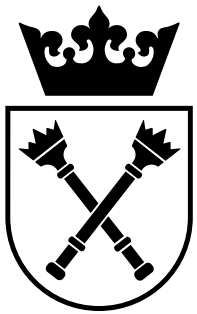
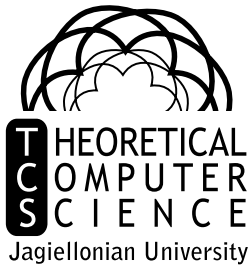
AMPPZ 2019: Prezentacja rozwiązań

Uniwersytet Jagielloński

27 października 2019





Statystyki

Zgłoszeń: 594
Zaakceptowanych: 208

Statystyki

Zgłoszeń: 594
Zaakceptowanych: 208

Pierwsze wbite zadanie: (C) Koło (z) Matematyków
uw7 [Partyka, Wiśniewski, Wasilewski]

Statystyki

Najbardziej zdeterminowani:

uw7 [Partyka, Wiśniewski, Wasilewski]
16 prób na zadaniu K

Zadanie A
Asymilacja

Zgłoszeń: 84
Zaakceptowanych: 36

Pierwsze rozwiązanie:
uj8 [Kapała, Tokarz, Deka]
00:21:23

Autor: Lech Duraj

Mamy n planet do opanowania, o wartościach $a_1; : : : ; a_n$ oraz k statków asymilacyjnych. W jednym ruchu możemy dokonać inwazji na planetę o wartości m, lądując na niej s m statkami. Planeta robi się wtedy opanowana, a jej nowa wartość to $m + s$. Z takiej planety można w przyszłości (ale tylko raz) podnieść $m + s$ statków. Ile co najmniej razy trzeba podnieść, by podbić wszystko?

Mamy n planet do opanowania, o wartościach $a_1; : : : ; a_n$ oraz k statków asymilacyjnych. W jednym ruchu możemy dokonać inwazji na planetę o wartości m, lądując na niej s m statkami. Planeta robi się wtedy opanowana, a jej nowa wartość to $m + s$. Z takiej planety można w przyszłości (ale tylko raz) podnieść $m + s$ statków. Ile co najmniej razy trzeba podnieść, by podbić wszystko?

Zadanie inspirowane mechaniką migracji barbarzyńców w grze *Imperator: Rome*.

Mamy n planet do opanowania, o wartościach $a_1; : : : ; a_n$ oraz k statków asymilacyjnych. W jednym ruchu możemy dokonać inwazji na planetę o wartości m, lądując na niej s m statkami. Planeta robi się wtedy opanowana, a jej nowa wartość to $m + s$. Z takiej planety można w przyszłości (ale tylko raz) podnieść $m + s$ statków. Ile co najmniej razy trzeba podnieść, by podbić wszystko?

Zadanie inspirowane mechaniką migracji barbarzyńców w grze *Imperator: Rome*.

(dobrze zastosowana poniższa strategia = lawina barbarzyńców!)

Jeśli liczba statków na początku starczy na opanowanie wszystkich planet od razu, to odpowiedź jest oczywiście 0.

Jeśli liczba statków na początku starczy na opanowanie wszystkich planet od razu, to odpowiedź jest oczywiście 0.
Ale co, jeśli nie jest?

Jeśli liczba statków na początku starczy na opanowanie wszystkich planet od razu, to odpowiedź jest oczywiście 0.
Ale co, jeśli nie jest?

Na pewno jest jakaś planeta, którą będziemy mobilizować. Co więcej, na pewno jest planeta P, którą będziemy mobilizować jako pierwszą.

Jeśli liczba statków na początku starczy na opanowanie wszystkich planet od razu, to odpowiedź jest oczywiście 0.
Ale co, jeśli nie jest?

Na pewno jest jakaś planeta, którą będziemy mobilizować. Co więcej, na pewno jest planeta P, którą będziemy mobilizować jako pierwszą.

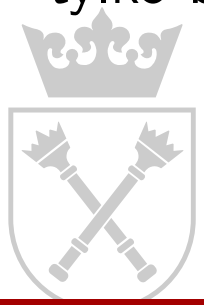
Ale w takim razie można *od razu* opanować P i *od razu* ją zmobilizować – dalej będzie tylko łatwiej.

Jeśli liczba statków na początku starczy na opanowanie wszystkich planet od razu, to odpowiedź jest oczywiście 0.
Ale co, jeśli nie jest?

Na pewno jest jakaś planeta, którą będziemy mobilizować. Co więcej, na pewno jest planeta P, którą będziemy mobilizować jako pierwszą.

Ale w takim razie można *od razu* opanować P i *od razu* ją zmobilizować – dalej będzie tylko łatwiej.

Czyli wiemy, że pierwszym naszym ruchem będzie opanowanie jakiejś planety i natychmiastowa mobilizacja. To teraz oczywiste jest, którą planetę wybrać – największą, na jaką możemy się porwać. Inny wybór tylko by nam utrudnił sprawę, bo mielibyśmy dalej mniej statków.



Pełny algorytm wygląda tak:

Q – zbiór jeszcze nie opanowanych planet
s – suma wartości planet w Q
k – liczba naszych statków
wynik – liczba mobilizacji, jakie wykonaliśmy.

- Jeżeli k s, to opanuj wszystkie planety i zakończ algorytm.
- W przeciwnym razie, weź p – największą planetę z Q mniejszą lub równą k.
- $Q = Q$ fpg; $s = s$ p; $k = k + p$; $wynik = wynik + 1$.

Pełny algorytm wygląda tak:

Q – zbiór jeszcze nie opanowanych planet
s – suma wartości planet w Q
k – liczba naszych statków
wynik – liczba mobilizacji, jakie wykonaliśmy.

- Jeżeli k s, to opanuj wszystkie planety i zakończ algorytm.
- W przeciwnym razie, weź p – największą planetę z Q mniejszą lub równą k.
- $Q = Q$ fpg; $s = s$ p; $k = k + p$; $wynik = wynik + 1$.

Najprościej implementować Q jako `multiset`, co daje algorytm $O(n \log n)$
Ale można też użyć posortowanej tablicy – nieco dłuższy kod, nieco lepsza stała w złożoności.

```
multiset<int> s{10, 20, 30};
long long x = 123000123000LL;
auto it = s.lower_bound(x);
if(it != s.end()) {
    cout << *it;
}
```

(ten kod wypisze liczbę 10)

Zadanie B
Robaczek

Zgłoszeń: 38

Zaakceptowanych: 5

Pierwsze rozwiązanie:
uw1 [Kondraciuk, Skiba, Paluszek]
01:36:15*

Autor: Adam Polak

- Weźmy dowolny wierzchołek na najkrótszej ścieżce pomiędzy pozycją początkową i końcową Robaczka. Nazwijmy ten wierzchołek v.

- Weźmy dowolny wierzchołek na najkrótszej ścieżce pomiędzy pozycją początkową i końcową Robaczka. Nazwijmy ten wierzchołek v.
- Zadanie teraz jest następujące: mamy pozycję Robaczka i chcemy dotrzeć końcem Robaczka do wierzchołka v. Jeśli mamy maszynkę, która potrafi rozwiązywać takie zadanie, to możemy odpalić ją dla pozycji początkowej i końcowej robaczka – potem sklejamy wyniki.

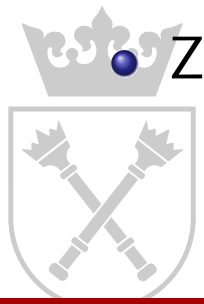
- Weźmy dowolny wierzchołek na najkrótszej ścieżce pomiędzy pozycją początkową i końcową Robaczka. Nazwijmy ten wierzchołek v.
- Zadanie teraz jest następujące: mamy pozycję Robaczka i chcemy dotrzeć końcem Robaczka do wierzchołka v. Jeśli mamy maszynkę, która potrafi rozwiązywać takie zadanie, to możemy odpalić ją dla pozycji początkowej i końcowej robaczka – potem sklejamy wyniki.
- Załóżmy, że robaczek leży na ścieżce $(x; y)$. Wtedy tak naprawdę chcemy, żeby robaczek dotknął jednym ze swoich końców wierzchołka $z := lca(x; y)$.

