardem:

ASD2_[Nazwisko][Inicjał imienia]_[tytuł zadania]_[data w formacie rrmmdd].cpp

Przykładowo:

ASD2_NowakJ_Hashtable_160420.cpp

2. Program musi wczytywać z wejścia standardowego i zapisywać na wyjście standardowe.

- Wiele zadań wymaga przeprowadzenia operacji na tablicach:

obsługi tablicy symboli (słownika) z jak najszybszym czasem dostępu

- Słowa kluczowe dla kompilatora
- Numery identyfikacyjne klientów w bazie zamówień

- Wymagamy by struktura danych obsługiwała opcje Search, Insert, Delete w

Średnim czasie O(1) bez zakładania czegokolwiek o elementach

Spotykamy się z tym problemem dla wszystkich elementów (słowników) w postaci [klucz, wartość]

TABLICE Z ADRESOWANIEM BEZPOŚREDNIM

- Tablica z haszowaniem ma rozmiar m, T[0, m-1]
- Liczba kluczy n jest bliska m lub m jest małe w porównaniu z dostępnością pamięci
- Klucz k staje się indeksem T, czyli funkcja haszująca ma postać h(k)=k
- Nie ma kolizji , czas dostępu wynosi O(1)
- Nie ma potrzeby przechowywania kluczy
- Niestety metoda wymaga dużych zasobów pamięci i jest niemożliwa do zrealizowania dla dużych m

TABLICE Z HASZOWANIEM

- Uogólnienie tablic A[0 .. n-1).
- Nie używamy klucza jako indeksu w tablicy A, zamiast tego obliczamy indeks jako wartość pewnej funkcji haszowania h(k): A[k] – A[h(k)].
- Rozmiar tablicy z haszowaniem jest proporcjonalny do ilości elementów nie do ich zakresu.
- Funkcja h(k) nie musi odpowiadać jednej wartości.
- Jeśli h(k1)=h(k2) to dwa klucze są w kolizji. Potrzebny jest mechanizm do rozwiązywania kolizji.

I adresowanie otwarte

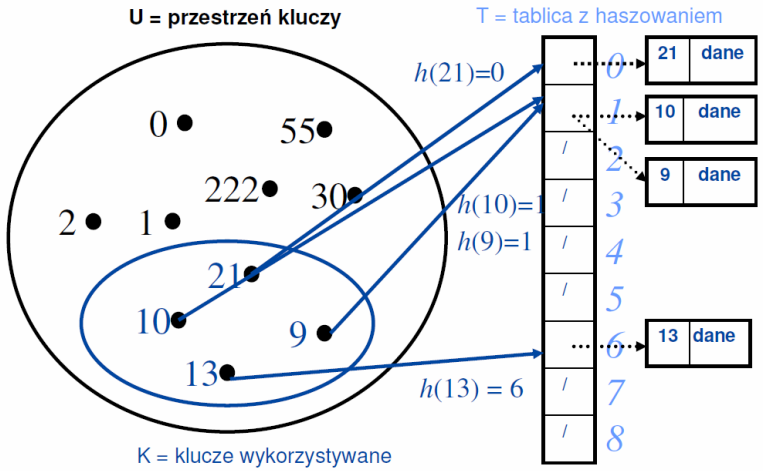
II Łańcuchy

PRZYKŁADY

- Tablice rejestracyjne
- numery są ustalone (nie wszystkie muszą być wykorzystane)
- Można poszukiwać przy wykorzystaniu tablic z haszowaniem o ustalonym rozmiarze
- Funkcja haszująca: numer rejestracyjny modulo maksymalny rozmiar tablicy z haszowaniem

h(EL55080)=55080 mod 10000

- Loginy i hasła użytkowników systemu.



FUNKCJA HASZUJĄCA

- Jakie funkcje mieszania są dobre?
- Co robić w przypadku kolizji?
- Jakie założenia są niezbędne dla czasu O(1) w średnim przypadku?
- Co z przypadkiem najgorszym?

GŁÓWNE PODEJŚCIA:

- Adresowanie bezpośrednie
- Adresowanie otwarte
- Łańcuchy

Celem jest dobra funkcja mieszająca, uniwersalne haszowanie.

