**Jakub Łącki**
Treść zadania, Opracowanie

**Dawid Dąbrowski, Karol Pokorski**
Program

**Dostępna pamięć: 64 MB.**

*OI, etap I, 6.10–3.11.2014*

# Pieczęć

*Bajtek znalazł dziś w swojej poczcie dziwny dokument. Było to zawiadomienie o tym, że w spadku po swoim wuju Bajtazarze otrzymał gigantyczną sumę pieniędzy. Pismo jest opieczętowane wielokrotnie pieczęcią Królestwa Bajtocji. Bajtek woli się upewnić, że nie pada ofiarą naciągaczy. Chce więc sprawdzić, czy pieczęć jest prawdziwa.*

*Bajtek wie, jak wygląda pieczęć Królestwa Bajtocji. Na otrzymanym piśmie jest jednak tyle tuszu, że trudno powiedzieć, czy skrupulatny urzędnik przystawił pieczęć bardzo wiele razy, czy też jest to nieudolna próba zmylenia Bajtka przez oszustów. Pomóż Bajtkowi i napisz program, który mając dany wzór z otrzymanego pisma oraz opis pieczęci, stwierdzi, czy Bajtek jest w posiadaniu autentycznego zawiadomienia o spadku.*

*Pieczęć ma specjalne zabezpieczenia, w związku z czym: (1) nie można jej podczas przystawiania obracać, (2) nie można jej przyłożyć tak, żeby pozostawiła tusz poza stemplowanym dokumentem, oraz (3) każde miejsce na dokumencie może być pokryte tuszem z pieczęci co najwyżej raz.*

## Wejście

*W pierwszym wierszu standardowego wejścia znajduje się jedna liczba całkowita $q$ ($1 \leqslant q \leqslant 10$), określająca liczbę zestawów danych. W kolejnych wierszach znajdują się opisy kolejnych zestawów danych.*

*W pierwszym wierszu opisu pojedynczego zestawu danych znajdują się cztery liczby całkowite $n$, $m$, $a$ oraz $b$ ($1 \leqslant n, m, a, b \leqslant 1000$) pooddzielane pojedynczymi odstępami.*

*W kolejnych $n$ wierszach znajduje się opis wzoru z pisma. Każdy z tych wierszy składa się z $m$ znaków, z których każdy to* `.` *(kropka) lub* `x`*. Kropka oznacza, że w danym miejscu kartki nie ma śladów tuszu, a* `x`*, że tusz pozostawił ślad.*

*Następnie opisany jest wygląd przykładowego dokumentu po jednokrotnym opieczętowaniu go pieczęcią Królestwa Bajtocji. Jest on podany w takim samym formacie jak opis dokumentu otrzymanego przez Bajtka, w $a$ wierszach zawierających po $b$ znaków* `.` *oraz* `x`*. Możesz założyć, że zarówno opis wzoru z pisma, jak i opis przykładowego dokumentu zawiera jakiś ślad tuszu.*

*W testach wartych łącznie 44% punktów dla każdego zestawu danych zachodzi: $n, m, a, b \leqslant 150$.*

## Wyjście

*Twój program powinien wypisać na standardowe wyjście dokładnie $q$ wierszy. W $i$-tym wierszu powinna się znaleźć odpowiedź dla $i$-tego zestawu danych.*

*Jeśli dokument otrzymany przez Bajtka mógł zostać opieczętowany pieczęcią, należy wypisać jedno słowo* `TAK`*. Jeśli zaś pismo otrzymane przez Bajtka na pewno jest fałszerstwem, należy wypisać jedno słowo* `NIE`*.*

### Przykład

| *Dla danych wejściowych:* | *poprawnym wynikiem jest:* |
|---|---|
| `2` | `TAK` |
| `3 4 4 2` | `NIE` |
| `xx..` | |
| `.xx.` | |
| `xx..` | |
| `x.` | |
| `.x` | |
| `x.` | |
| `..` | |
| `2 2 2 2` | |
| `xx` | |
| `xx` | |
| `.x` | |
| `x.` | |

**Testy „ocen”:**

`1ocen`: *Jeden przypadek: dokument o wymiarach $2 \times 3$ cały pokryty tuszem, pieczątka o wymiarach $1 \times 2$. Odpowiedź* `NIE`.

`2ocen`: *Jeden przypadek: dokument–szachownica o wymiarach $8 \times 8$, pieczątka–szachownica o wymiarach $3 \times 3$. Odpowiedź* `NIE`.

`3ocen`: *$q = 10$, wszystkie przypadki takie same: dokumenty $1000 \times 1000$, pieczątki losowe, wymiaru $20 \times 20$, zawsze na dokumencie pieczątka odciśnięta jest dokładnie raz. Wszystkie odpowiedzi* `TAK`.