- Niech x oznacza jeden z końców Robaczka. Wtedy spróbujmy pójść do najgłębszego wierzchołka w poddrzewie x. Po tej operacji mamy dwie możliwości:
  1. Drugi koniec Robaczka jest w tym samym poddrzewie wierzchołka c, co i wierzchołek v. Wtedy wygraliśmy.
  2. Robaczek leży na ścieżce $(x^0; y^0)$. Wtedy powtarzamy dokładnie to, co wcześniej, tylko z innym końcem Robaczka.

- Niech x oznacza jeden z końców Robaczka. Wtedy spróbujmy pójść do najgłębszego wierzchołka w poddrzewie x. Po tej operacji mamy dwie możliwości:
  1. Drugi koniec Robaczka jest w tym samym poddrzewie wierzchołka c, co i wierzchołek v. Wtedy wygraliśmy.
  2. Robaczek leży na ścieżce $(x^0; y^0)$. Wtedy powtarzamy dokładnie to, co wcześniej, tylko z innym końcem Robaczka.
- Jeśli w jakimś momencie Robaczek przestał "podnosić się" jednym ze swoich końców – przerywamy i mówimy "NIE".

- Niech x oznacza jeden z końców Robaczka. Wtedy spróbujmy pójść do najgłębszego wierzchołka w poddrzewie x. Po tej operacji mamy dwie możliwości:
  1. Drugi koniec Robaczka jest w tym samym poddrzewie wierzchołka c, co i wierzchołek v. Wtedy wygraliśmy.
  2. Robaczek leży na ścieżce $(x^0; y^0)$. Wtedy powtarzamy dokładnie to, co wcześniej, tylko z innym końcem Robaczka.
- Jeśli w jakimś momencie Robaczek przestał "podnosić się" jednym ze swoich końców – przerywamy i mówimy "NIE".
- Zauważmy, że za każdym razem, kiedy schodzimy Robaczkiem do najgłębszego poddrzewa – odwiedzamy nowy wierzchołek. Zatem można zaimplementować to rozwiązanie tak, żeby Robaczek wykonał nie więcej niż n ruchów.

- Niech x oznacza jeden z końców Robaczka. Wtedy spróbujmy pójść do najgłębszego wierzchołka w poddrzewie x. Po tej operacji mamy dwie możliwości:
  1. Drugi koniec Robaczka jest w tym samym poddrzewie wierzchołka c, co i wierzchołek v. Wtedy wygraliśmy.
  2. Robaczek leży na ścieżce $(x^0; y^0)$. Wtedy powtarzamy dokładnie to, co wcześniej, tylko z innym końcem Robaczka.
- Jeśli w jakimś momencie Robaczek przestał "podnosić się" jednym ze swoich końców – przerywamy i mówimy "NIE".
- Zauważmy, że za każdym razem, kiedy schodzimy Robaczkiem do najgłębszego poddrzewa – odwiedzamy nowy wierzchołek. Zatem można zaimplementować to rozwiązanie tak, żeby Robaczek wykonał nie więcej niż n ruchów.
- Złożoność czasowa: $O(n)$.

