SPOJ Problem Set (challenge)

854. Ustawione sojusze

Problem code: BSPIES

W ostatnich latach rywalizacja pomiędzy miastami Bajtlandii, tradycyjnie przejawiająca się w konkursach ogrodniczych, przybrała bardziej militarny charakter. Wybuch wojny domowej wydaje się wręcz nieuchronny. Trochę kłopotliwe jest jednak często stawiane pytanie, kto będzie walczył przeciwko komu; wydaje się, że nawet burmistrzowie zainteresowanych miast nie do końca znają na nie odpowiedź. Aby uporządkować sytuację, w stolicy zwołano Walny Zjazd Emisariuszy Miast Bajtlandii (WZEMB). Jego głównym celem jest powołanie do życia frakcji, które będą walczyły przeciwko sobie.

Zasada obrad WZEMBu jest prosta. Jeden po drugim, emisariusze podchodzą do Karty z wypisaną na niej listą frakcji. Karta początkowo jest pusta. Każdy emisariusz, czytając Kartę od góry do dołu, stara się znaleźć na niej pierwszą frakcję, w skład której wchodzą wyłącznie miasta nie będące w stanie konfliktu z miastem reprezentowanym przez emisariusza. Jeżeli taka frakcja istnieje, emisariusz dołącza do niej nazwę swojego miasta; jeżeli nie -- zakłada dla swojego miasta nową frakcję, dodając wpis nowej frakcji pod wszystkimi innymi.

Sprawami wewnętrznymi Bajtlandii są żywotnie zainteresowani jej, nie zawsze życzliwie nastawieni, sąsiedzi. Agentom bitlandzkiego wywiadu udało się uzyskać dostęp do tajnych planów bajtlandzkiego WZEMBu. Mają teraz możliwość wpłynięcia na jego program poprzez zmodyfikowanie kolejności, w której emisariusze podchodzą do Karty. Zamierzonym celem jest doprowadzenie do sytuacji, w której liczba powstających w Bajtlandii frakcji jest możliwie jak największa. Spróbuj napisać program automatyzujący to zadanie.

Wejście

W pierwszej linii pliku wejściowego podano liczbę przypadków testowych t (t=100). Każdy przypadek testowy zaczyna się od linii zawierającej liczbę całkowitą n, oznaczającą liczbę miast w Bajtlandii (1<=n<=5000). W kolejnej linii podana jest liczba całkowita m (1<=m<=10000). Każda z kolejnych m linii zawiera parę liczb całkowitych a b oddzielonych spacją, oznaczającą, że miasta a i b znajdują się w stanie konfliktu ze sobą (1<=a,b<=n).

Wyjście

Dla każdego przypadku testowego wypisz ciąg n liczb identyfikujących miasta, których emisariusze powinni kolejno podchodzić do Karty.

Wynik

Wynik uzyskany przez program jest równy całkowitej liczbie frakcji, które powstaną w Bajtlandii (zsumowanej po wszystkich przypadkach testowych).

Przykład

Wejście:

```
2

3
2
1 2
3 2

4
3
2 3
4 3
1 2
```

Wyjście:

```
1 2 3
1 4 3 2
```

Wynik:

```
2pkt. + 3pkt. = 5pkt.
```

Added by: Adrian Kosowski
Date: 2006-05-28
Time limit: 20s
Source limit:50000B
Languages: All

SPOJ Problem Set (challenge)

857. Operator Jaś

Problem code: TOFFMILL

Jaś, odkąd pamiętają jego rodzice, przejawiał zainteresowanie do wszelkich pojazdów na czterech kółkach i do... łamigłówek matematycznych. Rodzina Jasia, podobnie jak i ich sąsiedzi, mają duże ogrody wymagające częstej pielęgnacji. Postanowili zatem połączyć zamiłowania Jasia i sprezentowali mu najnowszy model kosiarki do trawy. Odtąd Jaś wytycza nowe trasy w ogrodach swoich rodziców i sąsiadów, optymalizując czasy ich przejazdów, przy okazji kosząc im trawę...

