MINISTERSTWO EDUKACJI NARODOWEJ
FUNDACJA ROZWOJU INFORMATYKI
KOMITET GŁÓWNY OLIMPIADY INFORMATYCZNEJ

XVIII OLIMPIADA INFORMATYCZNA

Drodzy Czytelnicy,

zapraszamy wszystkich do rozwiązywania informatycznych zadań olimpijskich, ale tych, którzy chcą wziąć udział w Olimpiadzie, zobowiązujemy do przeczytania „Zasad organizacji zawodów XVIII OI", które zawierają ważne wymagania dotyczące rozwiązań i nie tylko. „Zasady..." znajdują się na stronie OI: www.oi.edu.pl. Programy rozwiązujące przedstawione poniżej zadania należy nadsyłać przez stronę http://sio.mimuw.edu.pl, na którą skądinąd warto zaglądać, gdyż znajdują się tam różne ogłoszenia dotyczące przebiegu I etapu Olimpiady, wybrane odpowiedzi na pytania zawodników, narzędzia do sprawdzania rozwiązań pod względem formalnym i wysyłania rozwiązań oraz forum służące do wymiany doświadczeń między zawodnikami.

Zachęcam wszystkich interesujących się informatyką do uczestnictwa w Olimpiadzie. Nawet jeśli startując pierwszy raz, nie potrafimy rozwiązać wszystkich zadań, spróbujmy zrobić tylko niektóre z nich. Startując w Olimpiadzie, stajemy się natychmiast członkami prestiżowego klubu olimpijczyków-informatyków.

Termin nadsyłania rozwiązań przedstawionych tu zadań I etapu Olimpiady to **15 listopada 2010**.

Młodzi polscy olimpijczycy to ścisła czołówka światowa. Potwierdzają to chociażby wyniki laureatów ubiegłorocznej, XVII Olimpiady Informatycznej na zawodach międzynarodowych. W Olimpiadzie Informatycznej Krajów Bałtyckich, która odbyła się w maju w Estonii, nasi olimpijczycy zdobyli cztery medale: Paweł Lipski — złoto, Łukasz Jocz i Krzysztof Leszczyński — medale srebrne, a Stanisław Barzowski — medal brązowy. Ten znakomity wynik osiągnęli uczniowie młodsi, którzy w tym roku mają szansę wystartować raz jeszcze w Olimpiadzie. Nasza pierwsza reprezentacja wspaniale spisała się podczas Olimpiady Informatycznej Krajów Europy Środkowej (Słowacja, lipiec) oraz podczas Międzynarodowej Olimpiady Informatycznej (Kanada, sierpień). Zwycięzczynią tegorocznej Olimpiady Informatycznej Krajów Europy Środkowej została Anna Piekarska. Jan Kanty Milczek zdobył srebrny medal, a Adrian Jaskółka i Igor Adamski medale brązowe. W Kanadzie najlepiej z naszych reprezentantów wypadł Adrian Jaskółka, zajmując trzecie miejsce w świecie i zdobywając złoty medal, medale srebrne zdobyli Anna Piekarska i Jan Kanty Milczek, a brąz przypadł w udziale Igorowi Adamskiemu. Te wszystkie sukcesy potwierdzają to, że *Informatyka jest specjalnością młodych Polaków.*

Korzystając z serwisu Olimpiady i portalu main.edu.pl, mamy szansę uczyć się od najlepszych i doskonalić swoje umiejętności tak, żeby w przyszłości osiągać olimpijskie sukcesy. Awans do finału Olimpiady to możliwość wyboru najlepszych uczelni w Polsce do studiowania informatyki. To, w dobie kierunków zamawianych, wysokie stypendia od początku studiów. Sukcesy olimpijskie są też brane pod uwagę przez liczne firmy podczas rekrutacji na atrakcyjne staże wakacyjne.