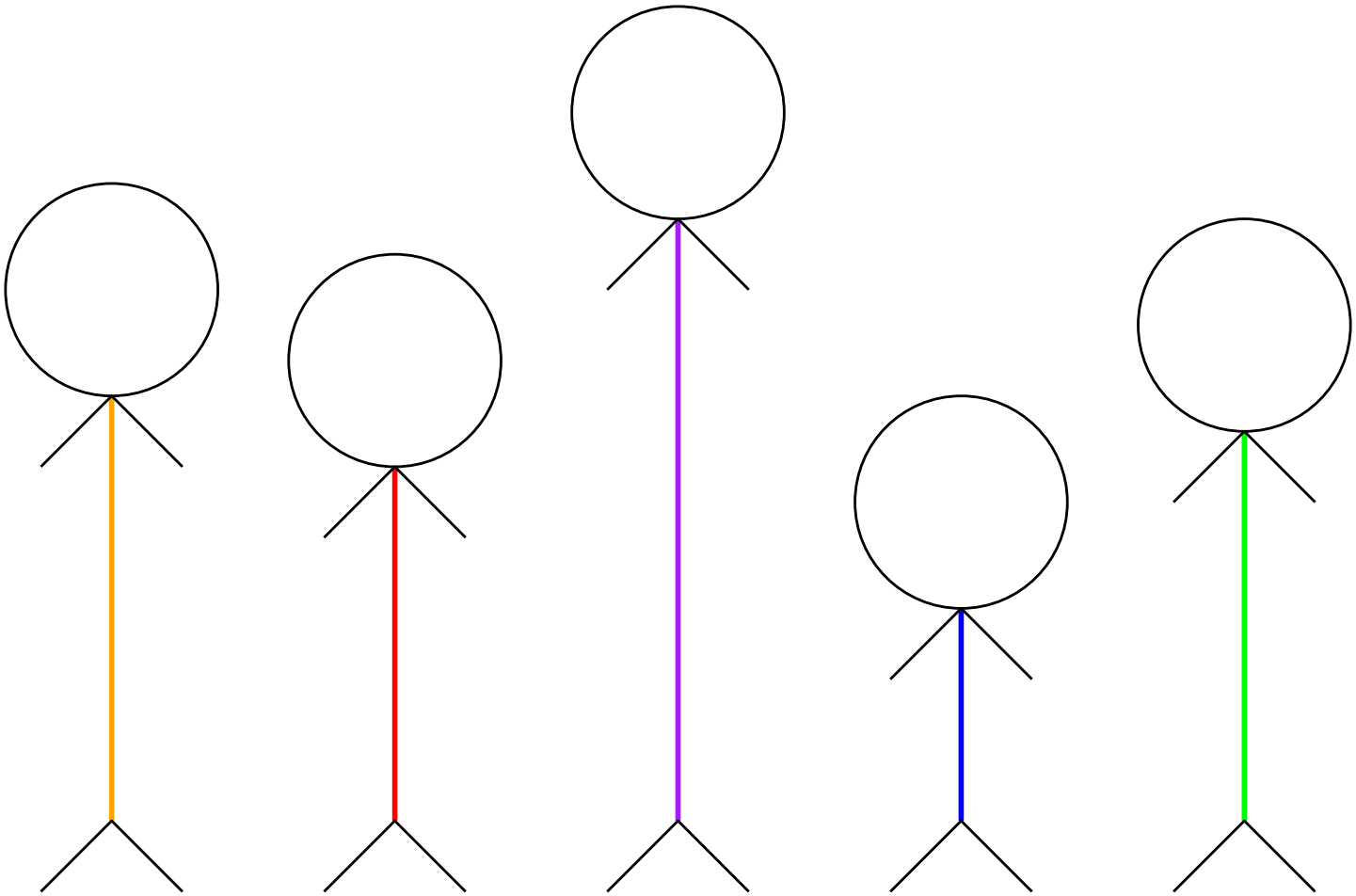


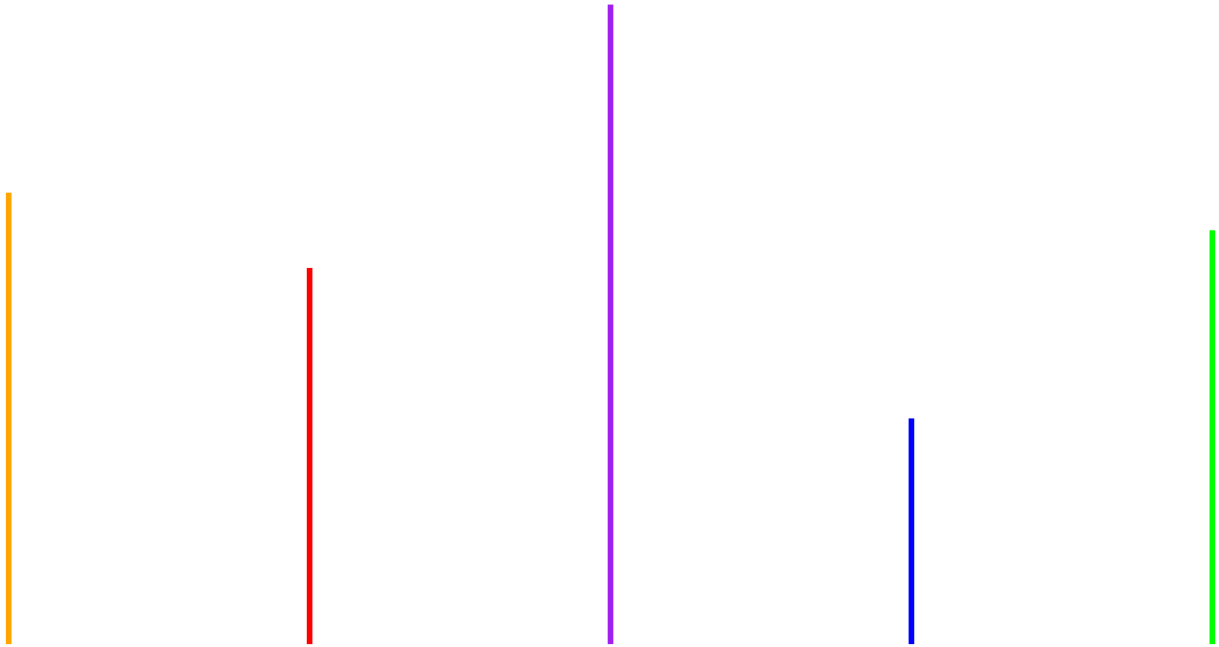
Zadanie C
Koło (z) Matematyków

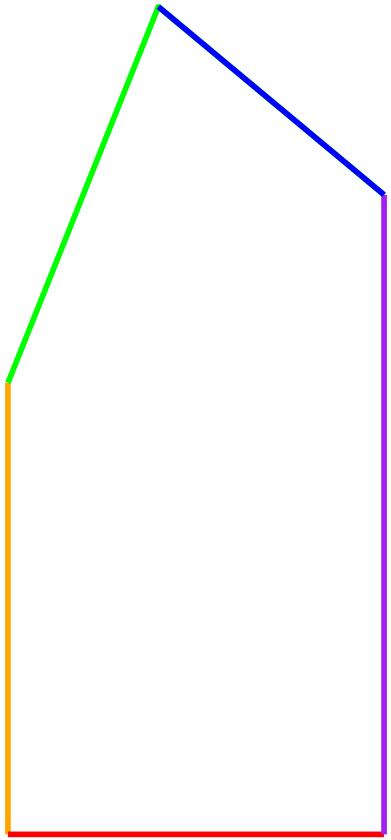
Zgłoszeń: 64
Zaakceptowanych: 42

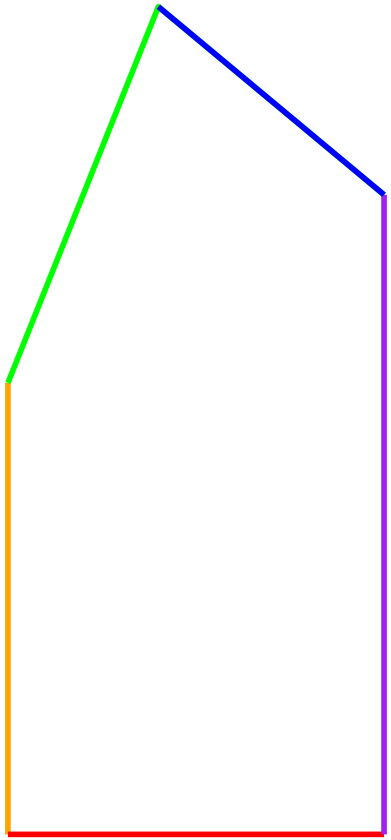
Pierwsze rozwiązanie:
uw7 [Partyka, Wiśniewski, Wasilewski]
00:07:36

Autor: Vladyslav Hlembotskyi









Zadanie D
›aby

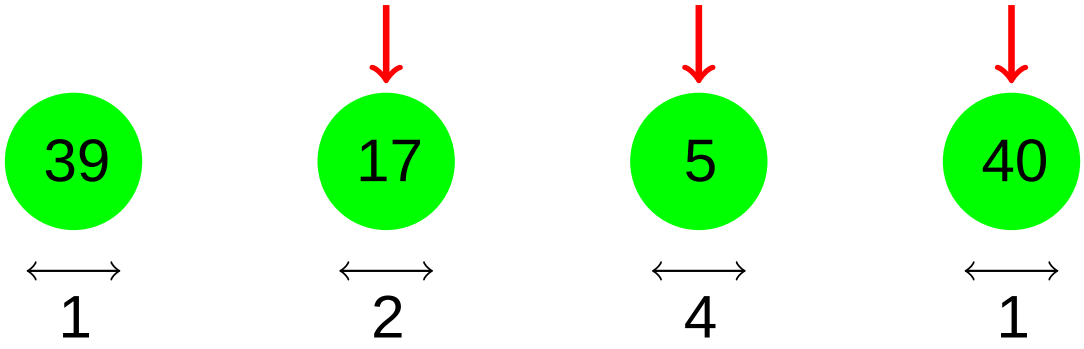
Zgªosze«: 74
Zaakceptowanych: 35

Pierwsze rozwi¡zanie:
uj2 [Zub, Orap, Denha]
00:08:57

Autor: Kamil D¦bowski

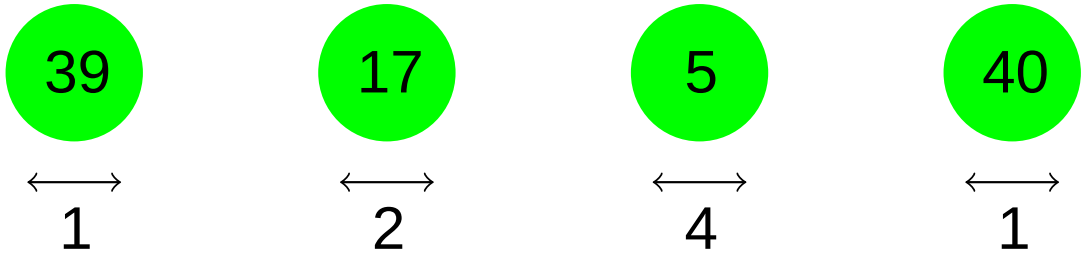
Jest n kamieni w rz¦dzie, na ka»dym mieszka jedna »aba. Ka»da »aba m
pewien poziom s_i oraz mo»e skoczy¢ raz w lewo lub w prawo o co najwy»
r_i. Chcemy wybra¢ dru»yn¦ za³o»on¡ z trzech »ab, dla której:

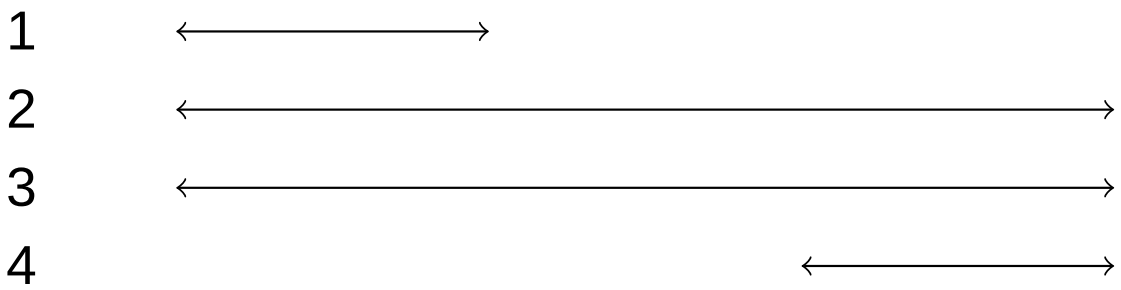
- ca³a dru»yna mo»e trenowa¢ na jednym kamieniu, tj. istnieje kamie
  na który mog¡ skoczy¢ wszystkie trzy »aby,
- suma poziomów tych »ab jest jak najwi¦ksza.



