**Szymon Acedański, Jakub Radoszewski** — Treść zadania, Opracowanie
**Tomasz Kulczyński** — Program

**Dostępna pamięć: 64 MB.** *OI, Etap III, dzień drugi, 15.04.2010*

# Mosty

*San Bajcisko to pięknie położone nadmorskie miasto. Składa się ono z n niewielkich, ale gęsto zaludnionych wysp, ponumerowanych od 1 do n. Pewne pary wysp są połączone mostami, którymi poprowadzono dwukierunkowe drogi. Każda para wysp może być połączona co najwyżej jednym mostem. Z każdej wyspy San Bajciska można dotrzeć do dowolnej innej, przemieszczając się pomiędzy wyspami wyłącznie za pomocą mostów.*

*Bajtazar i Bajtek wybierają się na przejażdżkę rowerową po San Bajcisku. Swoją wycieczkę zaczną na wyspie nr 1. Chcą zwiedzić miasto, przejeżdżając każdym mostem dokładnie raz, i zakończyć podróż na wyspie nr 1. Niestety, w wycieczce będzie im przeszkadzał... wiatr. Otóż na każdym moście strasznie wieje, co* **w różnym stopniu** *może utrudniać przejazd rowerowy mostem* **w jedną bądź w drugą stronę**. *Dla uproszczenia zakładamy, że przy ustalonym kierunku przejazdu przez dany most siła wiatru jest stała.*

*Pomóż Bajtazarowi i Bajtkowi znaleźć taką trasę spełniającą ich wymagania, po przejechaniu której będą najmniej zmęczeni. Jako miarę tego, na ile trasa jest męcząca, przyjmujemy maksymalną siłę wiatru, z jaką nasi bohaterowi będą musieli zmagać się na trasie.*

## Wejście

*W pierwszym wierszu standardowego wejścia znajdują się dwie liczby całkowite oddzielone pojedynczym odstępem: n oraz m ($2 \leqslant n \leqslant 1\,000$, $1 \leqslant m \leqslant 2\,000$), oznaczające odpowiednio liczbę wysp oraz liczbę mostów w San Bajcisku. Wyspy są ponumerowane od 1 do n, a mosty od 1 do m. W kolejnych m wierszach znajdują się opisy mostów. $(i+1)$-szy wiersz zawiera cztery liczby całkowite $a_i$, $b_i$, $l_i$, $p_i$ ($1 \leqslant a_i, b_i \leqslant n$, $a_i \neq b_i$, $1 \leqslant l_i, p_i \leqslant 1\,000$), pooddzielane pojedynczymi odstępami. Liczby te opisują most nr i łączący wyspy o numerach $a_i$ oraz $b_i$. Siła wiatru, jaki przeszkadza w trakcie przejazdu rowerem z wyspy $a_i$ do $b_i$, to $l_i$; jeżeli zaś przejechać tym samym mostem z wyspy $b_i$ do $a_i$, to przeszkadzający wiatr będzie miał siłę $p_i$.*

## Wyjście

*Jeżeli nie istnieje żadna trasa spełniająca wymagania naszych bohaterów, to pierwszy i jedyny wiersz standardowego wyjścia powinien zawierać jedno słowo* `NIE`. *W przeciwnym przypadku wyjście powinno składać się z dwóch wierszy, opisujących najmniej męczącą trasę wycieczki po San Bajcisku. Pierwszy z nich powinien zawierać jedną liczbę całkowitą: największą siłę wiatru, jaki będzie wiał w trakcie wycieczki. Właśnie tę liczbę należy zminimalizować. W drugim wierszu powinno znaleźć się m liczb całkowitych, pooddzielanych pojedynczymi odstępami i oznaczających numery mostów, którymi kolejno przejeżdża się na najmniej męczącej trasie.*

*Jeśli istnieje więcej niż jedna poprawna odpowiedź, Twój program powinien wypisać dowolną z nich.*

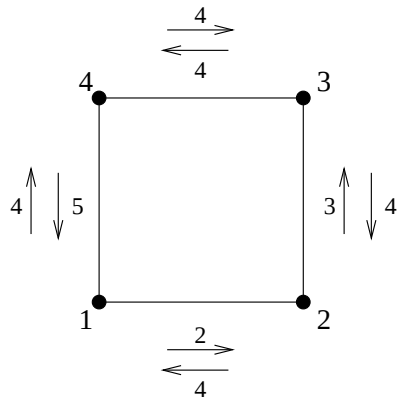
## Przykład

*Dla danych wejściowych:*

```
4 4
1 2 2 4
2 3 3 4
3 4 4 4
4 1 5 4
```

*poprawnym wynikiem jest:*

```
4
4 3 2 1
```




**Wyjaśnienie do przykładu:** *Optymalna trasa Bajtazara i Bajtka ma postać:* $1 \xrightarrow{4} 4 \xrightarrow{4} 3 \xrightarrow{4} 2 \xrightarrow{4} 1$. *Maksymalna siła wiatru na tej trasie wynosi 4.*

# Rozwiązanie

## Wprowadzenie

Z każdym rokiem Olimpiady Informatycznej dowiadujemy się coraz to nowych rzeczy o Bajtocji, Bajtazarze i jego przyjaciołach. Z niecierpliwością czekamy na czasy, w których już w szkołach podstawowych będzie się recytować na lekcjach ogólnie znane fakty z geografii fizycznej, gospodarczej, a teraz szczególnie z klimatologii Bajtocji.