Na koniec chciałbym podziękować wypróbowanym przyjaciołom Olimpiady: Fundacji Rozwoju Informatyki — formalnemu organizatorowi, współorganizatorowi — firmie Asseco Poland SA, firmie Combidata Poland, Ogólnopolskiej Fundacji Edukacji Komputerowej, wydawnictwom — PWN i WNT.

prof. dr hab. Krzysztof Diks

Zadania

KONSPIRACJA

plik źródłowy kon.*, dostępna pamięć 128 MB

Wroga Bitocja napadła zdradziecko na Bajtocję i okupuje znaczną jej część. Król Bajtocji, Bajtazar, chce zorganizować na okupowanych terenach ruch oporu. Bajtazar rozpoczął od wytypowania osób, które utworzą strukturę organizacyjną ruchu oporu. Należy je podzielić na dwie grupy: **konspiratorów**, którzy będą prowadzić bezpośrednią działalność na terenie okupowanym, oraz **grupę wsparcia**, która będzie działać z terytorium wolnej Bajtocji.

Pojawił się jednak pewien problem — podział taki musi spełniać następujące warunki:

- Każde dwie osoby z grupy wsparcia powinny się znać — zagwarantuje to, że będą stanowiły zwartą i zgraną grupę.
- Konspiratorzy nie powinni się znać nawzajem.
- Żadna z grup nie może być pusta, tzn. musi być przynajmniej jeden konspirator i jedna osoba w grupie wsparcia.

Bajtazar zastanawia się, ile jest różnych sposobów podziału wytypowanych osób na dwie grupy zgodnie z powyższymi warunkami, a przede wszystkim, czy taki podział jest w ogóle możliwy. Jako że sam nie całkiem umie sobie z tym poradzić, poprosił Cię o pomoc.

Wejście

W pierwszym wierszu standardowego wejścia znajduje się jedna liczba całkowita n ($2 \leqslant n \leqslant 5\,000$), oznaczająca liczbę osób zaangażowanych w organizację ruchu oporu. Osoby te są ponumerowane od 1 do n. W kolejnych n wierszach opisane jest, które osoby się znają. W i-tym z tych wierszy znajduje się opis znajomych osoby nr i, złożony z liczb całkowitych pooddzielanych pojedynczymi odstępami. Pierwsza z tych liczb, k_i ($0 \leqslant k_i \leqslant n-1$), oznacza liczbę znajomych osoby nr i. Dalej w wierszu znajduje się k_i liczb całkowitych $a_{i,1}, a_{i,2}, \ldots, a_{i,k_i}$. Liczby $a_{i,j}$ spełniają $1 \leqslant a_{i,j} \leqslant n$, $a_{i,j} \neq i$ oraz są podane w kolejności rosnącej. Możesz założyć, że jeśli w ciągu liczb a_i występuje liczba x, to także w ciągu liczb a_x występuje liczba i.

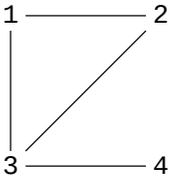
Wyjście

W pierwszym i jedynym wierszu standardowego wyjścia Twój program powinien wypisać jedną liczbę całkowitą: liczbę sposobów, na które osoby mające utworzyć ruch oporu mogą zostać podzielone na grupę wsparcia i grupę działającą na terenach okupowanych. Jeżeli nie istnieje żaden podział wytypowanych osób na dwie grupy spełniający podane warunki, wówczas poprawnym wynikiem jest 0.

Przykład

Dla danych wejściowych:

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



poprawnym wynikiem jest:

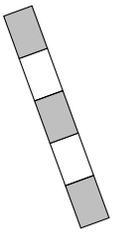
```
3
```

Wyjaśnienie do przykładu: Dla tej grupy spiskowców są możliwe trzy podziały na grupy. Grupę konspiratorów mogą stanowić uczestnicy o numerach 1 i 4, o numerach 2 i 4 lub sam uczestnik o numerze 4.

LIZAK

plik źródłowy liz.*, dostępna pamięć 64 MB

Bajtazar prowadzi w Bajtogrodzie sklep ze słodyczami. Wśród okolicznych dzieci najpopularniejszymi słodyczami są lizaki waniliowo-truskawkowe. Składają się one z wielu segmentów jednakowej długości, z których każdy ma jeden smak — waniliowy lub truskawkowy. Cena lizaka jest równa sumie wartości jego segmentów; segment waniliowy kosztuje jednego bajtalara, a truskawkowy dwa bajtalary.



Rys. 1: Przykładowy lizak o pięciu segmentach, trzech truskawkowych i dwóch waniliowych, ułożonych na przemian. Cena tego lizaka wynosi 8 bajtalarów.

