Omówienie zadań

Potyczki Algorytmiczne 2017

Galeria handlowa

Najszybsze rozwiązanie:

Mateusz Radecki (0:04)

Galeria handlowa

- Jest n sklepów; każdy oferuje niektóre spośród m produktów.
- W sklepie możemy kupić część spośród oferowanych produktów; możemy w szczególności nie wziąć żadnego.
- Nie możemy wejść do sklepu, zakupiwszy wcześniej produkt będący na stanie tego sklepu.
- Chcemy odwiedzić wszystkie sklepy i kupić wszystkie produkty.

Galeria handlowa

Algorytm

- Jeśli istnieje produkt niedostępny w żadnym sklepie, odpowiedzią jest `NIE`.

Galeria handlowa

Algorytm

- Jeśli istnieje produkt niedostępny w żadnym sklepie, odpowiedzią jest `NIE`.
- W przeciwnym razie zawsze istnieje sposób na zakup wszystkich produktów.

Galeria handlowa

Algorytm

- Jeśli istnieje produkt niedostępny w żadnym sklepie, odpowiedzią jest `NIE`.
- W przeciwnym razie zawsze istnieje sposób na zakup wszystkich produktów.
- Odwiedzamy wszystkie sklepy w dowolnej kolejności, a każdy produkt kupujemy w najpóźniejszym możliwym sklepie.

Złożoność: $\mathcal{O}(nm)$.

Niebanalne podróże 2

Najszybsze rozwiązanie:

Kamil Dębowski (0:21)

Niebanalne podróże

Ustalmy największy wierzchołek cyklu v. Dla każdego podzbioru A wierzchołków $\{1, \ldots, v-1\}$ i ustalonego wierzchołka $k \leq v-1$ chcemy wyznaczyć liczbę prostych ścieżek zaczynających się w v, używających jedynie wierzchołków z podzbioru A i kończących się w wierzchołku k. Robimy to standardowym programowaniem dynamicznym. Daje to $2^{v-1}(v-1)$ stanów programowania dynamicznego, które można obliczyć w czasie $O(2^v m)$. Stąd łatwo wyznaczyć liczbę szukanych cykli.
Wykonując to otrzymamy rozwiązanie w złożoności $O(m(2^n + 2^{n-1} + \ldots 2^0)) = O(2^n m)$.

Klawiatura

Najszybsze rozwiązanie:

Jarosław Kwiecień (0:28)

Klawiatura

- Bajtosz wpisał swoje hasło, wpisując małe i duże litery. Kasował pomyłki przyciskiem *CapsLock*, który działa jak *Backspace*.
- Haker przechwycił klawisze naciśnięte przez Bajtosza i zapisał wprowadzony tekst do edytora. Klawisz *CapsLock* w edytorze hakera zachował jednak standardowe zachowanie.
- Dla każdego z wielu popularnych haseł chcemy dowiedzieć się, czy mogło zostać ono wpisane przez Bajtosza.

Klawiatura

Czy i-ty znak tekstu pasuje do j-tego znaku wzorca?

- Jeśli zignorujemy rozmiar znaków, to dopasowywane znaki muszą być sobie równe.

Klawiatura

Czy i-ty znak tekstu pasuje do j-tego znaku wzorca?

- Jeśli zignorujemy rozmiar znaków, to dopasowywane znaki muszą być sobie równe.
- Przed wprowadzeniem i-tego znaku tekstu musieliśmy nacisnąć *CapsLock* dokładnie $i - j$ razy.

Klawiatura

Czy i-ty znak tekstu pasuje do j-tego znaku wzorca?

- Jeśli zignorujemy rozmiar znaków, to dopasowywane znaki muszą być sobie równe.
- Przed wprowadzeniem i-tego znaku tekstu musieliśmy nacisnąć *CapsLock* dokładnie $i - j$ razy.
- Niech $\mathrm{case}(c)$ oznacza wielkość litery: 1, gdy jest wielka, a 0, gdy mała. Musi wtedy dodatkowo zajść:

$$\mathrm{case}(\mathrm{text}[i]) \oplus ((i-j) \bmod 2) = \mathrm{case}(\mathrm{pattern}[j]).$$

Czy i-ty znak tekstu pasuje do j-tego znaku wzorca?