Z historii wiemy, że za czasów dobrobytu sieć połączeń drogowych w Bajtocji była grafem gęstym, później nieraz starano się zredukować ją do drzewa, a lepiej nie myśleć, co stało się w związku z niedawnym kryzysem na światowych rynkach. Często podawano, że Bajtocja jest położona na wyspie w kształcie wielokąta wypukłego, niejednokrotnie wspominano też coś o estakadach, tunelach i mostach. Analizowany tekst źródłowy umacnia nas w przekonaniu o istnieniu mostów, co więcej, zdradza kolejne tajemnice: że Bajtocja składa się z wielu wysp (!), które w samym San Bajcisku tworzą wraz z mostami graf spójny[1] $G = (V_G, E_G)$, o $n$ wierzchołkach i $m$ krawędziach. Co więcej, Bajtazar postanowił każdej z krawędzi $e_i$ przypisać dwie wagi: $l_i$, $p_i$, oznaczające wysiłek potrzebny do pokonania jej rowerem w każdą ze stron. Wraz z Bajtkiem poszukuje on takiego cyklu Eulera[2], żeby maksymalna waga krawędzi napotkanej na tym cyklu, w kierunku przechodzenia po tym cyklu, była możliwie najmniejsza.

[1] Mówimy, że graf nieskierowany jest *spójny*, jeżeli z każdego wierzchołka da się dojść do każdego innego.

[2] *Cykl Eulera* to cykl przechodzący przez każdą krawędź grafu dokładnie raz (ta definicja obowiązuje niezależnie od tego, czy graf jest skierowany, czy nie).

Co prawda z tekstu nie wiemy na 100%, czy San Bajcisko w ogóle jest miastem, ani tym bardziej czy leży w Bajtocji, ale czymże innym mogłoby być!?

## Zarys algorytmu wzorcowego

Zacznijmy od przytoczenia klasycznego kryterium istnienia cyklu Eulera w grafach nieskierowanych. Dodajmy na wstępie, iż w całym opracowaniu będziemy zakładać, że wszystkie rozważane grafy nie zawierają wierzchołków izolowanych. Ich usunięcie z grafu nie wpływa bowiem na istnienie cykli Eulera, a ułatwia zwięzłe formułowanie twierdzeń.

**Twierdzenie 1 (Eulera dla grafów nieskierowanych).** *Graf nieskierowany jest* eulerowski *(tzn. ma cykl Eulera) wtedy i tylko wtedy, gdy jest spójny oraz każdy jego wierzchołek ma parzysty stopień*[3].

A zatem, jeśli wejściowy graf nie spełnia tego kryterium (przypomnijmy, że warunku spójności sprawdzać w zadaniu nie trzeba), możemy od razu być pewni, że za żadne skarby Bajtazar nie przejedzie po San Bajcisku tak, jak to sobie wymarzył. Co jeśli jednak cykl Eulera istnieje? Jak wybrać ten poszukiwany przez Bajtazara? Nazwijmy *k-cyklem* każdy cykl Eulera, w którym na wszystkich mostach wieje nie bardziej niż $k$ (w kierunku przechodzenia po cyklu). Zadanie polega na znalezieniu najmniejszego $k$ takiego, że w danym grafie istnieje $k$-cykl.

**Obserwacja 1.** Załóżmy, że w treści pewnego zadania znajduje się jedno z określeń: „znajdź minimum", „maksimum", „najmniejszą" bądź „największą wartość". Wówczas szansa na rozwiązanie tego zadania nie maleje, jeśli w głowie rozwiązującego pojawi się pomysł zastosowania techniki wyszukiwania binarnego po wyniku.

Technika ta była już kilkukrotnie opisywana w książeczkach Olimpiady. Polega w tym przypadku na znalezieniu w ciągu liczb $1, 2, \ldots, 1\,000$ szukanej najmniejszej liczby $k$ przy użyciu wyszukiwania binarnego. Potrzebujemy więc wykonać $O(\log w)$ zapytań o to, czy w grafie istnieje $k$-cykl, przy czym $w$ jest maksymalną wagą krawędzi na wejściu.

Gdy już znajdziemy najmniejsze $k$ oraz będziemy wiedzieć, w którą stronę należy przejechać każdym z mostów (o tym w kolejnym punkcie), pozostanie wyznaczyć cykl Eulera w grafie skierowanym. W tym miejscu pozwolimy sobie polecić książkę [26], w której można znaleźć więcej informacji na temat cykli Eulera, jak również algorytm ich znajdowania[4] o złożoności czasowej liniowej względem liczby wierzchołków i krawędzi grafu.

## Wykonywanie zapytań o $k$-cykle

Sprowadziliśmy nasze zadanie do problemu ustalenia, czy w grafie istnieje $k$-cykl dla danego $k$. Zacznijmy od wyeliminowania prostego przypadku:

[3] *Stopniem* wierzchołka $v$ w grafie nieskierowanym nazywamy liczbę krawędzi incydentnych z tym wierzchołkiem (oznaczenie: $deg(v)$).

[4] Można o tym także posłuchać na stronie `http://was.zaa.mimuw.edu.pl/?q=node/31`

**Obserwacja 2.** Jeśli $k$ jest mniejsze niż zarówno $l_i$, jak i $p_i$ dla pewnego $i$, to w grafie nie istnieje $k$-cykl.