Obecnie Bajtazarowi został na składzie tylko jeden (za to być może bardzo długi) lizak. Sprzedawca zdaje sobie sprawę, że być może nikt nie będzie chciał go kupić w całości, dlatego dopuszcza możliwość łamania go na granicach segmentów w celu uzyskania lizaka o mniejszej długości. Fragment lizaka przeznaczony ostatecznie do sprzedaży musi pozostać niepołamany.

Doświadczenie pokazuje, że klienci najczęściej chcą kupić lizaka za całe swoje kieszonkowe. Bajtazar zastanawia się, dla wielu możliwych wartości k, jak przełamać posiadany lizak tak, aby otrzymać lizak o cenie równej dokładnie k bajtalarów. Ponieważ zadanie nie jest wcale proste, poprosił Cię o pomoc.

Wejście

W pierwszym wierszu standardowego wejścia znajdują się dwie liczby całkowite n oraz m ($1 \leqslant n, m \leqslant 1\,000\,000$) oddzielone pojedynczym odstępem. Oznaczają one odpowiednio liczbę segmentów ostatniego pozostałego w sklepie lizaka oraz liczbę rozpatrywanych wartości k. Segmenty lizaka są ponumerowane kolejno od 1 do n. W drugim wierszu znajduje się n-literowy opis lizaka, złożony z liter `T` i `W`, przy czym `T` oznacza segment truskawkowy, zaś `W` — waniliowy; i-ta z tych liter opisuje smak i-tego segmentu. W kolejnych m wierszach znajdują się kolejne wartości k do rozpatrzenia ($1 \leqslant k \leqslant 2\,000\,000$), po jednej w wierszu.

Wyjście

Twój program powinien wypisać na standardowe wyjście dokładnie m wierszy zawierających wyniki dla kolejnych wartości k, po jednym wyniku w wierszu. Jeśli dla danej wartości k nie da się wyłamać z lizaka spójnego fragmentu o wartości równej k bajtalarów, należy wypisać słowo `NIE`. W przeciwnym przypadku należy wypisać dwie liczby l oraz r ($1 \leqslant l \leqslant r \leqslant n$) oddzielone pojedynczym odstępem, takie że fragment lizaka złożony z segmentów o numerach od l do r włącznie ma wartość dokładnie k bajtalarów. Jeśli istnieje wiele możliwych odpowiedzi, Twój program może podać dowolną z nich.

Przykład

Dla danych wejściowych:	poprawnym wynikiem jest:

`5 3`
`TWTWT`
`5`
`1`
`7`
	`1 3`
	`2 2`
	`NIE`

Wyjaśnienie do przykładu: Przykład opisuje lizak z rys. 1. Segmenty o numerach od 1 do 3 tworzą lizak postaci `TWT`, wart 5 bajtalarów. Segment numer 2 ma smak waniliowy i kosztuje 1 bajtalara. Z tego lizaka nie da się w żaden sposób uzyskać lizaka wartego 7 bajtalarów.

PIORUNOCHRON

plik źródłowy pio.*, dostępna pamięć 128 MB

Postępujące zmiany klimatu zmusiły władze Bajtogrodu do wybudowania dużego piorunochronu, który chroniłby wszystkie budynki w mieście. Wszystkie budynki stoją w rzędzie przy jednej prostej ulicy i są ponumerowane kolejno od 1 do n.

Zarówno wysokości budynków, jak i wysokość piorunochronu wyrażają się nieujemnymi liczbami całkowitymi. Bajtogród dysponuje funduszami na wybudowanie tylko jednego piorunochronu. Co więcej, im wyższy ma być piorunochron, tym będzie droższy.

Aby piorunochron o wysokości p umieszczony na dachu budynku i (o wysokości h_i) mógł skutecznie chronić wszystkie budynki, dla każdego innego budynku j (o wysokości h_j) musi zachodzić następująca nierówność:

$$h_j \leqslant h_i + p - \sqrt{|i-j|}.$$

Tutaj $|i-j|$ oznacza wartość bezwzględną różnicy liczb i oraz j.

Bajtazar, burmistrz Bajtogrodu, poprosił Cię o pomoc. Napisz program, który dla każdego budynku i obliczy, jaka jest minimalna wysokość piorunochronu, który umieszczony na budynku i będzie chronił wszystkie budynki.

