Omówienie zadań AMPPZ 2016

Jury zawodów

Instytut Informatyki Uniwersytetu Wrocławskiego

4 grudnia 2016

Zadania

Zadanie	**AC**
A: Słaby generator pseudolosowy	27
B: Jaś sam w Nowym Jorku	0
C: Generowanie figury	46
D: Winogrona	53
E: Taksówki	3
F: Sieć antyspołecznościowa	15
G: Parking	23
H: Anagramy	6
I: Górska wędrówka	11
J: Kamień, papier, nożyce	61
K: Model Jasia–Bohra	56
L: Trybunał konstytucyjny	3

J – Kamień, papier, nożyce (Karol Pokorski)

Zadanie

Mając dany ciąg (długości $\leqslant 1\,000\,000$) kolejno rzucanych przez przeciwnika symboli w grze kamień, papier, nożyce, wyznaczyć ciąg symboli, który wygra ostro więcej pojedynków i ma jak najmniej zmian symbolu na inny.

J – Kamień, papier, nożyce (Karol Pokorski)

Zadanie

Mając dany ciąg (długości $\leqslant 1\,000\,000$) kolejno rzucanych przez przeciwnika symboli w grze kamień, papier, nożyce, wyznaczyć ciąg symboli, który wygra ostro więcej pojedynków i ma jak najmniej zmian symbolu na inny.

Obserwacja

Wynik to zawsze 0 lub 1:

- jeśli liczby wystąpień symboli `K`, `P`, `N` są wszystkie równe to wynikiem jest 1,
- w p.p. 0.

D – Winogrona (Karol Pokorski)

Zadanie

Mając dane drzewo ukorzenione ($n \leqslant 1\,000\,000$), wyznaczyć jedną krawędź, której usunięcie maksymalizuje liczbę liści w spójnej składowej, która ma mniej liści.

D – Winogrona (Karol Pokorski)

Zadanie

Mając dane drzewo ukorzenione ($n \leqslant 1\,000\,000$), wyznaczyć jedną krawędź, której usunięcie maksymalizuje liczbę liści w spójnej składowej, która ma mniej liści.

- Liczymy rozmiary poddrzew każdego wierzchołka:
  - $S[u] = 1$, gdy u jest liściem (owocem),
  - $S[u] = \sum S[v]$, w p.p. (v to synowie u).

D – Winogrona (Karol Pokorski)

Zadanie

Mając dane drzewo ukorzenione ($n \leqslant 1\,000\,000$), wyznaczyć jedną krawędź, której usunięcie maksymalizuje liczbę liści w spójnej składowej, która ma mniej liści.

- Liczymy rozmiary poddrzew każdego wierzchołka:
  - $S[u] = 1$, gdy u jest liściem (owocem),
  - $S[u] = \sum S[v]$, w p.p. (v to synowie u).
- Po usunięciu krawędzi pomiędzy u oraz jego rodzicem, otrzymujemy dwa poddrzewa mające odpowiednio $S[u]$ oraz $S[1] - S[u]$ liści.

D – Winogrona (Karol Pokorski)

Zadanie

Mając dane drzewo ukorzenione ($n \leqslant 1\,000\,000$), wyznaczyć jedną krawędź, której usunięcie maksymalizuje liczbę liści w spójnej składowej, która ma mniej liści.

- Liczymy rozmiary poddrzew każdego wierzchołka:
  - $S[u] = 1$, gdy u jest liściem (owocem),
  - $S[u] = \sum S[v]$, w p.p. (v to synowie u).
- Po usunięciu krawędzi pomiędzy u oraz jego rodzicem, otrzymujemy dwa poddrzewa mające odpowiednio $S[u]$ oraz $S[1] - S[u]$ liści.
- Wybieramy ten wierzchołek u, dla którego minimum z tych wartości jest największe możliwe.