Dla ka»dej »aby mo»emy znale¹¢ jej si¦g, to jest znale¹¢ pierwszy i ostatni kamie«, na który mo»e skoczy¢:

$$[\max(1; i - r_i); \min(n; i + r_i)]$$





Pocz¡tki i ko«ce zasi¦gów nazwiemy zdarzeniami i b¦dziemy rozpatrywa¢ od lewej do prawej (wcze±niej je sortuj¡c):

- kiedy rozpoczyna si¦ zasi¦g nowej »aby, dorzucamy jej poziom do struktury (np. multiseta),
- je»eli w strukturze mamy co najmniej trzy »aby, liczymy sum¦ poziomów trzech najlepszych »ab i aktualizujemy wynik,
- kiedy ko«czy si¦ zasi¦g danej »aby, usuwamy jej poziom.

Warto zwróci¢ uwag¦, »e je»eli kamie« jest zarówno pocz¡tkiem i ko«cem zasi¦gu dla niektórych »ab, to najpierw trzeba rozwa»y¢ pocz¡tki (dodawanie do struktury), a pó¹niej ko«ce (usuwanie ze struktury), tak aby te »aby mog³y si¦ spotka¢ na tym kamieniu.

Z³o»ono±¢ czasowa $O(n \log n)$.

Zadanie E
Drony

Zgªosze«: 21
Zaakceptowanych: ??

Pierwsze rozwi¡zanie:
?

Autor: Krzysztof Maziarz

W przestrzeni trójwymiarowej znajduje si¦ dronów, pocz¡tkowo w pozycjach$(x_i; y_i; 0)$ dla $i \in \{1, 2, \ldots, n\}$. Pomi¦dzy dronami jestm poª¡cze«, które tworz¡ spójny graf planarny. Dla ka»dego z poª¡cze« znana jest maksymalna dªugo±¢ { je±li drony które ª¡czy oddal¡ si¦ bardziej ni» ta odlegªo±¢, poª¡czenie si¦ zrywa.

Otrzymujemyk manewrów; jeden manewr to zmiana trzeciej wspóªrz¦dne jednego z dronów. Podczas manewrów niektóre poª¡czenia si¦ zrywaj¡.

Dla ka»dej zq par dronów$(u_i; v_i)$ rozstrzygn¡¢, kiedy przestaªa istnie¢ ±cie»ka pomi¦dzyu_i orazv_i (prowadz¡ca po niezerwanych poª¡czeniach).

Zauwa»my najpierw, »e aby odpowiedzie¢ na zapytania, wystarczy stwierdzi¢ dla ka»dej kraw¦dzi kiedy si¦ zerwa ª a.

Zauwa»my najpierw, »e aby odpowiedzie¢ na zapytania, wystarczy stwierdzi¢ dla ka»dej kraw¦dzi kiedy si¦ zerwa ª a.

Mo»emy przewtorzy¢ kraw¦dzie od najpó ¹ niej do najwcze ± niej zerwanej ("odwróci¢ czas").

Zauwa»my najpierw, »e aby odpowiedzie¢ na zapytania, wystarczy stwierdzi¢ dla ka»dej kraw¦dzi kiedy si¦ zerwa³a.

Mo»emy przewtorzy¢ kraw¦dzie od najpó¹niej do najwcze±niej zerwanej ("odwróci¢ czas").

Nasze zapytania przyjmuj¡ posta¢: dla ka»dej pary (u_i, v_i), podaj kiedy wierzcho³ki u_i oraz v_i znalaz³y si¦ w jednej spójnej sk³adowej.

Zauwa»my najpierw, »e aby odpowiedzie¢ na zapytania, wystarczy stwierdzi¢ dla ka»dej kraw¦dzi kiedy si¦ zerwa³a.

Mo»emy przewtorzy¢ kraw¦dzie od najpó¹niej do najwcze±niej zerwanej ("odwróci¢ czas").

Nasze zapytania przyjmuj¡ posta¢: dla ka»dej pary (v_i, y), podaj kiedy wierzcho³ki v_i oraz v_i znalaz³y si¦ w jednej spójnej sk³adowej.

Utrzymujemy obecny podzia³ wierzcho³ków na sk³adowe, oraz dla ka»dej z nich zbiór identyfikatorów zapyta« o jednym ko«cu w tej sk³adowej. Š¡cz¡c przerzucamy "mniejszy do wi¦kszego".

Zauwa»my najpierw, »e aby odpowiedzie¢ na zapytania, wystarczy stwierdzi¢ dla ka»dej kraw¦dzi kiedy si¦ zerwaªa.

Mo»emy przewtorzy¢ kraw¦dzie od najpó¹niej do najwcze±niej zerwanej ("odwróci¢ czas").

Nasze zapytania przyjmuj¡ posta¢: dla ka»dej pary (u_i, v_i), podaj kiedy wierzchoªki u_i oraz v_i znalazªy si¦ w jednej spójnej skªadowej.

Utrzymujemy obecny podziaª wierzchoªków na skªadowe, oraz dla ka»dej z nich zbiór identyfikatorów zapytaº o jednym ko«cu w tej skªadowej. Šª¡czenie przerzucamy "mniejszy do wi¦kszego".

Ta cz¦±¢ rozwi¡zania działa w czasie $\mathcal{O}(m \log^2 m)$.

Pozostaje wyznaczy¢ dla ka»dej kraw¦dzi kiedy si¦ zerwa ª a.

Pozostaje wyznaczy¢ dla ka»dej kraw¦dzi kiedy si¦ zerwa ª a.

Dla uproszczenia zauwa»my, »e skoro pierwsze dwie wspó ª rz¦dne dron
pozostaj¡ sta ª e, to znaj¡c maksymaln¡ d ª ugo±¢ na któr¡ mo»na rozci¡gn¡
kraw¦d¹ (u; v) mo»emy wyliczy¢ maksymaln¡ ró»nic¦ trzecich
wspó ª rz¦dnych wierzcho ª ków u i v.

Pozostaje wyznaczy¢ dla ka»dej kraw¦dzi kiedy si¦ zerwa^a.

Dla uproszczenia zauwa»my, »e skoro pierwsze dwie wspó^arz¦dne dronó
pozostaj¡ sta^e, to znaj¡c maksymaln¡ d^ugo±¢ na któr¡ mo»na rozci¡gn¡
kraw¦d¹ (u; v) mo»emy wyliczy¢ maksymaln¡ ró»nic¦ trzecich
wspó^arz¦dnych wierzcho^ków u i v.

Chcemy zasymulowa¢ manewry, wykrywaj¡c zrywaj¡ce si¦ kraw¦dzie.
Niestety, naiwne zasymulowanie manewru na wierzchołku v zaj¦^aby
czasO(deg(v)).

Fakt: w gra e planarnym istnieje wierzcho^{a}ek o stopniu co najwy»ej 5. Wynika to na przyk^{a}ad z tego, »e liczba kraw¦dzi grafu planarnego jest mniejsza ni» $3n$ (lub równowa»nie ze wzoru Eulera).