Wejście

W pierwszym wierszu standardowego wejścia znajduje się jedna liczba całkowita n ($1 \leqslant n \leqslant 500\,000$) oznaczająca liczbę budynków w Bajtogrodzie. W każdym z kolejnych n wierszy znajduje się jedna liczba całkowita h_i ($0 \leqslant h_i \leqslant 1\,000\,000\,000$), oznaczająca wysokość i-tego budynku.

Wyjście

Twój program powinien wypisać na standardowe wyjście n wierszy. W i-tym wierszu powinna znaleźć się nieujemna liczba całkowita p_i, oznaczająca minimalną wysokość piorunochronu na i-tym budynku.

Przykład

Dla danych wejściowych:	poprawnym wynikiem jest:

`6`
`5`
`3`
`2`
`4`
`2`
`4`
	`2`
	`3`
	`5`
	`3`
	`5`
	`4`

PRZEKŁADANKA

plik źródłowy prz.*, dostępna pamięć 64 MB

Bajtazar kupił synkowi Bajtkowi zestaw klocków ponumerowanych od 1 do n i ustawił je w rzędzie w pewnej kolejności. Zadaniem Bajtka jest ustawienie klocków w kolejności numerów tych klocków, od najmniejszego do największego. Jedyne ruchy, jakie może wykonywać Bajtek, to:

- przełożenie ostatniego klocka na początek (ruch typu `a`), oraz
- przełożenie trzeciego klocka na początek (ruch typu `b`).

Pomóż Bajtkowi i napisz program, który sprawdzi, czy dany układ klocków da się w ogóle ustawić w żądanej kolejności, a jeżeli tak, to poda, jak to zrobić.

Wejście

W pierwszym wierszu standardowego wejścia znajduje się jedna liczba całkowita n, $1 \leqslant n \leqslant 2\,000$. W drugim wierszu znajduje się n liczb całkowitych z zakresu od 1 do n pooddzielanych pojedynczymi odstępami. Liczby te nie powtarzają się i reprezentują początkowe ustawienie klocków.

Wyjście

Jeśli nie istnieje sekwencja ruchów prowadząca do ustawienia klocków w porządku rosnącym numerów, Twój program powinien wypisać na standardowe wyjście „`NIE DA SIE`" (bez cudzysłowów).

W przeciwnym przypadku, w pierwszym wierszu wyjścia powinna znaleźć się jedna liczba całkowita m ($m \leqslant n^2$), oznaczająca liczbę wykonywanych operacji. Przez **operację** rozumiemy k-krotne wykonanie jednego z ruchów `a` lub `b`.

Jeżeli $m > 0$, to w drugim wierszu powinien znaleźć się ciąg m liczb całkowitych z dodanymi pojedynczymi znakami `a` lub `b`. Zapis postaci k`a` (dla $0 < k < n$) oznacza k-krotne wykonanie ruchu `a`. Zapis postaci k`b` (dla $0 < k < n$) oznacza k-krotne wykonanie ruchu `b`.

Dodatkowo, znaki stowarzyszone z liczbami znajdującymi się w drugim wierszu muszą być ułożone na przemian.

Jeśli istnieje więcej niż jedno rozwiązanie, Twój program może wypisać dowolne z nich.

Przykład

Dla danych wejściowych:	poprawnym wynikiem jest:
`4` `1 3 2 4`	`4` `3a 2b 2a 2b`
natomiast dla danych:	poprawnym wynikiem jest:
`7` `1 3 2 4 5 6 7`	`NIE DA SIE`
a dla danych:	poprawnym wynikiem jest:
`3` `1 2 3`	`0`

WYKRES

plik źródłowy wyk.*, dostępna pamięć 64 MB

Wykresem nazywamy dowolny ciąg punktów na płaszczyźnie. Dany wykres $(P_1, \ldots, P_n)$ zamierzamy zastąpić innym wykresem, który będzie miał co najwyżej m punktów ($m \leqslant n$), ale w taki sposób, aby „przypominał" jak najbardziej oryginalny wykres.