K – Model Jasia–Bohra (Maciej Duleba)

Zadanie

Dany jest mały (multi)zbiór liczb naturalnych B ($|B| \leqslant 10$). Zbiór A zdefiniowany jest następująco:

- $n \in A$ ($n \leqslant 10^{15}$),
- $\left(\forall_{x \in \mathbb{N} \cup \{0\}}\right) (x \in A) \Rightarrow (\forall_{b \in B}) \left(\frac{x}{b} \in A\right)$.

Podać minimalną możliwą moc zbioru A.

K – Model Jasia–Bohra (Maciej Duleba)

Zadanie

Dany jest mały (multi)zbiór liczb naturalnych B ($|B| \leqslant 10$). Zbiór A zdefiniowany jest następująco:

- $n \in A$ ($n \leqslant 10^{15}$),
- $\left(\forall_{x \in \mathbb{N} \cup \{0\}}\right) (x \in A) \Rightarrow (\forall_{b \in B}) \left(\frac{x}{b} \in A\right)$.

Podać minimalną możliwą moc zbioru A.

Obserwacja

Szukany wynik jest mały.

K – Model Jasia–Bohra (Maciej Dulęba)

- Najgorszy przypadek, gdy B to dziesięć najmniejszych liczb pierwszych. Wówczas $|A| = 458123$.

K – Model Jasia–Bohra (Maciej Duleba)

- Najgorszy przypadek, gdy B to dziesięć najmniejszych liczb pierwszych. Wówczas $|A| = 458123$.
- Można brutalnie wygenerować wszystkie elementy zbioru A.

K – Model Jasia–Bohra (Maciej Duleba)

- Najgorszy przypadek, gdy B to dziesięć najmniejszych liczb pierwszych. Wówczas $|A| = 458123$.
- Można brutalnie wygenerować wszystkie elementy zbioru A.
- Należy umieć stwierdzać ile **różnych** liczb napotkano (np. `std::unordered_set`), uważać na powtórzenia w zbiorze B oraz sytuację, gdy $1 \in B$.

G – Parking (Karol Pokorski)

Zadanie

Zaproponować strukturę danych, która umożliwia obsługę parkingu równoległego na prostej:

- parkowanie w najkrótszej wolnej luce (w przypadku remisu: najwcześniejszej),
- wyjeżdżanie z parkingu.

G – Parking (Karol Pokorski)

Zadanie

Zaproponować strukturę danych, która umożliwia obsługę parkingu równoległego na prostej:

- parkowanie w najkrótszej wolnej luce (w przypadku remisu: najwcześniejszej),
- wyjeżdżanie z parkingu.

- Numery rejestracyjne zamieniamy na identyfikatory numerowe (np. za pomocą `std::unordered_map`).

G – Parking (Karol Pokorski)

Zadanie

Zaproponować strukturę danych, która umożliwia obsługę parkingu równoległego na prostej:

- parkowanie w najkrótszej wolnej luce (w przypadku remisu: najwcześniejszej),
- wyjeżdżanie z parkingu.

- Numery rejestracyjne zamieniamy na identyfikatory numerowe (np. za pomocą `std::unordered_map`).
- Luki zapisujemy dwojako (na przykład na `std::set` z odpowiednim komparatorem):
  - posortowane po pozycji początku luki,
  - posortowane po długości luki a w przypadku remisu po pozycji początku luki.

G – Parking (Karol Pokorski)

Zadanie

Zaproponować strukturę danych, która umożliwia obsługę parkingu równoległego na prostej:

- parkowanie w najkrótszej wolnej luce (w przypadku remisu: najwcześniejszej),
- wyjeżdżanie z parkingu.

- Numery rejestracyjne zamieniamy na identyfikatory numerowe (np. za pomocą `std::unordered_map`).
- Luki zapisujemy dwojako (na przykład na `std::set` z odpowiednim komparatorem):
  - posortowane po pozycji początku luki,
  - posortowane po długości luki a w przypadku remisu po pozycji początku luki.
- Oprócz tego tablica identyfikatorów samochodów stojących na parkingu (wraz z wskaźnikiem na luki w w/w zbiorach).