## Rozwiązanie

Naszym celem jest sprawdzenie, czy dokument otrzymany przez Bajtka mógł zostać wykonany przez wielokrotne odciskanie na nim pieczęci Królestwa Bajtocji. Dokument podany jest w postaci tablicy rozmiaru $n \times m$. Będziemy mówić, że składa się on z *$n$ wierszy* oraz *$m$ kolumn*. W każdym wierszu mamy *$m$ pól*, a każde pole jest albo zamalowane, albo białe. Pieczęć jest podana w taki sam sposób jako tablica złożona z $a$ wierszy oraz $b$ kolumn.

W trakcie odciskania pieczęci Królestwa Bajtocji na dokumencie należy przestrzegać trzech ograniczeń. Po pierwsze, pieczęci nie wolno obracać. Po drugie, za każdym razem należy ją przykładać tak, by zamalowywała ona jedynie pola leżące w granicach dokumentu. Wreszcie po trzecie, każde pole dokumentu może być pokryte tuszem co najwyżej jednokrotnie. Jeśli końcowy wygląd pewnego dokumentu da się uzyskać za pomocą pewnej liczby przystawień pieczęci, przy zachowaniu podanych reguł, dokument taki nazwiemy *poprawnym*. W szczególności pusty dokument, tj. taki, na którym nie ma ani jednego zamalowanego pola, uznajemy za poprawny (choć opis takiego dokumentu nie może się pojawić na wejściu).

Aby sprawdzić, czy dokument jest poprawny, spróbujemy odtworzyć sekwencję odciśnięć pieczęci. Spójrzmy na proces pieczętowania dokumentu od końca. Wyobraźmy sobie, że nagraliśmy film, który przedstawia cały proces przykładania pieczęci królestwa Bajtocji do dokumentu, i teraz oglądamy go wstecz, tj. od końca do początku. Na początku filmu dokument jest zamalowany tuszem. Każde przyłożenie pieczęci wymazuje ślady tuszu z niektórych pól, a w końcu cały dokument jest pusty. Chcielibyśmy teraz wymazać wszystkie ślady tuszu z dokumentu, postępując podobnie jak na puszczonym w tył filmie.

Wyobraźmy sobie *pieczęć wymazującą*, która może posłużyć do cofania wcześniejszych odciśnięć pieczęci. Ma ona taki sam kształt, jak opisana na wejściu pieczęć Królestwa Bajtocji, tj. działa na dokładnie tak samo rozmieszczone pola dokumentu i wymaga przestrzegania analogicznych reguł: nie wolno jej obracać ani przykładać tak, by działała na pola poza dokumentem. Ponadto, każde wymazywane pole musi być zaciemnione w momencie wymazywania. Aby rozwiązać zadanie, wystarczy sprawdzić, czy za pomocą pieczęci wymazującej możemy w podanym dokumencie zamienić wszystkie zamalowane pola na pola białe.

Na początku przyjrzyjmy się najprostszemu możliwemu algorytmowi: przykładamy pieczęć wymazującą w dowolnym miejscu (w którym wolno nam ją przyłożyć) i powtarzamy tę operację tak długo, jak się da. Jeśli na końcu otrzymamy pusty dokument, mamy dowód, że podany na wejściu dokument był poprawny. Mógł on przecież powstać przez odciskanie pieczęci Królestwa Bajtocji w dokładnie tych samych miejscach, w których użyliśmy pieczęci wymazującej. Jednak co w przypadku, gdy na kartce pozostało jeszcze trochę śladów tuszu i nie możemy już więcej użyć pieczęci wymazującej?

Spójrzmy na następujący dokument rozmiaru $1 \times 4$:

```
xxxx
```

oraz pieczęć zamalowującą dwa sąsiednie pola:

```
xx
```

Skorzystajmy z naszego algorytmu i w początkowym dokumencie wymażmy dwa pola – drugie i trzecie od lewej:

```
x..x
```

Dostaliśmy dokument, na którym nie da się już użyć naszej pieczęci wymazującej. Widać jednocześnie, że gdybyśmy przykładali pieczęć wymazującą w inny sposób, np. najpierw do pierwszego i drugiego pola dokumentu, a potem do trzeciego i czwartego, udałoby się nam uzyskać pusty dokument.

Nie możemy zatem przykładać pieczęci wymazującej w dowolnym miejscu. Potrzebujemy nieco sprytniejszego sposobu. Łatwo przekonać się, że jeśli nasza pieczęć ma taki kształt, jak w powyższym przykładzie, wystarczy przykładać lewe pole pieczęci do pierwszego od lewej zamalowanego pola na dokumencie. Jeśli w ten sposób uda nam się wymazać cały tusz, dokument jest poprawny, a w przeciwnym razie – nie. Jak zaraz pokażemy, to podejście daje się rozszerzyć na dowolne dokumenty i pieczęcie.