Nowy wykres tworzymy w ten sposób, że dzielimy ciąg punktów $P_1, \ldots, P_n$ na s ($s \leqslant m$) spójnych podciągów:

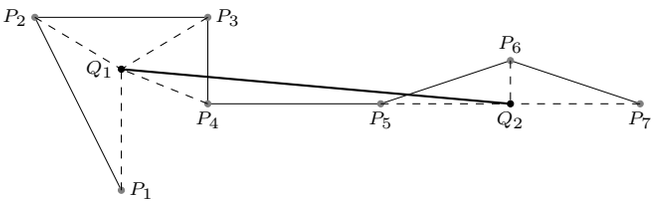
$$(P_{k_0+1}, \ldots, P_{k_1}), \quad (P_{k_1+1}, \ldots, P_{k_2}), \quad \ldots, \quad (P_{k_{s-1}+1}, \ldots, P_{k_s}),$$

przy czym $0 = k_0 < k_1 < k_2 < \ldots < k_s = n$, a następnie każdy podciąg $(P_{k_{i-1}+1}, \ldots, P_{k_i})$, dla $i = 1, \ldots, s$, zastępujemy jednym nowym punktem Q_i. Mówimy wtedy, że każdy z punktów $P_{k_{i-1}+1}, \ldots, P_{k_i}$ został **ściągnięty** do punktu Q_i. W rezultacie otrzymujemy nowy wykres reprezentowany przez ciąg $Q_1, \ldots, Q_s$. Miarą podobieństwa tak utworzonego wykresu do oryginalnego jest maksimum odległości każdego z punktów $P_1, \ldots, P_n$ do punktu, do którego został on ściągnięty:

$$\max_{i=1,\ldots,s} \left(\max_{j=k_{i-1}+1,\ldots,k_i} \big(d(P_j, Q_i)\big) \right),$$

przy czym $d(P_j, Q_i)$ jest odległością między P_j i Q_i i wyraża się standardowym wzorem:

$$d\big((x_1, y_1), (x_2, y_2)\big) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}.$$



Rys. 1: Przykładowy wykres $(P_1, \ldots, P_7)$ i nowy wykres (Q_1, Q_2), gdzie $(P_1, \ldots, P_4)$ ściągamy do Q_1, natomiast (P_5, P_6, P_7) do Q_2.

Dla danego wykresu składającego się z n punktów należy znaleźć wykres najbardziej go przypominający, który zawiera co najwyżej m punktów, przy czym podział wykresu na spójne podciągi jest dowolny. Ze względu na skończoną precyzję obliczeń, za poprawne będą uznawane wyniki, których podobieństwo do danego wykresu jest co najwyżej o 0.000001 większe od optymalnego wyniku.

Wejście

W pierwszym wierszu standardowego wejścia znajdują się dwie liczby całkowite n oraz m oddzielone pojedynczym odstępem, $1 \leqslant m \leqslant n \leqslant 100\,000$. Każdy z następnych n wierszy zawiera po dwie liczby całkowite oddzielone pojedynczym odstępem. W $(i+1)$-szym wierszu znajdują się liczby x_i, y_i, $-1\,000\,000 \leqslant x_i, y_i \leqslant 1\,000\,000$, reprezentujące współrzędne (x_i, y_i) punktu P_i.

Wyjście

W pierwszym wierszu standardowego wyjścia należy wypisać jedną liczbę rzeczywistą d, będącą miarą podobieństwa znalezionego wykresu do wykresu oryginalnego. W drugim wierszu standardowego wyjścia należy wypisać jedną liczbę całkowitą s, $1 \leqslant s \leqslant m$. Następnie, w kolejnych s wierszach powinny zostać wypisane współrzędne punktów $Q_1, \ldots, Q_s$, po jednym punkcie w wierszu. W $(i+2)$-gim wierszu powinny znaleźć się liczby rzeczywiste u_i i v_i oddzielone pojedynczym odstępem i określające współrzędne (u_i, v_i) punktu Q_i. Wszystkie liczby rzeczywiste na wyjściu należy wypisać z rozwinięciem do co najwyżej 15 cyfr po kropce dziesiętnej.