Jeśli ta obserwacja nie wydaje się Czytelnikowi oczywista, prosimy o ponowne jej przeczytanie, z niniejszym zdaniem włącznie. Skorzystanie z tego spostrzeżenia wyeliminuje nam zbyt małe wartości $k$, ale nie pozwoli na udzielenie odpowiedzi na zapytanie w każdym przypadku. Rozważmy więc dla danego $k$ częściowo skierowany graf $G_k$, który otrzymujemy z wyjściowego grafu $G$, skierowując te krawędzie, przez które można przejechać tylko w jednym kierunku, bo w przeciwnym wieje mocniej niż $k$. Na rys. 1 po lewej stronie przedstawiono graf $G_4$ dla przykładowych danych wejściowych z treści zadania. Zauważmy, że w oryginalnym grafie $G$ istnieje $k$-cykl wtedy i tylko wtedy, gdy da się skierować wszystkie pozostałe krawędzie grafu $G_k$, tak aby otrzymany graf skierowany był eulerowski. Intuicyjnie, w $G_k$ odrzuciliśmy tylko takie kierunki przechodzenia wzdłuż krawędzi, które nie mogą wystąpić w żadnym $k$-cyklu.

**Definicja 1.** *Orientacją* grafu częściowo skierowanego $H$ nazwijmy dowolny graf skierowany, który możemy otrzymać, skierowując nieskierowane krawędzie grafu $H$.

Odtąd będziemy zastanawiali się już tylko nad tym, jak znaleźć eulerowską orientację $G'_k$ grafu $G_k$. Chcemy zatem, aby graf $G'_k$ spełniał następujące, również klasyczne kryterium.

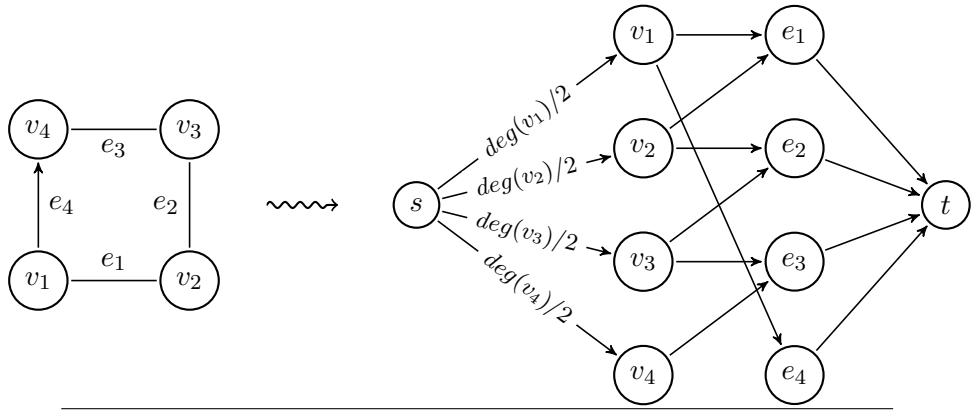
**Twierdzenie 2 (Eulera dla grafów skierowanych).** *Graf skierowany jest eulerowski wtedy i tylko wtedy, gdy jest silnie spójny*[5] *oraz każdy jego wierzchołek ma stopień wchodzący równy stopniowi wychodzącemu*[6].

W powyższym kryterium pojawiają się dwa warunki, analogiczne do tych z kryterium dla grafów nieskierowanych (Twierdzenie 1). Przypomnijmy jednak, że w oryginalnym grafie warunek spójności mieliśmy zagwarantowany w treści zadania. Na pierwszy rzut oka nie widać, czy pomaga to w jakiś sposób w kwestii silnej spójności wersji skierowanej grafu. Okazuje się jednak, że pomaga, a nawet gwarantuje tę silną spójność, o czym świadczy poniższa obserwacja. W jej dowodzie posługujemy się pojęciem silnie spójnych składowych (patrz np. książka [21]), którego znajomość na pewno ułatwia jego lekturę, ale można spróbować przeczytać ten dowód także bez tej wiedzy.

**Obserwacja 3.** Załóżmy, że $G$ jest nieskierowanym grafem eulerowskim, a $G'$ jest jego orientacją, w której każdy wierzchołek ma stopień wchodzący równy stopniowi wychodzącemu. Wówczas $G'$ jest silnie spójny, a więc, w szczególności (patrz Twierdzenie 2), $G'$ jest eulerowski.

[5] Mówimy, że graf skierowany jest *silnie spójny*, jeżeli z każdego wierzchołka da się dojść do każdego innego.

[6] *Stopniem wejściowym* (*wyjściowym*) wierzchołka $v$ w grafie skierowanym nazywamy liczbę krawędzi wchodzących do tego wierzchołka (odpowiednio, wychodzących z niego). Oznaczenia: $indeg(v)$ — stopień wejściowy, $outdeg(v)$ — stopień wyjściowy.




Rys. 1: Przykład konstrukcji sieci przepływowej z grafu częściowo skierowanego. Krawędzie nieoznaczone mają przepustowość jednostkową.

**Dowód:** Załóżmy przez sprzeczność, że $G'$ nie jest silnie spójny. Podzielmy graf $G'$ na tzw. *silnie spójne składowe* — są to maksymalne podgrafy grafu, które są silnie spójne. Silnie spójne składowe każdego grafu są, oczywiście, rozłączne wierzchołkowo. Wiadomo, że w podziale muszą wystąpić co najmniej dwie takie składowe, gdyż w przeciwnym przypadku $G'$ byłby silnie spójny.