G – Parking (Karol Pokorski)

- Przyjazd samochodu:
  - znalezienie najkrótszej luki (lub zwrócenie `NIE`),
  - usunięcie z obu struktur,
  - ewentualne wstawienie nowej, krótszej luki.

G – Parking (Karol Pokorski)

- Przyjazd samochodu:
  - znalezienie najkrótszej luki (lub zwrócenie `NIE`),
  - usunięcie z obu struktur,
  - ewentualne wstawienie nowej, krótszej luki.
- Odjazd samochodu:
  - sprawdzenie czy samochód był w tablicy samochodów (+ ewentualne wypisanie `BRAK`),
  - usunięcie starej luki z obu struktur,
  - wstawienie nowej luki (być może sklejenie luk, jeśli to był jedyny samochód pomiędzy lukami).

C – Generowanie figury (Karol Pokorski)

Zadanie

Wygenerować figurę bez dziur o bokach prostopadłych do osi układu współrzędnych o ustalonym polu ($1 \leqslant a \leqslant 10^{12}$) i ustalonym obwodzie ($4 \leqslant p \leqslant 10^{6}$).

C – Generowanie figury (Karol Pokorski)

Zadanie

Wygenerować figurę bez dziur o bokach prostopadłych do osi układu współrzędnych o ustalonym polu ($1 \leqslant a \leqslant 10^{12}$) i ustalonym obwodzie ($4 \leqslant p \leqslant 10^6$).

- Zaczynamy od upewnienia się, że p jest parzyste.

C – Generowanie figury (Karol Pokorski)

Zadanie

Wygenerować figurę bez dziur o bokach prostopadłych do osi układu współrzędnych o ustalonym polu ($1 \leqslant a \leqslant 10^{12}$) i ustalonym obwodzie ($4 \leqslant p \leqslant 10^6$).

- Zaczynamy od upewnienia się, że p jest parzyste.
- Zauważamy, że pole nie może być mniejsze niż $p/2 - 1$.

C – Generowanie figury (Karol Pokorski)

Zadanie

Wygenerować figurę bez dziur o bokach prostopadłych do osi układu współrzędnych o ustalonym polu ($1 \leqslant a \leqslant 10^{12}$) i ustalonym obwodzie ($4 \leqslant p \leqslant 10^6$).

- Zaczynamy od upewnienia się, że p jest parzyste.
- Zauważamy, że pole nie może być mniejsze niż $p/2 - 1$.
- ...a także nie może być większe niż $\lfloor p/4 \rfloor \cdot \lceil p/4 \rceil$.

C – Generowanie figury (Karol Pokorski)

- Znajdujemy $w \geqslant h$ o takiej własności, że $w + h = p/2$ oraz $w \cdot h \geqslant a$ i $(w + 1) \cdot (h - 1) < a$.

C – Generowanie figury (Karol Pokorski)

