Omówienie zadań AMPPZ 2018

Jury

II UWr

27 października 2018

H: Hobby

Zadanie (Autor: Karol Pokorski)

Rozwiązać łamigłówkę Suko.

- W pola planszy 3×3 należy wpisać permutację.
- Testujemy wszystkie 9! permutacji naiwnie i po prostu sprawdzamy warunki.
- Alternatywnie: testujemy dużo losowych permutacji.
- Alternatywnie: przeszukiwanie z nawrotami.

C: Chińskie twierdzenie o resztach

Zadanie (Autor: Karol Pokorski)

Znaleźć największe rozwiązanie układu kongruencji:

$$\begin{cases} a_1 \equiv b_1 \pmod{m} \\ a_2 \equiv b_2 \pmod{m} \\ \vdots \quad \vdots \quad \vdots \\ a_n \equiv b_n \pmod{m} \end{cases}$$

- Przekształcamy każdą kongruencję do postaci $a_i - b_i \equiv 0 \pmod{m}$.
- Zauważamy, że znaczenie powyższego to: m jest dzielnikiem $a_i - b_i$.
- Wynik to największy wspólny dzielnik wszystkich różnic $a_i - b_i$. Obliczamy go algorytmem Euklidesa.

J: Jednakowe szaliki

Zadanie (Autor: Jakub Tarnawski)

Dany ciąg liczb a_i i parametr k. Zmienić liczby ciągu o łącznie nie więcej niż k, żeby jak najwięcej z nich było równych.

- Sortujemy ciąg i zauważamy, że możemy skupić się jedynie na wyrównywaniu spójnych kawałków w posortowanym ciągu.
- Wyszukujemy binarnie wynik: liczbę równych liczb po wykonaniu operacji.
- Dla ustalonego kawałka, który chcemy zrównać, zawsze chcemy wyrównać do mediany.
- Obliczamy w czasie liniowym koszty sprowadzenia wszystkich spójnych kawałków ustalonej długości do mediany za pomocą metody gąsiennicy: utrzymujemy zawsze sumę wartości mniejszych od mediany i większych od mediany.

F: Foremki na ciasta

Zadanie (Autor: Karol Pokorski)

Sprawdzić czy $\sqrt{x} + \sqrt{y} < \sqrt{z}$.

- Pomysł zrobienia tego na typach zmiennoprzecinkowych, nawet `long double`, jest bardzo niepewny.
- Podnosimy całość do kwadratu: dostajemy $x + y + 2\sqrt{xy} < z$.
- Chcemy sprawdzić czy $2\sqrt{xy} < z - x - y$.
- Na dobry początek zauważamy, że jeśli $z < x + y$ to wynik zadania to `NIE`.
- Jeśli to nie rozstrzygnęło, podnosimy nierówność $2\sqrt{xy} < z - x - y$ do kwadratu i pozbywamy się pierwiastków (za cenę dużych liczb).
- Używamy `__int128_t`, piszemy własne mini-bignumy lub zachowujemy dużą ostrożność przy używaniu typów 64-bitowych.

K: Kieszonkowe

Zadanie (Autor: Karol Pokorski)

Dla danego ciągu wzrostów lub spadków kieszonkowego z dziurami wyznaczyć wypełnienie dziur, aby sumaryczne kieszonkowe było najmniejsze lub największe (przy zachowaniu warunku o zawsze nieujemnym kieszonkowym startującym od 0 i kończącym się na 0).

- Gdyby nie zera, mielibyśmy do rozwiązania problem wypełnienia dziur, aby uzyskać poprawne nawiasowanie.
- Bieżące kieszonkowe utożsamiamy z bieżącym balansem nawiasów.

K: Kieszonkowe

Zadanie (Autor: Karol Pokorski)

Dla danego ciągu wzrostów lub spadków kieszonkowego z dziurami wyznaczyć wypełnienie dziur, aby sumaryczne kieszonkowe było najmniejsze lub największe (przy zachowaniu warunku o zawsze nieujemnym kieszonkowym startującym od 0 i kończącym się na 0).

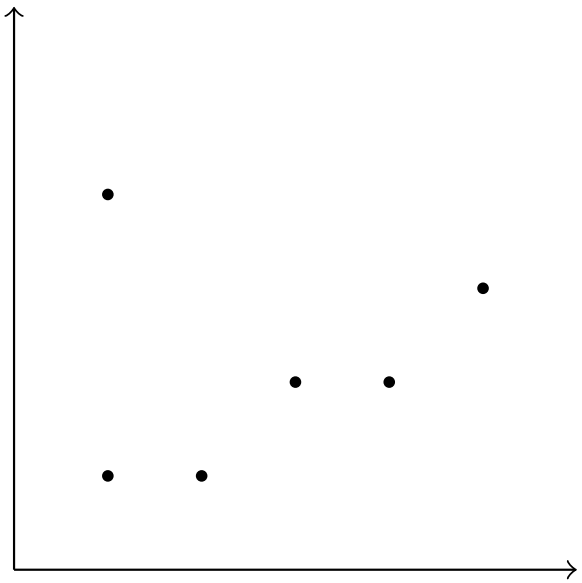
- Zauważamy, że najmniejsze/największe kieszonkowe jest dla najmniejszego/największego leksykograficznie nawiasowania.
- Dla każdego prefiksu i każdego sufiksu obliczamy minimalny/maksymalny możliwy balans nawiasów jaki możemy tam uzyskać.
- Mając te informacje jesteśmy w stanie wypełniać rozwiązanie zachłannie: wiemy jaki nawias chcemy postawić w pierwszej kolejności (otwierający dla problemu maksymalizacyjnego, zamykający dla problemu minimalizacyjnego) i potrafimy ustalić czy będzie się dało dobrze dokończyć nawiasowanie po wypełnieniu kolejnej dziury.
- Zera traktujemy jako typ nawiasu, który nie zmienia balansu.

I: Indeks

Zadanie (Autor: Artur Kraska)

Ile dni potrzeba na zebranie wszystkich wpisów od wykładowców znajdujących się na ustalonych pozycjach, dających wpisy w określonych momentach dnia? Prędkość ruchu studenta wynosi jeden pokój na minutę.

- Spójrzmy na problem geometrycznie. Wykładowcę dającego wpis w pokoju p w minucie t utożsamiamy z punktem (p, t) na płaszczyźnie.

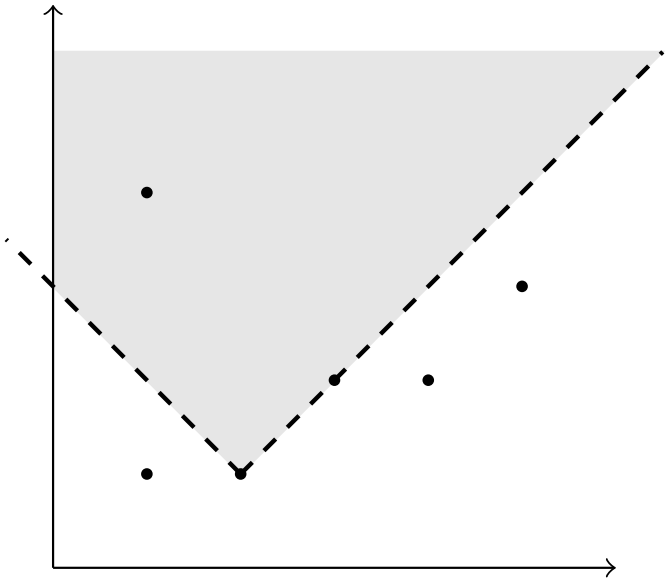


I: Indeks

Zadanie (Autor: Artur Kraska)

Ile dni potrzeba na zebranie wszystkich wpisów od wykładowców znajdujących się na ustalonych pozycjach, dających wpisy w określonych momentach dnia? Prędkość ruchu studenta wynosi jeden pokój na minutę.

- Jeśli student znajduje się w punkcie (p, t), to po upływie t' sekund może się znaleźć w punktach $(p - t', t + t'), (p - t' + 1, t + t'), \ldots, (p + t', t + t')$.

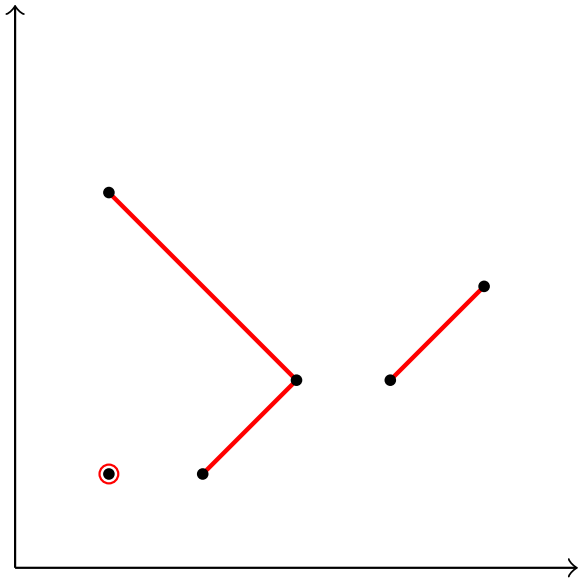


I: Indeks

Zadanie (Autor: Artur Kraska)

Ile dni potrzeba na zebranie wszystkich wpisów od wykładowców znajdujących się na ustalonych pozycjach, dających wpisy w określonych momentach dnia? Prędkość ruchu studenta wynosi jeden pokój na minutę.

- Chcemy pokryć wszystkie punkty dozwolonymi ścieżkami (każdy odcinek musi zawierać się w trójkącie wyznaczanym przez poprzedni trójkąt). Każda ścieżka to osobny dzień zbierania wpisów.



I: Indeks

Zadanie (Autor: Artur Kraska)

Ile dni potrzeba na zebranie wszystkich wpisów od wykładowców znajdujących się na ustalonych pozycjach, dających wpisy w określonych momentach dnia? Prędkość ruchu studenta wynosi jeden pokój na minutę.

- Obracamy płaszczyznę o 45 stopni.
- Ignorujemy jeden wymiar.
- Korzystamy z właściwości, że minimalna liczba podciągów nierosnących pokrywających wszystkie elementy ciągu jest równa długości najdłuższego podciągu rosnącego.
- Rozwiązujemy problem LIS z użyciem wyszukiwania binarnego lub drzewa przedziałowego.

B: Bony Fibonacciego

Zadanie (Autor: Karol Pokorski)

Wyznacz n-tą najmniejszą liczbę dającą się zapisać jako suma dokładnie k (niekoniecznie różnych) liczb Fibonacciego.

- Aby zapisać liczbę jako sumę jak najmniejszej liczby liczb Fibonacciego, wystarczy użyć algorytmu zachłannego.
- Jeśli użyto więcej niż k liczb Fibonacciego to tej liczby nie da się zapisać jako sumy dokładnie k liczb Fibonacciego.
- Jeśli użyto mniej – można rozdrobnić dowolne F_m na $F_{m-1} + F_{m-2}$ (poza sytuacją gdy $m = 1$).
- Zadanie sprowadza się więc do nieco prostszego: wyznaczyć n-tą liczbę, większą lub równą n, dla której algorytm zachłanny użyje co najwyżej k liczb Fibonacciego.
- Jeszcze prościej: po prostu $(n + k)$-te rozwiązanie (już bez warunku o liczbie większej lub równej n).

B: Bony Fibonacciego

Zadanie (Autor: Karol Pokorski)

Wyznacz n-tą najmniejszą liczbę dającą się zapisać jako suma dokładnie k (niekoniecznie różnych) liczb Fibonacciego.

- Programowanie dynamiczne: stanem jest $DP[m][k]$: liczba liczb mniejszych niż F_m, które dają się zapisać jako suma co najwyżej k liczb Fibonacciego.
- Ustalamy najmniejsze m, dla którego $DP[m][k] \geqslant n$. Jeśli $m > 86$ to odpowiedź to `NIE` (bo $F_{86} > 10^{18}$).
- Wiemy już, że wynik ma mieć dokładnie m cyfr w systemie pozycyjnym o wagach Fibonacciego.
- Ustalamy kolejne bity reprezentacji w zależności od stanów DP.

G: Gusta malarskie

Zadanie (Autor: Karol Pokorski)

Wybrać największy dobry podzbiór T podanego zbioru S. Zbiór T jest dobry, gdy dla każdego x w T znajdują się co najwyżej dwa elementy ze zbioru $\{x, 2x, 3x\}$.

- Dla każdego x należy ze zbioru S wyrzucić co najmniej jedną z liczb x, $2x$, $3x$.
- Rozważmy graf, w którym wierzchołkami są elementy S, a krawędzie są między parami $(x, 2x)$, $(x, 3x)$, $(2x, 3x)$.
- Każdy element zbioru S zapiszmy jako $2^\alpha \cdot 3^\beta \cdot r$, gdzie r nie dzieli się przez 2 ani 3.
- Elementy o różnych r są w różnych składowych grafu i mogą być rozpatrywane niezależnie. Dla każdej składowej chcemy wybrać jak najmniej wierzchołków, żeby usatysfakcjonować wszystkie trójkąty $(x, 2x, 3x)$ w tej samej spójnej.

G: Gusta malarskie

Zadanie (Autor: Karol Pokorski)

Wybrać największy dobry podzbiór T podanego zbioru S. Zbiór T jest dobry, gdy dla każdego x w T znajdują się co najwyżej dwa elementy ze zbioru $\{x, 2x, 3x\}$.

- Dla każdej spójnej podzielmy ją na warstwy według $\alpha + \beta$.
- Warstw jest co najwyżej $\log_2 10^9 \approx 30$, a na każdej z nich co najwyżej $\log_3 10^9 \approx 19$ wierzchołków. W dodatku tylko jedna spójna może mieć taką charakterystykę (ta z $r = 1$, a następna jest już z $r = 5$).
- Programowanie dynamiczne po warstwach. Na każdej warstwie stanem jest maska bitowa usuniętych wierzchołków. Wynikiem stanu sumaryczna liczba usuniętych wierzchołków. Kompatybilność masek można sprawdzać naiwnie.

L: Linie

Zadanie (Autor: Karol Pokorski)

Ile jest całkowicie wypełnionych remisujących plansz $n \times n$ do uogólnionej gry w kółko i krzyżyk?

- Obliczmy plansze, które nie są remisujące.
- Skorzystamy z metody włączeń i wyłączeń. Potrzebujemy zliczać ile jest plansz w których jest $\geqslant k$ linii pionowych i $\geqslant l$ linii poziomych oraz zero, jedna lub dwie przekątne z tego samego (nieustalonego) symbolu.
- Przypadki, w których są tylko linie pionowe lub tylko linie poziome są odrobinę inne (każda linia może być z innego symbolu).

L: Linie

Zadanie (Autor: Karol Pokorski)

Ile jest całkowicie wypełnionych remisujących plansz $n \times n$ do uogólnionej gry w kółko i krzyżyk?

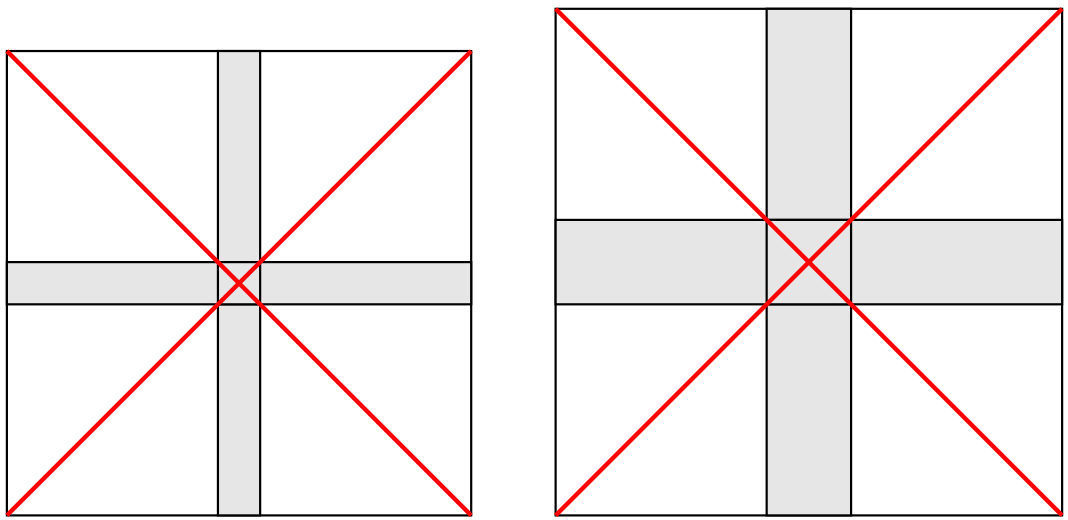
- Wyniki (liczby konfiguracji dla rozmiaru $n \times n$ i zadanej liczby linii poziomych i pionowych) obliczamy z użyciem programowania dynamicznego. Przypadki dla różnej liczby wybranych przekątnych (d) trzeba rozpatrywać osobno.
- Dla $d = 0$: liczymy $DP_0[n][k][l]$ na podstawie $DP_0[n-1][k][l]$, $DP_0[n-1][k-1][l]$, $DP_0[n-1][k][l-1]$, $DP_0[n-1][k-1][l-1]$.
- Dla $d = 1$ można podobnie policzyć $DP_1[n][k][l]$.

L: Linie

Zadanie (Autor: Karol Pokorski)

Ile jest całkowicie wypełnionych remisujących plansz $n \times n$ do uogólnionej gry w kółko i krzyżyk?