- Jeśli zignorujemy rozmiar znaków, to dopasowywane znaki muszą być sobie równe.
- Przed wprowadzeniem i-tego znaku tekstu musieliśmy nacisnąć *CapsLock* dokładnie $i - j$ razy.
- Niech $\mathrm{case}(c)$ oznacza wielkość litery: 1, gdy jest wielka, a 0, gdy mała. Musi wtedy dodatkowo zajść:

$$\mathrm{case}(\mathrm{text}[i]) \oplus ((i-j) \bmod 2) = \mathrm{case}(\mathrm{pattern}[j]).$$

- Równoważnie,

$$\mathrm{case}(\mathrm{text}[i]) \oplus (i \bmod 2) = \mathrm{case}(\mathrm{pattern}[j]) \oplus (j \bmod 2).$$

Klawiatura

Algorytm

- Zamieniamy rozmiar co drugiego znaku w tekście oraz każdym ze wzorców.
- Po takiej zamianie po prostu sprawdzamy dla każdego ze wzorców, czy jest podciągiem tekstu.

Klawiatura

Algorytm

- Zamieniamy rozmiar co drugiego znaku w tekście oraz każdym ze wzorców.
- Po takiej zamianie po prostu sprawdzamy dla każdego ze wzorców, czy jest podciągiem tekstu.
- Możemy odpowiedzieć na wszystkie zapytania off-line w czasie $\mathcal{O}(n + \sum |z_i|)$ albo on-line w czasie $\mathcal{O}(n + \sum |z_i| \log n)$ lub $\mathcal{O}(n + \sum |z_i| A)$, gdzie A – rozmiar alfabetu.

Sumowanie

Najszybsze rozwiązanie:

Michał Seweryn (2:13)

Dany jest język skryptowy, który wykonuje następujący fragment kodu:

```
while true:
  foundReplacement = false
  for i = 1, 2, ..., k:
    if a(i) jest podsłowem s:
      zastąp najwcześniejsze wystąpienie a(i) przez b(i)
      foundReplacement = true
      break
  if not foundReplacement:
    break
print(s)
```

Należy napisać w tym języku skrypt, który dodaje dwie liczby zapisane binarnie. Dokładniej, jeśli na wejściu znajduje się napis postaci `101111+1001`, to należy wypisać `111000`. Będziemy omawiali rozwiązanie na podstawie tego przykładu.

Szkic rozwiązania jest następujący. Będziemy na końcu wejściowego napisu utrzymywać znak reprezentujący cyfrę przeniesienia, a po nim napis reprezentujący wynik (na początku jest on pusty).

Szkic rozwiązania jest następujący. Będziemy na końcu wejściowego napisu utrzymywać znak reprezentujący cyfrę przeniesienia, a po nim napis reprezentujący wynik (na początku jest on pusty).

Następnie będziemy przenosić cyfrę stojącą bezpośrednio przed plusem na koniec drugiej liczby z wejścia i stosować tam reguły dodawania, dopisując kolejną cyfrę do wyniku.

Szkic rozwiązania jest następujący. Będziemy na końcu wejściowego napisu utrzymywać znak reprezentujący cyfrę przeniesienia, a po nim napis reprezentujący wynik (na początku jest on pusty).

Następnie będziemy przenosić cyfrę stojącą bezpośrednio przed plusem na koniec drugiej liczby z wejścia i stosować tam reguły dodawania, dopisując kolejną cyfrę do wyniku.

Trzeba oczywiście rozważyć przypadki, gdy cyfry w którejś z liczb wejściowych się skończą

Będziemy reprezentować cyfrę przeniesienia przez `ab`, a cyfry przenoszone przez `cd`.

Będziemy reprezentować cyfrę przeniesienia przez ab, a cyfry przenoszone przez cd.

- W pierwszym kroku wykorzystujemy znak + do zapisania cyfry przeniesienia. Mamy napis 101111@*e*1001.
- W drugim kroku przenosimy *e* na koniec.
- W trzecim kroku zamieniamy *e* na *a*, aby więcej ta cyfra się nie ruszała. Mamy napis 101111@1001*a*.

