AAL 2019Z Michał Kowieski

Treść zadania:

Przedmiotem analizy jest tablica mieszająca: tablica przechowuje rekordy zawierające napisy. Długość tablicy jest ograniczona arbitralnie przez pewną stałą K. Dla danego napisu s obliczamy k=M(s) i umieszczamy strukturę reprezentującą napis w tablicy mieszającej: H[k]. W przypadku kolizji funkcji mieszającej (H[k] zajęte) reprezentujące napis s struktury danych zapisywane są w sposób alternatywny zobacz warianty). Przedmiotem implementacji powinno być: dodanie i usunięcie elementów w H[]. Wybór funkcji mieszającej M(s) do decyzji projektanta - ale patrz wariant 3.

W13:

tablica H[k,n] (gdzie n=0...N) jest dwuwymiarowe, elementy kolidujące zapisywane są w lokalizacjach H[k,0], H[k,1], itd. (Oczywiście, przekroczenie przez drugi indeks rozmiaru tablicy będzie powodować odrzucenie elementu)

W21:

Testy przeprowadzić dla: listy słów języka polskiego wygenerowanych z zadanych tekstów. Generator należy wykonać samodzielnie.

W31:

Zastosować jedną funkcję mieszającą; dodatkowo przeprowadzić analizę dla enumeracji tablicy (wydobycia wszystkich elementów).

Opis funkcji mieszającej:

Zastosowana zostanie tak zwana "wielomianowa mieszająca funkcja krocząca" (ang. polynomial rolling hash function).

Wzór:

```
hash(s) = s[0] + s[1] * p + s[2] * p^2 + ... + s[n-1] * p^(n-1) mod m
```

- s - napis dla którego chcemy wyliczyć wartość skrótu
- n - długość napisu s
- m - duża stała
- p - duża stała

Działanie programu:

Na wejście program dostaje ciąg napisów. Najpierw dodaje je wszystkie do tablicy. Następnie generuje wewnętrzny stan tablicy mieszającej (jakie napisy się tam znajdują oraz w których są "kubełkach"). Ten stan zwraca użytkownikowi. Jako ostatni krok usuwa wszystkie elementy wcześniej dodane do tablicy.

Argumenty i sposób uruchomienia programów:

Program główny:

```
hashtablecli [--version] [--help] <command> [<args>]
```

Komendy:

IO

Opis:

Pobiera dane z stdio i zwraca wynik na stdout.

Przykład użycia:

```
hashtablecli io <<in.txt >>out.txt
```

GENERATE

Opis:

Generuje instancję problemu, rozwiązuje go i zwraca wynik.

Argumenty:

- data - ścieżka do tekstu
- n - ilość generowanych słów

Przykład użycia:

```
hashtablecli generate --data ./pan_tadeusz.txt -n 100
```

BENCHMARK

Opis:

Przeprowadza cały proces testowania z pomiarem czasu dla rosnącego n i porównuje ze złożonością teoretyczną.

Argumenty:

- data - ścieżka do tekstu
- n - rozmiar pierwszego problemu
- k - ilość problemów
- step - krok między problemami
- r - ilość instancji dla każdego problemu

Przykład użycia:

```
hashtablecli benchmark --data ./dziady.txt -n 1000 -k 30 --step 500 -r 10
```

Generator napisów:

```
strgeneratorcli [--version] [--help] [<args>]
```

Argumenty:

- data - ścieżka do tekstu
- n - ilość generowanych słów

Przykład użycia:

```
strgeneratorcli --data ./dziady.txt -n 10 >>out.txt
```

Program oczyszczający tekst:

```
textcleanercli [--version] [--help]
```

Przykład użycia:

```
textcleanercli <<in.txt >>out.txt
```

Wykorzystywane technologie:

- Język - Python 3
- Biblioteki - numpy, matplotlib, pandas, sphinx (dokumentacja)

Opis zachowania w warunkach brzegowych:

- Jeżeli dodanie napisu nie jest możliwe ponieważ tablica w wymiarze N jest pełna rzucany jest odpowiedni wyjątek.
- Jeżeli istnieje już dany napis w tablicy rzucany jest odpowiedni wyjątek.
- Jeżeli usunięcie napisu nie jest możliwe ponieważ danego napisu nie ma rzucany jesty odpowiedni wyjątek.