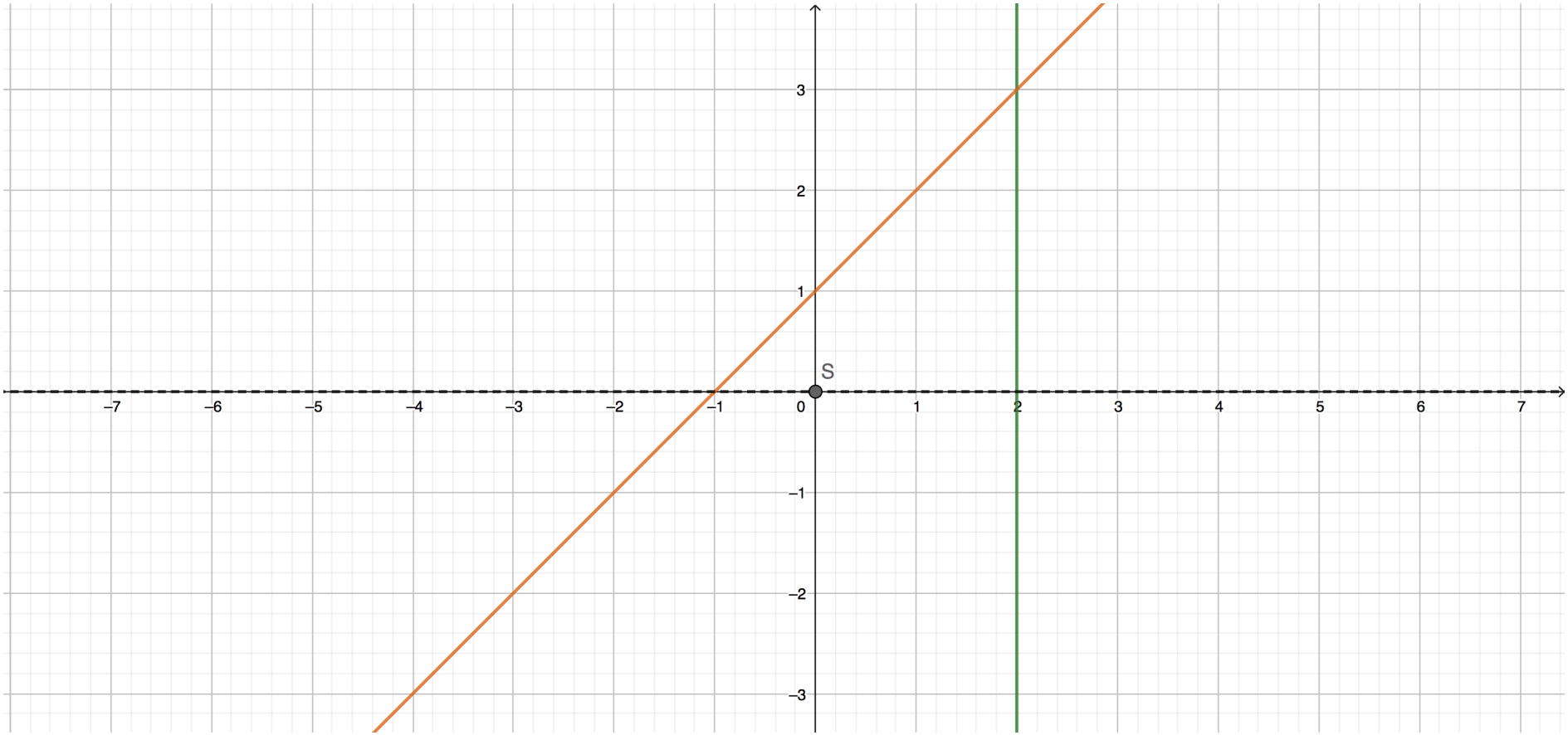
- Dla $d = 2$ rozpatrujemy przypadki z n parzystym i nieparzystym osobno: obliczenie $DP_2[2n][k][l]$ sprowadzamy do obliczenia $DP_2[2n-2][k'][l']$ (ustalając środkowe wiersze i środkowe kolumny), zaś obliczenie $DP_2[2n-1][k][l]$ sprowadzamy (też) do obliczenia $DP_2[2n-2][k'][l']$ (ustalając środkowy wiersz i środkową kolumnę).



A: Aparat na dronie

Zadanie (Autor: Bartłomiej Dudek)

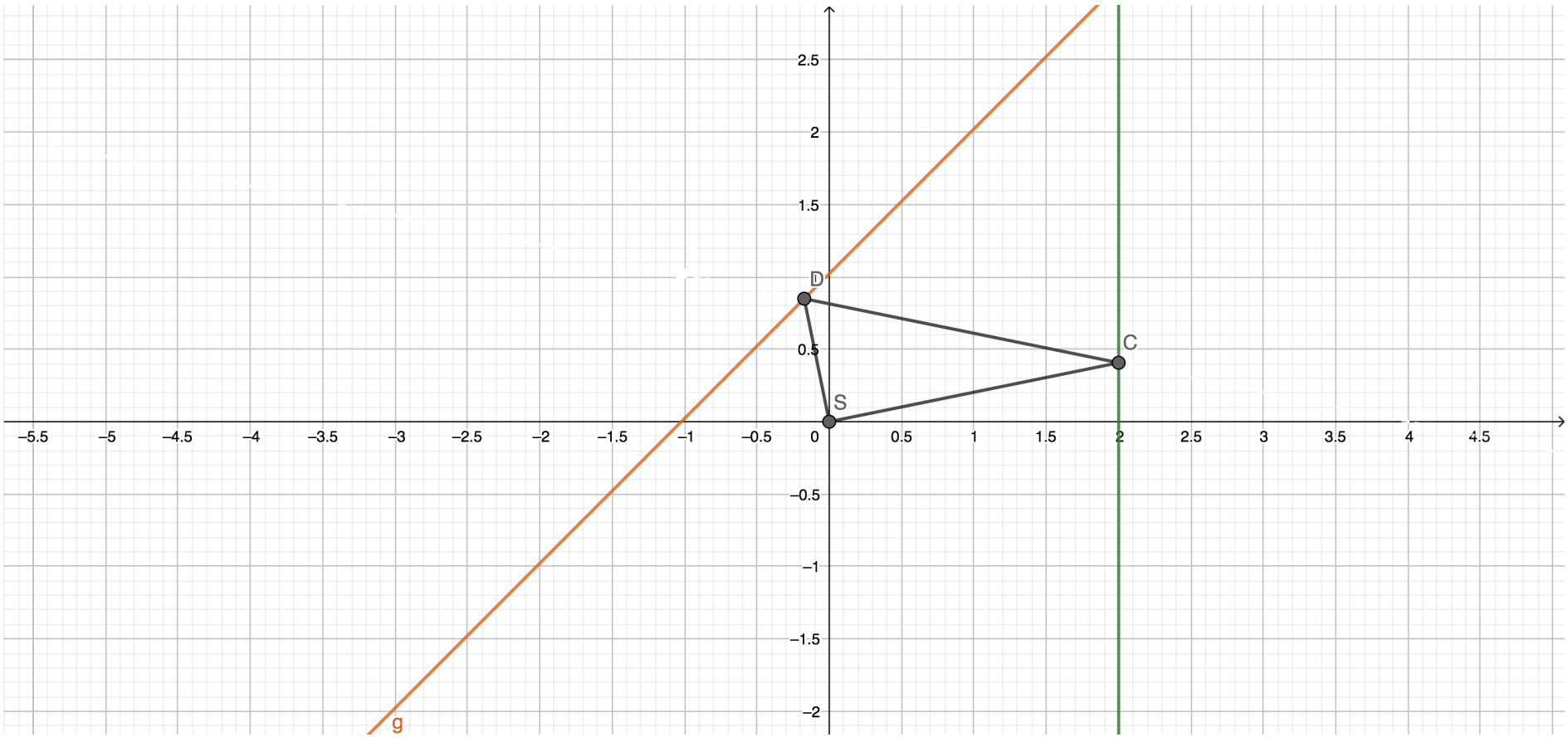
Dane są dwie proste na płaszczyźnie, znajdź najkrótszą trasę zaczynającą i kończącą się w $S = (0, 0)$, która odwiedza obydwie proste.



A: Aparat na dronie

Zadanie (Autor: Bartłomiej Dudek)

Dane są dwie proste na płaszczyźnie, znajdź najkrótszą trasę zaczynającą i kończącą się w $S = (0, 0)$, która odwiedza obydwie proste.

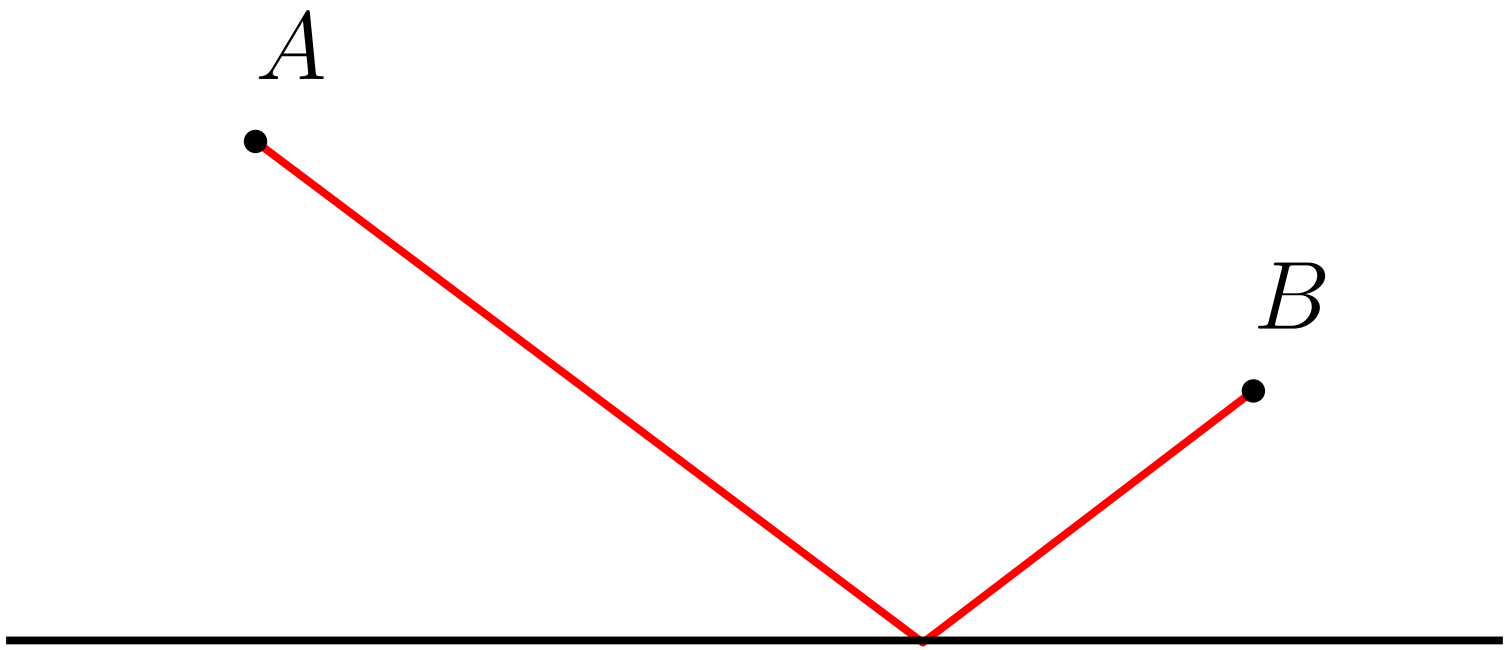


Prostsze zadanie

Znajdź najkrótszą trasę zaczynającą się w punkcie A, kończącą się w punkcie B, oraz odwiedzającą daną prostą ℓ.

Prostsze zadanie

Znajdź najkrótszą trasę zaczynającą się w punkcie A, kończącą się w punkcie B, oraz odwiedzającą daną prostą ℓ.

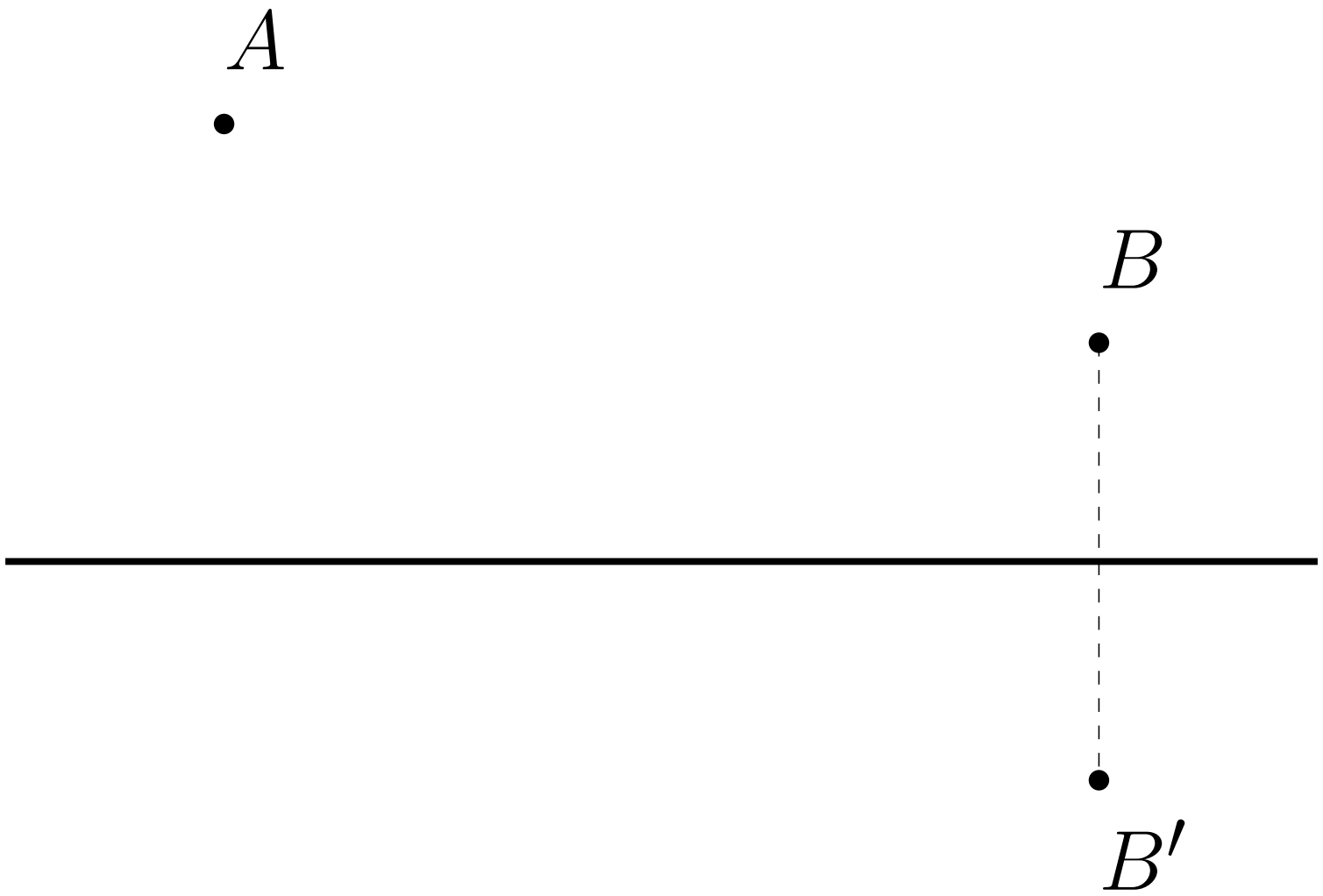


Prostsze zadanie

Znajdź najkrótszą trasę zaczynającą się w punkcie A, kończącą się w punkcie B, oraz odwiedzającą daną prostą ℓ.