Mamy napis 101111@1001a. Bierzemy teraz pierwszą cyfrę przed @ i przenosimy ją aż do a. Mamy napis 10111@1001da. Następnie stosujemy reguły dodawania, aktualizując cyfrę w wyniku oraz cyfrę przeniesienia. Otrzymujemy w tym przypadku napis 10111@100b0.

Po pewnej liczbie iteracji dochodzimy do sytuacji w której:

- Liczby na wejściu były równej długości – należy wtedy usunąć @ i ewentualnie dopisać jedynkę jeśli cyfra przeniesienia była dodatnia.
- Zostały cyfry tylko w lewej liczbie – jest ona wtedy postaci 101010@b1010101. Należy wtedy usunąć @ i zastosować ewentualne reguły przeniesienia, gdy cyfrą przeniesienia było b.
- Zostały cyfry tylko w prawej liczbie – jest ona wtedy postaci @101010b1010101. Należy wtedy zastosować reguły przeniesienia, gdy cyfrą przeniesienia było b i na końcu usunąć @.

Całkowita liczba obrotów zewnętrznej pętli jest ograniczona przez kwadrat długości wejścia – zatem zmieści się ona w limicie. Również długość słowa nie przekroczy ok. dwukrotoności napisu wejściowego – zatem słowo nigdy nie będzie dłuższe niż ograniczenia w zadaniu. Wzorcowe skrypty mają odpowiednio 31 i 37 reguł.

Trenerzy

Najszybsze rozwiązanie:

??? (?:??)

Uproszczenia i oznaczenia

Bez straty ogólności załóżmy, że $1 \leq n \leq \mathrm{nww}(a, b)$ oraz $a \leq b$. W co najwyżej jednym dniu obaj trenerzy jednocześnie zmienią swój plan.
Oznaczmy przez A, B liczby dni kiedy Marcin i Michał zmieniają swoje plany, a przez C dni, czy zmieniają go obaj.

Uproszczenia i oznaczenia

Bez straty ogólności załóżmy, że $1 \leq n \leq \mathrm{nww}(a, b)$ oraz $a \leq b$. W co najwyżej jednym dniu obaj trenerzy jednocześnie zmienią swój plan.
Oznaczmy przez A, B liczby dni kiedy Marcin i Michał zmieniają swoje plany, a przez C dni, czy zmieniają go obaj.

Wynik wynosi: $n - A - B + 2 \cdot C$.

Uproszczenia i oznaczenia

Bez straty ogólności załóżmy, że $1 \leq n \leq \mathrm{nww}(a, b)$ oraz $a \leq b$. W co najwyżej jednym dniu obaj trenerzy jednocześnie zmienią swój plan.
Oznaczmy przez A, B liczby dni kiedy Marcin i Michał zmieniają swoje plany, a przez C dni, czy zmieniają go obaj.

Wynik wynosi: $n - A - B + 2 \cdot C$.

Ile wynoszą A, B, C?

$A \in \{\lfloor \frac{n}{a} \rfloor, \lceil \frac{n}{a} \rceil\}$,
$B \in \{\lfloor \frac{n}{a} \rfloor, \lceil \frac{n}{a} \rceil\}$,
$C = 1$ *(opłaca się zwiększyć C kosztem zmniejszenia A, B).*

Oznaczmy przez x numer dnia, gdy obaj trenerzy zmieniają plan (dni numerujemy od 0).

Co powinien spełniać x?

Aby zachodziło $A = \lfloor \frac{n}{a} \rfloor$, musi zachodzić $x \geq n \bmod a$.
Aby zachodziło $B = \lfloor \frac{n}{b} \rfloor$, musi zachodzić $x \geq n \bmod b$.

Oznaczmy przez x numer dnia, gdy obaj trenerzy zmieniają plan (dni numerujemy od 0).

Co powinien spełniać x?

Aby zachodziło $A = \lfloor \frac{n}{a} \rfloor$, musi zachodzić $x \geq n \bmod a$.
Aby zachodziło $B = \lfloor \frac{n}{b} \rfloor$, musi zachodzić $x \geq n \bmod b$.

Kiedy nie można spełnić żadnego warunku?

Żadnego z tych dwóch warunków nie możemy zapewnić wtedy i tylko wtedy, gdy $a > n$ oraz $b > n$.

Czy możemy spełnić oba warunki?