Jeśli klucze mają postać numeryczną wówczas pozycję w talicy (indeks) można obliczyć z wykorzystaniem funkcji haszującej postaci : adres (indeks) = klucz Modulo rozmiar tablicy (liczbę dostępnych adresów)
Jeśli klucz jest złożony z liter, wówczas można przypisać każdej literze kod tablicy ASCII i utworzyć wartość numeryczną W postaci sumy kodów ASCII dla danego wyrażenia. Długie ciągi takie jak numery telefonów mogą być dzielone na części A następnie sumowane: 014528345650 przeprowadzamy w: 01+45+28+34+56+54=218

ROZWIĄZYWANIE KOLIZJI METODĄ ADRESOWANIA OTWARTEGO

- Klucze mapowane na tą samą pozycję są wstawiane do tablicy w pierwsze wolne miejsce.
- Funkcja haszująca powinna być dwuargumentowa, przy czym drugi argument jest numerem próby

$$H(k,i) = (h(k) + i) \bmod m$$

m nie powinno być potęgą dwójki (wtedy bierzemy najmłodsze m bitów) – jeżeli nie mamy gwarancji, że ich rozkład jest równomierny, lepiej wybrać funkcję która zależy od wszystkich bitów klucza

m nie powinno być postaci $2^p - 1$ — wartość funkcji haszujących różniących się kolejnością dwóch sąsiednich cyfr binarnych będzie taka sama

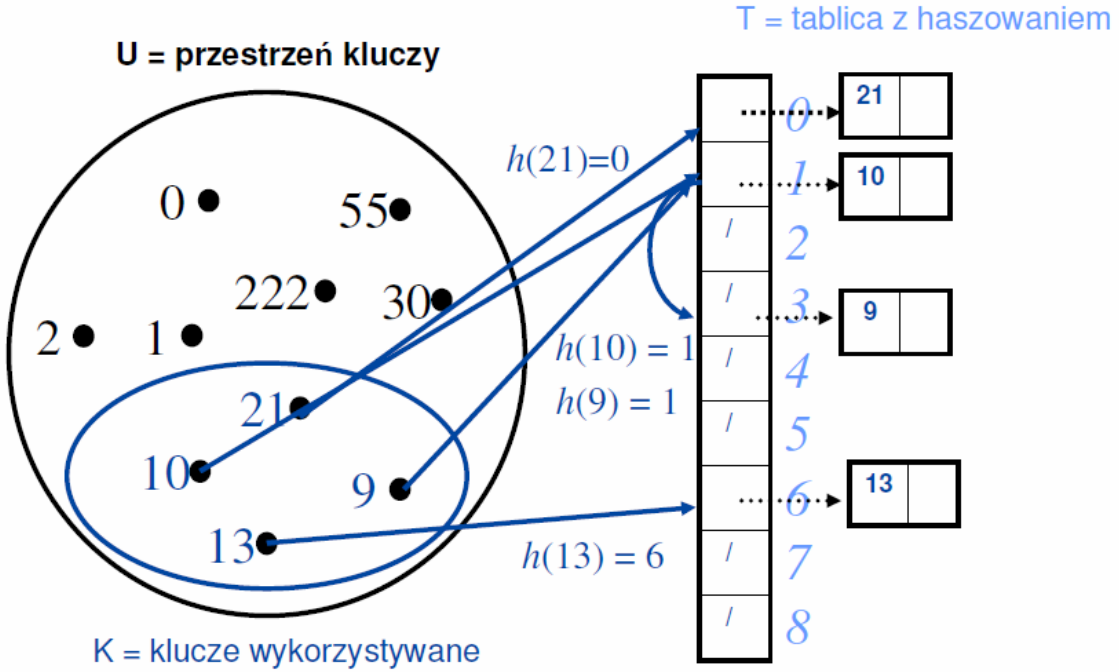
podobnie dla innych podstaw (np. 10, czyli $m \neq 10^p$, $m \neq 10^p - 1$)

dobrymi wartościami m są liczby pierwsze niezbyt bliskie potęgom 2

Weźmy $|U| = n =$ 2000 i załóżmy, że dopuszczamy maksymalnie 3 kolizje dla klucza.