Spójrzmy na dokument i znajdźmy w nim pierwszy od góry wiersz, w którym co najmniej jedno pole jest zamalowane. Następnie w tym wierszu znajdźmy pierwsze od

lewej zamalowane pole. To pole nazwiemy *pierwszym* polem dokumentu. Analogicznie zdefiniujmy pierwsze pole pieczęci. Czas na kluczową obserwację.

**Obserwacja 1.** Rozważmy dokument, który został poprawnie opieczętowany pewną liczbą odciśnięć pieczęci Królestwa Bajtocji. Wówczas pierwsze pole dokumentu zostało zamalowane pierwszym polem pieczęci.

Uzasadnienie obserwacji jest bardzo proste. Niech $p$ będzie pierwszym polem otrzymanego dokumentu. Rozważmy moment, w którym zamalowane zostało pole $p$. Zastanówmy się, gdzie mogła wówczas zostać przyłożona pieczęć. Ponieważ $p$ to pierwsze pole ostatecznego dokumentu, nie mogliśmy zamalować żadnego pola w wierszach powyżej $p$ oraz żadnego pola na lewo od $p$ w tym samym wierszu. Stąd już wprost wnioskujemy, że dopuszczalne jest tylko takie przyłożenie pieczęci, w którym pole $p$ zostaje zamalowane przez pierwsze pole pieczęci.

## Prosty algorytm

Poczyniona obserwacja wskazuje, w których miejscach pieczęć została przyłożona do dokumentu. W naszym rozwiązaniu odtwarzamy sekwencję przyłożeń pieczęci i zgodnie z tą sekwencją używamy pieczęci wymazującej na podanym na wejściu dokumencie. W tym celu znajdujemy pierwsze pole dokumentu i przykładamy w nim pierwsze pole pieczęci wymazującej. Krok ten powtarzamy tak długo, jak to możliwe. Jeśli w pewnym momencie otrzymamy pusty dokument, wiemy, że podany na wejściu dokument był poprawny. Druga opcja jest taka, że dokument zawiera jeszcze zamalowane pola, jednak nie możemy na nim w żaden sposób odcisnąć pieczęci wymazującej. Taki dokument uznamy za niepoprawny.

Zapiszmy nasze dotychczasowe ustalenia w postaci algorytmu. Dopóki co najmniej jedno pole dokumentu jest zamalowane, powtarzamy kolejno następujące kroki:

1. Znajdź pierwsze pole dokumentu.
2. Sprawdź, czy można odcisnąć pieczęć wymazującą na dokumencie, przykładając pierwsze pole pieczęci do pierwszego pola dokumentu (należy sprawdzić, czy przyłożona pieczęć nie wystaje poza dokument oraz czy pola, które mają zostać wymazane, są aktualnie zamalowane).
    - Jeśli nie jest to możliwe, zgłoś wynik „dokument niepoprawny”.
    - W przeciwnym razie odciśnij pieczęć wymazującą na dokumencie, tj. wymaż tusz z odpowiednich pól dokumentu.

Jeśli po zakończeniu wykonania algorytmu otrzymamy pusty dokument, dokument uznajemy za poprawny.

Zastanówmy się teraz, jaki jest czas działania naszego rozwiązania. Znalezienie pierwszego pola dokumentu realizujemy w najprostszy możliwy sposób: po prostu przeglądamy dokument od samej góry wiersz po wierszu. Wymaga to czasu $O(nm)$. Podobnie w czasie $O(ab)$ znajdujemy pierwsze pole pieczęci. Wreszcie sprawdzenie, czy pieczęć można przyłożyć, łatwo zrealizować w czasie $O(ab)$, porównując wszystkie pola

opisu pieczęci z odpowiadającymi polami dokumentu. W takim samym czasie działa też wymazywanie z dokumentu odpowiednich pól. Zatem jednokrotne wykonanie opisanych powyżej kroków wymaga czasu $O(nm+ab)$. Dla uproszczenia dalszej analizy tego algorytmu załóżmy, że $ab \leqslant nm$, co pozwala nam uprościć czas działania do $O(nm)$.

Ile razy będziemy musieli powtórzyć kroki 1 i 2? Oznaczmy przez $d$ liczbę pól, które zamalowuje pieczęć. Za każdym razem, gdy wykonamy obydwa kroki, wymażemy z dokumentu dokładnie $d$ zamalowanych pól. Zatem łącznie co najwyżej $O(nm/d)$ razy wykonamy obydwa kroki algorytmu.

W pesymistycznym przypadku mamy jednak $d = 1$ i wtedy najlepsze ograniczenie na czas działania całego algorytmu to $O(n^2m^2)$. Jeśli poprawnie zaimplementujemy takie rozwiązanie, otrzymamy 44% punktów.