Zauważmy, że w $G'$ istnieje jakaś krawędź $(u, v)$ łącząca dwa wierzchołki z różnych silnie spójnych składowych. Faktycznie, gdyby żadna taka krawędź nie istniała, to oryginalny graf $G$ nie byłby spójny.

Skonstruujmy w grafie $G'$ maksymalną, czyli nieprzedłużalną, ścieżkę $p$, rozpoczynającą się od krawędzi $(u, v)$, która nie przechodzi żadną krawędzią więcej niż raz (przez jeden wierzchołek już może). Ścieżka $p$ musi kończyć się w wierzchołku $u$, jako że z każdego innego wierzchołka grafu możemy zawsze wyjść jakąś krawędzią, gdyż zawsze mamy tyle samo krawędzi wychodzących co wchodzących. To oznacza, że $p$ jest cyklem, który przechodzi przez wierzchołki $u$ i $v$ z dwóch różnych silnie spójnych składowych. To jednak nie jest możliwe (dlaczego?). Mamy żądaną sprzeczność. ■

Dzięki tej obserwacji możemy skupić się już tylko na warunku dotyczącym stopni wejściowych i wyjściowych wierzchołków w wynikowej orientacji $G'_k$. Postaramy się tak skierować krawędzie grafu $G_k$, aby:

- każdej krawędzi przypisać jeden z wierzchołków z nią incydentnych — będzie to jej początek we w pełni skierowanym grafie,
- zapewnić, że każdy z wierzchołków będzie przypisany do tylu krawędzi, ile wynosi połowa jego stopnia w wyjściowym grafie $G$; w ten sposób każdy z wierzchołków będzie miał stopień wejściowy równy wyjściowemu.

Skonstruujemy w tym celu sieć przepływową[7] $H_k$, patrz rys. 1. Składa się ona z:

- wierzchołków $v_i$ (dla $i \in \{1, 2, \ldots, n\}$) odpowiadających wierzchołkom $G_k$,

[7] *Sieć przepływowa* to graf skierowany, w którym każda krawędź ma przypisaną liczbę (zwaną przepustowością) oraz istnieją dwa wyróżnione wierzchołki: *źródło* i *ujście*.

- wierzchołków $e_j$ (dla $j \in \{1, 2, \ldots, m\}$) odpowiadających krawędziom $G_k$,
- krawędzi $v_i \to e_j$ o jednostkowych przepustowościach, jeśli $e_j$ jest krawędzią nieskierowaną, a $v_i$ jest z nią incydentny,
- krawędzi $v_i \to e_j$ o jednostkowych przepustowościach, jeśli $e_j$ jest krawędzią skierowaną wychodzącą z $v_i$,
- wierzchołka $s$ (źródła) połączonego z każdym $v_i$ krawędzią o przepustowości równej połowie stopnia $v_i$ w wyjściowym grafie $G$,
- ujścia $t$ połączonego ze wszystkimi $e_j$ krawędziami jednostkowymi.

Znajdźmy *maksymalny przepływ* w tak skonstruowanej sieci, np. przy użyciu algorytmu Edmondsa-Karpa opisanego w książce [21] [8]. Jeśli w znalezionym przepływie krawędź $v_i \to e_j$ jest nasycona, a $e_j$ jest nieskierowana w $G_k$, to skierujmy ją tak, by wychodziła z $v_i$. Jeśli wszystkie krawędzie incydentne ze źródłem (lub, równoważnie, z ujściem) są nasycone, to w grafie $G'_k$ stopnie wejściowe i wyjściowe wszystkich wierzchołków są równe. Wówczas i tylko wówczas (zachęcamy Czytelnika do samodzielnego uzasadnienia tej równoważności!) graf $G'_k$ ma cykl Eulera, czyli początkowy graf posiada $k$-cykl.

## Rozwiązanie wzorcowe – podsumowanie

To kończy opis rozwiązania wzorcowego, zaimplementowanego w plikach `mos.cpp`, `mos1.pas` oraz `mos[3-5].cpp`. Warto zastanowić się jeszcze nad jego złożonością czasową. W tym celu skupimy się przede wszystkim na analizie algorytmu przepływowego.

Zauważmy, że graf $H_k = (V_k, E_k)$ ma $O(n + m)$ wierzchołków i krawędzi, czyli $|V_k| = O(n + m)$ i $|E_k| = O(n + m)$. Podręcznikowa złożoność czasowa algorytmu Edmondsa-Karpa to $O(|V_k| \cdot |E_k|^2)$, czyli całkiem sporo. Ponieważ przepustowości są w naszym przypadku całkowitoliczbowe, więc możemy skorzystać z innego oszacowania złożoności czasowej tego algorytmu, a mianowicie $O((|V_k| + |E_k|) \cdot P)$, przy czym $P$ to wynikowa wartość maksymalnego przepływu. W przypadku grafu $H_k$ mamy oczywiście $P \leqslant m$, co pozwala oszacować koszt czasowy algorytmu przepływowego przez $O(m \cdot (n + m))$.

Przypomnijmy, że w rozwiązaniu wzorcowym wykonujemy $O(\log w)$ kroków, w każdym z których konstruujemy graf $H_k$ (czas $O(n + m)$) i uruchamiamy nasz algorytm przepływowy. Po wykonaniu wszystkich kroków znamy optymalne skierowanie krawędzi, na podstawie którego możemy znaleźć wynikowy cykl Eulera w czasie $O(n+m)$. Stąd, złożoność czasową rozwiązania wzorcowego możemy oszacować przez $O(m \cdot (n + m) \cdot \log w)$. Dodajmy, że złożoność pamięciowa to $O(n + m)$.