Mamy *floor*(2000/3) = 666; liczba pierwsza bliska tej wartości, a jednocześnie daleka od potęg 2 to np. 701.

$$h(k) = k \bmod 701$$

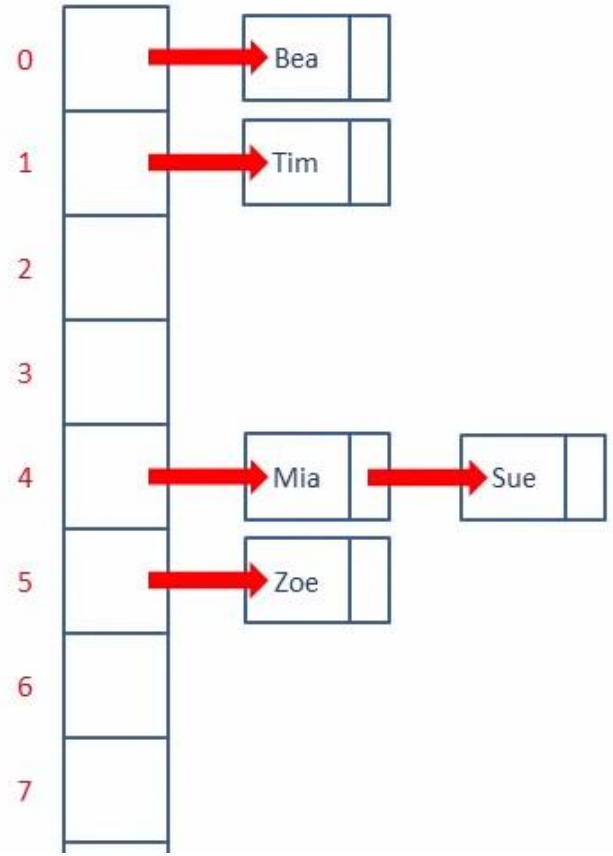


ROZWIĄZYWANIE KOLIZJI METODĄ ADRESOWANIA OTWARTEGO -STRATEGIE PRÓBKOWANIA

Najczęściej stosowane są trzy podejścia:

- Próbkowanie liniowe
- próbkowanie kwadratowe
- Haszowanie dwukrotne

ROZWIĄZYWANIE KOLIZJI - ŁAŃCUCHY

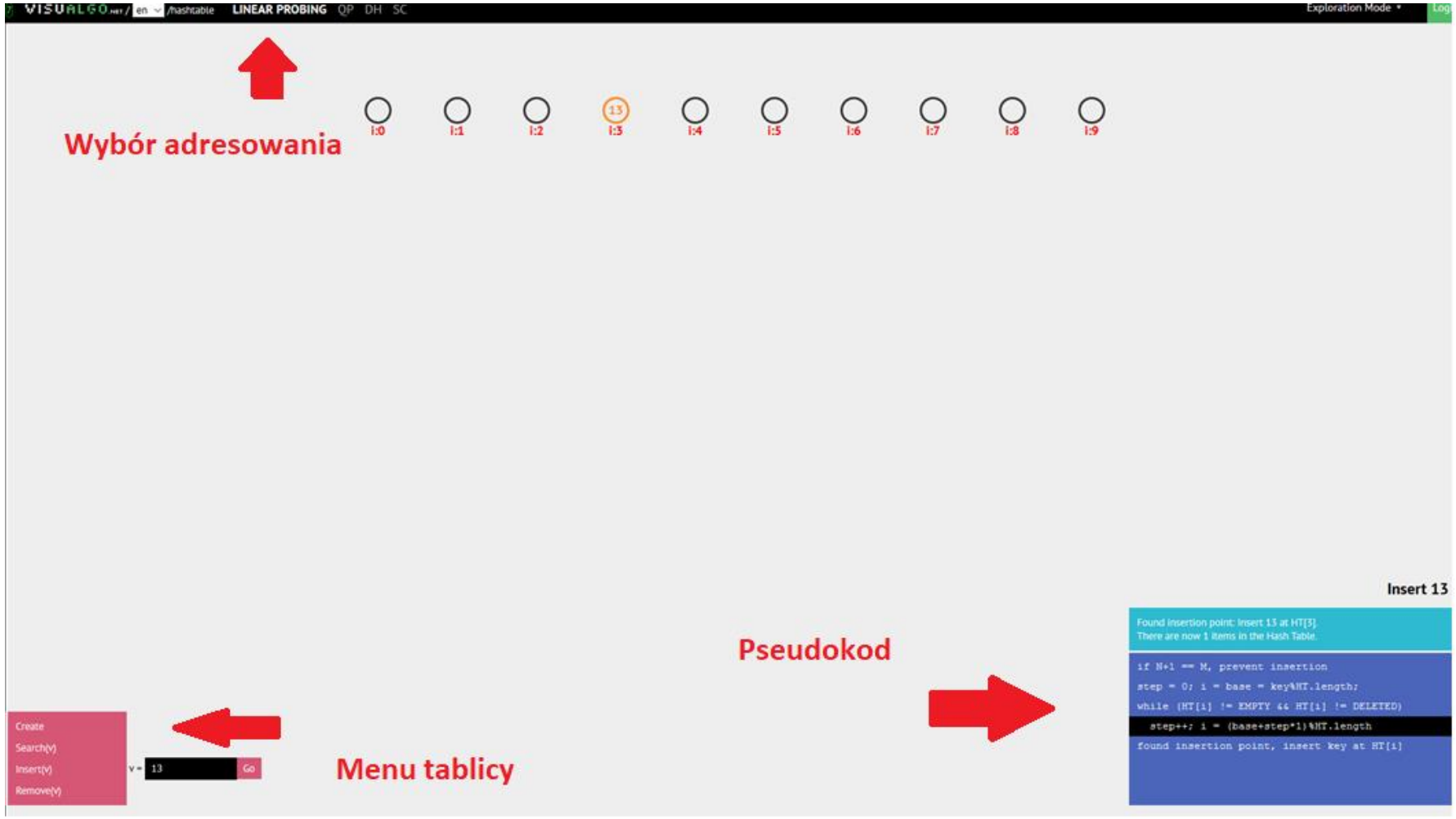


Mia	M	77	i	105	a	97	279	4
Tim	T	84	i	105	m	109	298	1
Bea	B	66	e	101	a	97	264	0
Zoe	Z	90	o	111	e	101	302	5
Sue	S	83	u	117	e	101	301	4

PRZYDATNE MATERIAŁY

Krótkie przedstawienie problemu tablic z haszowaniem w języku angielskim.
Osoby z niewystarczającą znajomością tego języka proszę o włączenie napisów i w ustawieniach zmianę języka napisów na polski – w tym przypadku automatyczne tłumaczenie daje sobie radę nieźle.
https://www.youtube.com/watch?v=KyUTuwz_b7Q

Strona z aplikacją tworzącą tablicę, przeszukującą ją i usuwającą elementy*
*UWAGA- proces usuwania różni się w przypadku zadania domowego!
Jednakże całość daje pewne wyobrażenie jak taka tablica funkcjonuje. Menu generatora tablicy w lewym dolnym rogu strony (czerwone). Po wykonaniu operacji w prawym dolnym rogu wyświetla się kod. Najpierw w pasku na górze należy wybrać
LP – Linear probing a następnie w wysuwanym menu na dole określić rozmiar i dodać elementy o danym kluczu.
https://visualgo.net/en/hashtable



https://visualgo.net/en/hashtable

ZADANIE DOMOWE 1 HASHTABLE

Proszę utworzyć tablicę z mieszaniem umożliwiającą przechowanie danych obiektu składającego się z ***klucza (typ long)*** *oraz* ***krótkiego łańcucha tekstowego*** *(można przyjąć, że liczba znaków nie przekroczy 8).*
W tablicy należy zastosować ***adresowanie otwarte z przyrostem liniowym równym 1.***

Wejście

W pierwszej linii pliku wejściowego znajduje się liczba całkowita określająca liczbę przypadków testowych. Następnie znajduje się opis kolejnych przypadków testowych, przy czym opis każdego przypadku ma formę wykonywalnego skryptu, w którym może wystąpić jedno z przedstawionych poniżej poleceń.