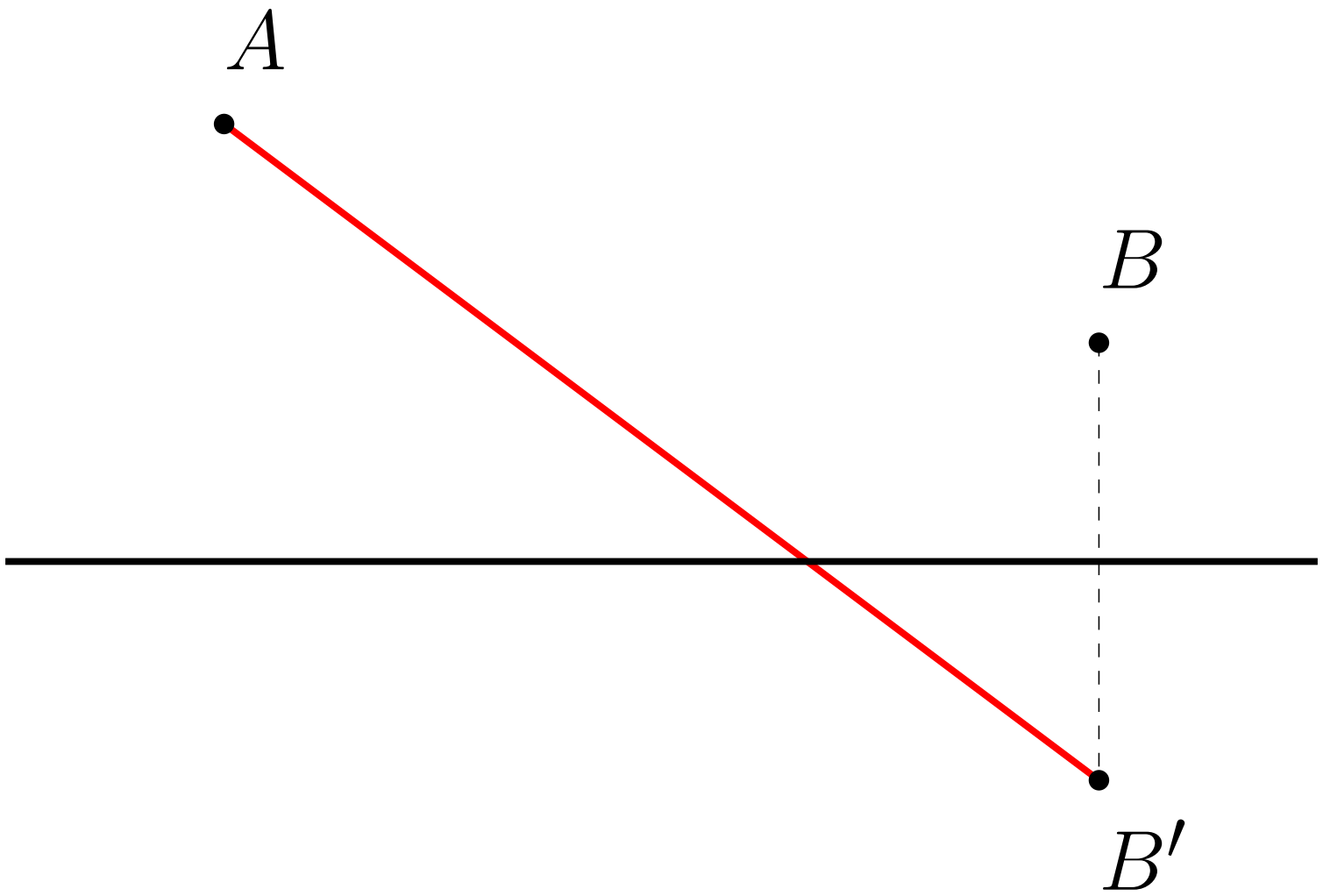
Prostsze zadanie

Znajdź najkrótszą trasę zaczynającą się w punkcie A, kończącą się w punkcie B, oraz odwiedzającą daną prostą ℓ.



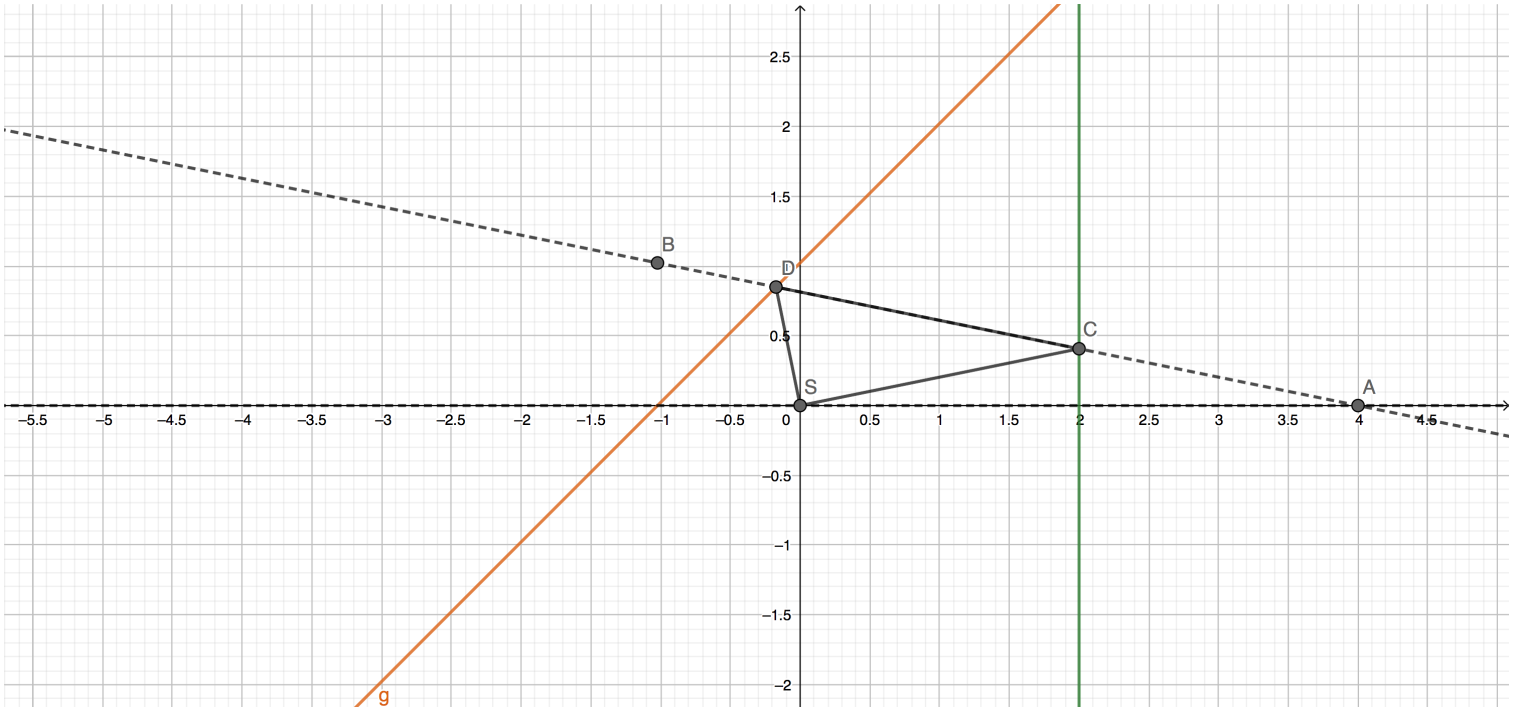
Prostsze zadanie

Znajdź najkrótszą trasę zaczynającą się w punkcie A, kończącą się w punkcie B, oraz odwiedzającą daną prostą ℓ.



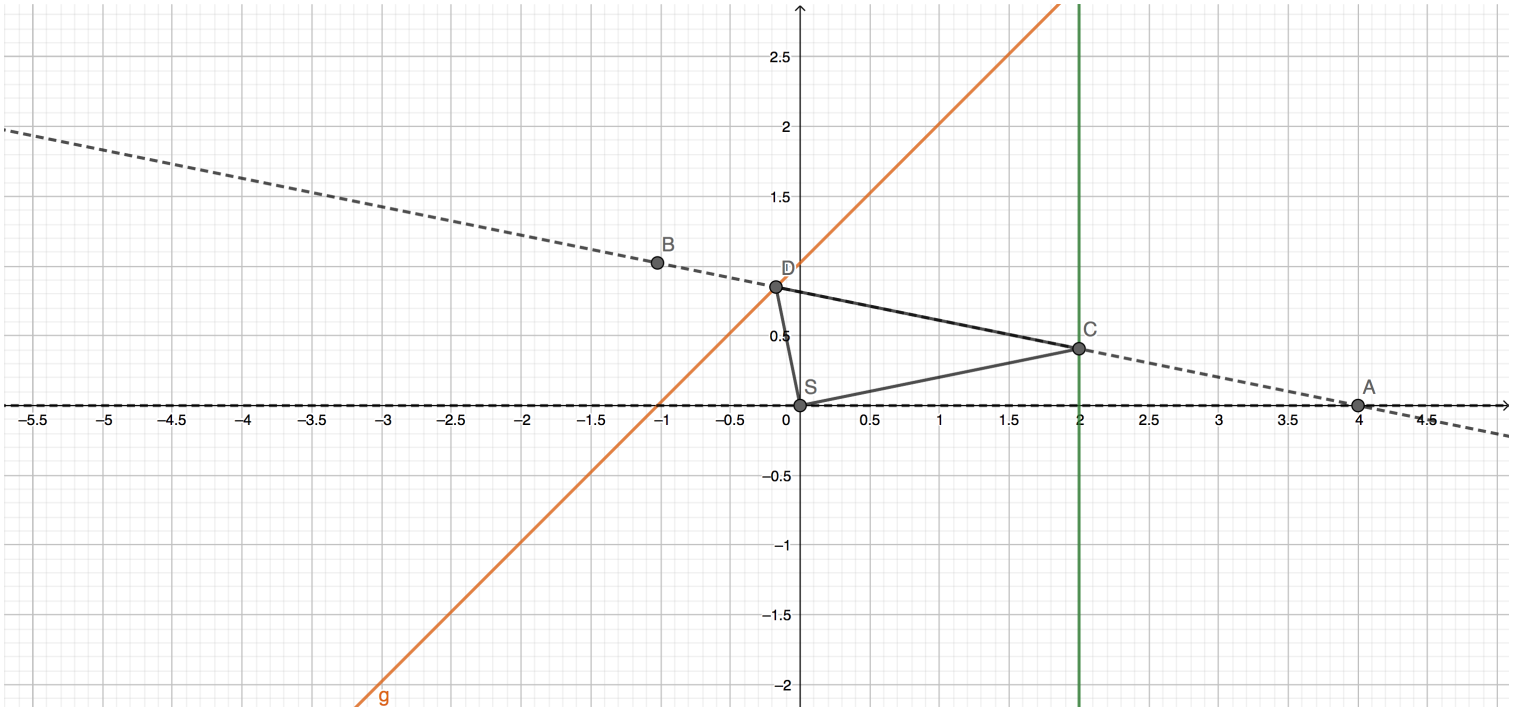
..a teraz jeszcze raz przykład z tego trudniejsze

Odbijamy S względem pierwszej prostej otrzymując punkt B oraz względem drugiej prostej otrzymując punkt A.



..a teraz jeszcze raz przykład z tego trudniejsze

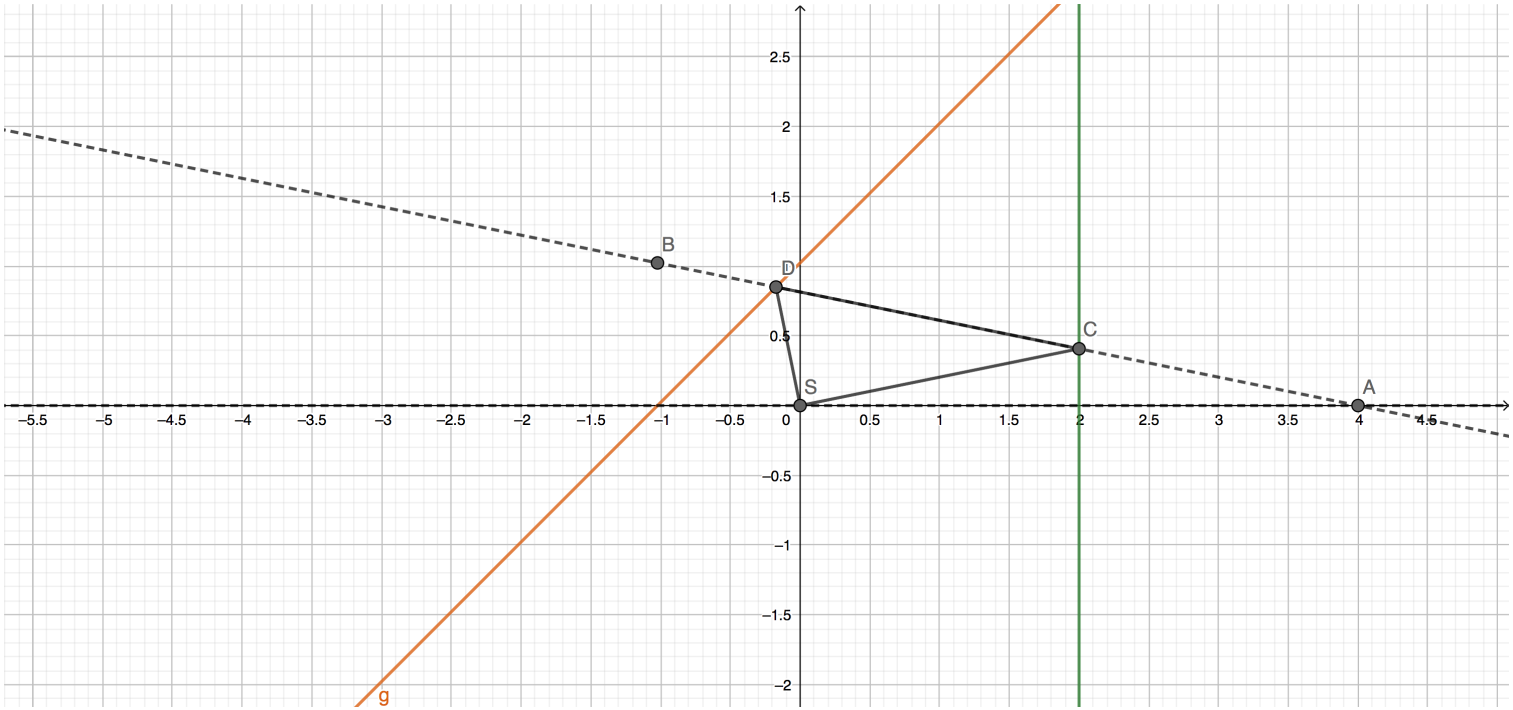
Odbijamy S względem pierwszej prostej otrzymując punkt B oraz względem drugiej prostej otrzymując punkt A.



Przecinamy prostą przechodzącą przez punkty A i B z naszymi prostymi. Otrzymane punkty C i D są tymi, które minimalizują sumaryczną długość trasy.

..a teraz jeszcze raz przykład z tego trudniejsze

Odbijamy S względem pierwszej prostej otrzymując punkt B oraz względem drugiej prostej otrzymując punkt A.



Przecinamy prostą przechodzącą przez punkty A i B z naszymi prostymi. Otrzymane punkty C i D są tymi, które minimalizują sumaryczną długość trasy.

...czy tak jest zawsze?

Całe rozwiązanie

Niech C będzie punktem, w którym odwiedzamy prostą f, a D punktem, w którym odwiedzamy g.

Całe rozwiązanie

Niech C będzie punktem, w którym odwiedzamy prostą f, a D punktem, w którym odwiedzamy g.

1. Być może $C = D$, trzeba więc sprawdzić punkt przecięcia obydwu prostych.

Całe rozwiązanie

Niech C będzie punktem, w którym odwiedzamy prostą f, a D punktem, w którym odwiedzamy g.

1. Być może $C = D$, trzeba więc sprawdzić punkt przecięcia obydwu prostych.
2. C, D i S są współliniowe. Załóżmy, że D jest między C a S, wtedy C jest rzutem S na f (bo w przeciwnym wypadku moglibyśmy lekko przesunąć C po f i poprawić rozwiązanie).

Całe rozwiązanie

Niech C będzie punktem, w którym odwiedzamy prostą f, a D punktem, w którym odwiedzamy g.

1. Być może $C = D$, trzeba więc sprawdzić punkt przecięcia obydwu prostych.
2. C, D i S są współliniowe. Załóżmy, że D jest między C a S, wtedy C jest rzutem S na f (bo w przeciwnym wypadku moglibyśmy lekko przesunąć C po f i poprawić rozwiązanie).
3. ...szukana droga "odbija" się po jednym razie od obydwu prostych tak jak w przykładzie.

Całe rozwiązanie

Niech C będzie punktem, w którym odwiedzamy prostą f, a D punktem, w którym odwiedzamy g.

1. Być może $C = D$, trzeba więc sprawdzić punkt przecięcia obydwu prostych.
2. C, D i S są współliniowe. Załóżmy, że D jest między C a S, wtedy C jest rzutem S na f (bo w przeciwnym wypadku moglibyśmy lekko przesunąć C po f i poprawić rozwiązanie).
3. ...szukana droga "odbija" się po jednym razie od obydwu prostych tak jak w przykładzie. Niech A będzie odbiciem symetrycznym S względem f, a B odbiciem symetrycznym S względem g.

Całe rozwiązanie

Niech C będzie punktem, w którym odwiedzamy prostą f, a D punktem, w którym odwiedzamy g.

1. Być może $C = D$, trzeba więc sprawdzić punkt przecięcia obydwu prostych.
2. C, D i S są współliniowe. Załóżmy, że D jest między C a S, wtedy C jest rzutem S na f (bo w przeciwnym wypadku moglibyśmy lekko przesunąć C po f i poprawić rozwiązanie).
3. ...szukana droga "odbija" się po jednym razie od obydwu prostych tak jak w przykładzie. Niech A będzie odbiciem symetrycznym S względem f, a B odbiciem symetrycznym S względem g. Wtedy punkty C oraz D leżą na prostej przechodzącej przez punkty A oraz B, możemy je więc wyznaczyć i wyliczyć długość otrzymanej trasy.

Całe rozwiązanie

Niech C będzie punktem, w którym odwiedzamy prostą f, a D punktem, w którym odwiedzamy g.

1. Być może $C = D$, trzeba więc sprawdzić punkt przecięcia obydwu prostych.
2. C, D i S są współliniowe. Załóżmy, że D jest między C a S, wtedy C jest rzutem S na f (bo w przeciwnym wypadku moglibyśmy lekko przesunąć C po f i poprawić rozwiązanie).
3. ...szukana droga "odbija" się po jednym razie od obydwu prostych tak jak w przykładzie. Niech A będzie odbiciem symetrycznym S względem f, a B odbiciem symetrycznym S względem g. Wtedy punkty C oraz D leżą na prostej przechodzącej przez punkty A oraz B, możemy je więc wyznaczyć i wyliczyć długość otrzymanej trasy.

Należy pamiętać, że dwie proste niekoniecznie się przecinają.