Fakt: w grafie planarnym istnieje wierzchołek o stopniu co najwyżej 5. Wynika to na przykład z tego, że liczba krawędzi grafu planarnego jest mniejsza niż $3n$ (lub równoważnie ze wzoru Eulera).

Istnieje więc porządek wierzchołków, taki że każdy wierzchołek ma tylko 5 krawędzi "w prawo".

Symulujemy manewry, ale gdy wykonujemy manewr na wierzchołku v, to przeglądamy tylko krawędzie z v w prawo.

Symulujemy manewry, ale gdy wykonujemy manewr na wierzchoªku v, to przegl¡damy tylko krawªdzie z v w prawo.

Patrz¡c na krawªd¹ (v; u), to mo»e by¢ tak, »e zerwaª j¡ wªa±nie wykonany manewr - ale ta sytuacja jest prosta. Musimy jeszcze uwzgl¦dni mo»liwo±¢, »e zerwaª j¡ pewien manewr wykonywany gdzie± pomiªdzy poprzednim manewrem na a obecnym.

Symulujemy manewry, ale gdy wykonujemy manewr na wierzchoªku v, to przegl¡damy tylko kraw¦dzie z v w prawo.

Patrz¡c na kraw¦d¹ $(v;u)$, to mo»e by¢ tak, »e zerwaª j¡ wªa±nie wykonany manewr - ale ta sytuacja jest prosta. Musimy jeszcze uwzgl¦dni mo»liwo±¢, »e zerwaª j¡ pewien manewr wykonywany wcze±niej gdzie± pomi¦dzy poprzednim manewrem na a obecnym.

Pomi¦dzy dwoma kolejnymi manewrami jego trzecia wspóªrz¦dna byªa staªa. ›eby kraw¦d¹ $(v;u)$ si¦ nie zerwaªa, trzecia wspóªrz¦dna musiaªa caªy czas znajdowa¢ si¦ w odpowiednim przedziale.

Chcemy wiedzie¢, czy w pewnym przedziale czasu $[l, r]$, trzecia wspó^{a}rz¦dnau wysz^{a}a poza przedzia^a $[h_l; h_r]$, i je±li tak, to jaki by^a pierwszy moment kiedy to si¦ sta^{a}o.

Chcemy wiedzie¢, czy w pewnym przedziale czasu $[h_l, h_r]$, trzecia wspó³rz¦dnau wysza poza przedzia³ $[h_l; h_r]$, i je±li tak, to jaki by³ pierwszy moment kiedy to si¦ sta³o.

Dla ka»dego u, mo»emy rozwa»y¢ sekwencj¦ warto±ci jego trzecej wspó³rz¦dnej, i nad t¡ sekwencj¡ zbudowa¢ drzewo przedzia³owe z minimami i maksimami. Wtedy na powy»sze pytanie mo»emy odpowiedzie¢ w czasie $\mathcal{O}(\log k)$ chodz¡c po drzewie przedzia³owym.

Chcemy wiedzie¢, czy w pewnym przedziale czasu $[l, r]$, trzecia wspóªrz¦dnau wyszªa poza przedziaª $[h_l; h_r]$, i je±li tak, to jaki byª pierwszy moment kiedy to si¦ staªo.

Dla ka»degœu, mo»emy rozwa»y¢ sekwencj¦ warto±ci jego trzecej wspóªrz¦dnej, i nad t¡ sekwencj¡ zbudowa¢ drzewo przedziaªowe z minimami i maksimami. Wtedy na powy»sze pytanie mo»emy odpowiedzie¢ w czasie $\mathcal{O}(\log k)$ chodz¡c po drzewie przedziaªowym.

Finalna zªo»ono±¢: $\mathcal{O}((n + k) \log(n + k) + m \log^2 m)$.

Zadanie F
Fantastyczna kompresja

Zgªosze«: 64
Zaakceptowanych: 20

Pierwsze rozwi¡zanie:
uw6 [Kowalska, Boguta, Kaszuba]
00:56:46*

Autor: Krzysztof Maziarz

Dana jest skompresowana permutacja w postaci sumy wszystkich jej spójnych podci¡gów o d³ugo±ciach (k 6). Znale¹¢ wszystkie permutacje pasuj¡ce do te skompresowanej postaci.

Dla skompresowanej postacikz= 3:

$$[8; 10; 12]$$

pasuj¡ce permutacje to:

$$[1; 2; 5; 3; 4] \text{ oraz } [2; 1; 5; 4; 3]$$

Zauwa»my, »e maj¡c s¡siednie sumy, tj. s_j oraz s_{j+1}, mo»emy znale¹¢ ró»nic¦ mi¦dzy elementem a_i oraz a_{i+k}:

$$s_j = a_i + a_{i+1} + \ldots + a_{i+k-1}$$
$$s_{j+1} = a_{i+1} + a_{a+2} + \ldots + a_{i+k}$$
$$s_{j+1} - s_j = a_{i+k} - a_i$$

Mo»emy zatem znale¹¢ t¦ ró»nic¦ dla ka»dej pary elementów odleg³ych o dok³adniek.

Mamy zatemk ci¡gów, dla których znamy wszystkie ró»nice pomi¦dzy elementami:

$$(a_0; a_k; a_{2k}; : : : ;)$$
$$(a_1; a_{k+1}; a_{2k+1}; : : : ;)$$
$$(a_2; a_{k+2}; a_{2k+2}; : : : ;)$$
$$: : :$$

Wystarczy, »e znajdziemy warto±¢ jednego elementu w takim ci¡gu i znamy warto±ci wszystkich elementów.

Wiemy jednak, »e wszystkie te elementy musz¡ tworzy¢ permutacj¦, zatem na pewno który± z tych ci¡gów musi zawiera¢ 1.

Mo»emy sprawdzi¢ ka»dy z tych k ci¡gów (k jest niewielkie) jako potencjalny ci¡g zawieraj¡cy 1.

Wtedy poznajemy wszystkie inne warto±ci liczb z tego ci¡gu. Teraz mo»emy znale¹¢ kolejn¡ najmniejsz¡ liczb¦ i wybra¢ który± z pozosta³ych $k - 1$ ci¡gów, jako ten który zawiera t¦ liczb¦.

To prowadzi do rekurencji, która sprawdza wszystkie $k!$ mo»liwo±ci. Warto zauwa»y¢, »e istnieje co najwy»ej $k!$ permutacji, które pasuj¡ do zakodowanej postaci.

Na ko«cu warto sprawdzi¢, czy uzyskana permutacja faktycznie pasuje, co mo»emy ³atwo zrobi¢ licz¡c wszystkie sumy (okienkiem, b¡d¹ brutalnie).

Z³o»ono±¢ czasowa $O(n \cdot k!)$.

Zadanie G
Antykwariat

Zgªosze«: 54
Zaakceptowanych: 8

Pierwsze rozwi¡zanie:
uwr01 [Rzepecki, Górniak, Agrawal]
01:02:36*

Autor: Krzysztof Maziarz

Mamy dany ci¡g n liczb $a_1; :::; a_n$ z przedzia³u $[1; M]$. Dostajemy du»o zapyta« $(L; R)$, dla ka»dego musimy poda¢ liczb¦ spójnych przedzia³ów naszego ci¡gu które maj¡ wszystkie warto±ci w przedziale $[L; R]$.

Mamy dany ci¡g n liczb $a_1, \ldots, a_n$ z przedzia³u $[1; M]$. Dostajemy du»o zapyta« $(L; R)$, dla ka»dego musimy poda¢ liczb¦ spójnych przedzia³ów naszego ci¡gu które maj¡ wszystkie warto±ci w przedziale $[L; R]$.

W zadaniu mieli±my jeszcze jeden bardzo wa»ny warunek: elementy ci¡gu a_i zosta³y wygenerowane losowo (niezale»nie i jednostajnie z przedzia³u $[1; M]$). Dok³adniej, u»yli±my do tego funkcji rand48{ ale do rozwi¡zania nie trzeba wiedzie¢ jak ta funkcja dzia³a.