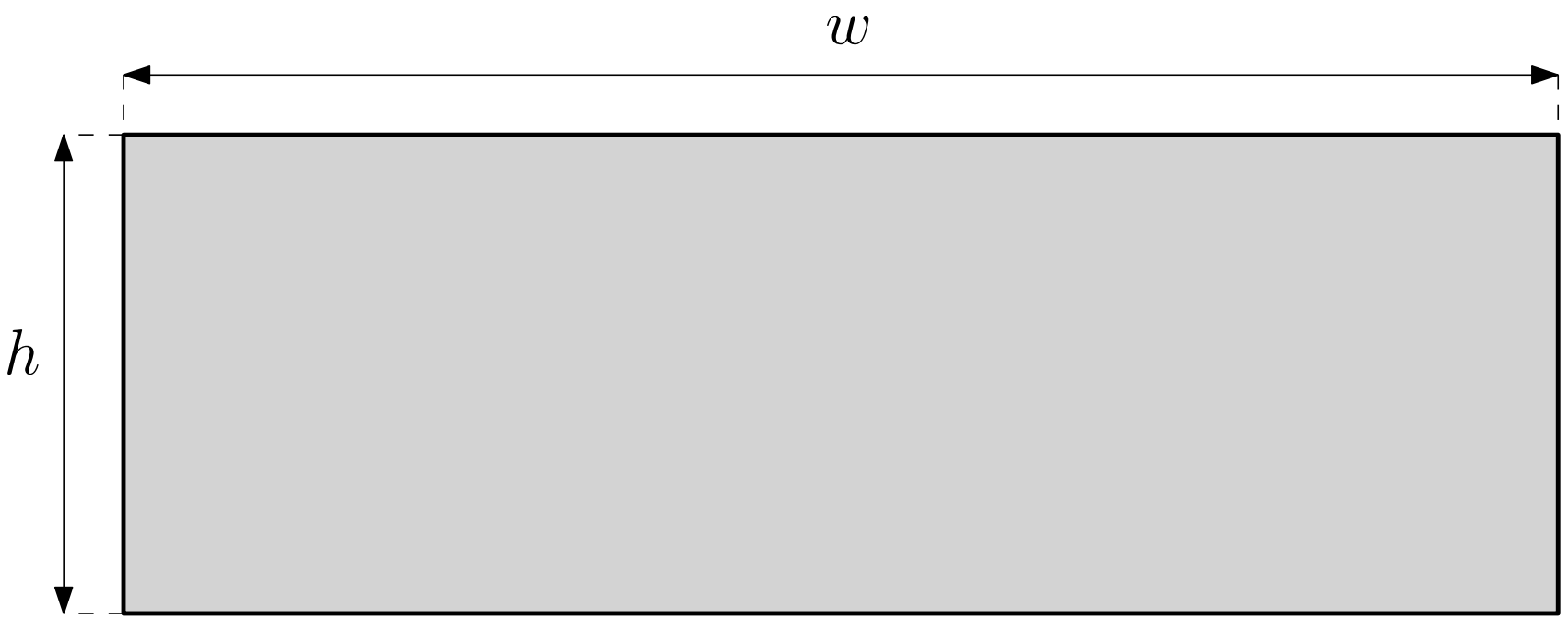
- Znajdujemy $w \geqslant h$ o takiej własności, że $w + h = p/2$ oraz $w \cdot h \geqslant a$ i $(w + 1) \cdot (h - 1) < a$.
- Takie w i h zawsze istnieją!

C – Generowanie figury (Karol Pokorski)

- Znajdujemy $w \geqslant h$ o takiej własności, że $w + h = p/2$ oraz $w \cdot h \geqslant a$ i $(w + 1) \cdot (h - 1) < a$.
- Takie w i h zawsze istnieją!
- Zauważamy, że można lekko zmodyfikować prostokąt $w \times h$ tak, aby nie zmienić jego obwodu, a jednocześnie zmniejszyć jego pole o $w \cdot h - a \leqslant w - h < w$.