D: Droga

...po uważnym przeczytaniu treści:

D: Droga

...po uważnym przeczytaniu treści:

Zadanie (Autor: Michał Łowicki)

Zaimplementuj strukturę danych, która pozwoli na wykonywanie q operacji:

1. ustaw $A[x] = \max(A[x], m)$ dla każdego $x = a, a+1, \ldots, b$,
2. znajdź $\min\{A[a], A[a+1], \ldots, A[b]\}$.

D: Droga

...po uważnym przeczytaniu treści:

Zadanie (Autor: Michał Łowicki)

Zaimplementuj strukturę danych, która pozwoli na wykonywanie q operacji:

1. ustaw $A[x] = \max(A[x], m)$ dla każdego $x = a, a+1, \ldots, b$,
2. znajdź $\min\{A[a], A[a+1], \ldots, A[b]\}$.

1. $A[x]$ odpowiada ostatniej minucie, w której x-ty odcinek drogi jest czysty (być może $A[x]$ jest większe niż aktualny czas).

D: Droga

...po uważnym przeczytaniu treści:

Zadanie (Autor: Michał Łowicki)

Zaimplementuj strukturę danych, która pozwoli na wykonywanie q operacji:

1. ustaw $A[x] = \max(A[x], m)$ dla każdego $x = a, a+1, \ldots, b$,
2. znajdź $\min\{A[a], A[a+1], \ldots, A[b]\}$.

1. $A[x]$ odpowiada ostatniej minucie, w której x-ty odcinek drogi jest czysty (być może $A[x]$ jest większe niż aktualny czas).
2. Po znalezieniu minimum należy jeszcze wyznaczyć ile śniegu napadało od wyliczonej minuty.

D: Droga

...po uważnym przeczytaniu treści:

Zadanie (Autor: Michał Łowicki)

Zaimplementuj strukturę danych, która pozwoli na wykonywanie q operacji:

1. ustaw $A[x] = \max(A[x], m)$ dla każdego $x = a, a+1, \ldots, b$,
2. znajdź $\min\{A[a], A[a+1], \ldots, A[b]\}$.

1. $A[x]$ odpowiada ostatniej minucie, w której x-ty odcinek drogi jest czysty (być może $A[x]$ jest większe niż aktualny czas).
2. Po znalezieniu minimum należy jeszcze wyznaczyć ile śniegu napadało od wyliczonej minuty.
3. Kompresujemy współrzędne x tak, żeby tablica A miała długość $\leqslant 600\,000$.

D: Droga

...po uważnym przeczytaniu treści:

Zadanie (Autor: Michał Łowicki)

Zaimplementuj strukturę danych, która pozwoli na wykonywanie q operacji:

1. ustaw $A[x] = \max(A[x], m)$ dla każdego $x = a, a+1, \ldots, b$,
2. znajdź $\min\{A[a], A[a+1], \ldots, A[b]\}$.

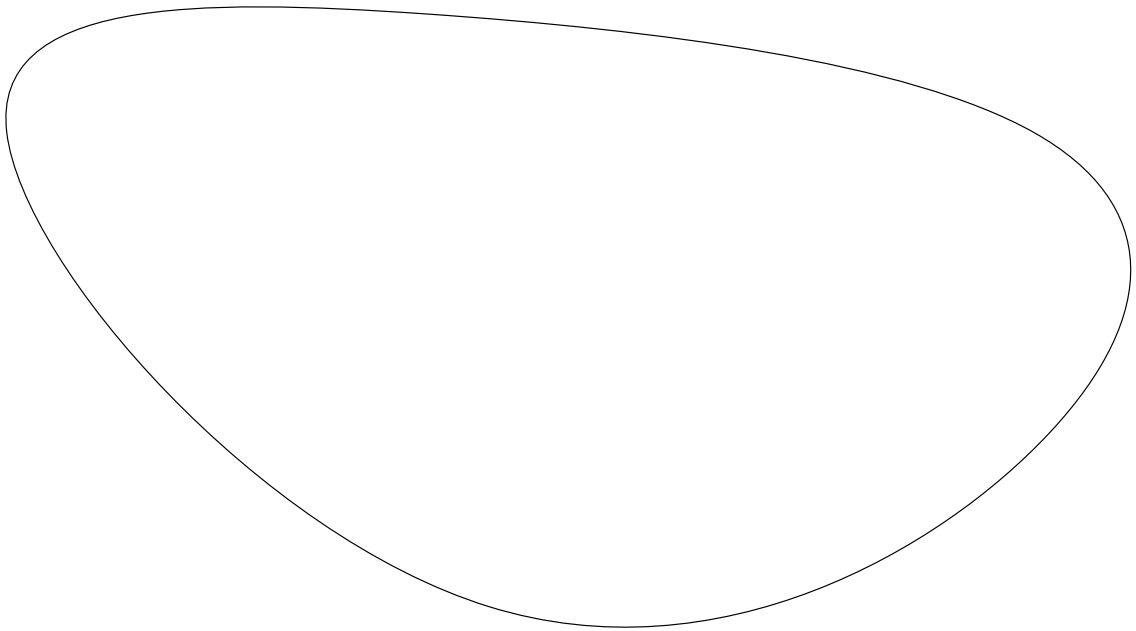
1. $A[x]$ odpowiada ostatniej minucie, w której x-ty odcinek drogi jest czysty (być może $A[x]$ jest większe niż aktualny czas).
2. Po znalezieniu minimum należy jeszcze wyznaczyć ile śniegu napadało od wyliczonej minuty.
3. Kompresujemy współrzędne x tak, żeby tablica A miała długość $\leqslant 600\,000$.
4. Budujemy drzewo przedziałowe przedział-przedział z leniwą propagacją, każda operacja zajmuje $O(\log q)$.

E: Ewaluacja

Zadanie (Autor: Jakub Tarnawski)

Dla danego grafu nieskierowanego z wagami na krawędziach wyznacz, dla każdej krawędzi e, największe x takie, że po zmianie kosztu e na x ta krawędź należy do **jakiegoś** MST.

Zacznijmy od znalezienia dowolnego MST, nazwijmy je T.

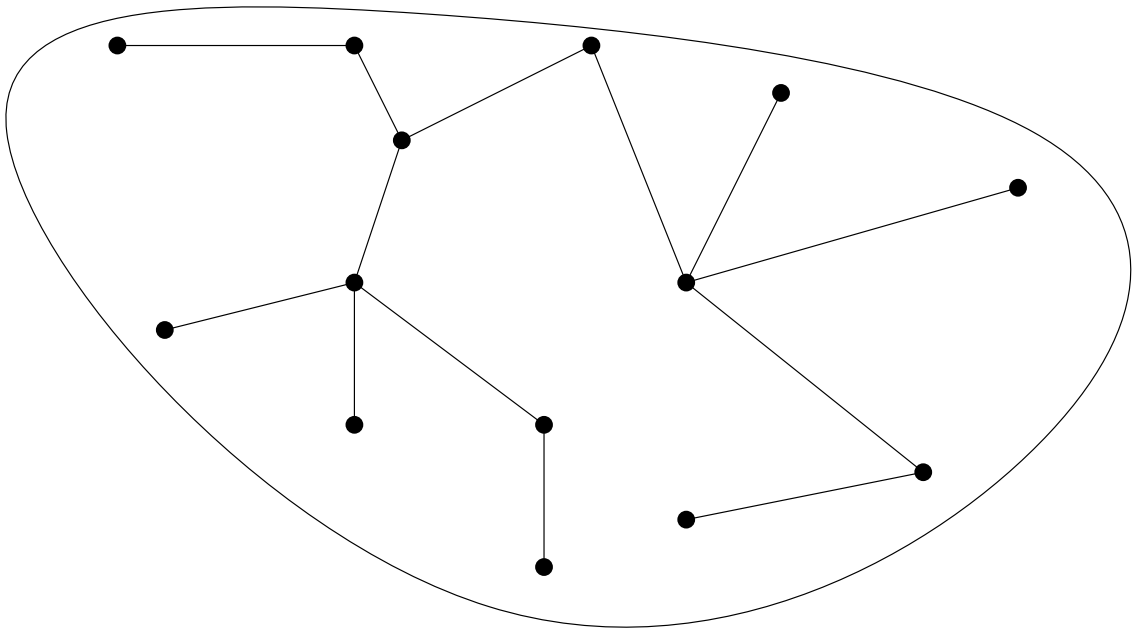


E: Ewaluacja

Zadanie (Autor: Jakub Tarnawski)

Dla danego grafu nieskierowanego z wagami na krawędziach wyznacz, dla każdej krawędzi e, największe x takie, że po zmianie kosztu e na x ta krawędź należy do **jakiegoś** MST.

Zacznijmy od znalezienia dowolnego MST, nazwijmy je T.

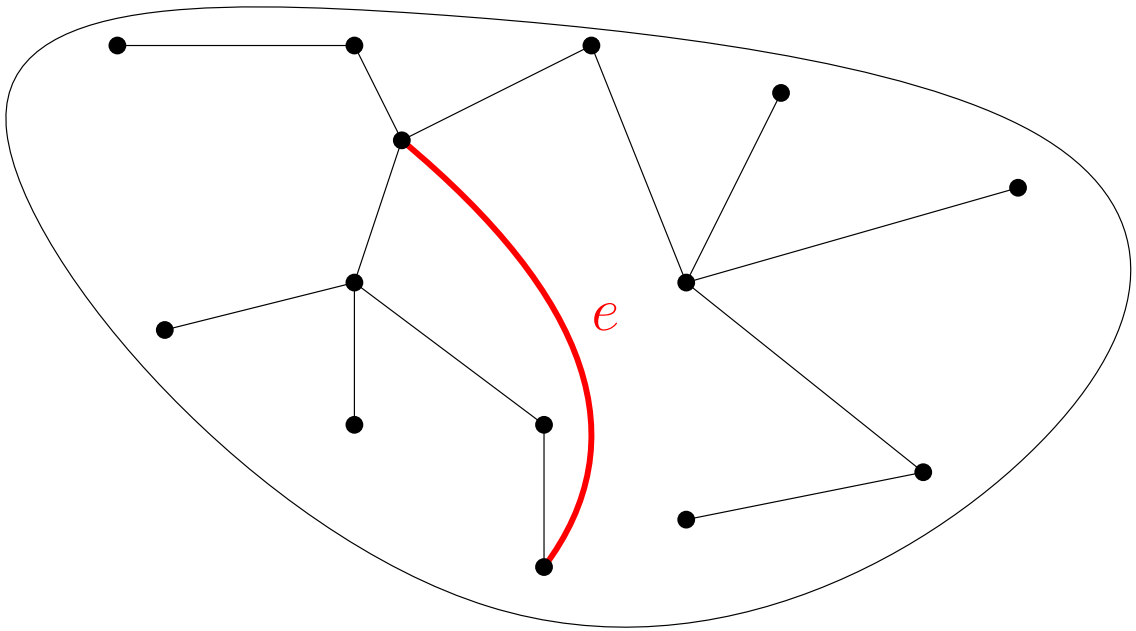


E: Ewaluacja

Zadanie (Autor: Jakub Tarnawski)

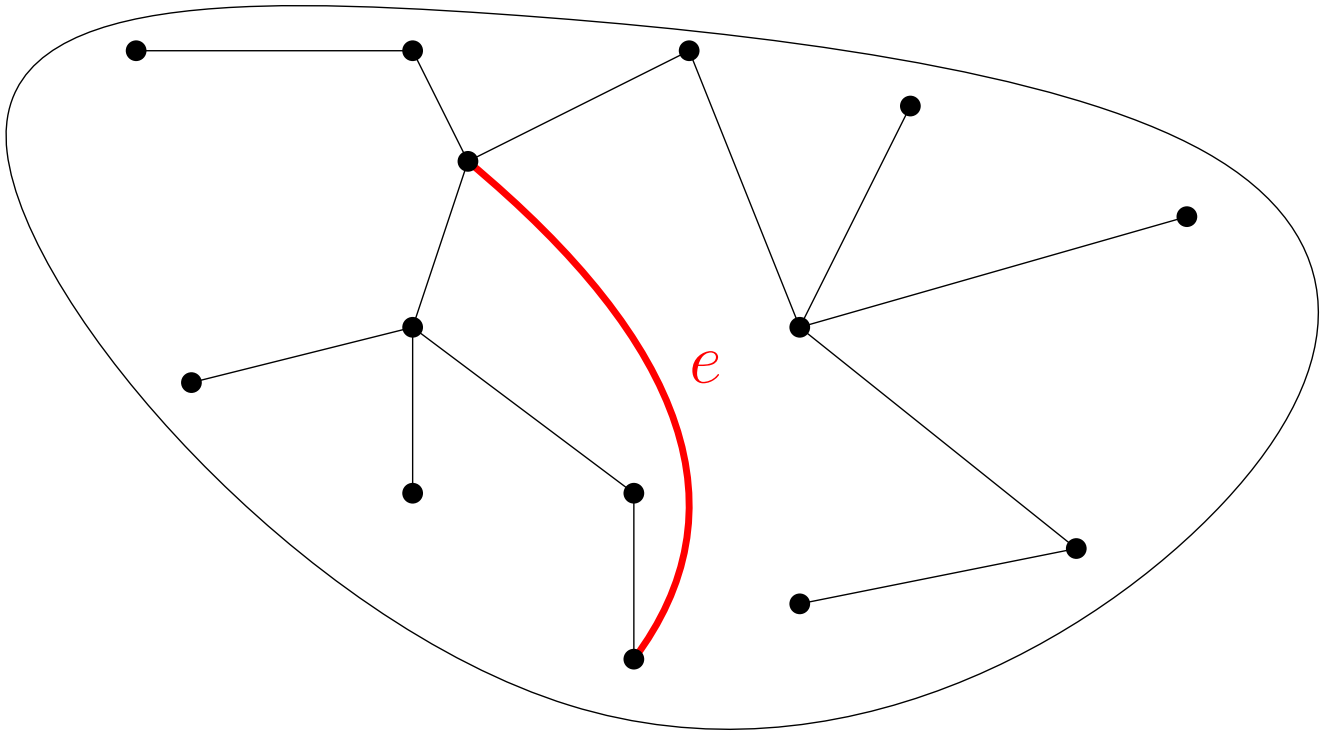
Dla danego grafu nieskierowanego z wagami na krawędziach wyznacz, dla każdej krawędzi e, największe x takie, że po zmianie kosztu e na x ta krawędź należy do **jakiegoś** MST.

Zacznijmy od znalezienia dowolnego MST, nazwijmy je T.



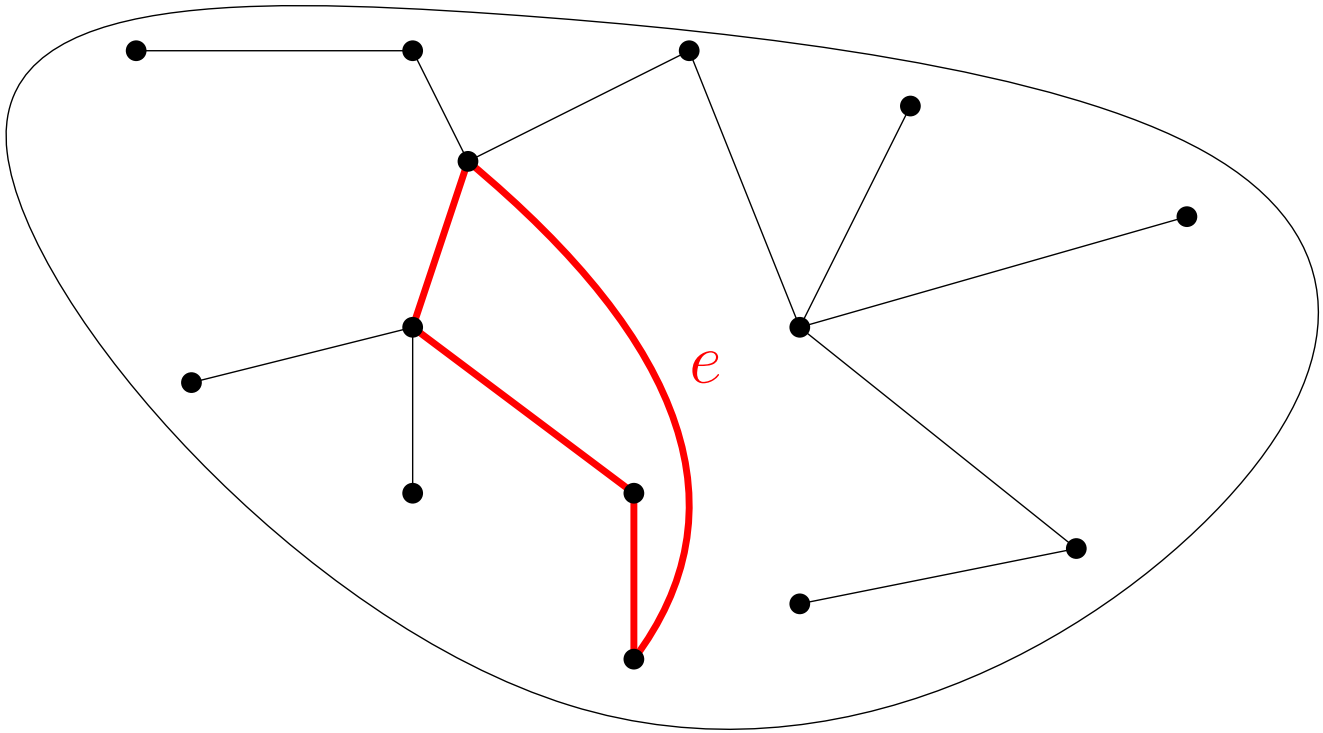
Jak wyznaczyć odpowiedź dla $e \notin T$?

