Zadania przed kolokwium III

Zadanie 1
Jest dana tablica A zawierająca N liczb całkowitych. Szukamy par identycznych elementów, które zajmują różne pola w tablicy. Para indeksów (P, Q) jest *identyczna*, jeżeli 0 <= P < Q < N oraz A[P] = A[Q]. Napisz algorytm, który zwróci liczbę identycznych par indeksów.
Przykładowo: A=[3, 5, 6, 3, 3, 5]. Są tutaj 4 pary identycznych indeksów (0,3); (0,4); (1,5); (3,4). Jako że P < Q, to (2,2) oraz (5, 1) nie są brane pod uwagę.

Zadanie 2
Pewna firma przechowuje dużo liczb pierwszych w postaci binarnej jako stringi "10101...". Zaimplementuj strukturę danych Set do przechowywania tych danych. Powinna wspierać metody:

- konstruktor Set(A), który tworzy Set z listy stringów A
- contains(s), która sprawdza, czy dana liczba jest w Secie

Oszacuj złożoność czasową i pamięciową powyższych funkcji.

Zadanie 3
Dana jest tablica T[N] zawierająca liczby naturalne (-1 oznacza puste pole) będąca tablicą hashującą z funkcją hash(x) i liniowym rozwiązywaniem konfliktów. Napisz algorytm, który policzy średnią liczbę skoków po tej tablicy jaka musi zostać wykonana, żeby odszukać jakąś zawartą w niej liczbę.

Zadanie 4
Mamy dany zbiór x1,x2,...,xn. Zależy nam na tym, aby funkcja, która otrzyma jako argumenty 2 indeksy (z tego zbioru) w czasie jak najmniejszym (optymalnie O(log(N)) znalazła minimalną wartość znajdującą się w zbiorze [x[indeks1], …, x[indeks2]]. Wybierz optymalną strukturę danych, która spełni postawiony warunek.
Uwaga: interesuje nas minimalny czas wyszukiwania, nie konstrukcji!

Zadanie 5
Zaprojektuj system autouzupełniania. Powinien on działać w następujący sposób:
System tworzymy konstruktorem AutoComplete(sentences, counts). sentences to tablica stringów zawierająca znaki a-Z i spacje, counts to tablica intów mówiąca o tym jaka jest częstość wyszukiwań zdania sentences[i]. Użytkownik wpisuje znaki a-Z + spacja i po każdym wpisanym znaku system powinien wypisać k zdań o najwyższym priorytecie, których prefixem jest dotychczas wpisany przez użytkownika ciąg znaków. Priorytet jest tym wyższy, im wyższy jest wskaźnik sentences[i] danego zdania. Jeśli dwa zdania mają identyczne sentences[i] to lepsze jest to niższe leksykograficznie. Gdy użytkownik poda znak #, to znaczy, że zakończył wyszukiwania i prefix znaków wprowadzonych przez użytkownika budowany jest od nowa.
Interfejs:
AutoComplete(sentences: String[], counts: Int[])
String[] getUserChar(letter: Char, k: Int)

Przykład:
sentences= [“bit”, “algo”, “bitalgo”]
counts = [3,2,7]
k = 2
Użytkownik podaje b - zwracamy [“bitalgo”, “bit”]
teraz podaje a - zwracamy []
teraz podaje # - od nowa budujemy prefix
podaje a - zwracamy [“algo”] itp.

Zadanie 6
Zaproponuj strukturę danych, którą tworzymy na podstawie tablicy słów words (składają się tylko z liter a-z) i tablicy weights, gdzie weights[i] mówi o wadze słowa words[i]. Struktura ma szybko realizować zapytania: query(prefix, suffix) - zwraca słowo z words o największym priorytecie, które ma jako prefiks prefix i sufiks suffix.

Zadanie 7 (skiplista była na PI)
Opisz jak zmodyfikować strukturę SkipListy, aby możliwe było wyszukiwanie i-tego elementu w czasie O(log(n)).