Zajęcia 28

Temat: Algorytmy tekstowe

Czas trwania: 2x45 min

Cel zajęć:

projektuje i programuje proste problemy z różnych dziedzin, stosuje przy tym: instrukcje wejścia/wyjścia, wyrażenia arytmetyczne i logiczne, instrukcje warunkowe, instrukcje iteracyjne, tablice, rekurencję, pisze własne funkcje rekurencyjne, struktury danych, biblioteka STL, algorytmy tekstowe KMP, haszowanie, łańcuchy znaków, testuje poprawność programów dla różnych danych, posługuje się zintegrowanym środowiskiem programistycznym przy pisaniu, uruchamianiu i testowaniu programów;

Efekty:

- umie uruchomić potrzebne oprogramowanie,
- umie napisać program z wykorzystaniem algorytmów tekstowych,
- potrafi wyszukiwać wzorzec w tekście,
- zna algorytm naiwny i KMP, haszowanie.

Formy i metody pracy: praca samodzielna, omówienie, wykład

Zadania do wykonania na zajęciach	Treści programowe
1. Czołgiści	M.3, P.2.15, A.3.3,
2. Ści(ą)gany	M.3, P.2.15, A.3.3,

Materiały do zajęć:

http://informatykaplus.edu.pl/upload/list/czytelnia/Zaawansowane_algorytmy_ALL.pdf ss. 13-15

Zadania do wykonania w domu:

Pociągi

https://szkopul.edu.pl/problemset/problem/_aNlLfqZBTY6FRf2_IGScTLn/site/

ZADANIA I ROZWIĄZANIA

Zadanie 1. Czołgiści

Limit pamięci: 128MB

W Skaryszewie odbywa się doroczny ogólnoświatowy zjazd czołgistów. Przybyło na niego swoimi czołgami n czołgistów. Każdy czołg ma nazwę (typ czołgu, nazwa nadana przez czołgistę itp.). Znając wszystkie nazwy czołgów musisz stwierdzić, ile jest par czołgów o tej samej nazwie.

Wejście

W pierwszym wierszu wejścia znajduje się jedna liczba całkowita n ($1 \leq n \leq 1000$) oznaczająca liczbę czołgistów (i tym samym liczbę czołgów). W kolejnych n wierszach znajdują się nazwy czołgów. Nazwa składa się wyłącznie z małych liter alfabetu angielskiego i ma nie więcej niż 2000 znaków.

Wyjście

Na wyjście należy wypisać jedną liczbę całkowitą, oznaczającą ilość par czołgów o tych samych nazwach.

Przykład

Wejście	Wyjście
5 rudy tygrys rudy rudy tygrys	4

Rozwiązanie

Rozwiązanie wolne: Możemy posortować wszystkie napisy i policzyć takie same pary czołgów. Sortowanie długich napisów będzie jednak dość powolne.

Rozwiązanie szybkie: Dla każdego napisu obliczmy hasz (sumę kontrolną) dla wszystkich jego liter. Teraz możemy po prostu dla każdego elementu sprawdzić, ile jest takich samych po lewo od niego. Aby uniknąć kolizji trzymajmy dwa różne hasze dla każdego czołgu.

Jak obliczyć sumę kontrolną dla nazwy czołgu?

```
napis s
p ← 263
q ← 10^9+7
w ← 0
dla i=0,1,2,..., |s| wykonuj
    w ← (p · p + s[i]) mod q
```

Teraz możemy po prostu dla każdego elementu sprawdzić, ile jest takich samych po prawo od niego. Aby uniknąć kolizji trzymajmy dwa różne hasze dla każdego czołgu.

```
hasz[n] // obliczone hasze dla każdej nazwy czołgu
dla i=0,1,2,...,n-1 wykonuj
    dla j=0,1,2,...,i-1 wykonuj
        jeżeli (hasz[i]=hasz[j])
            wynik ← wynik + 1
```

Aby uniknąć kolizji wskazane jest obliczanie dwóch różnych haszy dla każdej nazwy czołgu.

Zadanie 2. Ści(ą)gany

Dostępna pamięć: 64MB

Janek wymyślił nowy aforyzm. Ostatnio stało się to jego hobby. Zauważył jednak, że niektórzy zaczęli wykorzystywać jego sentencję w swoich wypracowaniach bez pytania go o zgodę. Trochę go to irytuje, więc postanowił sprawdzić, kto ściąga i wypisuje jego maksymę.

Wejście

W pierwszym wierszu wejścia znajduje się jedna liczba całkowita n ($1 \le n \le 100$), oznaczająca liczbę wypracowań do sprawdzenia. W drugim wierszu zapisana jest sentencja Janka: ciąg znaków nie dłuższy niż 1000000 złożony wyłącznie z dużych liter alfabetu łacińskiego. W kolejnych n liniach znajdują się wypracowania do sprawdzenia (ciągi znaków nie dłuższe niż 1000000 złożony wyłącznie z dużych liter alfabetu łacińskiego, po jednym wypracowaniu w linii).

Wyjście

W kolejnych n znajdują się komunikaty *TAK* lub *NIE* - informacja, czy w danym wypracowaniu zawarta jest sentencja Janka.

Przykład

Wejście	Wyjście
3 ABC ABABABCAB ABABABABAB AB	TAK NIE NIE

Rozwiązanie

Naszym zadaniem jest wyszukanie wzorca w tekście. Spróbujmy sprawdzać to z użyciem haszowania. W tym celu najpierw wyznaczmy potęgę $p^{|wzorzec|}$ oraz hasz wzorca (oczywiście ze względu na wartości pamiętamy tylko resztę dzielenia przez dużą liczbę pierwszą q):

```
p←127, q←10^9+7LL
pn←1
dla i=1,2,3,…,|wzorzec| wykonuj
    pn←(pn · p) mod q
dla i=|wzorzec|-1,…,2,1,0 wykonuj
    h←(pn · p + wzorzec[i]) mod q
```

Liczymy również hasze sufiksowe dla przeszukiwanego tekstu:

```
H[] // hasze sufiksowe
H[|tekst|]←0
dla i=|tekst|-1,…,2,1,0 wykonuj
    H[i]←(H[i+1] · p + tekst[i]) mod q
```

Teraz dla wskazanego tekstu sprawdźmy, czy hasz wycinka tekstu jest równy haszowi wzorca (pamiętajmy o mnożeniu przez potęgę `pn`). Gdybyśmy odejmując otrzymali wartość ujemną, dodajmy do wyniku `q`).

```
dla i=0,1,2,…,|tekst|-|wzorzec| wykonuj
    haszWycinka ← (H[i]-H[i+|tekst|*pn]) mod q
    dopóki (haszWycinka<0) haszWycinka ← haszWycinka + q
    jeżeli (h = haszWycinka)
            wypisz TAK
```

Oczywiście hasze sufiksowe obliczamy i sprawdzamy dla każdego z tekstów.