$e \notin T$



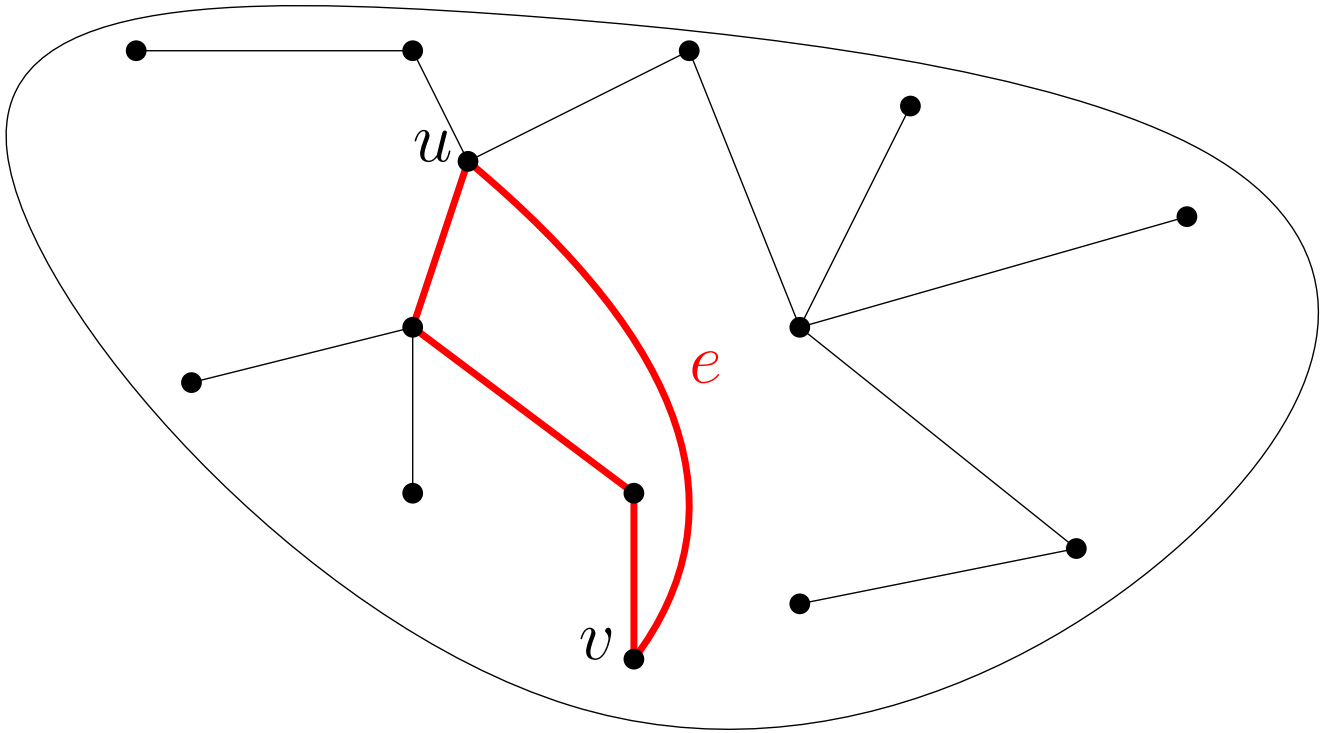
Jeśli e jest ściśle cięższa niż wszystkie inne krawędzie na pewnym cyklu C takim, że $e \in C$, to e nie należy do *żadnego* MST.

$e \notin T$



Jeśli e jest ściśle cięższa niż wszystkie inne krawędzie na pewnym cyklu C takim, że $e \in C$, to e nie należy do *żadnego* MST.

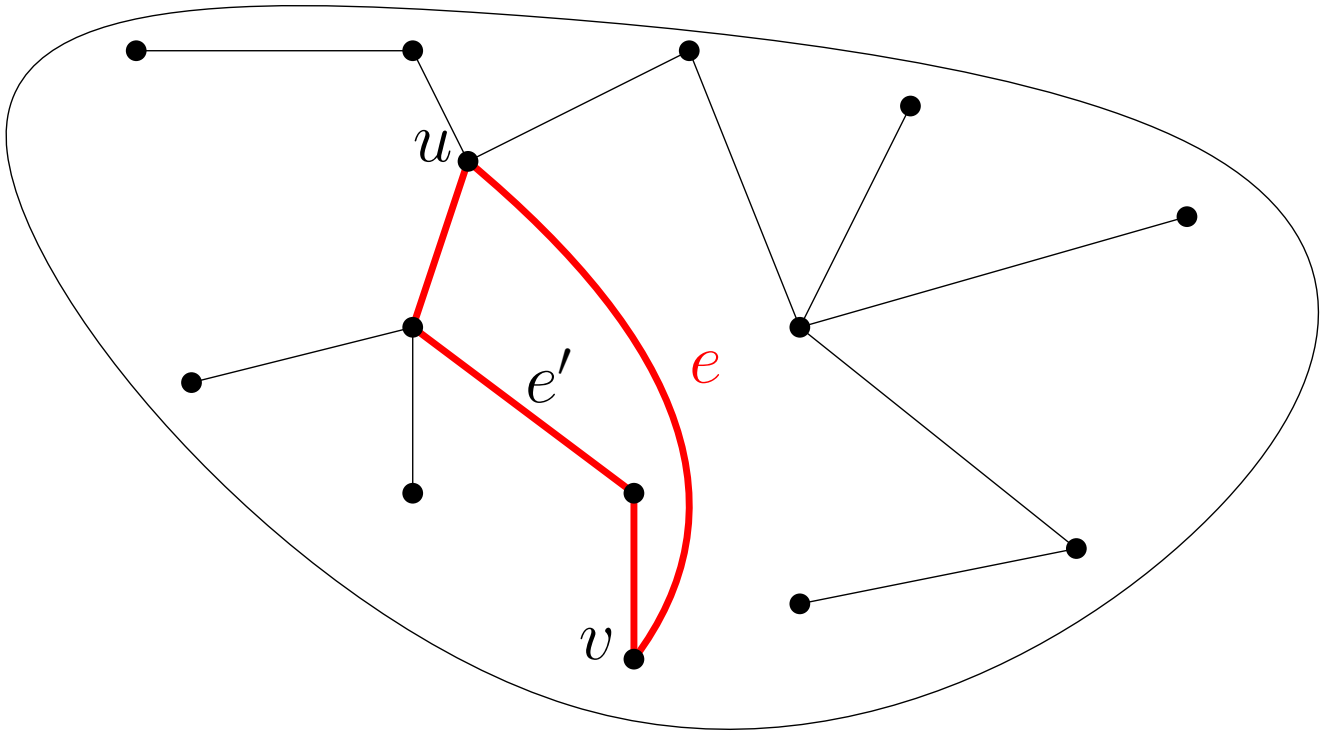
$e \notin T$



Jeśli e jest ściśle cięższa niż wszystkie inne krawędzie na pewnym cyklu C takim, że $e \in C$, to e nie należy do *żadnego* MST.

Widać też, że jeśli $e = (u, v)$ jest tak samo ciężko jak najcięższa krawędź e' na ścieżce łączącej u i v w T, to możemy podmienić e' na e i otrzymać inne MST.

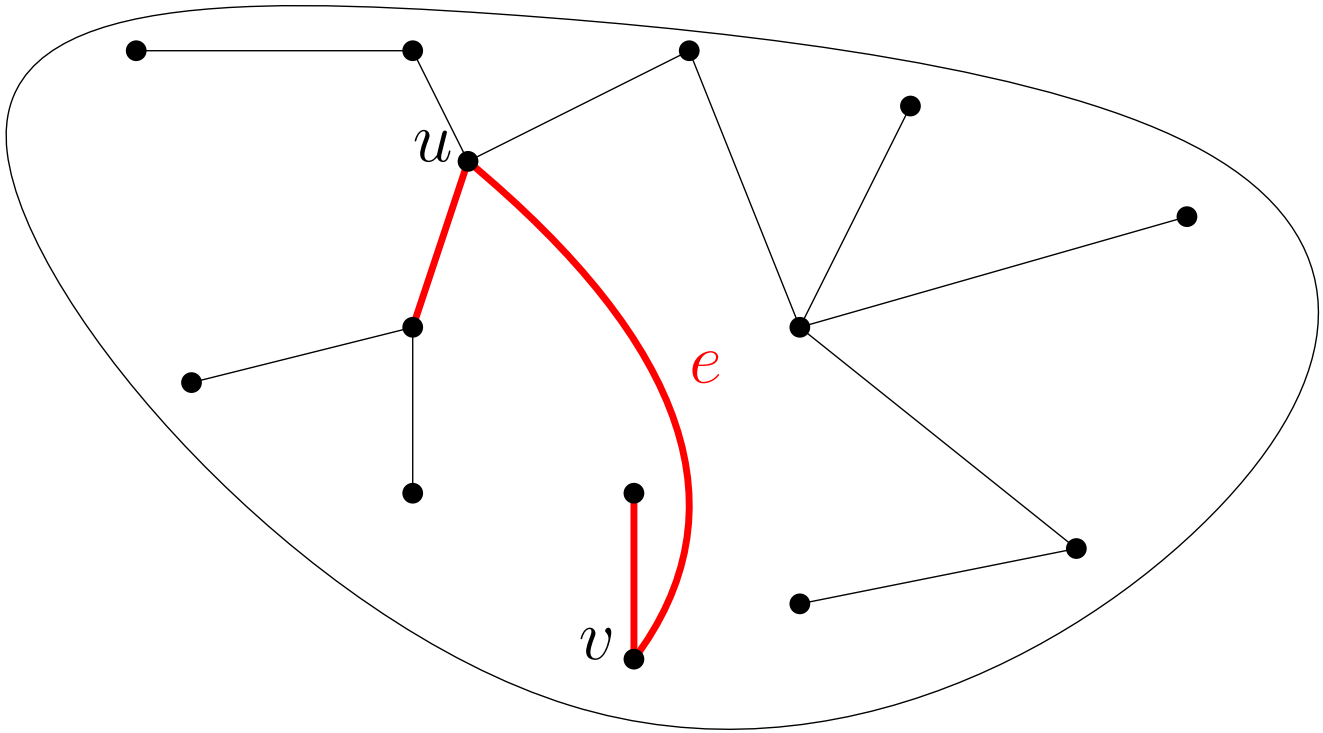
$e \notin T$



Jeśli e jest ściśle cięższa niż wszystkie inne krawędzie na pewnym cyklu C takim, że $e \in C$, to e nie należy do *żadnego* MST.

Widać też, że jeśli $e = (u, v)$ jest tak samo ciężko jak najcięższa krawędź e' na ścieżce łączącej u i v w T, to możemy podmienić e' na e i otrzymać inne MST.

$e \notin T$

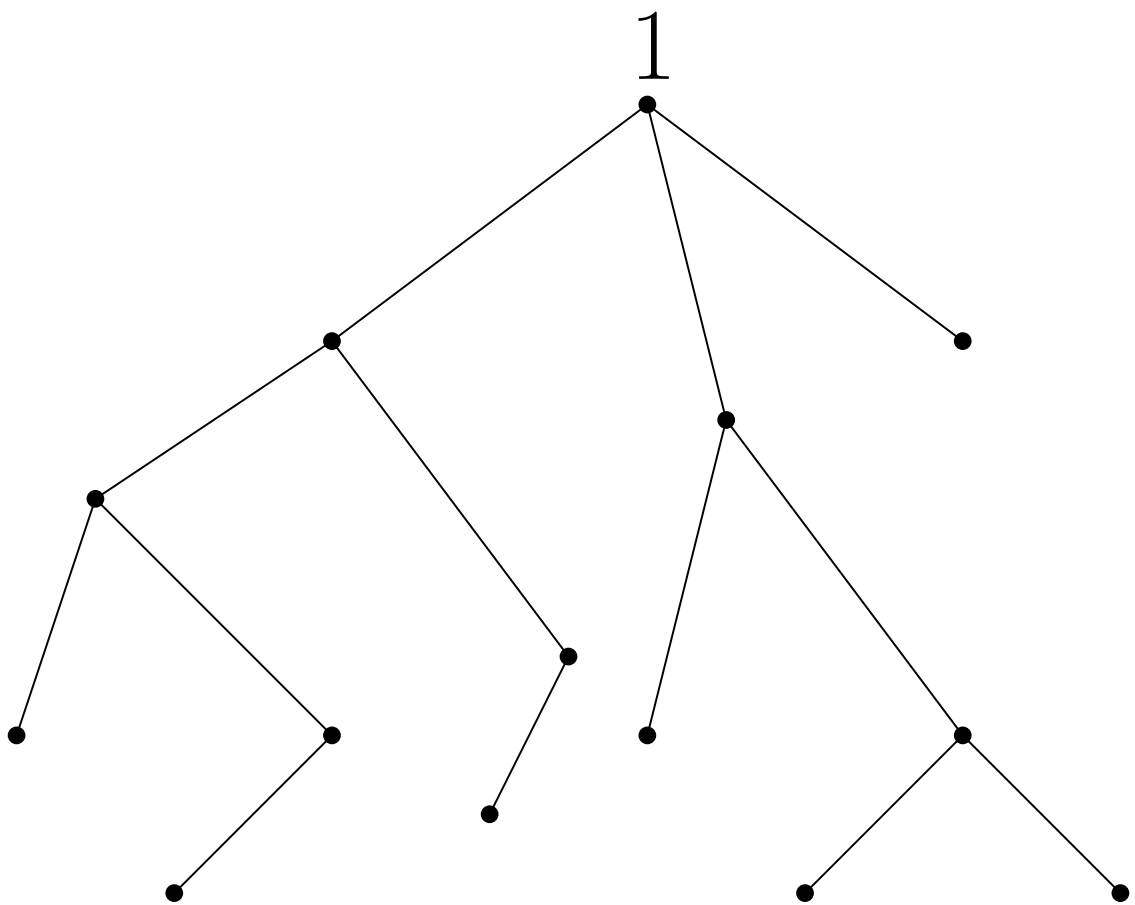


Jeśli e jest ściśle cięższa niż wszystkie inne krawędzie na pewnym cyklu C takim, że $e \in C$, to e nie należy do *żadnego* MST.

Widać też, że jeśli $e = (u, v)$ jest tak samo ciężko jak najcięższa krawędź e' na ścieżce łączącej u i v w T, to możemy podmienić e' na e i otrzymać inne MST.

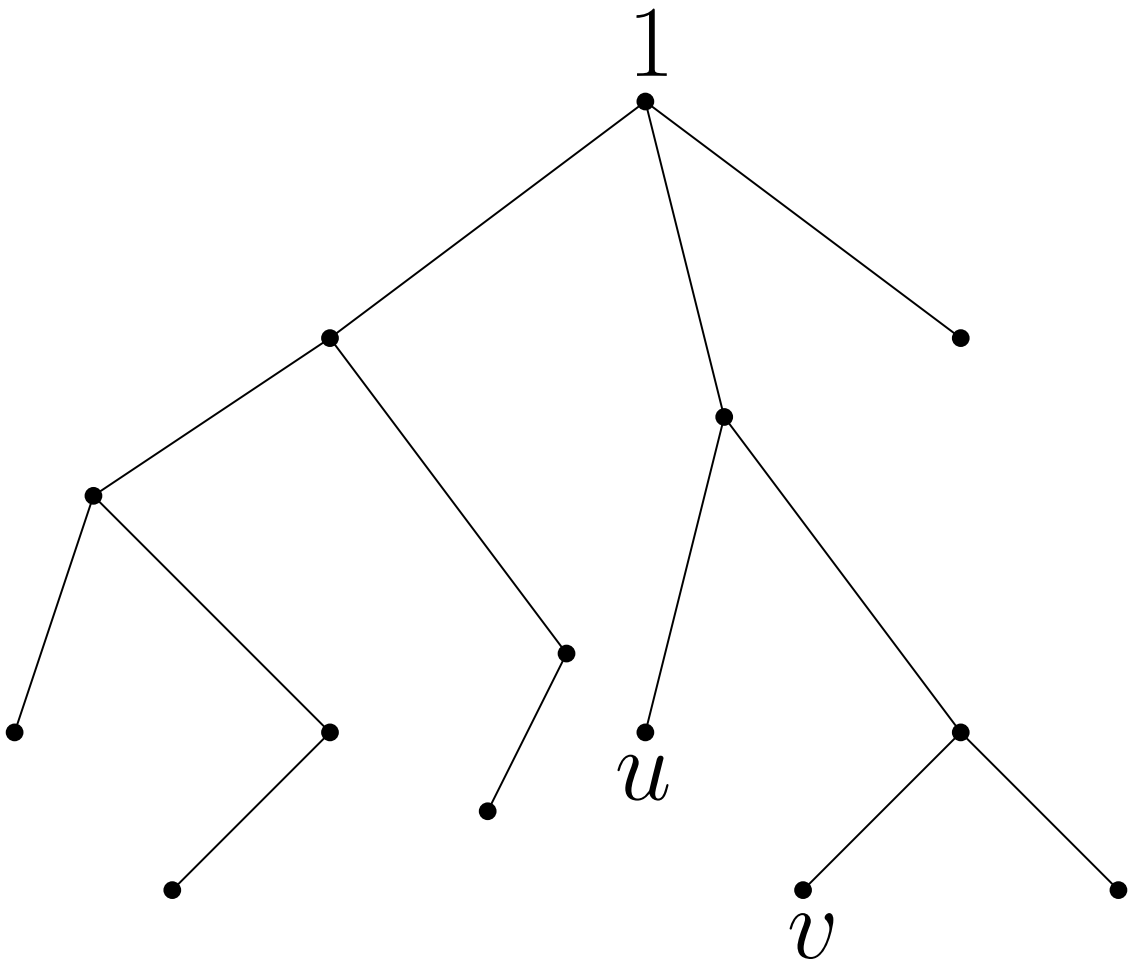
$e \notin T$

Czyli dla każdej krawędzi $e \notin T$ chcemy policzyć minimum na ścieżce łączącej u i v w T. Ukorzeńmy T w wierzchołku 1.