Spostrze»enie: je±li wylosujemy liczb z $[1; M]$, to prawdopodobie«stwo »e ostatnia z ich jest najwi¦ksza nie przekracza $\frac{1}{n}$

Spostrze»enie: je±li wylosujemy liczb z $[1; M]$, to prawdopodobie«stwo »e ostatnia z ich jest najwi¦ksza nie przekracza $\frac{1}{n}$.

Z tego wynika, »e je±li czytamy losowych liczb, i utrzymujemy obecne maksimum, to oczekiwana liczba zmian tego maksimum to

$$\sum_{i=1}^{n} \frac{1}{i} = O(\log n)$$

Spostrze»enie: je±li wylosujemy liczb z $[1; M]$, to prawdopodobie«stwo »e ostatnia z ich jest najwi¦ksza nie przekracza $\frac{1}{n}$.

Z tego wynika, »e je±li czytamy losowych liczb, i utrzymujemy obecne maksimum, to oczekiwana liczba zmian tego maksimum to

$$\sum_{i=1}^{n} \frac{1}{i} = O(\log n)$$

Oczywi±cie je±li utrzymujemy par¦ $(\min; \max)$, to oczekiwana liczba zmian któregokolwiek komponentu tej pary to równie» $O(\log n)$.

Czytamy ci¡g od prawej do lewej. Gdy jeste±my na pozycji j, utrzymujemy stos zawieraj¡cy takie pozycje i, na których zmienia si¦ maksimum ci¡gu $a[i;:::;j]$. Analogicznie utrzymujemy drugi stos dla minimum.

Czytamy ci¡g od prawej do lewej. Gdy jeste±my na pozycji j, utrzymujemy stos zawieraj¡cy takie pozycje i, na których zmienia si¦ maksimum ci¡gu $a[i;:::;j]$. Analogicznie utrzymujemy drugi stos dla minimum.

Šatwo mo»emy aktualizowa¢ nasze stosy gdy przesuwamy si¦ z obserwacji i na $i-1$ z poprzedniego slajdu ich rozmiar zawsze b¦dzie $O(\log n)$.

Czytamy ci¡g od prawej do lewej. Gdy jeste±my na pozycji j, utrzymujemy stos zawieraj¡cy takie pozycje i, na których zmienia si¦ maksimum ci¡gu $a[i; :::; j]$. Analogicznie utrzymujemy drugi stos dla minimum.

Šatwo mo»emy aktualizowa¢ nasze stosy gdy przesuwamy si¦ z obserwacji i na $i-1$ z poprzedniego slajdu ich rozmiar zawsze b¦dzie $O(\log n)$.

Stosy pozwalaj¡ nam rozwa»y¢ wszystkie przedziaªy o lewym ko«cu i i w pogrupowa¢ je po parze $(\min; \max)$ na $O(\log n)$ grup. Grupuj¡c tak dla ka»dego i otrzymujemy podziaª wszystkich $O(n^2)$ podprzedziaªów na $O(n \log n)$ grup (ka»da grupa ma to samo minimum i maksimum).

Zamiatamy warto±ci od M do 1. Gdy rozwa»amy warto±¢ x, to najpierw dodajemy do naszej struktury wszystkie grupy o min = x, a nast¦pnie odpowiadamy na zapytania b = x.

Zamiatamy warto±ci od M do 1. Gdy rozwa»amy warto±¢ x, to najpierw dodajemy do naszej struktury wszystkie grupy y o $min = x$, a nast¦pnie odpowiadamy na zapytania $b = x$.

Nasza struktura musi obs³ugiwa¢ operacj¦ dodania grupy o zadanym max i pewnej liczno±ci, oraz zapytania o sum¦ liczno±ci grup o $max \leq x$. Do tego wystarczy proste drzewo przedzia³owe.

Zamiatamy warto±ci od M do 1. Gdy rozwa»amy warto±¢ x, to najpierw dodajemy do naszej struktury wszystkie grupy y dla $n = x$, a nast¦pnie odpowiadamy na zapytania $b = x$.

Nasza struktura musi obs³ugiwa¢ operacj¦ dodania grupy o zadanym y dla pewnej liczno±ci, oraz zapytania o sum¦ liczno±ci grup dla x. Do tego wystarczy proste drzewo przedzia³owe.

Wykonujemy $O(n \log n)$ operacji na drzewie, wi¦c z³o»ono±¢ rozwi¡zania to $O(n \log^2 n)$.

Zadanie H
Oscypki

Zgªosze«: 81
Zaakceptowanych: 31

Pierwsze rozwi¡zanie:
uwr01 [Rzepecki, Górniak, Agrawal]
00:12:29*

Autor: Krzysztof Maziarz

Zadanie I
Henryk Portier i Promie« Palindromiczny

Zgªosze«: 78
Zaakceptowanych: 24

Pierwsze rozwi¡zanie:
uj4 [Grzybowska, Rajtar, Burczy«ski]
00:29:06

Autor: Krzysztof Maziarz

Jest pewne sªowo A zªo»one ze znaków o wadze 0,1 g. Nie znamy jednak tego sªowa, a tylko promienie palindromiczne, czyli dla ka»dego i { dªugo±¢ najdªu»szego palindromu o ±rodku A[i]. Zadaniem jest odtworzy¢ sªowo A { je±li jest wi¦cej rozwi¡za«, trzeba wypisa¢ wszystkie.

Jest pewne sªowo A zªo»one ze znaków 0,1 g. Nie znamy jednak tego sªowa, a tylko promienie palindromiczne, czyli dla ka»dego i { dªugo±¢ najdªu»szego palindromu o ±rodku A[i]. Zadaniem jest odtworzy¢ sªowo A { je±li jest wi¦cej rozwi¡za«, trzeba wypisa¢ wszystkie.

Pierwsze dwa znaki to 00, 01, 10 lub 11. Sprawd¹my wszystkie te mo»liwo±ci.

Jest pewne s^aowo A z^ao»one ze znakówf0,1 g. Nie znamy jednak tego s^aowa, a tylko promienie palindromiczne, czyli dla ka»dego i { d^augo±¢ najd^au»szego palindromu o ±rodkuA[i]. Zadaniem jest odtworzy¢ s^aowo A { je±li jest wi¦cej rozwi¡za«, trzeba wypisa¢ wszystkie.

Pierwsze dwa znaki to 00, 01, 10 lub 11. Sprawd¹my wszystkie te mo»liwo±ci.

Dla i 3 mo»emy ^atwo wyznaczy¢A[i] { je±li promie« wA[i 1] jest równy co najmniej 1, to musi by¢A[i] = A[i 2]. Je±li jest równy 0, to musi by¢A[i] 6= A[i 2], co te» wyznacza A[i] jednoznacznie.

Czyli na podstawie pierwszych dwóch znaków i promieni palindromiczny wyznaczamy ca³e s³owo. Ale nie mamy gwarancji, »e nasze s³owo faktycznie pasuje do podanych promieni.

Dla ka»dego z 4 otrzymanych s³ów sprawdzamy zatem, czy dla niego wysz³yby takie promienie, jak podane w zadaniu. S³u»y do tego tzw. algorytm Manachera, ale mo»na te» radzi¢ sobie bez niego (np. hashowaniem).

Ciekawostka: zawsze jest albo 0, albo 4 rozwi¡zania.

Zadanie J
Anteny

Zgªosze«: 4
Zaakceptowanych: ??

Pierwsze rozwi¡zanie:
?

Autor: Daniel Goc

Dany jest wielok¡t wypuk²y im punktów w ±rodku. Mamy ci¡g zapyta« postaci ÿdla danej pary boków wyznaczyc liczb¦ prostych przechodz¡cyc przez pewne dwa spo±ród tych punktów".

›adne 3 punkty nie s¡ wspó²liniowe.