Przykład

Dla danych wejściowych:	poprawnym wynikiem jest:

`7 2`
`2 0`
`0 4`
`4 4`
`4 2`
`8 2`
`11 3`
`14 2`
	`3.00000000`
	`2`
	`2.00000000 1.76393202`
	`11.00000000 1.99998199`

USTALENIA TECHNICZNE

Polecenia używane do kompilacji rozwiązań (np. zadania `abc`):

Dla `c`	`gcc -O2 -static abc.c -lm`
Dla `cpp`	`g++ -O2 -static abc.cpp -lm`
Dla `pas`	`ppc386 -O2 -XS -Xt abc.pas`

Do kompilowania rozwiązań używane będą następujące kompilatory:

- Pascal — Free Pascal Compiler 2.2.2
- C/C++ — GCC 4.4.1

Ograniczenia:

- kod źródłowy rozwiązania nie powinien przekraczać 100 KB, a kod wykonywalny 5 MB,
- czas kompilacji rozwiązania nie powinien przekraczać 30 s,
- wielkość pamięci operacyjnej dostępnej programom jest podana w treściach zadań (wartość ta dotyczy sumarycznego zapotrzebowania na pamięć, a więc zawiera m.in. rozmiar kodu wykonywalnego, stosu, sterty itp.).

Rozwiązania powinny:

- składać się z jednego pliku źródłowego o nazwie podanej w treści zadania,
- czytać dane ze standardowego wejścia, zapisywać wynik na standardowe wyjście, chyba że dla danego zadania wyraźnie napisano inaczej,
- kończyć działanie kodem wyjścia 0 (inne kody wyjścia uznawane są za błąd wykonania).

Rozwiązania nie mogą:

- otwierać jakichkolwiek plików,
- tworzyć nowych procesów,
- korzystać z funkcji sieciowych,
- korzystać z zewnętrznych bibliotek (np. crt, graph),
- uruchamiać innych programów.

Więcej informacji (w tym przykłady) można znaleźć w witrynie internetowej Olimpiady.

Wskazówki dla uczestników

1. Aby Twoje rozwiązanie mogło zostać właściwie ocenione, zastosuj się do ustaleń zawartych w „Zasadach organizacji zawodów" i treściach zadań.
2. W Systemie Internetowym Olimpiady sio.mimuw.edu.pl znajdują się odpowiedzi na pytania zawodników dotyczące Olimpiady. Ponieważ odpowiedzi mogą zawierać ważne informacje dotyczące toczących się zawodów, polecamy Ci regularnie zapoznawać się z ukazującymi się odpowiedziami. Dalsze pytania należy przysyłać poprzez System Internetowy Olimpiady.
3. Przestrzegaj dokładnie warunków określonych w treści zadania.
4. Należy założyć, że dane testowe są bezbłędne, zgodne z warunkami zadania i podaną specyfikacją wejścia. Twój program nie musi tego sprawdzać.
5. Nie przyjmuj żadnych założeń, które nie wynikają z treści zadania.
6. Staraj się dobrać taką metodę rozwiązania zadania, która jest nie tylko poprawna, ale daje wyniki w jak najkrótszym czasie.
7. Ocena za rozwiązanie zadania jest określana na podstawie wyników testowania programu i uwzględnia poprawność oraz efektywność metody rozwiązania użytej w programie. W szczególności programy poprawne, lecz działające zbyt długo — zwłaszcza dla dużych rozmiarów danych — mogą zostać ocenione nisko.
8. Koniecznie zapoznaj się z materiałami dostępnymi w witrynie Olimpiady: www.oi.edu.pl.
9. **Nie udostępniaj innym swoich rozwiązań zadań przed upływem godziny zakończenia zawodów. Chroń je przed niepowołanym dostępem. Jeśli ktoś wyśle skradzione Tobie lub udostępnione przez Ciebie rozwiązania zadań, możesz zostać zdyskwalifikowany.**
10. Z uwagi na zwiększenie efektywności strumieni w kompilatorze G++ od wersji 3.4, istnieje możliwość ich używania w rozwiązaniach zadań, bez obawy przekroczenia limitu czasu na operacjach wejścia/wyjścia. W tym celu, należy na samym początku programów używających strumieni wyłączać synchronizację wejścia/wyjścia przy użyciu `ios_base::sync_with_stdio(0);`

Przewodniczący Komitetu Głównego
Olimpiady Informatycznej

prof. dr hab. Krzysztof Diks