Rozpatrzmy tabelkę $a \times b$. W polu (n mod a, n mod b) wpiszmy wartość n. Jeśli w prostokącie o lewym dolnym rogu w polu o wartości n możemy znaleźć mniejszą wartość, jest ona wartością x spełniającą oba warunki.

14	8	2	**17**	**11**	**5**	**20**
7	1	16	**10**	**4**	**19**	**13**
0	15	9	3	18	12	6

Dla przykładu powyżej ($n = 10$, $a = 3$, $b = 7$), poszukiwanymi wartościami x są 4 i 5.

Czy możemy spełnić oba warunki?

14	**8**	**2**	17	11	5	20
7	**1**	**16**	10	4	19	13
0	**15**	**9**	3	18	12	6

W początkowym kwadracie $a \times a$, przekątna jest zbiorem takich n, dla których nie da się spełnić obu warunków.
Jeśli w tym kwadracie nie ma pola odpowiadającego n, ograniczmy się do prawego prostokąta $a \times (b - a)$.

8	5	2	11
4	1	10	7
0	9	6	3

Kolejność wartości na pozostałych polach się nie zmieniła!

Czy możemy spełnić oba warunki?

14	**8**	**2**	17	11	5	20
7	**1**	**16**	10	4	19	13
0	**15**	**9**	3	18	12	6

W początkowym kwadracie $a \times a$, przekątna jest zbiorem takich n, dla których nie da się spełnić obu warunków.
Jeśli w tym kwadracie nie ma pola odpowiadającego n, ograniczmy się do prawego prostokąta $a \times (b - a)$.

8	5	2	11
4	1	10	7
0	9	6	3

Kolejność wartości na pozostałych polach się nie zmieniła!

Algorytm

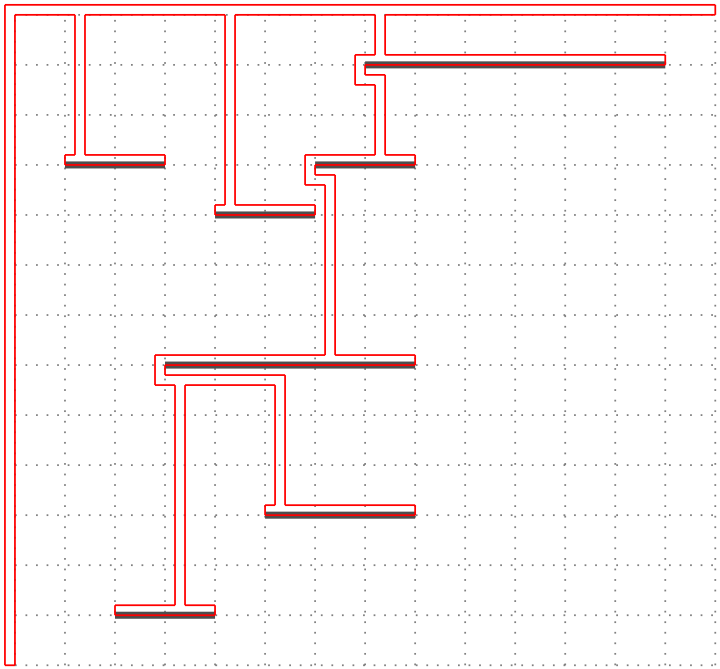
Redukujemy prostokąt algorytmem Euklidesa. Otrzymujemy rozwiązanie o złożoności $O(log(n))$ dla jednego zapytania.

Wielokąt

Najszybsze rozwiązanie:

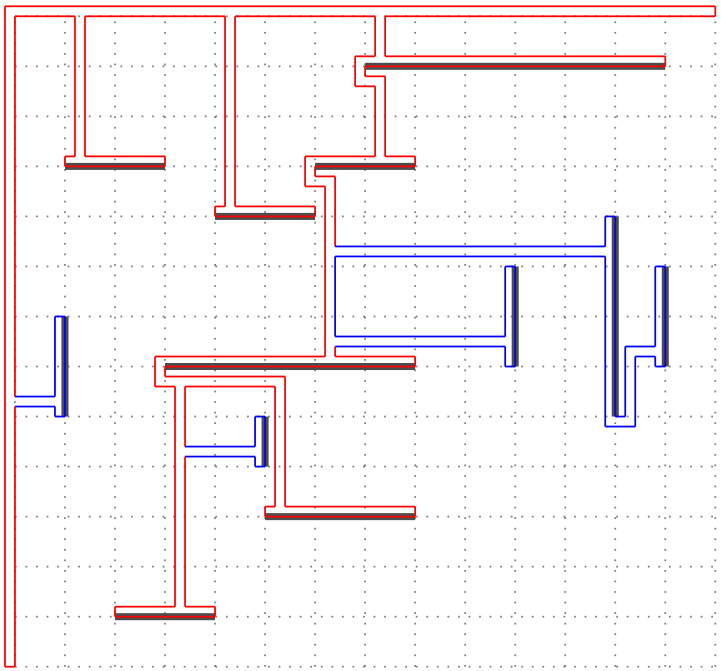
brak

Wielokąt zawsze da się skonstruować. Na początek rozważamy jedynie odcinki poziome, łączymy je w wielokąt tak, aby później możliwe było dodanie odcinków pionowych:



Każdy odcinek „podłączamy” do odcinka bezpośrednio nad nim; wszystkie odcinki „nad” możemy wyznaczyć w czasie $O(n \log n)$ prostym zamiataniem.

Analogiczną metodą rozważamy odcinki pionowe, podłączając je do boku wielokąta leżącego bezpośrednio na lewo:



Konstrukcja wymaga $10n + O(1)$ boków.

Przesył

Najszybsze rozwiązanie:

??? (?:??)

Dany jest graf nieskierowany oraz q zapytań. Każde zapytanie wskazuje r_i krawędzi, które zostaną usunięte lub staną się jednokierunkowe. Na każde zapytanie należy *online* odpowiedzieć, do ilu wierzchołków da się dojechać ze wszystkich innych.

Graf jest spójny, więc możemy utworzyć drzewo rozpinające.

Graf jest spójny, więc możemy utworzyć drzewo rozpinające.

Będziemy rozważać teraz proces udzielania odpowiedzi na pojedyncze zapytanie. Mamy danych r krawędzi, które chcemy usunąć lub zmienić na jednokierunkowe. Krawędzie te możemy podzielić na dwie grupy – te, które należały wcześniej do drzewa rozpinającego i te, które do niego nie należały.

Usuńmy najpierw krawędzie, które były w drzewie rozpinającym. Krawędzie te powodują rozspójnienie drzewa na $\leq r + 1$ spójnych. Główną ideą rozwiązania będzie stworzenie nowego grafu, w którym wierzchołki reprezentują powyższe spójne.

Usuńmy najpierw krawędzie, które były w drzewie rozpinającym. Krawędzie te powodują rozspójnienie drzewa na $\leq r + 1$ spójnych. Główną ideą rozwiązania będzie stworzenie nowego grafu, w którym wierzchołki reprezentują powyższe spójne.

Graf spójnych

Istnieje krawędź między składowymi $A \to B$ w nowym grafie $\iff$ $\exists a \in A, b \in B$ w pierwotnym grafie połączone bezpośrednią krawędzią $a \to b$.

Będziemy teraz chcieli opisać zbiory wierzchołków w każdej składowej.

Będziemy teraz chcieli opisać zbiory wierzchołków w każdej składowej.

Lemat

Rozważmy sortowanie drzewa rozpinającego po *POST ORDER*. Wtedy wszystkie spójne możemy opisać przy pomocy nie więcej niż $2r + O(1)$ przedziałów liczb (łącznie).