Dany jest wielok¡t wypuk²y im punktów w ±rodku. Mamy ci¡g zapyta« postaci ÿdla danej pary boków wyznaczyc liczb¦ prostych przechodz¡cyc przez pewne dwa spo±ród tych punktów".

›adne 3 punkty nie s¡ wspó²liniowe.

Dany jest wielok¡t wypuk²y im punktów w ±rodku. Mamy ci¡g zapyta« postaci ÿdla danej pary boków wyznaczyc liczb¦ prostych przechodz¡cyc przez pewne dwa spo±ród tych punktów".

›adne 3 punkty nie s¡ wspó²liniowe.

Dany jest wielok¡t wypuk ay im punktów w ±rodku. Mamy ci¡g zapyta« postaci ÿdla danej pary boków wyznaczy¢ liczb¦ prostych przechodz¡cyc przez pewne dwa spo±ród tych punktów".

›adne 3 punkty nie s¡ wspó aliniowe.

Rozwi¡zanie: obs ugujemy ka»de zapytanie osobno w czasie $O(n \log m)$.

$$\#\text{ prostych przecinaj¡cych bokiaa}^0\text{ oraz bb}^0 =$$
$$\#\text{ prostych poni»ej odcinkæab}$$
$$-\ \#\text{ prostych poni»ej odcinkæab}^0$$
$$-\ \#\text{ prostych poni»ej odcinkæa}^0\text{b}$$
$$+\ \#\text{ prostych poni»ej odcinkaa}^0\text{b}^0$$

Ile prostych jest poni»ej odcinka ab?

~~Ile prostych jest ponizej odcinka ab?~~

Zostawmy tylko punkty poni»ej odcinka ab.

Ile prostych przecina odcinek ab?

~~Ile prostych jest ponizej odcinka ab?~~

Zostawmy tylko punkty poni»ej odcinka ab.

Ile prostych przecina odcinek ab?

Jaki warunek musi spe³niaæ punkt x, by prosta xy przecina³a odcinek ab?

~~Ile prostych jest ponizej odcinka ab?~~

Zostawmy tylko punkty poni»ej odcinka ab.

Ile prostych przecina odcinek ab?

Jaki warunek musi spe^{a}nia¢ punkt x, by pó^{a}prosta xy przecina^{a}a odcinek ab?

~~Ile prostych jest ponizej odcinka ab?~~

Zostawmy tylko punkty poni»ej odcinka ab.

Ile prostych przecina odcinek ab?

Jaki warunek musi spe^{a}nia¢ punkt x, by pó^{a}prosta xy przecina^{a}a odcinek ab?

Punkt y musi nale»e¢ do trójk¡tæaxb.

WKW na przeci¦cie pó³prostej xy z odcinkiem ab:
Punkt y musi nale»e¢ do trójk¡ta axb. ($\angle$)
W porz¡dku k¡towym wokó³ a punkt y musi by¢ na lewo od punktu x,
za± w porz¡dku k¡towym wokó³ b punkt y musi by¢ na prawo od punktu x.

WKW na przeci¦cie prostej xy z odcinkiem ab:
w porz¡dku k¡towym wokó³ a punkty x, y musz¡ le»e¢ INACZEJ ni» w porz¡dku k¡towym wokó³ b.

Chcemy policzy¢ pary x, y, le»¡ce wzgl¦dem siebie w jednym porz¡dku inaczej ni» w drugim.

Problem sprowadza si¦ do policzenia inwersji w ci¡gu, który rozwi¡zuje si w czasie $O(m \log m)$ np. za pomoc¡ algorytmu mergesort.

Zadanie K
Duch

Zgªosze«: 21
Zaakceptowanych: ??

Pierwsze rozwi¡zanie:
?

Autor: Krzysztof Maziarz

Dane s¡ prostok¡ty, ka»dy poruszaj¡cy si¦ ruchem liniowym z jakim± wektorem pr¦dko±ci. Policz najwi¦ksze pole przeci¦cia prostok¡tów w ka»dym z zadanych przedziaªów czasu.

Ka»dy prostok¡t daje cztery niezale»ne ograniczenia (pó ª p ª aszczyzny) n
przeci¦cie prostok¡tów: dolne i górne ograniczenie na x oraz na y.

Skupmy si¦ na analizowaniu i wyznaczaniu ogranicze« na x.

Ka»dy prostok¡t wyznacza nast¦puj¡ce ograniczenia w momencie t:

$$x \quad x_1 + t \quad v_x$$

$$x \quad x_2 + t \quad v_x$$

N ogranicze« dolnych daje nam N funkcji liniowych (pó³p³aszczyzn), to samo dla górnych.

Wysoko±¢ obszaru pomi¦dzy ograniczeniami dolnymi a górnymi jest rów
szeroko±ci przeci¦cia prostok¡tów z zadania, co nazwiesz$szer(t)$ dla
momentu$t$.

Wysoko±¢ obszaru pomi¦dzy ograniczeniami dolnymi a górnymi jest rów
szeroko±ci przeci¦cia prostok¡tów z zadania, co nazwiesz(t) dla
momentu t.

Funkcja szer(t) jest bitoniczna, kawa³kami liniowa oraz wkl¦s³a, o ile
ograniczymy si¦ do przedzia³u czasu, gdzie jest to funkcja niezerowa.

Funkcja wys(t) (dla wspó^{a}rz¦dnych y) spe^{a}nia podobne w^{a}asno±ci. Interesuje nas maksymalizowanie pola przeci¦cia czyli:

$$pole(t) = szer(t) \quad wys(t)$$

co jest bitoniczn¡ funkcj¡ kawa^{a}kami kwadratow¡.

Funkcja wys(t) (dla wspó^{a}rz¦dnych y) spe^{a}nia podobne w^{a}asno±ci. Interesuje nas maksymalizowanie pola przeci¦cia czyli:

$$pole(t) = szer(t) \quad wys(t)$$

co jest bitoniczn¡ funkcj¡ kawa^{a}kami kwadratow¡.

Dowód bitoniczno±ci pozostawiamy czytelnikowi.

Funkcja pole(t) nie musi by¢ ani wypuk³a ani wkl¦s³a.
(a nawet ró»nie bywa)

Funkcja pole(t) nie musi by¢ ani wypuk³a ani wkl¦s³a.
(a te schodki nie s¡ idealnie poziome)

Rozwi¡zanie I (bez bitoniczno±ci)

Zbieramy ograniczenia na x i wyznaczamy przeci¦cie pó³p³aszczyzn, to samo dla y. Dzielimy czas na $O(n)$ przedzialików, takich »e w ka»dym $szer(t)$ oraz $wys(t)$ s¡ funkcjami liniowymi { wtedy ich iloczyn jest funkcj¡ kwadratow¡.

W ka»dym przedzialiku szukamy maksimum wzorem lub wyszukiwaniem ternarnym, budujemy na tych warto±ciach drzewo przedzia³owe i w $O(\log n)$ odpowiadamy na zapytania, uwa»aj¡c na ko«cówki przedzia³u z zapytania.

Da si¦ ªatwiej. Skoro funkcja pole(t) jest bitoniczna, to kusi wyszukiwanie ternarne, by znale¹¢ globalne maksimum.

Ale nie da si¦ wyszukiwa¢ ternarnie po funkcji, która ma remisy poza maksimum!

Da si¦ ªatwiej. Skoro funkcja pole(t) jest bitoniczna, to kusi wyszukiwanie ternarne, by znale¹¢ globalne maksimum.

Ale nie da si¦ wyszukiwa¢ ternarnie po funkcji, która ma remisy poza maksimum!

...wi¦c ograniczamy si¦ do przedziaªu czasu, gdzie warto±ci s¡ niezerowe. Tam funkcja pole(t) ±ci±le ro±nie, potem by¢ mo»e ma remis na maksimum, potem ±ci±le maleje.