## Rozwiązanie wzorcowe

Opiszemy teraz, jak przyspieszyć nasz algorytm. Zastanówmy się, jak działa aktualna wersja algorytmu w pesymistycznym przypadku. Jak wcześniej zauważyliśmy, zachodzi on wtedy, gdy podana na wejściu pieczęć zamalowuje tylko jedno pole dokumentu. W tej sytuacji opis pieczęci podany na wejściu może być bardzo nieefektywny. Przyjmijmy, że $n = m = a = b = 1000$. Wtedy opis pieczęci to tablica $1000 \times 1000$, w której tylko jedno pole jest zamalowane. Co więcej, przy każdym przyłożeniu pieczęci przeglądamy tę tablicę w całości!

Skoro interesują nas jedynie zamalowane pola pieczęci, będziemy ją reprezentować w inny sposób. Przyjmijmy ponownie, że pieczęć zamalowuje dokładnie $d$ pól. Wówczas do opisu pieczęci wystarczy nam lista składająca się z $d$ elementów: każdy element opisywać będzie współrzędne jednego pola, które zamalowuje pieczęć. Ponumerujmy wiersze i kolumny pieczęci kolejnymi liczbami naturalnymi, poczynając od 1. Wiersze numerujemy od góry do dołu, a kolumny od lewej do prawej. Przykładowo, pieczęć

```
..
.x
xx
```

możemy zapisać w postaci listy $(2,2),(3,1),(3,2)$. Każdy element na tej liście określa numer wiersza oraz numer kolumny jednego zamalowanego pola. Na potrzeby implementacji prościej nam jednak będzie zapamiętać położenia zamalowanych pól *względem* położenia pierwszego pola. Dlaczego? Gdy znajdziemy pierwsze pole dokumentu, chcemy do niego przyłożyć pierwsze pole pieczęci. A zatem potrzebujemy wiedzieć, gdzie względem niego leżą inne zamalowane pola pieczęci. Pieczęć z powyższego przykładu będziemy więc reprezentować w postaci listy $(0,0),(1,-1),(1,0)$. Jeśli położenie pierwszego pola to $(x,y)$, wystarczy zmodyfikować pierwotną reprezentację, odejmując od każdej pierwszej współrzędnej $x$, a od każdej drugiej $y$. Dzięki temu drugi krok algorytmu możemy wykonać w czasie proporcjonalnym do liczby zamalowanych pól na pieczęci, czyli $O(d)$. Oczywiście na samym początku algorytmu musimy jeszcze przetworzyć reprezentację pieczęci do nowej postaci, co można jednak prosto wykonać w czasie $O(ab)$. Rozwiązanie oparte na tym pomyśle znaleźć można w pliku `pies1.cpp`.

Nadal jednak sporo czasu zużywamy w pierwszym kroku algorytmu na znalezienie pierwszego pola dokumentu. Używamy tu oczywistego algorytmu, który przegląda dokument wiersz za wierszem i zatrzymuje się, gdy tylko znajdzie zamalowane pole. Zauważmy jednak, że w trakcie wykonania algorytmu możemy jedynie zamieniać zamalowane pola na pola białe. A zatem pierwsze pole dokumentu może się jedynie przesuwać coraz „dalej". Jeśli więc w pewnym momencie stwierdzimy, że pierwszym polem dokumentu jest $p$, to gdy następnym razem szukamy pierwszego pola dokumentu, możemy pominąć wszystkie pola leżące powyżej pola $p$ oraz w tym samym wierszu na lewo od niego. Dzięki temu w trakcie *wszystkich* wykonań pierwszego kroku algorytmu każde pole przejrzymy co najwyżej jednokrotnie. W efekcie sumaryczny czas spędzony w pierwszym kroku to $O(nm)$.

Implementację tego algorytmu znaleźć można w pliku `pie.cpp`. Zajmijmy się teraz czasem działania. Na początku budujemy listową reprezentację pieczęci w czasie $O(ab)$, otrzymując listę długości $d$. Następnie, algorytm wykonuje $O(\frac{nm}{d})$ faz. Krok drugi zajmuje za każdym razem czas $O(d)$, co łącznie daje $O(nm)$, czyli tyle samo ile wynosi łączny czas wykonywania pierwszego kroku. Musimy jeszcze umieć szybko stwierdzać, czy dokument posiada zamalowane pole. W tym celu wystarczy stale utrzymywać w pomocniczej zmiennej liczbę aktualnie zamalowanych pól. Dopóki zmienna ta ma dodatnią wartość, dokument posiada zamalowane pola. Tym samym poprawność dokumentu potrafimy sprawdzić w czasie $O(nm + ab)$, czyli w czasie liniowym od rozmiaru danych wejściowych.

# Zawody II stopnia

opracowania zadań