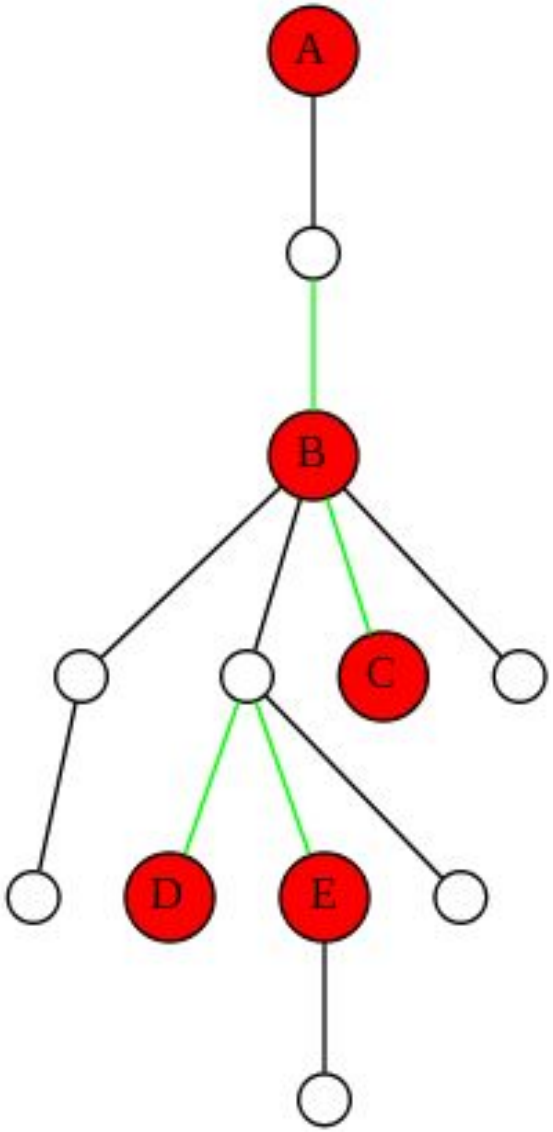
Dowód

Dla każdej spójnej możemy wybrać wierzchołek, który jest najbliższy korzenia drzewa rozpinającego (tzn. jest korzeniem pewnego poddrzewa w drzewie rozpinającym) – nazwijmy go reprezentantem.

Dowód

Dla każdej spójnej możemy wybrać wierzchołek, który jest najbliższy korzenia drzewa rozpinającego (tzn. jest korzeniem pewnego poddrzewa w drzewie rozpinającym) – nazwijmy go reprezentantem.

Możemy teraz w naturalny sposób stworzyć drzewo reprezentantów. Dokładniej dwaj reprezentanci są połączeni krawędzią jeśli można w drzewie rozpinającym przejść od jednego do drugiego, nie przechodząc przez innego reprezentanta. Przykład zaznaczono na następującym obrazku:


A
B
C
D
E

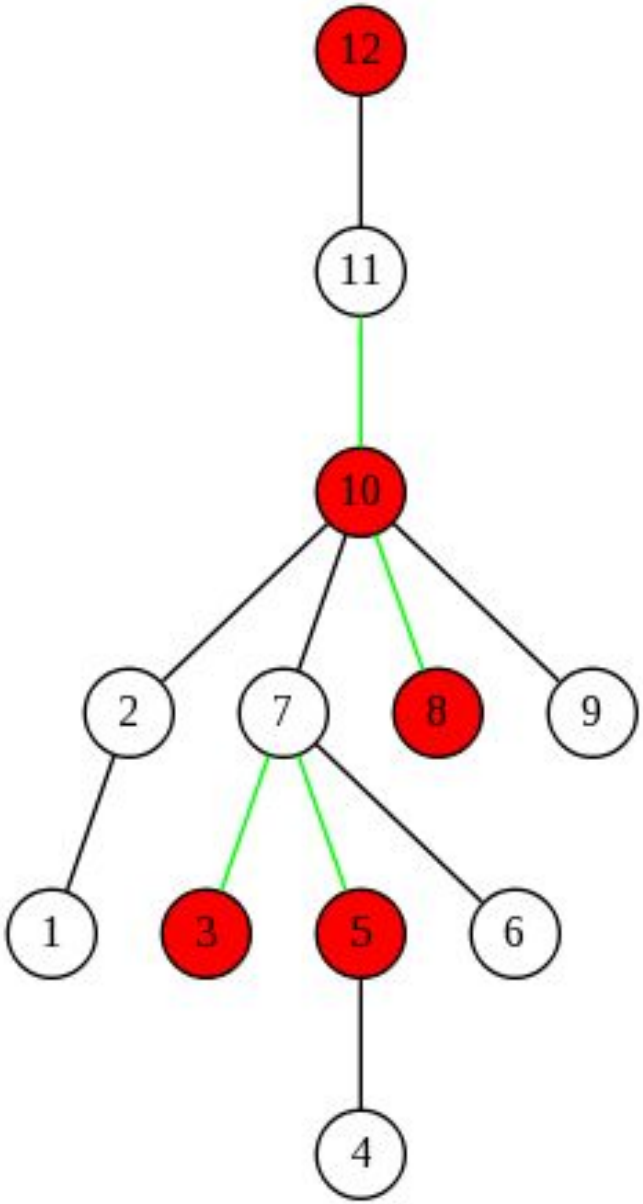
Dowód

Niech I_v oznacza przedział numerów wierzchołków w poddrzewie wierzchołka v w drzewie rozpinającym.

Dowód

Niech I_v oznacza przedział numerów wierzchołków w poddrzewie wierzchołka v w drzewie rozpinającym.

Wtedy jeśli v jest reprezentantem, to spójna, którą on reprezentuje, może być opisana przez $I_v \setminus \bigcup_{v_k} I_{v_k}$, gdzie v_k są bezpośrednimi synami w drzewie reprezentantów.


12
11
10
2
7
8
9
1
3
5
6
4

Dowód

Zbiór $I_v \setminus \bigcup_{v_k} I_{v_k}$ możemy opisać jako sumę pewnych $k_v + 1$ przedziałów, gdzie k_v to liczba synów danego węzła w drzewie reprezentantów. Wszystkie powstałe w ten sposób przedziały są parami rozłączne.

Dowód

Zbiór $I_v \setminus \bigcup_{v_k} I_{v_k}$ możemy opisać jako sumę pewnych $k_v + 1$ przedziałów, gdzie k_v to liczba synów danego węzła w drzewie reprezentantów. Wszystkie powstałe w ten sposób przedziały są parami rozłączne.

Zatem łączna liczba przedziałów, które opisują składowe to $\sum(k_v + 1)$ – przy czym suma przebiega po wierzchołkach będących reprezentantami. Ponieważ drzewo reprezentantów zawiera $r + 1$ wierzchołków, to $\sum(k_v + 1) = 2r + 2$.

Wróćmy zatem do pierwotnego pytania. Powiedzmy, że chcemy wiedzieć, czy między składowymi A i B istnieje krawędź. Przyjmijmy również, iż składowa A jest opisana przedziałami $P_1, P_2, ..., P_a$, a składowa B jest opisana przedziałami $Q_1, Q_2, ..., Q_b$. Wystarczy zatem odpowiedzieć czy

$$\exists_{1 \leq i \leq a, 1 \leq j \leq b} \quad \left| \{ p \to q | p \in P_i, q \in Q_j \} \right| > 0.$$

Wróćmy zatem do pierwotnego pytania. Powiedzmy, że chcemy wiedzieć, czy między składowymi A i B istnieje krawędź. Przyjmijmy również, iż składowa A jest opisana przedziałami $P_1, P_2, ..., P_a$, a składowa B jest opisana przedziałami $Q_1, Q_2, ..., Q_b$. Wystarczy zatem odpowiedzieć czy

$$\exists_{1 \leq i \leq a, 1 \leq j \leq b} \quad \left| \{ p \to q | p \in P_i, q \in Q_j \} \right| > 0.$$

Fakt

Na pytania o liczbę krawędzi między przedziałami (w pierwotnym grafie) możemy odpowiadać w czasie stałym po wcześniejszym preprocessingu.

Fakt

Na pytania o liczbę krawędzi między przedziałami (w pierwotnym grafie) możemy odpowiadać w czasie stałym po wcześniejszym preprocessingu.

Fakt

Na pytania o liczbę krawędzi między przedziałami (w pierwotnym grafie) możemy odpowiadać w czasie stałym po wcześniejszym preprocessingu.

Jest to problem równoważny następnemu: mając dany prostokąt, gdzie w każdej kratce znajduje się liczba, należy szybko odpowiadać na pytania o sumę liczb w pewnych podprostokątach.

Fakt

Na pytania o liczbę krawędzi między przedziałami (w pierwotnym grafie) możemy odpowiadać w czasie stałym po wcześniejszym preprocessingu.

Jest to problem równoważny następnemu: mając dany prostokąt, gdzie w każdej kratce znajduje się liczba, należy szybko odpowiadać na pytania o sumę liczb w pewnych podprostokątach.