Wejście

Każdy ogród podzielony jest na pola kwadratowe o wymiarze jednostkowym. Jaś porusza się pomiędzy sąsiednimi polami, w dowolnym z czterech kierunków. Zakładać będziemy, że ogród nie posiada wewnątrz żadnych innych pól niż trawnik, który musi zostać skoszony. Zaczynając ze wskazanego pola, Jaś musi skosić cały ogród i powrócić do punktu startu, minimalizując przy tym łączną długość drogi. W pierwszej linii podana jest informacja o liczbie ogrodów (zestawów testowych), t <= 10. W każdym zestawie testowym, w pierwszej linijce podana jest łączna ilość odcinków z brzegu ogrodu, 4 <= n <= 20000. Druga linia zawiera dokładnie n liczb całkowitych $a_1,...,a_n$ opisujących długości kolejnych odcinków z brzegu ogrodu (1 <= $|a_i|$ <= 250). Każde dwa kolejne odcinki z brzegu ogrodu są do siebie prostopadłe. Długość i-tego odcinka jest równa wartości bezwzględnej a_i, znak określa kierunek: "+" oznacza N lub E, "-" oznacza S lub W, zakładając, że ogród obchodzi się dookoła zgodnie ze wskazówkami zegara. Pole początkowe wskazane jest przez początek pierwszego odcinka z brzegu ogrodu, który jest podany w kierunku N/S. Przyjmiemy, że cały ogród mieści się w kwadracie 1000 x 1000.

Ilustracja pomocnicza treści

Wyjście

Na wyjściu należy podać liczbe oraz sekwencje ruchów kosiarki Jasia z wykorzystaniem informacji o kierunkach N, S, E oraz W, zaczynając od pierwszego pola (początek pierwszego odcinka z brzegu).

Punktacja

Każdy zestaw testowy będzie punktowany niezależnie. Za wynik punktowy danego zestawu testowego przyjęty będzie stosunek długości trasy Jasia do całkowitej liczby pól w ogrodzie. Łączny wynik będzie wartością średnią ze wszystkich wyników uzyskanych z poszczególnych testów.

Przykład

Wejście:

```
5
12
+1 +1 +1 +1 -1 +1 -1 -1 -1 -1 +1 -1
4
+2 +1 -2 -1
```

```
14
+2 +1 +2 +2 -1 +2 -3 -1 +1 -1 +1 -1 -2 -2
8
-2 -1 +1 -3 +2 +3 -1 +1
12
+7 +1 -3 +1 +3 +1 -7 -1 +1 -1 -1 -1
```

Wyjście:

```
8 ENSEWSNW
2 NS
18 NENNESEESSNWNWWSSW
10 WWWNEESESN
26 NNNNNNSSSEENNNSSSSSSNNWSWS
```

Wynik:

```
1.353
```

Added by: Michał Małafiejski
Date: 2006-05-28
Time limit: 1s-20s
Source limit:50000B
Languages: All

SPOJ Problem Set (challenge)

876. Kompresor prac domowych

Problem code: PZPI06_X

W ramach proekologicznej akcji pod hasłem "szanuj lasy, oszczędzaj papier", Jaś postanowił ograniczyć zużycie papieru w procesie swojej edukacji. Niestety, jego pierwszy pomysł polegający na nieodrabianiu prac domowych spotkał się ze zdecydowaną dezaprobatą nauczycieli. Dlatego też zasmucony Jaś ograniczył się tylko do kompresji pracy domowej i potrzebuje w tym celu Twojej pomocy.

Twoim zadaniem jest skompresowanie podanego na wejściu tekstu pracy domowej, złożonego wyłącznie z małych liter alfabetu 'a'-'z'. Postać skompresowana musi być czytelna dla dekompresora (w który wyposażony został każdy nauczyciel Jasia).

Dekompresor przetwarza tekst znak po znaku, interpretując poszczególne znaki jako polecenia zgodnie z następującymi zasadami:

- znak 'a'-'z': wypisz napotkaną literę,
- znak '+', po którym następuje liczba naturalna *n*: wypisz ponownie *n* ostatnio wypisanych liter,
- znak '-', po którym następuje liczba naturalna *n*: wymaż *n* ostatnio wypisanych liter.

Wartość liczby *n* nigdy nie może przekroczyć liczby dotychczas wypisanych znaków, a łączna liczba wymazywanych znaków nie może przekroczyć trzykrotności długości dekompresowanego pliku.

Wejście

Na wejściu podany jest ciąg znaków 'a'-'z' zakończony znakiem nowej linii, nie dłuższy niż 10^5 znaków.

Wyjście

Na wyjściu należy wypisać ciąg znaków w skompresowanej postaci. Postać skompresowana musi spełniać założenia podane w treści zadania.

Wynik

Liczba punktów przyznanych za rozwiązanie zadania jest równa stosunkowi długości tekstu skompresowanego do długości tekstu zdekompresowanego.

Przykład

Wejście:
```
abcabcabcabddddddd
```

Wyjście:

```
abc+3+6-1ddd+3
```

Wynik:
```
0.824
```

Added by: Adrian Kosowski
Date: 2006-06-02
Time limit: 1s-5s
Source limit:50000B
Languages: All

SPOJ Problem Set (lab1x)

1111. Suma

Problem code: SUMAN

Twoim zadaniem będzie obliczenie następującej sumy: 1+2+3+4+...+n. Zadania wygląda na proste ;)Ale chodzi w nim o napisaniu jak najkrótszego kodu, jest to zadanie typu challenge.