Ani pomiędzy poszczególnymi przypadkami, ani wewnątrz poszczególnych skryptów nie występują linie puste. Zestaw dopuszczalnych poleceń skryptu obejmuje następujące możliwości:

size - ustal rozmiar tablicy dla danego przypadku testowego na

add - dodaj obiekt o kluczu i łańcuchu do tablicy

delete - usuń z tablicy element o kluczu

print - wyświetl zawartość tablicy w następujący sposób: wyświetlane są tylko komórki wypełnione; każda z komórek zajmuje jeden wiersz i zawiera: indeks elementu w tablicy, klucz obiektu i łańcuch tekstowy rozdzielone spacjami; każde wywołanie funkcji print zakończone jest pustą linią

stop - koniec przypadku testowego

Wyjście

Wyjście zawiera kolejno rezultaty reakcji programu na polecenie "print".

Przykład

Dla danych wejściowych	Plik wyjściowy powinien zawierać
1 size 10 add 13 ala print add 23 ola print delete 13 print stop	3 13 ala 3 13 ala 4 23 ola 3 23 ola

OMÓWIENIE ZADANIA

Zasadniczą częścią problemu jest przyporządkowanie elementów w postaci [klucz, wartość] do odpowiedniego miejsca w tablicy. Na podstawie wcześniej przedstawionych informacji można zidentyfikować, że do elementów tych należy zastosować funkcję haszującą postaci klucz modulo rozmiar tablicy, ponieważ taka jest dostępna pula adresów. Osobną sprawą jest stworzenie środowiska, w którym operacja ta będzie wykonywana. Należy napisać program, który będzie w stanie obsłużyć polecenia skryptu i podać ich wynik wykorzystując koncepcję wyjścia i wejścia standardowego. Oznacza to, że domyślnie program oczekuje wprowadzania parametrów z klawiatury i wyprowadza je na ekran, ale jest w stanie obsłużyć przekierowania strumienia z/do pliku wg polecenia:
program.cpp << skrypt.in >> wynik.out, przy czym w skrypcie wejściowym znajdują się polecenia dla programu a w wyjściowym rozwiązanie (reakcja programu na polecenia).

OMÓWIENIE ZADANIA

Wspominałem Państwu, że proces usuwania przebiega inaczej niż w typowych tablicach tego typu.
Zacznijmy od tego, że w realnym przypadku, jeśli mamy do czynienia z procesem wielokrotnego usuwania oraz dodawania danych stosuje się raczej łańcuchy.
W przypadku adresowania otwartego powszechnym podejściem jest oznaczanie/maskowanie pola, z którego wartość usunięto by w trakcie wyszukiwania brak elementu nie zburzył całego procesu.
Jest to komunikacja dla algorytmu wyszukującego, że wcześniej coś tam było i należy szukać dalej wg ustalonego schematu tak jakby to pole było zajęte.
Taki proces spotkacie Państwo w zdecydowanej większości materiałów w Internecie – również tych, które podałem jako przykłady.
To zadanie ma charakter dydaktyczny i proces tutaj przebiega nieco inaczej.

OMÓWIENIE ZADANIA

Zwróćmy uwagę na to co stało się z pozycją elementu 23 ola po usunięciu elementu 13 ala. Widać ze zmienił on pozycję z 4 na 3.

Dla danych wejściowych	Plik wyjściowy powinien zawierać
1 size 10 add 13 ala print add 23 ola print delete 13 print stop	3 13 ala 3 13 ala 4 23 ola 3 23 ola

OMÓWIENIE ZADANIA

Ponieważ ten problem jest niestandardowy, pokazuję Państwu fragment innego skryptu testującego i jego wyniku. Przeanalizowanie tego przypadku powinno Państwu wystarczyć do poprawnego rozwiązania tego zadania.

```
size 11
add 27 A                5 27 A
add 38 B                6 38 B
add 116 C               7 116 C
add 161 D               8 161 D
add 96 E                9 96 E
print
delete 27               5 38 B
print                   6 116 C
delete 96               7 161 D
delete 116              8 96 E
delete 38
print                   7 161 D
delete 161
add 373 K               0 230 L
add 230 L               10 373 K
print
delete 373              10 230 L
print
add 361 M               0 119 N
add 119 N               9 361 M
print                   10 230 L
delete 361
print                   9 119 N
stop                    10 230 L
```

Wskazówka

Przy reorganizacji tabeli po usuwaniu elementów proszę zwrócić uwagę na elementy, które naturalnie zostały wstawione w dane miejsce i elementy, których pozycja została zmieniona na skutek tego, że ich podstawowy indeks był już wówczas zajęty.