Takie rozwiązanie można zaimplementować także w nieco lepszej złożoności czasowej: $O(m \cdot (n + m))$. W tym celu należy zastąpić wyszukiwanie binarne stopniowym dodawaniem do sieci przepływowej krawędzi typu $v_i \to e_j$ odpowiadających

[8] Musimy skorzystać z algorytmu, który dla całkowitych przepustowości konstruuje przepływ o całkowitych wartościach na każdej krawędzi. Tę własność mają jednak wszystkie powszechnie stosowane metody wyznaczania maksymalnego przepływu.

coraz większym wagom ($l_i$ lub $p_i$). Po każdej dodanej krawędzi próbujemy zwiększać maksymalny przepływ za pomocą ścieżek powiększających w algorytmie Edmondsa-Karpa (zwiększenie przepływu o jednostkę wykonujemy w czasie $O(n + m)$). Krok ten powtarzamy tak długo, aż osiągniemy maksymalny przepływ nasycający wszystkie krawędzie wchodzące do ujścia — wówczas stwierdzamy, że waga ostatnio dodanej krawędzi stanowi koszt optymalnej trasy Bajtazara, a samą trasę odczytujemy ze znalezionego przepływu dokładnie tak jak w rozwiązaniu wzorcowym. Dodajmy tylko, że to usprawnienie nie było wymagane do uzyskania maksymalnej punktacji za zadanie.

## Rozwiązanie bez użycia przepływu: skojarzenie

Jeszcze kilka lat temu algorytmy przepływowe nie mieściły się w zakresie kompetencji oczekiwanych od zawodników Olimpiady Informatycznej. Okazuje się, że i tutaj można obejść się bez nich.

Można bowiem zamiast sieci przepływowej rozważyć graf dwudzielny, podobny do przedstawionej sieci, lecz bez źródła i ujścia, oraz z wierzchołkami $v_i$ powielonymi $deg(v_i)/2$ razy. Taki graf dwudzielny ma $O(m)$ wierzchołków oraz

$$O\left(\sum_{v \in V_G} (deg(v))^2\right) \;=\; O\left(\sum_{v \in V_G} (n \cdot deg(v))\right) \;=\; O(n \cdot m)$$

krawędzi. Rozmiar najliczniejszego skojarzenia w tym grafie jest równy maksymalnemu przepływowi w opisywanej sieci przepływowej. Skojarzenie wskazuje również, w jaki sposób należy zorientować poszczególne krawędzie.

Najliczniejsze skojarzenie można wyznaczyć algorytmem Hopcrofta-Karpa (opisanym np. w książce [26]), uzyskując rozwiązanie zadania w czasie $O(m^{1,5} \cdot n \cdot \log w)$, które również powinno zmieścić się w limitach czasowych. Można także użyć algorytmu *turbo-matching* opisanego w opracowaniu zadania *Cło* z XV Olimpiady Informatycznej [15]. Jego wykorzystanie daje co prawda złożoność czasową $O(m^2 \cdot n \cdot \log w)$, lecz w praktyce ten algorytm często działa szybciej od algorytmu Hopcrofta-Karpa. I faktycznie, rozwiązanie naszego zadania oparte na tej metodzie, zaimplementowane w pliku `moss1.cpp`, formalnie wolniejsze, w tym zadaniu na wszystkich testach sprawdza się lepiej od wzorcowego.

## Jeszcze inne rozwiązanie

Część czytelników mogłaby nam zarzucić, że przecież skojarzenie w grafie dwudzielnym to prawie to samo co przepływ, więc w powyższym rozwiązaniu *de facto* nie unikamy „algorytmów przepływowych", czymkolwiek właściwie one są. Autorzy tego opracowania poniekąd podzielają tę opinię. Dlatego w tej sekcji opiszemy rozwiązania, które *naprawdę* unikają korzystania z pojęcia przepływu. Skorzystamy w nich z podobnych pomysłów, jak te, które pojawiły się w rozwiązaniu wzorcowym zadania *Kości* z XII Olimpiady Informatycznej [12].

Rozważania rozpocznijmy od tego miejsca w rozwiązaniu wzorcowym, w którym wykoncypowaliśmy już możliwość wykorzystania wyszukiwania binarnego i koncen-

trujemy się właśnie na grafie częściowo skierowanym $G_k$, dla którego poszukujemy eulerowskiej orientacji $G'_k$. Wiemy, że wystarczy w tym celu tak skierować nieskierowane krawędzie grafu $G_k$, aby w wynikowym grafie $G'_k$ każdy wierzchołek miał taki sam stopień wejściowy i wyjściowy. *Zrównoważeniem* wierzchołka $v$ w grafie skierowanym, $\Delta(v)$, nazwijmy różnicę między jego stopniem wyjściowym a wejściowym, tzn.

$$\Delta(v) \stackrel{\text{def}}{=} outdeg(v) - indeg(v).$$

Używając tego pojęcia, możemy po prostu powiedzieć, że w wynikowej orientacji $G'_k$ zrównoważenia wszystkich wierzchołków muszą być zerowe; nazwijmy każdy graf skierowany spełniający ten warunek *zbalansowanym*.