Wejście

Będziesz musiał rozwiązać dokładnie 10 przypadków testowych, w każdym z 10 wierszy znajduje sie dokładnie jedna liczba całkowita n (1<=n<=1000)

Wyjście

W każdym wierszu standardowego wyjścia powinna znaleźć się dokładnie jedna liczba, oznaczająca wartość sumy: 1+2+3+...+n.

Przykład

Wejście

```
1
4
(pozostałe 8 przypadków testowych)
```

Wyjście:

```
1
10
(pozostałe 8 przypadków testowych)
```

Added by: Marcin Sasinowski
Date: 2006-11-24
Time limit: 1s
Source limit:50000B
Languages: C99 strict
Resource: Potyczki algorytmiczne 2006

SPOJ Problem Set (challenge)

1369. Podstawówka

Problem code: PPODST

Bonifacy dumnie skonczył informatykę na uniwersytecie. Teraz pełen radości i entuzjazmu pracuje jako nauczyciel informatyki w Szkole Podstawowej im. S. Banacha w Bitowie Szestnastkowym. Właśnie skończył swój pierwszy semestr w nowej szkole i... napotkał pierwszy problem. Pewna polonistka poprosiła go o wyliczenie średniej ocen danej klasy na podstawie danych napisanych w OOWriterze. Jako, że Bonifacy niezbyt wie, jak się za to zabrać, zwrócił się do ciebie o pomoc.

Wejście

Na wejsciu znajduje sie w pierwszym wierszu liczba uczniów danej klasy, a następnie, w kolejnych wierszach imię i nazwisko ucznia, oraz srednia jaką osiągnął, oddzielone pojedyńczą spacją.

Wyjście

Na wyjściu ma się znajdować średnia ocen danej klasy

Przykład

Wejście:
```
4
Michał Zegarski 5.75
Arkadiusz Flegma 3
Monika Prawinska 4.85
Tomasz Patwa 4.24
```
Wyjście:
```
4.46
```

[gl&hf tr4m]

Added by: Damian Kaczmarek
Date: 2007-03-08
Time limit: 1s
Source limit:50000B
Languages: PERL

SPOJ Problem Set (challenge)

2644. Spotkania towarzyskie

Problem code: PLGCOL

Zgodnie z niepisaną tradycją Politechniki Bajtlandzkiej (PB) każdy ze studentów raz w roku powinien zorganizować Spotkanie Towarzyskie ku Uciesze Braci Studenckiej (STUBS). STUBS mają nieformalny charakter i cieszą się dużą popularnością. Organizowanie i uczestnictwo w STUBS pochłania wiele czasu i energii społeczności PB, gdyż przygotowanie udanego STUBS dla licznego grona przyjaciół i znajomych znakomicie podnosi prestiż towarzyski żaka.

Początkowo, gdy idea STUBS nie była jeszcze tak popularna, organizowano je całkowicie spontanicznie. W kolejnych latach okazało się, że konieczne są pewne uregulowania, powołano nawet Studencki Komitet Nadzoru nad STUBS (SKNS).

Obecnie SKNS przygotowuje harmonogram STUBS na kolejny rok akademicki. Nie jest to zadanie łatwe, gdyż należy wziąć pod uwagę szereg kryteriów. W tym miejscu należy się kilka słów wyjaśnienia. Otóż, każdy przedstawiciel Braci Studenckiej PB surowo przestrzega lojalności wobec swoich *ezer*. Nie wnikając w bliższe szczegóły rytuałów PB wystarczy powiedzieć, że jeśli student zorganizuje konkurencyjne STUBS w tym samym terminie co jej (jego) *ezer*, to jest to obraza i powód długotrwałych waśni (w przeszłości dochodziło nawet do zamieszek).

W fazie wstępnej planowania harmonogramu należy oszacować najmniejszą możliwą liczbę dopuszczalnych terminów aby wszystkie STUBS mogły się odbyć z poszanowaniem niepisanej tradycji. Zagadnienie zamodelowano w języku teorii grafów. Jeśli wierzchołki grafu (studenci PB) są połączone krawędzią (są swoimi *ezer*), to trzeba im przydzielić różne kolory (różne terminy). Liczba kolorów (im mniejsza, tym lepiej), użytych do legalnego pokolorowania grafu, da nam górne ograniczenie na minimalną liczbę terminów potrzebnych do zorganizowania wszystkich STUBS.