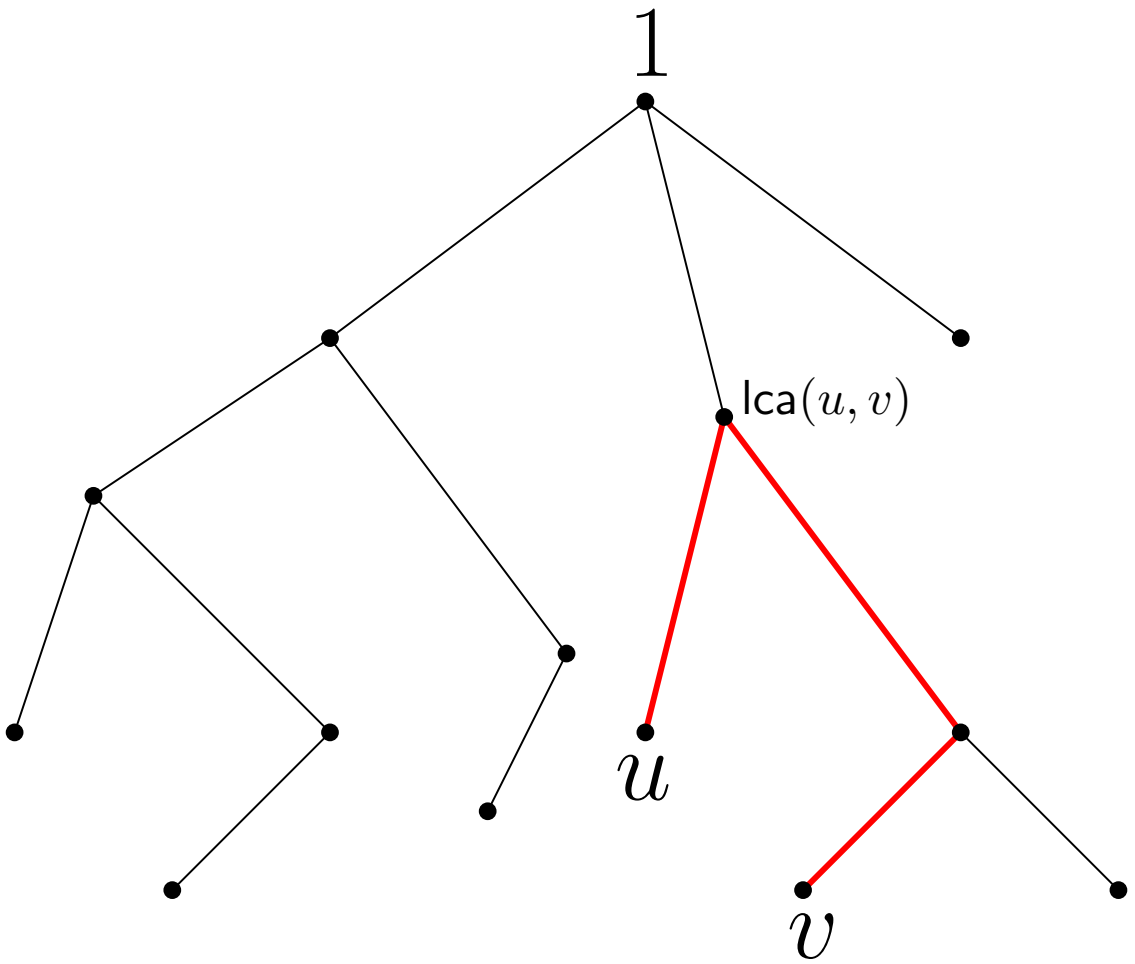
$e \notin T$

Czyli dla każdej krawędzi $e \notin T$ chcemy policzyć minimum na ścieżce łączącej u i v w T. Ukorzeńmy T w wierzchołku 1.



$e \notin T$

Czyli dla każdej krawędzi $e \notin T$ chcemy policzyć minimum na ścieżce łączącej u i v w T. Ukorzeńmy T w wierzchołku 1.



$e \notin T$

Czyli dla każdej krawędzi $e \notin T$ chcemy policzyć minimum na ścieżce łączącej u i v w T. Ukorzeńmy T w wierzchołku 1.

Wystarczy umieć liczyć:

1. $\text{lca}(u, v)$,
2. minimum na ścieżce od u do $\text{lca}(u, v)$,
3. minimum na ścieżce od v do $\text{lca}(u, v)$.

$e \notin T$

Czyli dla każdej krawędzi $e \notin T$ chcemy policzyć minimum na ścieżce łączącej u i v w T. Ukorzeńmy T w wierzchołku 1.

Wystarczy umieć liczyć:

1. $\text{lca}(u, v)$,
2. minimum na ścieżce od u do $\text{lca}(u, v)$,
3. minimum na ścieżce od v do $\text{lca}(u, v)$.

Algorytm odpowiadający offline na zapytania o lca przy użyciu struktury union-find może być zmodyfikowany, aby znajdować takie minima.

$e \notin T$

Czyli dla każdej krawędzi $e \notin T$ chcemy policzyć minimum na ścieżce łączącej u i v w T. Ukorzeńmy T w wierzchołku 1.

Wystarczy umieć liczyć:

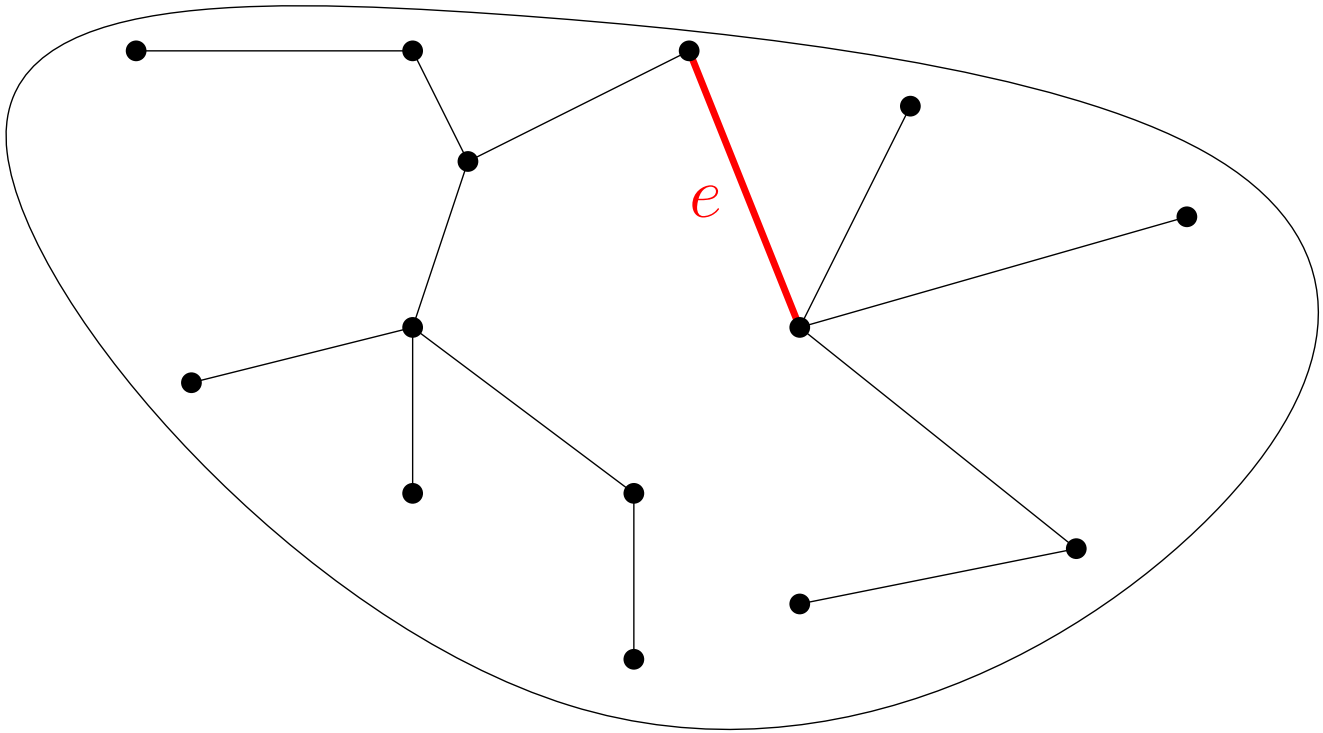
1. $\text{lca}(u, v)$,
2. minimum na ścieżce od u do $\text{lca}(u, v)$,
3. minimum na ścieżce od v do $\text{lca}(u, v)$.

Algorytm odpowiadający offline na zapytania o lca przy użyciu struktury union-find może być zmodyfikowany, aby znajdować takie minima.

Złożoność

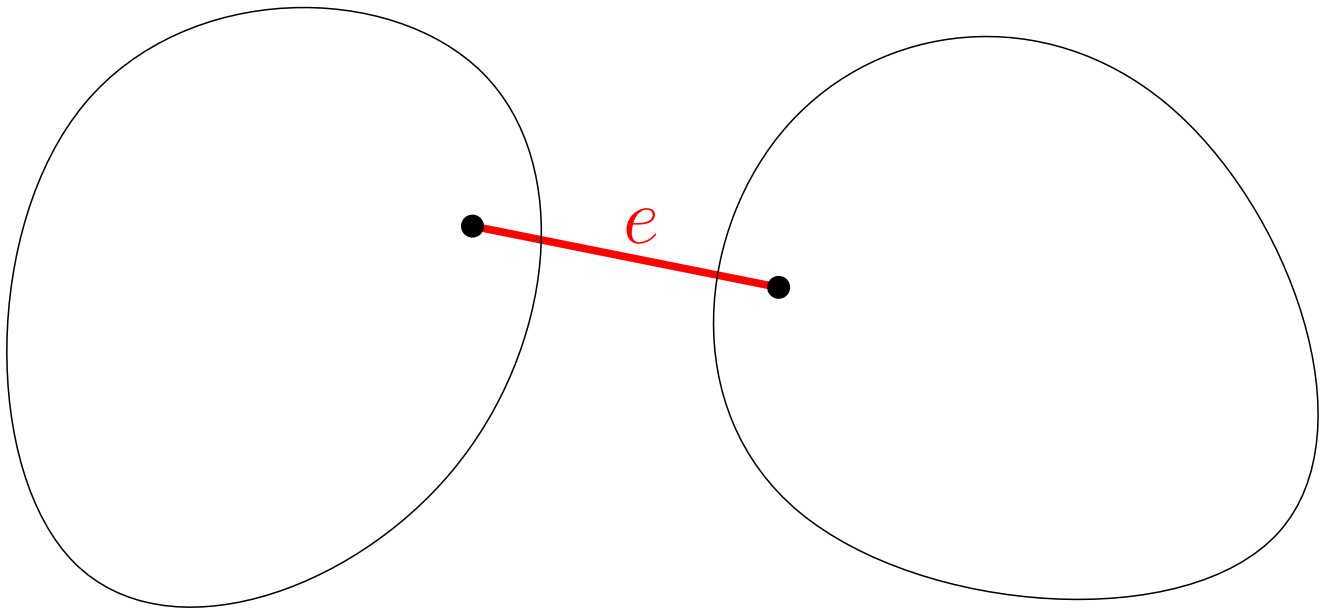
Sortowanie krawędzi + $O(m\alpha(m, n))$

$e \in T$



Jeśli e jest jedną z najlżejszych krawędzi w pewnym cięciu $(X, V \setminus X)$ oraz $u \in X$ oraz $v \notin X$, to e należy do *jakiegoś* MST.

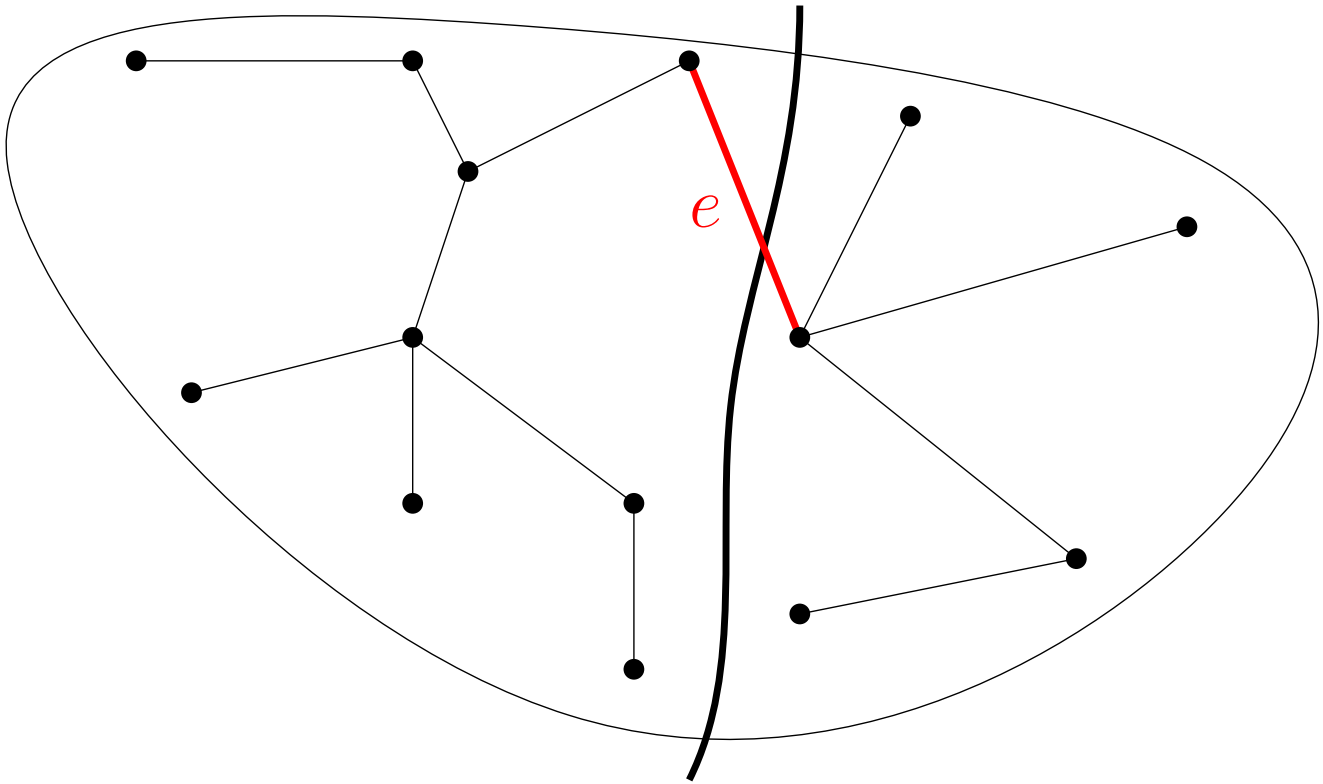
$e \in T$



Jeśli e jest jedną z najlżejszych krawędzi w pewnym cięciu $(X, V \setminus X)$ oraz $u \in X$ oraz $v \notin X$, to e należy do *jakiegoś* MST.

Usunięcie e z T definiuje pewne cięcie $(X, V \setminus X)$. Jeśli w tym cięciu istnieje ściśle tańsza krawędź e' to możemy podmienić e na e' i otrzymać tańsze drzewo rozpinające.

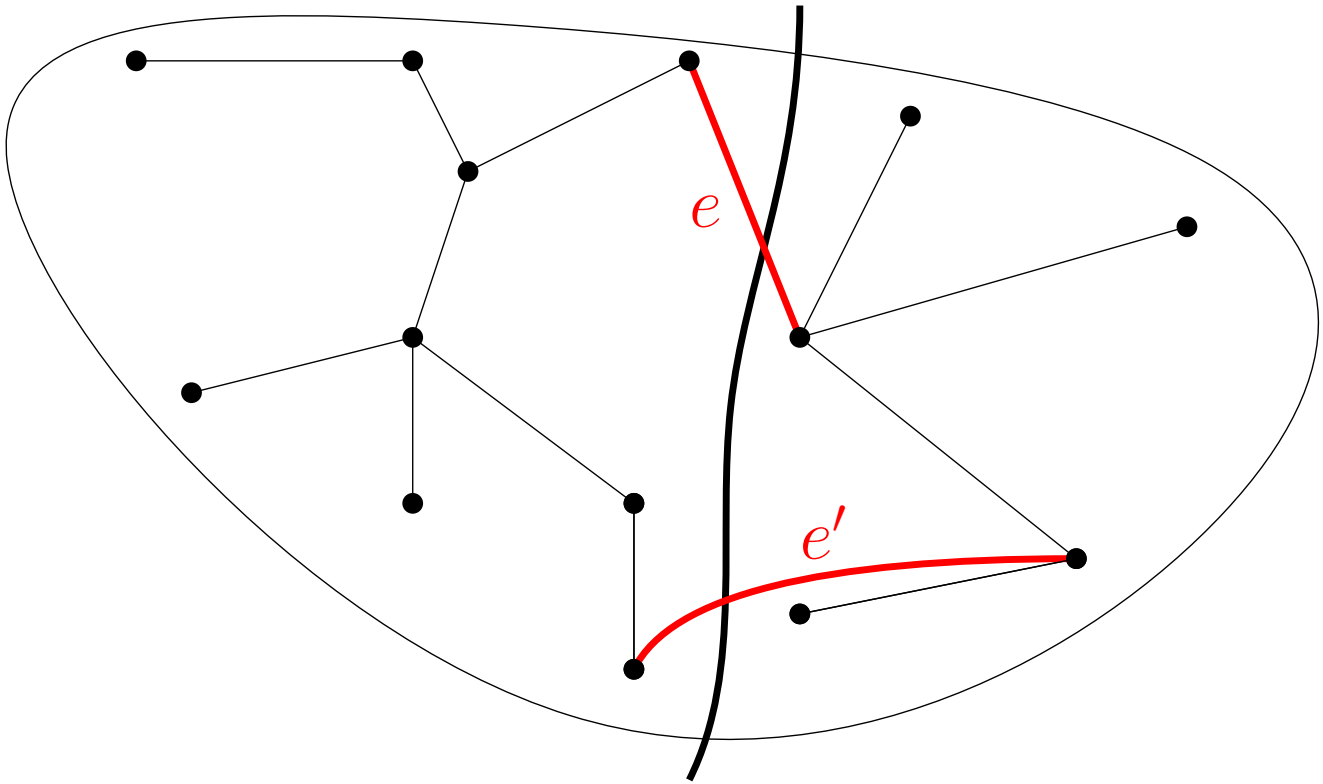
$e \in T$



Jeśli e jest jedną z najlżejszych krawędzi w pewnym cięciu $(X, V \setminus X)$ oraz $u \in X$ oraz $v \notin X$, to e należy do *jakiegoś* MST.