Poszukiwania takiego grafu skierowanego możemy rozpocząć w dość zabawny sposób, a mianowicie, losowo skierowując wszystkie nieskierowane krawędzie $G_k$. Istnieje wówczas szansa, że nam się poszczęściło i że otrzymany w ten sposób graf $G''_k$ jest zbalansowany. Liczenie na ten przypadek stanowi jednak nadmiar optymizmu: w $G''_k$ mogą wystąpić także wierzchołki o dodatnim bądź ujemnym zrównoważeniu. Możemy jednak założyć, że zrównoważenia wszystkich wierzchołków w $G''_k$ są parzyste — w przeciwnym przypadku od razu wiemy, że $G_k$ nie da się odpowiednio skierować.

Jeśli $G''_k$ nie jest zbalansowany, to możemy spróbować coś poprawić, tak aby stał się zbalansowany. Wykonajmy następujący krok:

(*) *Wybierzmy w $G''_k$ jakikolwiek wierzchołek $v^+$, taki że $\Delta(v^+) > 0$, i sprawdźmy, czy w $G''_k$ istnieje ścieżka z $v^+$ do jakiegokolwiek wierzchołka $v^-$ o ujemnym zrównoważeniu, przechodząca jedynie krawędziami, które w oryginalnym grafie $G_k$ były* nieskierowane*. Jeśli taka ścieżka istnieje, to odwróćmy wszystkie krawędzie w niej zawarte.*

W wyniku kroku (*) zmniejszamy zrównoważenie $v^+$ o 2, zwiększamy zrównoważenie $v^-$ również o 2, natomiast zrównoważenia wszystkich pozostałych wierzchołków pozostają niezmienione. Takie postępowanie ewidentnie przybliża nas do celu, czyli do uzyskania grafu zbalansowanego. Faktycznie, jeśli uda nam się powtórzyć krok (*) odpowiednio dużo razy, a mianowicie:

$$R \stackrel{\text{def}}{=} \sum_{v \,:\, \Delta(v)>0} \frac{\Delta(v)}{2} \tag{1}$$

razy, to otrzymany graf będzie zbalansowany. Pytanie brzmi, czy zachodzi implikacja odwrotna, czyli czy każdy graf niezbalansowany można sprowadzić do zbalansowanego, wykonując krok (*) odpowiednio dużo razy, zakładając oczywiście, że w ogóle istnieje jakakolwiek zbalansowana orientacja. Okazuje się, że ta implikacja zachodzi, co pokazuje następujące twierdzenie (zapisane w sposób dość abstrakcyjny). Od razu zaznaczmy, że wnioskiem z Twierdzenia 3 jest poprawność algorytmu poprawiającego losowo skierowany graf $G''_k$ za pomocą $R$-krotnego wykonania kroku (*) — złożonością tego algorytmu zajmiemy się zaraz po udowodnieniu twierdzenia.

**Twierdzenie 3.** *Niech $H = (V_H, E_H)$ będzie grafem częściowo skierowanym oraz niech $H''$ będzie dowolną niezbalansowaną orientacją grafu $H$. Niech dalej $v^+$ będzie dowolnym wierzchołkiem $H''$ o dodatnim zrównoważeniu. Jeśli $H$ ma jakąś zbalansowaną orientację, to w $H''$ istnieje ścieżka z $v^+$ do jakiegoś wierzchołka o ujemnym zrównoważeniu, przechodząca jedynie krawędziami, które w $H$ były nieskierowane.*

**Dowód:** Niech $H'$ będzie obiecaną, zbalansowaną orientacją grafu $H$. Niech dalej $F$ będzie zbiorem krawędzi, które są w $H''$, ale nie ma ich w $H'$. Zbiór $F$ reprezentuje zatem te krawędzie nieskierowane grafu $H$, których skierowanie jest inne w $H''$ niż w $H'$. Zdefiniujmy nowy graf $H''' = (V_H, F)$.

Ciekawą własnością grafu $H'''$ jest to, że znaki zrównoważeń wierzchołków są w nim takie same jak w $H''$ (dodatnie zrównoważenia przechodzą na dodatnie, ujemne na ujemne, zerowe na zerowe). Intuicyjnie, zbiór $F$ pokazuje, którym krawędziom z $H''$ należy odwrócić skierowanie, tak aby ten graf stał się zbalansowany — np. w przypadku wierzchołka o dodatnim zrównoważeniu więcej krawędzi wychodzących trzeba przerobić na wchodzące niż odwrotnie, wchodzących na wychodzące. Formalny dowód tego spostrzeżenia pomijamy.

W szczególności wiemy, że $v^+$ ma dodatnie zrównoważenie w $H'''$. Ponadto, jeśli w $H'''$ znajdziemy ścieżkę z $v^+$ do jakiegoś wierzchołka o ujemnym zrównoważeniu w $H'''$, to będzie ona reprezentować ścieżkę w $H''$ z $v^+$ do wierzchołka o ujemnym zrównoważeniu w $H''$, przechodzącą jedynie krawędziami, które w $H$ są nieskierowane. Znajdując taką ścieżkę, zakończymy dowód. Istnienie takiej ścieżki gwarantuje Obserwacja 4, dla niepoznaki zapisana i udowodniona tuż pod niniejszym dowodem. Używając tej obserwacji, kończymy dowód twierdzenia. ■

**Obserwacja 4.** Niech $H$ będzie grafem skierowanym oraz niech $v^+$ będzie wierzchołkiem tego grafu o dodatnim zrównoważeniu. Wówczas z $v^+$ istnieje ścieżka do jakiegoś wierzchołka o ujemnym zrównoważeniu.