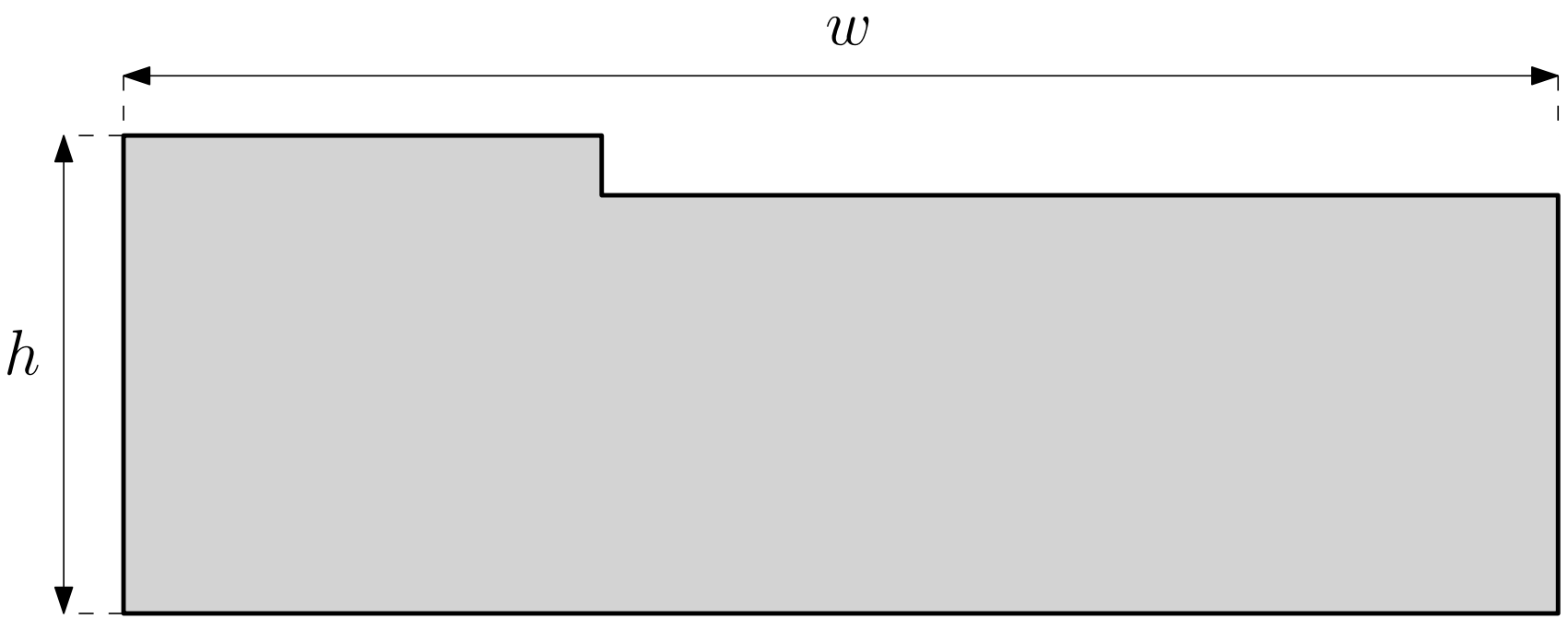
C – Generowanie figury (Karol Pokorski)

- Znajdujemy $w \geqslant h$ o takiej własności, że $w + h = p/2$ oraz $w \cdot h \geqslant a$ i $(w + 1) \cdot (h - 1) < a$.
- Takie w i h zawsze istnieją!
- Zauważamy, że można lekko zmodyfikować prostokąt $w \times h$ tak, aby nie zmienić jego obwodu, a jednocześnie zmniejszyć jego pole o $w \cdot h - a \leqslant w - h < w$.



C – Generowanie figury (Karol Pokorski)

- Znajdujemy $w \geqslant h$ o takiej własności, że $w + h = p/2$ oraz $w \cdot h \geqslant a$ i $(w + 1) \cdot (h - 1) < a$.
- Takie w i h zawsze istnieją!
- Zauważamy, że można lekko zmodyfikować prostokąt $w \times h$ tak, aby nie zmienić jego obwodu, a jednocześnie zmniejszyć jego pole o $w \cdot h - a \leqslant w - h < w$.



H – Anagramy (Karol Pokorski)

Zadanie

Stworzyć najkrótsze słowo, które ma dokładnie n anagramów. ($n \leqslant 10^{12}$)

H – Anagramy (Karol Pokorski)

Zadanie

Stworzyć najkrótsze słowo, które ma dokładnie n anagramów. ($n \leqslant 10^{12}$)

Wzór na liczbę anagramów słowa długości ℓ

$$m = \binom{\ell}{a_1} \cdot \binom{\ell - a_1}{a_2} \cdot \binom{\ell - a_1 - a_2}{a_3} \cdot \ldots$$

gdzie a_i oznacza liczbę wystąpień litery i.