Usunięcie e z T definiuje pewne cięcie $(X, V \setminus X)$. Jeśli w tym cięciu istnieje ściśle tańsza krawędź e' to możemy podmienić e na e' i otrzymać tańsze drzewo rozpinające.

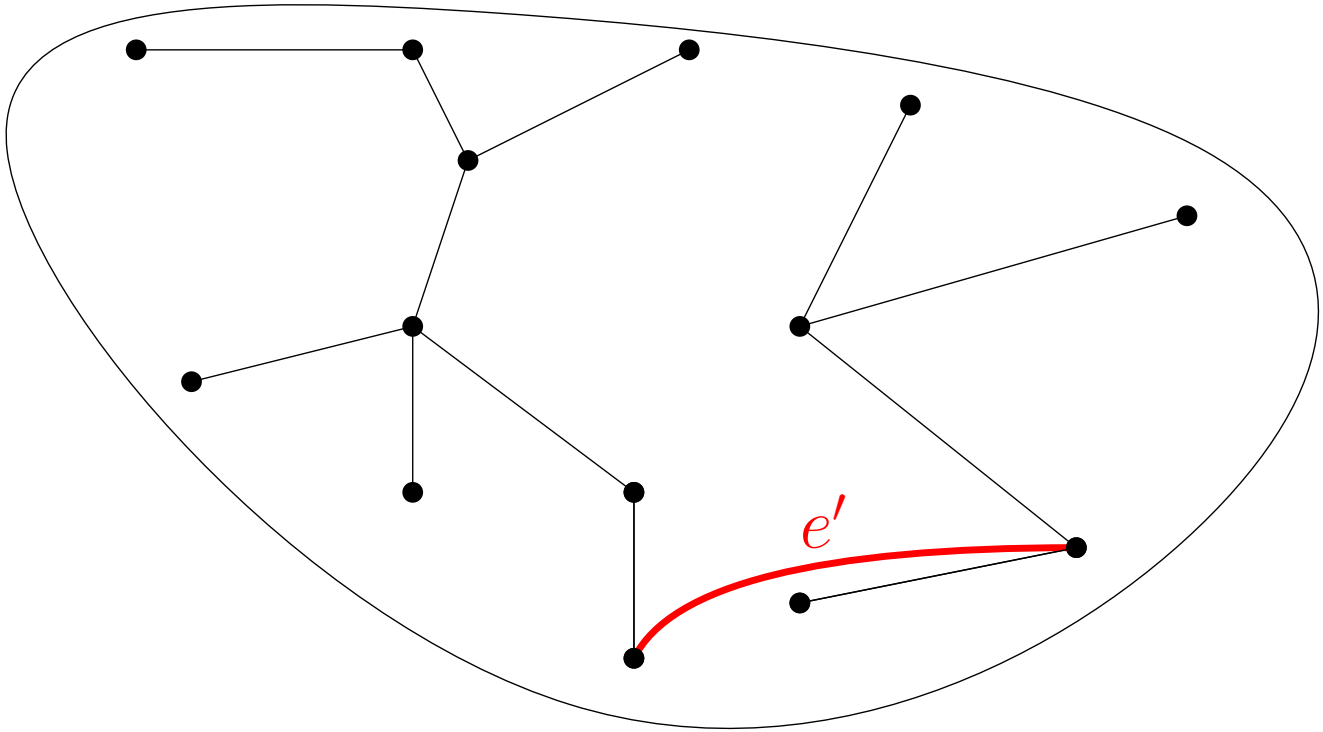
$e \in T$



Jeśli e jest jedną z najlżejszych krawędzi w pewnym cięciu $(X, V \setminus X)$ oraz $u \in X$ oraz $v \notin X$, to e należy do *jakiegoś* MST.

Usunięcie e z T definiuje pewne cięcie $(X, V \setminus X)$. Jeśli w tym cięciu istnieje ściśle tańsza krawędź e' to możemy podmienić e na e' i otrzymać tańsze drzewo rozpinające.

$e \in T$



Jeśli e jest jedną z najlżejszych krawędzi w pewnym cięciu $(X, V \setminus X)$ oraz $u \in X$ oraz $v \notin X$, to e należy do *jakiegoś* MST.

Usunięcie e z T definiuje pewne cięcie $(X, V \setminus X)$. Jeśli w tym cięciu istnieje ściśle tańsza krawędź e' to możemy podmienić e na e' i otrzymać tańsze drzewo rozpinające.

$e \in T$

Czyli dla każdej krawędzi $e \in T$ chcemy policzyć minimum po krawędziach spoza T o dokładnie jednym końcu w X, gdzie $(X, V \setminus X)$ jest cięciem definiowanym przez e.

$e \in T$

Czyli dla każdej krawędzi $e \in T$ chcemy policzyć minimum po krawędziach spoza T o dokładnie jednym końcu w X, gdzie $(X, V \setminus X)$ jest cięciem definiowanym przez e.

Krawędź $e' = (u, v) \notin T$ powinna być uwzględniona dla każdej krawędzi $e \in T$ takiej, że e leży na ścieżce łączącej u z v w T.

$e \in T$

Czyli dla każdej krawędzi $e \in T$ chcemy policzyć minimum po krawędziach spoza T o dokładnie jednym końcu w X, gdzie $(X, V \setminus X)$ jest cięciem definiowanym przez e.

Krawędź $e' = (u, v) \notin T$ powinna być uwzględniona dla każdej krawędzi $e \in T$ takiej, że e leży na ścieżce łączącej u z v w T.

1. Rozważamy krawędzie $e' \notin T$ w kolejności niemalejących wag.

$e \in T$

Czyli dla każdej krawędzi $e \in T$ chcemy policzyć minimum po krawędziach spoza T o dokładnie jednym końcu w X, gdzie $(X, V \setminus X)$ jest cięciem definiowanym przez e.

Krawędź $e' = (u, v) \notin T$ powinna być uwzględniona dla każdej krawędzi $e \in T$ takiej, że e leży na ścieżce łączącej u z v w T.

1. Rozważamy krawędzie $e' \notin T$ w kolejności niemalejących wag.
2. Utrzymujemy spójne składowe rozpięte przez krawędzie $e \in T$, dla których znamy już odpowiedź, w strukturze union-find.

$e \in T$

Czyli dla każdej krawędzi $e \in T$ chcemy policzyć minimum po krawędziach spoza T o dokładnie jednym końcu w X, gdzie $(X, V \setminus X)$ jest cięciem definiowanym przez e.

Krawędź $e' = (u, v) \notin T$ powinna być uwzględniona dla każdej krawędzi $e \in T$ takiej, że e leży na ścieżce łączącej u z v w T.

1. Rozważamy krawędzie $e' \notin T$ w kolejności niemalejących wag.
2. Utrzymujemy spójne składowe rozpięte przez krawędzie $e \in T$, dla których znamy już odpowiedź, w strukturze union-find.
3. To pozwala nam na szybkie wygenerowanie wszystkich krawędzi $e \in T$ leżących na ścieżce łączącej u z v, dla których nie znamy jeszcze odpowiedzi idąc od u do $\mathrm{lca}(u, v)$ (a potem od v do $\mathrm{lca}(u, v)$) i przeskakując maksymalne ciągi krawędzi ze znaną odpowiedzią.

$e \in T$

Czyli dla każdej krawędzi $e \in T$ chcemy policzyć minimum po krawędziach spoza T o dokładnie jednym końcu w X, gdzie $(X, V \setminus X)$ jest cięciem definiowanym przez e.

Krawędź $e' = (u, v) \notin T$ powinna być uwzględniona dla każdej krawędzi $e \in T$ takiej, że e leży na ścieżce łączącej u z v w T.

1. Rozważamy krawędzie $e' \notin T$ w kolejności niemalejących wag.
2. Utrzymujemy spójne składowe rozpięte przez krawędzie $e \in T$, dla których znamy już odpowiedź, w strukturze union-find.
3. To pozwala nam na szybkie wygenerowanie wszystkich krawędzi $e \in T$ leżących na ścieżce łączącej u z v, dla których nie znamy jeszcze odpowiedzi idąc od u do $\mathrm{lca}(u, v)$ (a potem od v do $\mathrm{lca}(u, v)$) i przeskakując maksymalne ciągi krawędzi ze znaną odpowiedzią.

Złożoność

Sortowanie krawędzi + $O(m\alpha(m, n))$

M: Magiczny labirynt

Zadanie (Autor: Paweł Gawrychowski)

Dany jest acykliczny graf planarny z wyróżnionym źródłem s i ujściem t, które są przyległe do ściany zewnętrznej. Ile jest par wierzchołków (u, v) takich, że istnieje ścieżka z s do u, z u do v, oraz z v do t? (być może $u = v$)

M: Magiczny labirynt

Zadanie (Autor: Paweł Gawrychowski)

Dany jest acykliczny graf planarny z wyróżnionym źródłem s i ujściem t, które są przyległe do ściany zewnętrznej. Ile jest par wierzchołków (u, v) takich, że istnieje ścieżka z s do u, z u do v, oraz z v do t? (być może $u = v$)

1. Usuńmy z grafu wierzchołki, do których nie da się dojść z s, a także te, z których nie da się dojść do t.

M: Magiczny labirynt

Zadanie (Autor: Paweł Gawrychowski)

Dany jest acykliczny graf planarny z wyróżnionym źródłem s i ujściem t, które są przyległe do ściany zewnętrznej. Ile jest par wierzchołków (u, v) takich, że istnieje ścieżka z s do u, z u do v, oraz z v do t? (być może $u = v$)

1. Usuńmy z grafu wierzchołki, do których nie da się dojść z s, a także te, z których nie da się dojść do t.
2. Teraz musimy policzyć pary wierzchołków (u, v) takie, że da się dojść z u do v.

M: Magiczny labirynt

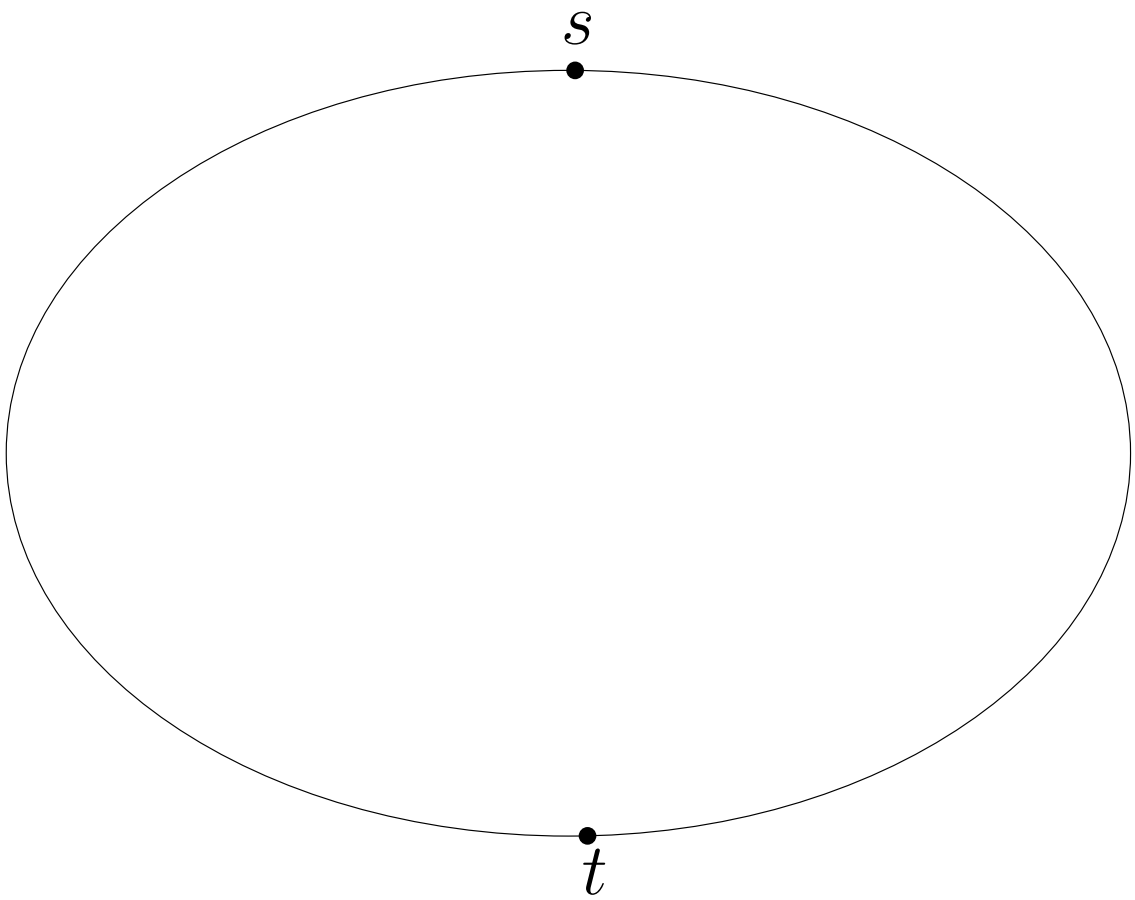
Zadanie (Autor: Paweł Gawrychowski)

Dany jest acykliczny graf planarny z wyróżnionym źródłem s i ujściem t, które są przyległe do ściany zewnętrznej. Ile jest par wierzchołków (u, v) takich, że istnieje ścieżka z s do u, z u do v, oraz z v do t? (być może $u = v$)

1. Usuńmy z grafu wierzchołki, do których nie da się dojść z s, a także te, z których nie da się dojść do t.
2. Teraz musimy policzyć pary wierzchołków (u, v) takie, że da się dojść z u do v.
3. Ponumerujmy wierzchołki w kolejności post-order. Jeśli istnieje ścieżka z u do v to $\text{post}[v] \leqslant \text{post}[u]$.

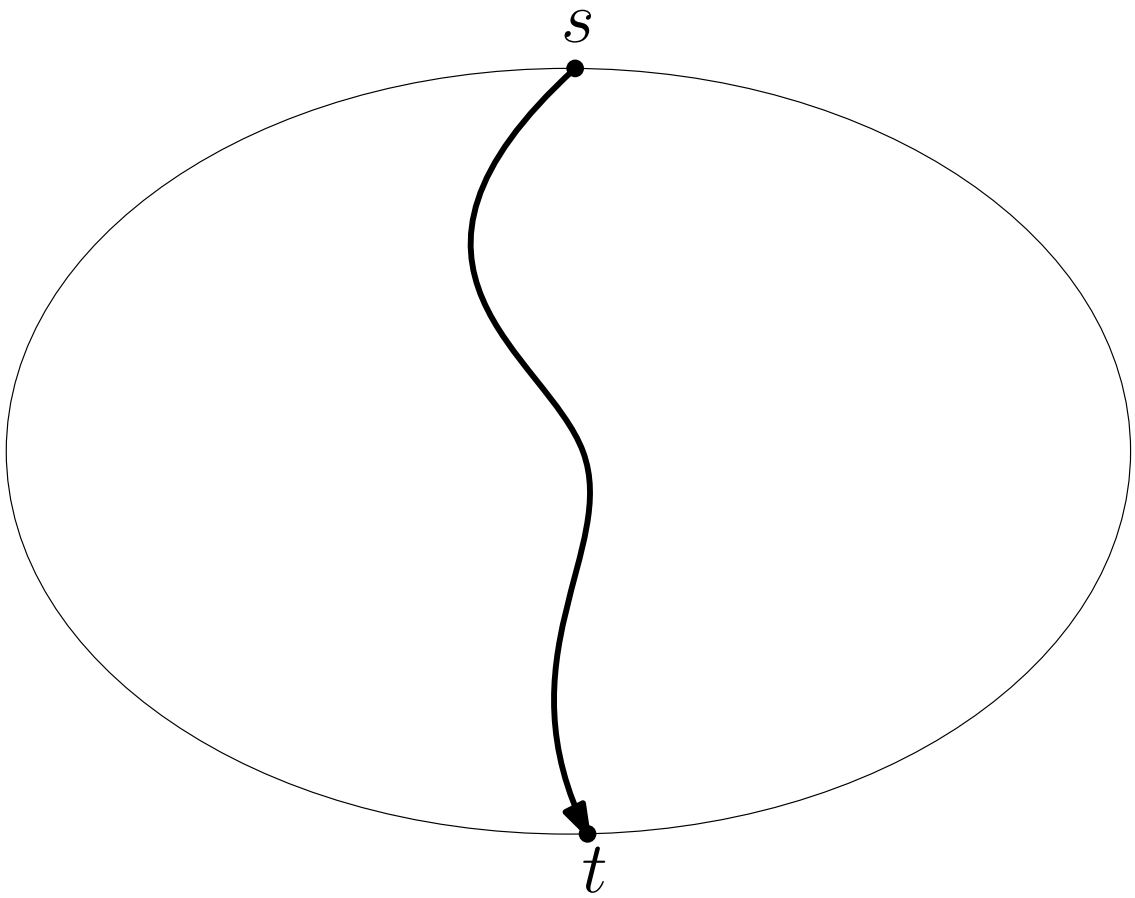
M: Magiczny labirynt

Narysujmy graf tak, aby wierzchołek s był na samej górze, wierzchołek t na samym dole, a każda krawędź prowadziła w dół.