**Dowód:** W dowodzie tej obserwacji, podobnie jak w dowodzie występującej kilka stron wcześniej Obserwacji 3, skonstruujemy w grafie $H$ maksymalną (nieprzedłużalną) ścieżkę $p$ prowadzącą z wierzchołka $v^+$, w której żadna krawędź nie powtarza się, choć wierzchołki już mogą. Zastanówmy się, gdzie może się ta ścieżka skończyć.

Na pewno $p$ *nie* może skończyć się w samym wierzchołku $v^+$, jako że ma on więcej krawędzi wychodzących niż wchodzących, więc po każdym powrocie do niego mamy jeszcze krawędź, którą możemy wyjść. Podobnie uzasadniamy, że ścieżka ta *nie* może skończyć się w żadnym innym wierzchołku o nieujemnym zrównoważeniu. Ponieważ $p$ nie może być nieskończona, więc musi skończyć się w wierzchołku o ujemnym zrównoważeniu. ■

Przyszła pora na obiecaną analizę złożoności czasowej podanego rozwiązania. Konstrukcja losowej orientacji $G''_k$ grafu $G_k$ wymaga, oczywiście, czasu $O(n+m)$. Równie szybko możemy obliczyć zrównoważenia wierzchołków w tej orientacji i sprawdzić, czy przypadkiem któreś z nich nie jest nieparzyste — a jeśli tak, natychmiast zwrócić odpowiedź `NIE`. Krok (*) łatwo zrealizować za pomocą dowolnego algorytmu przeszukiwania grafu (np. BFS, DFS, patrz książka [21]) — daje to także koszt czasowy $O(n+m)$. Krok ten musimy wykonać $R$ razy, przy czym łatwo widać, że $R \leqslant m$. Dokładając do tego zewnętrzne wyszukiwanie binarne, otrzymujemy oszacowanie złożoności czasowej algorytmu: $O(m \cdot (n+m) \cdot \log w)$, czyli takie samo jak w przypadku rozwiązania wzorcowego. Implementację tego rozwiązania można znaleźć w pliku `mos6.cpp`.

### Można lepiej

Okazuje się, że pomysły z poprzedniej sekcji również można wykorzystać do konstrukcji rozwiązania o złożoności czasowej $O(m \cdot (n+m))$. Istota tego podejścia jest jednak nieco inna niż w odpowiadającej modyfikacji rozwiązania wzorcowego.

Przedstawimy najpierw nieefektywną realizację tego innego podejścia. Składa się ono z następujących kroków:

1. Zaczynamy standardowo, czyli od sprawdzenia, czy wyjściowy graf $G$ (ten reprezentujący wyspy i mosty pomiędzy nimi) jest eulerowski, patrz Twierdzenie 1. Jeśli nie, to problem z głowy, a jeśli tak, to przechodzimy dalej.

2. Sortujemy krawędzie grafu $G$ *nierosnąco* względem *maksimum* z wag $l_i$, $p_i$ oznaczających siłę wiatru wiejącego w każdą ze stron. W ten sposób otrzymujemy uporządkowaną listę krawędzi $L$.

3. Niech $G'$ oznacza graf częściowo skierowany, początkowo równy $G$ (wszystkie krawędzie nieskierowane).

4. Przeglądamy kolejne krawędzie $e \in L$, próbując dla każdej z nich wybrać tańsze z dwóch skierowań, odpowiadające mniejszej sile wiatru (albo dowolne skierowanie, jeśli oba są równie tanie).

   a) W ten właśnie sposób skierowujemy krawędź $e$ w grafie $G'$, po czym sprawdzamy, czy po wykonaniu tego kroku $G'$ ma jakąś orientację eulerowską. To sprawdzenie możemy wykonać na dowolny z trzech przedstawionych w tym opracowaniu sposobów (przepływ, skojarzenie w grafie dwudzielnym albo iteracyjne poprawianie orientacji).

   b) Jeśli tak, to kontynuujemy przeglądanie listy $L$, a jeśli nie, to kończymy.

Twierdzimy, że ostatnia orientacja eulerowska, jaką otrzymamy — albo po przejrzeniu całej listy $L$, albo tuż przed skierowaniem jakiejś krawędzi w $G'$, które psuje możliwość znalezienia orientacji eulerowskiej — stanowi szukaną, najtańszą trasę Bajtazara.

Uzasadnienie tego stwierdzenia jest następujące. Niech $k_0$ oznacza koszt najtańszego cyklu Eulera w grafie $G$, tzn. maksimum z sił wiatru na krawędziach w kierunku przechodzenia wzdłuż cyklu. Jeśli $l_i > k_0$ dla pewnej, $i$-tej krawędzi, to musi zachodzić $p_i \leqslant k_0$ i wówczas musimy przejść tą krawędzią w kierunku odpowiadającym $p_i$. Symetrycznie, jeśli $p_i > k_0$, to $i$-tą krawędzią trzeba przejść w kierunku odpowiadającym $l_i$. To oznacza, że po przetworzeniu początkowego fragmentu listy $L$ odpowiadającego krawędziom o maksimum wag większym niż $k_0$, w grafie $G'$ musi wciąż istnieć orientacja eulerowska — i nasz algorytm ją znajdzie. Dalej będzie być może znajdował jeszcze jakieś inne orientacje, z których żadna nie będzie miała maksymalnej wagi większej niż $k_0$, gdyż krawędzie umożliwiające uzyskanie wagi przekraczającej $k_0$ już odpowiednio skierowaliśmy. Stąd, ostatnia znaleziona orientacja ma koszt nie gorszy niż $k_0$, czyli optymalny, co kończy nasze uzasadnienie.