H – Anagramy (Karol Pokorski)

Zadanie

Stworzyć najkrótsze słowo, które ma dokładnie n anagramów. ($n \leqslant 10^{12}$)

Wzór na liczbę anagramów słowa długości ℓ

$$m = \binom{\ell}{a_1} \cdot \binom{\ell - a_1}{a_2} \cdot \binom{\ell - a_1 - a_2}{a_3} \cdot \ldots$$

gdzie a_i oznacza liczbę wystąpień litery i.

Spoiler

Będziemy próbować przepchnąć przeszukiwanie z nawrotami.

H – Anagramy (Karol Pokorski)

Pytanie

Ile różnych liter może mieć szukane słowo?
W szczególności, czy to, że dysponujemy ograniczonym alfabetem (a–z) jest problemem?

H – Anagramy (Karol Pokorski)

Pytanie

Ile różnych liter może mieć szukane słowo?
W szczególności, czy to, że dysponujemy ograniczonym alfabetem (a–z) jest problemem?

Odpowiedź

Nie! Słowo, które ma k różnych liter, ma co najmniej $k!$ anagramów.

$$15! > 10^{12}$$

H – Anagramy (Karol Pokorski)

- Zauważamy, że zawsze istnieje słowo długości n postaci $\mathrm{a}^{n-1}\mathrm{b}$.
- Pozostaje sprawdzić słowa, w których występują przynajmniej trzy różne litery, bądź tylko dwie różne litery, z których każda występuje przynajmniej dwa razy.

H – Anagramy (Karol Pokorski)

- Zauważamy, że zawsze istnieje słowo długości n postaci $\mathrm{a}^{n-1}\mathrm{b}$.
- Pozostaje sprawdzić słowa, w których występują przynajmniej trzy różne litery, bądź tylko dwie różne litery, z których każda występuje przynajmniej dwa razy.
- Oznaczając przez ℓ długość takiego słowa, ma ono przynajmniej $\binom{\ell}{2}$ anagramów.
- Wystarczy więc sprawdzić kolejne długości $\ell = 1, 2, \ldots, 2\sqrt{n}$.

H – Anagramy (Karol Pokorski)

- Dokładamy kolejne litery po kolei, zakładając, że $a_1 \leqslant a_2 \leqslant a_3 \leqslant \ldots$.

H – Anagramy (Karol Pokorski)

- Dokładamy kolejne litery po kolei, zakładając, że $a_1 \leqslant a_2 \leqslant a_3 \leqslant \ldots$.
- Po dołożeniu kolejnej i-tej litery sprytnie aktualizujemy $\binom{\ell - a_1 - a_2 - \ldots - a_{i-1}}{a_i}$ jednocześnie zwiększając a_i.

H – Anagramy (Karol Pokorski)

- Dokładamy kolejne litery po kolei, zakładając, że $a_1 \leqslant a_2 \leqslant a_3 \leqslant \ldots$.
- Po dołożeniu kolejnej i-tej litery sprytnie aktualizujemy $\binom{\ell - a_1 - a_2 - \ldots - a_{i-1}}{a_i}$ jednocześnie zwiększając a_i.
- Przerywamy zwiększanie a_i gdy aktualna liczba anagramów przekroczy n.
- Przerywamy przeszukiwanie gdy aktualna liczba anagramów jest równa n bądź $i > 26$.

H – Anagramy (Karol Pokorski)

- Dokładamy kolejne litery po kolei, zakładając, że $a_1 \leqslant a_2 \leqslant a_3 \leqslant \ldots$.
- Po dołożeniu kolejnej i-tej litery sprytnie aktualizujemy $\binom{\ell - a_1 - a_2 - \ldots - a_{i-1}}{a_i}$ jednocześnie zwiększając a_i.
- Przerywamy zwiększanie a_i gdy aktualna liczba anagramów przekroczy n.
- Przerywamy przeszukiwanie gdy aktualna liczba anagramów jest równa n bądź $i > 26$.
- Odpalamy nasz program dla $n = 10^{12}$ i widzimy, że działa wystarczająco szybko, będzie więc wystarczająco wydajny także dla mniejszych n.