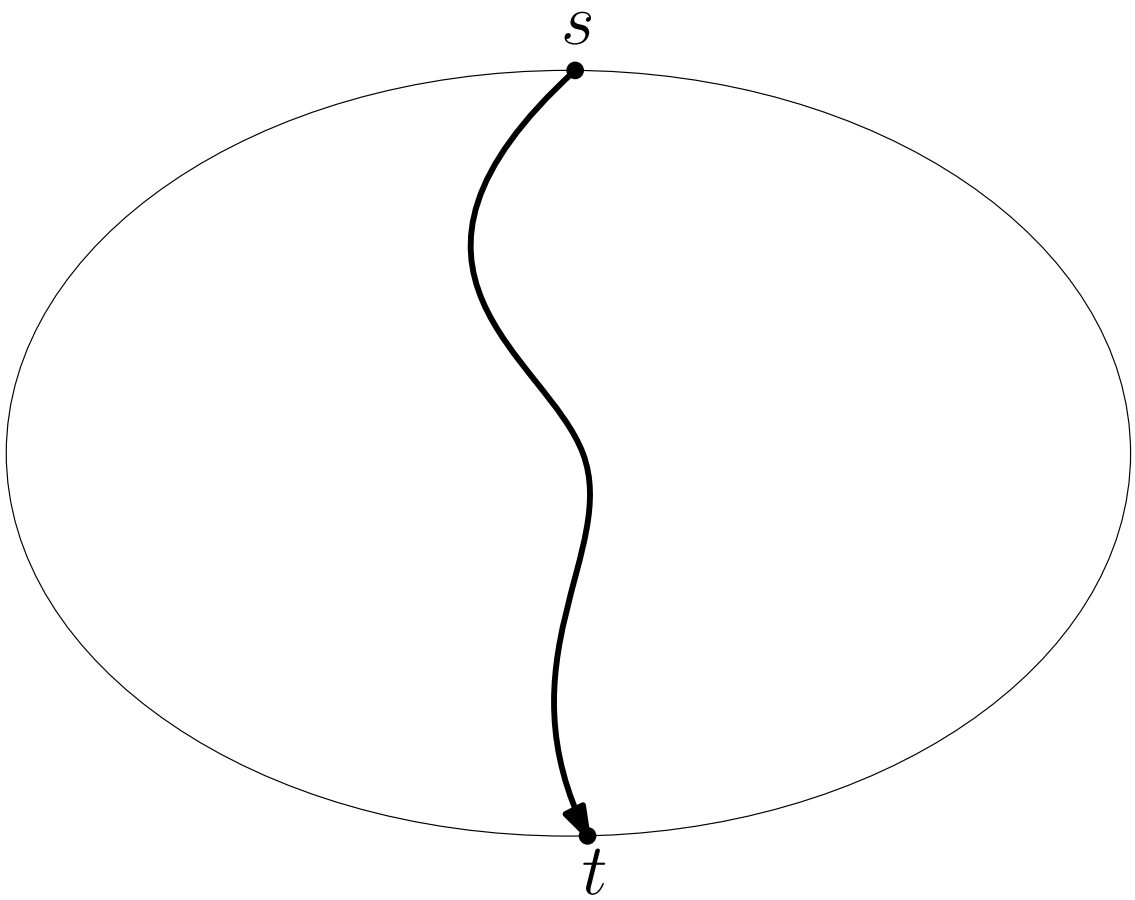
M: Magiczny labirynt

Narysujmy graf tak, aby wierzchołek s był na samej górze, wierzchołek t na samym dole, a każda krawędź prowadziła w dół.



M: Magiczny labirynt

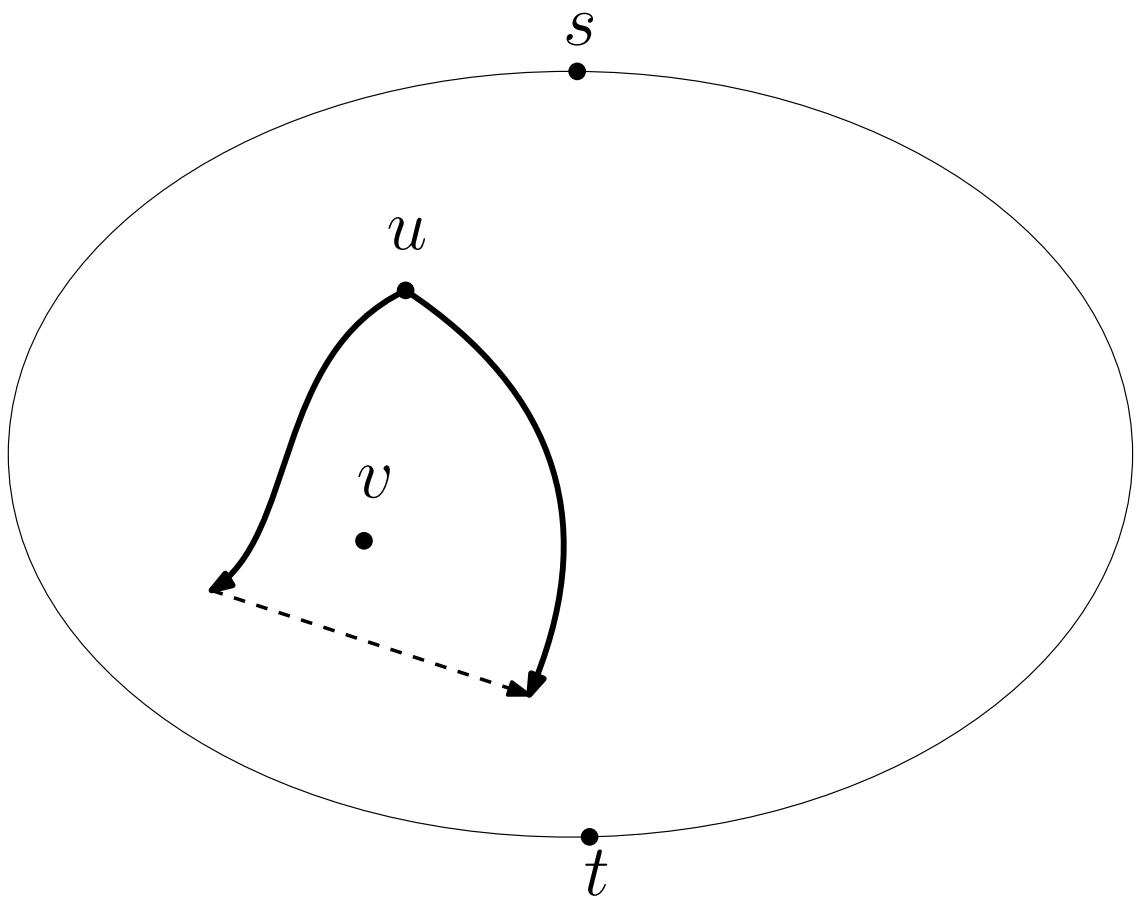
Narysujmy graf tak, aby wierzchołek s był na samej górze, wierzchołek t na samym dole, a każda krawędź prowadziła w dół.



Dwukrotnie przejdźmy graf w kolejności DFS: najpierw idąc zawsze "jak najbardziej w lewo", a potem zawsze "jak najbardziej w prawo".

M: Magiczny labirynt

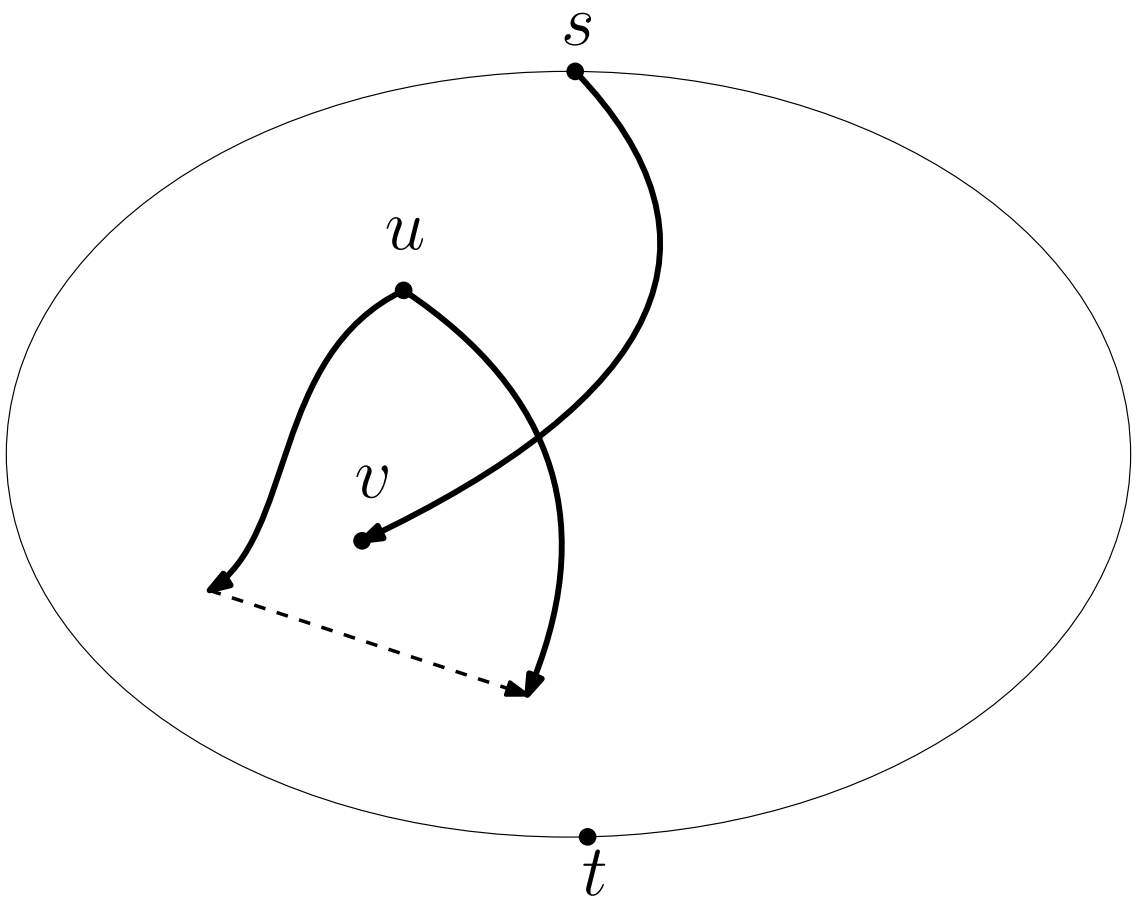
Narysujmy graf tak, aby wierzchołek s był na samej górze, wierzchołek t na samym dole, a każda krawędź prowadziła w dół.



Dwukrotnie przejdźmy graf w kolejności DFS: najpierw idąc zawsze "jak najbardziej w lewo", a potem zawsze "jak najbardziej w prawo".

M: Magiczny labirynt

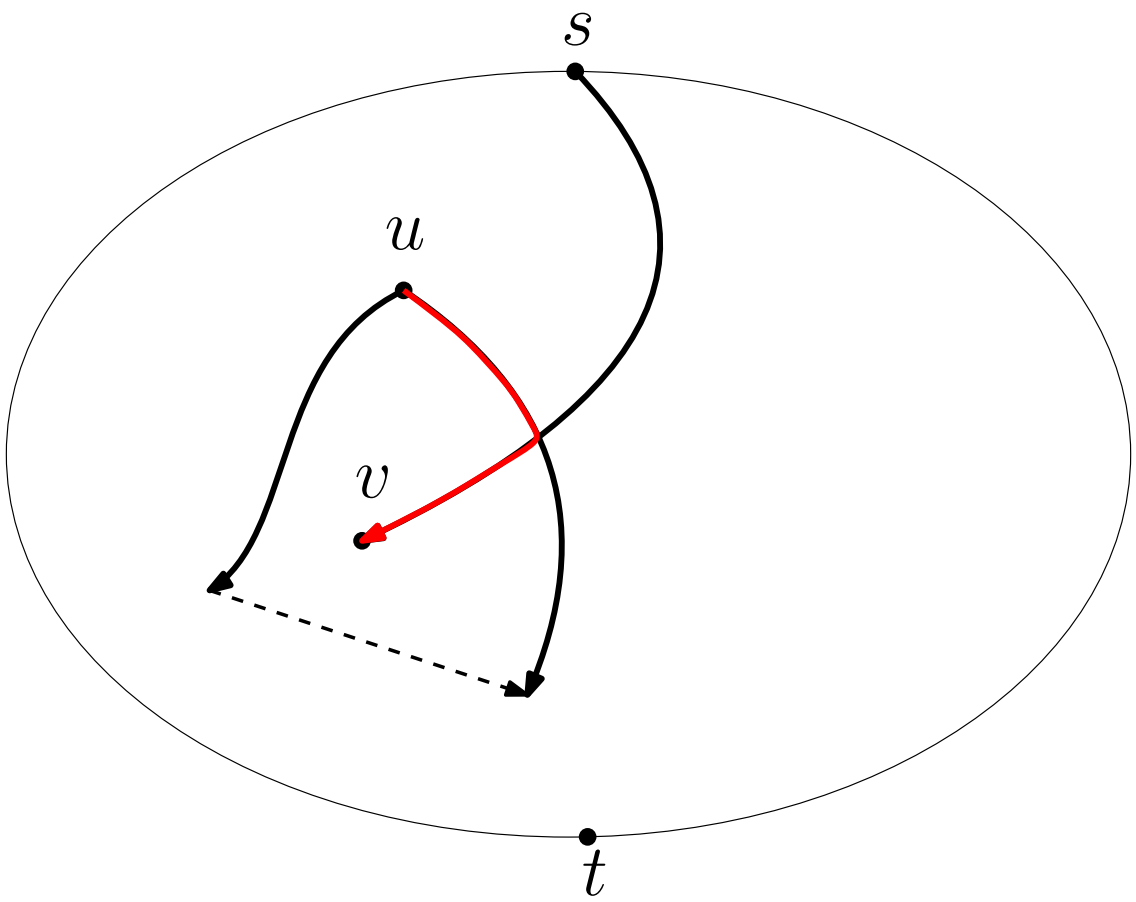
Narysujmy graf tak, aby wierzchołek s był na samej górze, wierzchołek t na samym dole, a każda krawędź prowadziła w dół.



Dwukrotnie przejdźmy graf w kolejności DFS: najpierw idąc zawsze "jak najbardziej w lewo", a potem zawsze "jak najbardziej w prawo".

M: Magiczny labirynt

Narysujmy graf tak, aby wierzchołek s był na samej górze, wierzchołek t na samym dole, a każda krawędź prowadziła w dół.



Dwukrotnie przejdźmy graf w kolejności DFS: najpierw idąc zawsze "jak najbardziej w lewo", a potem zawsze "jak najbardziej w prawo".

M: Magiczny labirynt

1. Dwukrotnie przejdźmy graf w kolejności DFS: najpierw idąc zawsze "jak najbardziej w lewo", a potem zawsze "jak najbardziej w prawo".

M: Magiczny labirynt

1. Dwukrotnie przejdźmy graf w kolejności DFS: najpierw idąc zawsze "jak najbardziej w lewo", a potem zawsze "jak najbardziej w prawo".
2. Oznaczmy otrzymany numery post-order przez post oraz post2.

M: Magiczny labirynt

1. Dwukrotnie przejdźmy graf w kolejności DFS: najpierw idąc zawsze "jak najbardziej w lewo", a potem zawsze "jak najbardziej w prawo".
2. Oznaczmy otrzymany numery post-order przez post oraz post2.
3. Można pokazać, że istnieje ścieżka z u do v wtedy i tylko wtedy gdy $\text{post}[v] \leqslant \text{post2}[u]$ oraz $\text{post2}[v] \leqslant \text{post2}[u]$.

M: Magiczny labirynt

1. Dwukrotnie przejdźmy graf w kolejności DFS: najpierw idąc zawsze "jak najbardziej w lewo", a potem zawsze "jak najbardziej w prawo".
2. Oznaczmy otrzymany numery post-order przez post oraz post2.
3. Można pokazać, że istnieje ścieżka z u do v wtedy i tylko wtedy gdy $\text{post}[v] \leqslant \text{post2}[u]$ oraz $\text{post2}[v] \leqslant \text{post2}[u]$.
4. To pozwala nam na zredukowanie zadania do zliczania liczby inwersji w permutacji.