Wiemy już, że algorytm opisany krokami 1-4 jest poprawny. Co więcej, pozbyliśmy się w nim wyszukiwania binarnego, ale za to przysporzyliśmy sobie dodatkowego

czynnika $m$, wynikłego z wielokrotnego wykonywania kroku 4a. Czynnika tego pozbędziemy się jednak równie łatwo, jak go wprowadziliśmy: wystarczy, że nie będziemy wyznaczać orientacji eulerowskiej za każdym razem od nowa.

W tym celu będziemy cały czas utrzymywać orientację eulerowską $G''$ grafu $G'$. Na początku (krok 3) wyznaczamy ją w czasie $O(n + m)$ za pomocą klasycznego algorytmu na cykl Eulera w grafie nieskierowanym. Musimy teraz pokazać, jak zmieniać orientację eulerowską po wykonaniu skierowania krawędzi $e$ w kroku 4a. Jeśli wykonane skierowanie jest takie samo jak skierowanie $e$ w $G''$, to nic nie musimy robić. Jeśli jest przeciwne, to musimy odwrócić odpowiadającą krawędź w $G''$, czym na pewno zaburzymy zbalansowanie orientacji $G''$: wówczas zrównoważenia końców tej krawędzi będą równe 2 i $-2$, ale zrównoważenia wszystkich pozostałych wierzchołków pozostaną zerowe. Po tej poprawce graf $G''$ jest więc niezbalansowaną orientacją nowego grafu $G'$. Jeśli da się ją poprawić do orientacji zbalansowanej, to na mocy Twierdzenia 3, wystarczy do tego jednokrotne wykonanie kroku (*). To oznacza, że krok 4a możemy wykonać w czasie $O(n + m)$, co pokazuje, że cały algorytm da się w ten sposób zaimplementować w złożoności czasowej $O(m \cdot (n + m))$.

## Testy

Rozwiązania tego zadania sprawdzano na 10 zestawach testów, po 3-5 testów w zestawie:

- Testy *a* to testy specyficzne, reprezentujące grafy, które albo są klikami (*3a*, *5a*, *9a*), albo mają mało cykli Eulera (a przynajmniej mało tanich cykli, cokolwiek to znaczy).
- Testy *b* reprezentują losowe grafy mające cykl Eulera.
- Testy *c* mają, w większości, odpowiedź `NIE`.
- Testy *d* (prócz *2d*) reprezentują grafy złożone z cykli prostych o jednym wspólnym wierzchołku. Na każdym z tych cykli w optymalnym rozwiązaniu tylko jedna krawędź ma wymuszone skierowanie.
- W testach *e* oraz w teście *2d* zbiory wag na krawędziach: $\{l_i\}$ oraz $\{p_i\}$, są rozłączne, co wpływa ujemnie na czas działania niektórych rozwiązań powolnych tego zadania.

| Nazwa | n | m |
|---|---|---|
| *mos1a.in* | 8 | 14 |
| *mos1b.in* | 8 | 12 |
| *mos1c.in* | 11 | 20 |
| *mos2a.in* | 19 | 28 |
| *mos2b.in* | 20 | 31 |
| *mos2c.in* | 20 | 34 |

| Nazwa | n | m |
|---|---|---|
| *mos2d.in* | 25 | 80 |
| *mos3a.in* | 5 | 10 |
| *mos3b.in* | 41 | 53 |
| *mos3c.in* | 38 | 60 |
| *mos3d.in* | 45 | 55 |
| *mos4a.in* | 35 | 60 |

| Nazwa | n | m |
|---|---|---|
| *mos4b.in* | 78 | 81 |
| *mos4c.in* | 80 | 100 |
| *mos4d.in* | 86 | 102 |
| *mos5a.in* | 25 | 300 |
| *mos5b.in* | 101 | 300 |
| *mos5c.in* | 100 | 250 |
| *mos5d.in* | 127 | 147 |
| *mos6a.in* | 210 | 1 567 |
| *mos6b.in* | 200 | 313 |
| *mos6c.in* | 250 | 400 |
| *mos6d.in* | 345 | 387 |
| *mos7a.in* | 500 | 2 000 |
| *mos7b.in* | 499 | 1 000 |
| *mos7c.in* | 502 | 510 |
| *mos7d.in* | 401 | 480 |
| *mos7e.in* | 45 | 990 |

| Nazwa | n | m |
|---|---|---|
| *mos8a.in* | 750 | 1 000 |
| *mos8b.in* | 658 | 800 |
| *mos8c.in* | 800 | 1 700 |
| *mos8d.in* | 871 | 900 |
| *mos8e.in* | 800 | 1 800 |
| *mos9a.in* | 63 | 1 953 |
| *mos9b.in* | 997 | 1 157 |
| *mos9c.in* | 1 000 | 2 000 |
| *mos9d.in* | 601 | 650 |
| *mos9e.in* | 1 000 | 1 800 |
| *mos10a.in* | 1 000 | 1 800 |
| *mos10b.in* | 1 000 | 2 000 |
| *mos10c.in* | 1 000 | 1 000 |
| *mos10d.in* | 801 | 1 000 |
| *mos10e.in* | 1 000 | 2 000 |