Jak wiadomo problem kolorowania grafów jest NP-trudny, a studentów PB jest sporo, więc prawdopodobnie nie uda się znaleźć optymalnego pokolorowania w rozsądnym czasie. W związku z tym SKNS ogłosił konkurs na najlepszy program implementujący algorytm przybliżony. Programy będą testowane na grafach odpowiadających systemowi rozgrywek stosowanemu w Lidze Fajnego Sportu Rozwojowego (narodowy sport Bajtlandii).

W tym miejscu znów należy się kilka słów wyjaśnienia. Otóż, tworzenie systemu rozgrywek LFSR jest dość skomplikowane i przebiega w dwóch etapach. W pierwszym etapie (i to jest istotne dla treści zadania) wykorzystuje się rejestr przesuwny ze sprzężeniem zwrotnym. Specjalna komisja ustala (wiążę się z tym skomplikowana procedura, której opis pominiemy) długość rejestru `l`, jego stan początkowy `seed` i wielomian charakterystyczny. Następnie wszystkim drużynom nadaje parami różne numery ze zbioru: {0, 1, ...,n-1} i wyznacza maksymalną liczbę spotkań `mmax` oraz dodatkowy parametr `k`. Kiedy wszystkie parametry zostaną ustalone, postępuje się w następujący sposób:

```
team1:=0;
team2:=0;
Powtarzaj  m_max razy:
  - do team1 dodaj shift(k) (modulo n);
  - do team2 dodaj shift(k) (modulo n);
  Jeśli team1 jest różne od  team2, to  drużyny o numerach
  team1 i team2 mają do rozegrania mecz.
```

Uwagi: - `shift(k)` oznacza liczbę powstałą z `k` bitów pobranych z rejestru, przy czym młodsze bity pobierane są najpierw; - może się zdarzyć, że para drużyn ma do rozegrania kilka meczów, nie ma to jednak wpływu na konstruowany graf.

Opis wejścia

Najpierw `t` - liczba grafów testowych.

Następnie opisy poszczególnych grafów:
`n` - liczba wierzchołków.
4 <= `l` <= 32 - długość rejestru
`seed` - stan początkowy rejestru (szesnastkowo)
struktura sprzężenia:
liczba `p` i `p` numerów niezerowych współczynników wielomianu charakterystycznego (taps) oraz liczby 1<= `k` <= 32 i `m_max`.

Opis wyjścia

Dla każdego testu W jednym wierszu liczba użytych kolorów `luc` i w następnym kolory kolejnych wierzchołków (liczby ze zbioru `{1, 2, ..., luc}`).

Punktacja

Dodatnia liczba punktów przyznawana jest wyłącznie za legalne pokolorowania. Jeśli oceniany algorytm użył do pokolorowania danego grafu `k` kolorów, a wzorcowa implementacja algorytmu SLF `k'` kolorów, to za pokolorowanie grafu przyznawane jest `(k'-2)/(k-2)` punktów. Wynik ten uśredniany jest po wszystkich grafach w teście, a następnie sumowany z wagami po wszystkich testach.

Uwaga: może się zdarzyć, że poprawny algorytm SLF da inną liczbę kolorów, niż SLF użyty w tym zadaniu jako wzorcowy.

Przydatne odnośniki

Rejestry przesuwne ze sprzężeniem zwrotnym: LFSR w Wikipedii i inny opis LFSR.

Kolorowanie grafów: Graph Coloring Page Josepha Culbersona, Vertex Coloring w encyklopedii MathWorld i książka Marka Kubale *Introduction to computational complexity and algorithmic graph coloring*.

Przykład

Wejście:

```
1
4 6 b 2 5 6 2 5
```

Interpretacja: Jeden graf, mający cztery wierzchołki, budowany w oparciu o rejestr długości 6 ze stanem początkowym 001011. Do sprzężenia zwrotnego brane są dwa bity: piąty i szósty.

Kolejno łączymy następujące wierzchołki (w nawiasie podano stan rejestru po wykonaniu operacji):

```
3       2       (111000)
3       0       (010011)
2       0       (101001)
3       2       (110110)
1       3       (110111)
```

W efekcie otrzymujemy graf (reprezentacja listowa):

```
0: 2, 3
1: 3
2: 0, 3
3: 0, 1, 2
```

Wyjście:

```
3
3 2 2 1
```

Interpretacja: Użyto trzech kolorów, wierzchołki 0, 1, 2 i 3 otrzymały odpowiednio kolory 3, 2, 2 i 1. Jest to pokolorowanie legalne (żadne dwa wierzchołki połączone krawędzią nie otrzymały tego samego koloru) i możliwe do otrzymania algorytmem SLF.

Added by: Łukasz Kuszner
Date: 2008-04-09
Time limit: 1s-10s
Source limit:50000B
Languages: All
Resource: Algorytmy i Struktury Danych, projekt 2008