Rozwiązanie II (z bitonicznością)

Po znalezieniu przecięcia półpłaszczyzn wyszukujemy ternarnie (lub binarnie po pochodnej) globalne maksimum.

Jeśli zapytanie zawiera ten optymalny moment, to znamy od razu wynik. Jeśli nie, to wystarczy rozważyć tylko początek i koniec przedziału.

W tych dwóch momentach liczymy wynik przez policzenie $szer(t)$ i $wys(t)$ (wyszukujemy binarnie odcinek otoczki w którym ląduje t).

Złożoność obu rozwiązań to zgrubsza $O(n \log n)$. Rozwiązanie II ma logarytm z trochę większej wartości, bo musimy zrobić ternary search po zakresie (i do zadanej dokładności).

Ciekawostka: Rozwiązanie I da się zaimplementować w arytmetyce liczb wymiernych.

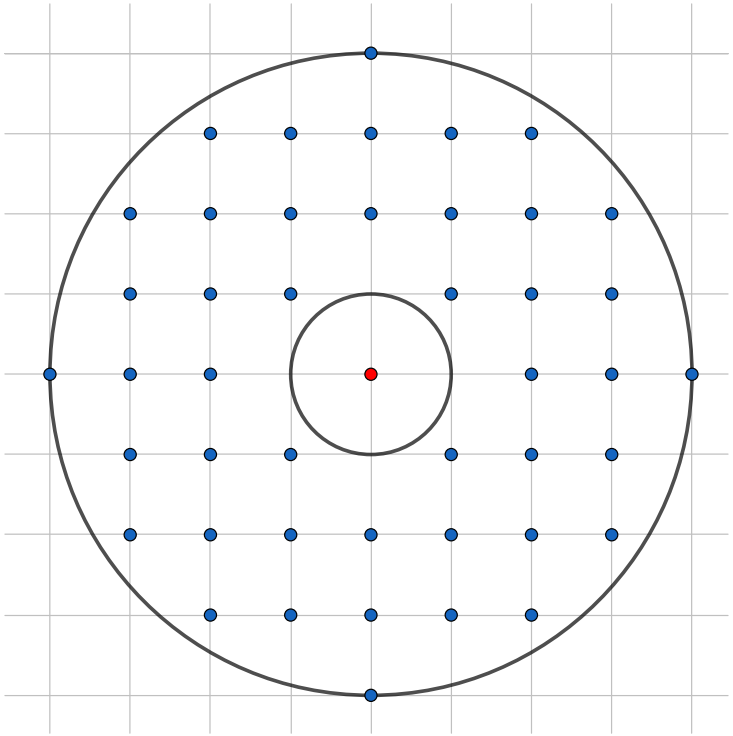
Zadanie L
Obwarzanek

Zgłoszeń: 11
Zaakceptowanych: 2

Pierwsze rozwiązanie:
uw1 [Kondraciuk, Skiba, Paluszek]
02:53:11

Autor: Krzysztof Maziarz

Dla punktu kratowego $(a; b)$ i liczb całkowitych nieujemnych L oraz R, rozważamy zbiór punktów kratowych których odległość od $(a; b)$ jest w przedziale $(L; R]$. Każdy zbiór który można uzyskać w opisany sposób dla pewnych $a; b; L; R$ nazywamy *obwarzankiem*.



Zaczynamy od pustego zbioru punktów S, i n razy dodajemy do niego nowy punkt kratowy. Po każdej takiej operacji należy stwierdzić, czy obecny zbiór S jest obwarzankiem.

Dodatkowo, mamy *bardzo mało dostępnej pamięci* – nie jesteśmy nawet w stanie utrzymywać w pamięci zbioru S.

Za to mamy gwarancję, że wszystkie punkty mają niezbyt duże współrzędne (na moduł nie przekraczające $C = 5000$).

Możemy utrzymywać najmniejszą i największą współrzędną x po punktach znajdujących się w S $(l_x; r_x)$, oraz analogicznie dla y $(l_y; r_y)$.

Warunkiem koniecznym aby obecny zbiór punktów był obwarzankiem jest $r_x \quad l_x = r_y \quad l_y$. Ponadto, możemy wyliczyć zewnętrzny promień

$$R = \frac{r_x \quad l_x}{2}$$

oraz środek

$$(a; b) = \quad \frac{l_x + r_x}{2}; \frac{l_y + r_y}{2}$$

Załóżmy, że dla każdego $r \in \{1; 2; \ldots; 5000\}$ znamy liczbę punktów kratowych zawartych w kole o środku w $(0; 0)$ i promieniu r. Wtedy mając wyliczony promień zewnętrzny R, i znając liczbę wczytanych punktów, możemy jednoznacznie wyznaczyć L.

Znamy już teraz wartości $a; b; L; R$ – albo S jest dokładnie takim obwarzankiem, albo nie jest obwarzankiem wcale.

By odpowiedzieć na pytanie czy S faktycznie jest obwarzankiem o zadanych parameterach, musimy utrzymywać jeszcze jeden rodzaj informacji. Niech p będzie liczbą pierwszą, a $h(x;y)$ pewną funkcją hashującą $\mathbb{Z}^2 \; ! \; \mathbb{Z}_p$.

Zbiór punktów S hashujemy sumując h po jego elementach:

$$hash(S) = \left(\sum_{(x,y) 2 S} h(x;y) \right) \bmod p$$

Zakładając, że $h(x;y)$ jesteśmy w stanie wyliczać w czasie stałym, możemy bez problemu utrzymywać wartość $hash(S)$.

Żeby S był zadanym obwarzankiem, musi zachodzić

$$hash(S) = (circHash(a; b; R) \quad circHash(a; b; L)) \mod p$$

gdzie $circleHash(a; b; r) = hash(f(x; y) : dist((x; y); (a; b)) \quad rg)$.

Załóżmy, że znane nam są wartości $circleHash(0; 0; r)$ dla $r\ 2\ f1; 2; :::; 5000g$. Jeśli funkcja h będzie miała taką własność, że znając sumaryczny hash zbioru możemy wyznaczyć hash tegoż zbioru po przesunięciu jego wszystkich punktów o zadany wektor, to z $circleHash(0; 0; r)$ wyznaczymy $circleHash(a; b; r)$.

Czas wybrać funkcję h. Niech

$$h(x;y) = r^x s^y \mod p$$

gdzie r i s są stałymi wylosowanymi jednostajnie z $[1; p \quad 1]$.

Po odpowiednim preprocessingu, $h(x;y)$ jestesmy w stanie liczyć w czasie stałym; ponadto, jeśli przesuniemy S o wektor $(a;b)$, to $hash(S)$ zmieni się na $r^a s^b hash(S)$.

Podczas naszego rozwiązania założylismy, że dla każdego $r \ 2 \ f1; 2; :::; 5000g$ znamy liczbę punktów kratowych w kole o promieniu r i środku w $(0; 0)$, oraz sumaryczny hash tych punktów.

Wartości te możemy policzyć na samym początku, wykonując preprocessing w czasie $O(C^2)$.

Złożoność: $O(C^2 + n \log C)$ (po drobnych optymalizacjach $O(C^2 + n)$).

Dlaczego to działa?

- szansa, że dla dwóch różnych zbiorów A, B zajdzie $hash(A) = hash(B)$ jest rzędu $O(\frac{C}{p})$ (z lematu Schwartza–Zippela)
- szansa, że zbiór S błędnie sklasyfikujemy jako obwarzanek, nie przekracza $O(\frac{C^2}{p})$

Jury zawodów

Kamil „Errichto” Dębowski
Lech Duraj
Grzegorz Guśpiel
Vladyslav Hlembotskyi
Bartosz Kostka
Krzysztof Maziarz
Adam Polak

Betatesterzy

Jan Gwinner
Witold Jarnicki
Mateusz Radecki
Maciej Wawro

Dziękujemy!