Wystarczy stworzyć dwuwymiarowe sumy prefiksowe na macierzy sąsiedztwa.

Tworzymy macierz sąsiedztwa pomiędzy spójnymi w drzewie rozpinającym. Wartość $M_{a,b}$ oznacza liczbę krawędzi ze spójnej a do spójnej b.
Możemy teraz uwzględnić usunięcie krawędzi niedrzewowych; jeśli usuwamy krawędź skierowaną ze spójnej a do spójnej b, wystarczy zmniejszyć $M_{a,b}$ o 1.

Na tej podstawie zatem graf G' powiązań między składowymi w pierwotnym grafie.

Uruchamiamy na nim algorytm znajdowania silnych spójnych składowych, który działa w czasie liniowym od liczby krawędzi w G'. Sprawdzamy, czy jest dokładnie jedna silnie spójna, z której nie wychodzą żadne krawędzie spoza niej i czy da się do niej dojść ze wszystkich innych wierzchołków.

Złożoność czasowa rozwiązania:

Preprocessing na stworzenie macierzy i stworzenie drzewa rozpinającego zajmuje $O(n^2 + m)$. Stworzenie drzewa reprezentantów można zrobić w $O(r \log r)$. Następnie tworzymy graf G' w czasie $O(r^2)$ i odpowiadamy na pytanie o istnienie dokładnie jednej silnej spójnej również w czasie $O(r^2)$.

Zatem całkowita złożoność to $O(n^2 + m + \sum r_i^2)$.

Machine learning

Najszybsze rozwiązanie:

??? (?:??)

Machine learning - rozgrzewka

Na początku rozwiążmy prostszą wersję zadania, w której przybliżamy f poprzez zwykłą funkcją liniową postaci $g(x) = ax + b$. Wtedy nasz błąd jest wielomianem kwadratowym zmiennych a oraz b, który możemy zminimalizować ze względu na te zmienne i wyznaczyć jego minimum globalne poprzez przyrównanie pochodnych po a i b do zera.
$nMSE = \sum_{i=1}^{n}(y_i - (ax_i + b))^2 =$
$\sum y_i^2 - 2a\sum x_i y_i - 2b\sum y_i + a^2\sum x_i^2 + 2ab\sum x_i + \sum b^2$.
Dostajemy, że $b = \frac{\sum y_i - a\sum x_i}{n}$ oraz $a = \frac{\sum x_i y_i - b\sum x_i}{\sum x_i^2}$, skąd dostajemy, że $a = \frac{\sum x_i y_i - \frac{\sum x_i \sum y_i}{n}}{\sum x_i^2 - \frac{(\sum x_i)^2}{n}}$ (gdy dane x_i oraz y_i wyrazimy w języku zmiennych losowych, to $a = \frac{Cov(X,Y)}{Var(X)}$).

Przypadek $c = x_i$

c - punkt złamania naszej funkcji.
Niech $y = g(c)$. Ustalmy y i w zależności od niego wyznaczmy optymalne nachylenia po obu stronach. Są one funkcjami liniowymi od y. Optymalny błąd po obu stronach jest funkcją kwadratową od y.

- Wyszukiwanie ternarne i ewaluacja błędu $\rightarrow$ proste wzory, czas $O(\log n)$
- Wyznaczenie wzoru funkcji kwadratowej i wyznaczenie jej minimum $\rightarrow$ trudne wzory, czas $O(1)$

Przypadek $x_i < c < x_{i+1}$

Załóżmy, że $x_i < x_{i+1}$.

Lemat

Jeżeli rozwiązanie optymalne ma punkt złamania ściśle wewnątrz przedziału, to wtedy jest ono przecięciem optymalnej prostej dla prefiksu punktów $x_1, \ldots, x_i$ oraz optymalnej prostej dla sufiksu punktów $x_{i+1}, \ldots, x_n$.

Dla każdego takiego przedziału wyznaczamy optymalne dopasowania prostych na prefiksie i sufiksie (za pomocą zwykłego przybliżenia jedną prostą) i sprawdzamy, czy ich przecięcie leży w przedziale (x_i, x_{i+1}). Zajmuje to czas $O(n)$.