H – Anagramy (Karol Pokorski)

- Dokładamy kolejne litery po kolei, zakładając, że $a_1 \leqslant a_2 \leqslant a_3 \leqslant \ldots$.
- Po dołożeniu kolejnej i-tej litery sprytnie aktualizujemy $\binom{\ell - a_1 - a_2 - \ldots - a_{i-1}}{a_i}$ jednocześnie zwiększając a_i.
- Przerywamy zwiększanie a_i gdy aktualna liczba anagramów przekroczy n.
- Przerywamy przeszukiwanie gdy aktualna liczba anagramów jest równa n bądź $i > 26$.
- Odpalamy nasz program dla $n = 10^{12}$ i widzimy, że działa wystarczająco szybko, będzie więc wystarczająco wydajny także dla mniejszych n.

Optymalizacje

Można zauważyć, że aktualna liczba anagramów musi zawsze dzielić n, a także ograniczyć zakres rozpatrywanych długości ℓ do $O(n^{1/3})$. Nie było to jednak konieczne.

A – Słaby generator pseudolosowy (Karol Pokorski)

Zadanie

Mając dany liniowy generator pseudolosowy:

$$t_n = (a \cdot t_{n-1} + b) \bmod p$$

znaleźć x, dla którego $t_x = N$.

A – Słaby generator pseudolosowy (Karol Pokorski)

Zadanie

Mając dany liniowy generator pseudolosowy:

$$t_n = (a \cdot t_{n-1} + b) \bmod p$$

znaleźć x, dla którego $t_x = N$.

- Zapomnijmy na chwilę o modulo.
- $t_1 = a \cdot t_0 + b$.
- $t_2 = a \cdot t_1 + b = a \cdot (a \cdot t_0 + b) + b = a^2 \cdot t_0 + ab + b$.
- $t_n = a^n \cdot t_0 + b \cdot (a^{n-1} + a^{n-2} + \cdots + a^0)$.

A – Słaby generator pseudolosowy (Karol Pokorski)

Wzór jawny na t_m

$$t_n = a^n \cdot t_0 + b \cdot \frac{a^n - 1}{a - 1}$$

A – Słaby generator pseudolosowy (Karol Pokorski)

Wzór jawny na t_m

$$t_n = a^n \cdot t_0 + b \cdot \frac{a^n - 1}{a - 1}$$

- Warto rachować dalej: znamy przecież t_n, t_0, a, b. Wystarczy rozwiązać równanie ze względu na a^n.

A – Słaby generator pseudolosowy (Karol Pokorski)

Wzór jawny na t_m

$$t_n = a^n \cdot t_0 + b \cdot \frac{a^n - 1}{a - 1}$$

- Warto rachować dalej: znamy przecież t_n, t_0, a, b. Wystarczy rozwiązać równanie ze względu na a^n.

Rozwiązanie równania

$$a^n = \frac{t_n \cdot (a-1) + b}{t_0 \cdot (a-1) + b}$$

A – Słaby generator pseudolosowy (Karol Pokorski)

- Przypominamy sobie o modulo i dzielenie zastępujemy odwrotnością modularną.
- n można uzyskać rozwiązując problem logarytmu dyskretnego, algorytmem Baby–step giant–step w czasie $O(\sqrt{p} \cdot \log p)$.
- Trzeba jeszcze uważać na dzielenie przez 0 i (niemałą) liczbę przypadków brzegowych, pamiętać o `long long`ach itd.

E – Taksówki (Karol Pokorski)

Zadanie

Dane jest n firm taksówkarskich ($\leqslant 10^5$). Każda dostarcza taksówki c_i osobowe ($\leqslant 15$), a podróż jedną z nich kosztuje $s_i + p_i \cdot (x-1)$ dla podróży na dystansie x kilometrów.
Danych jest też q zapytań ($\leqslant 10^5$) o przewiezienie m_i osób ($\leqslant 10^6$) na dystansie d_i kilometrów ($\leqslant 10^6$). Dla każdego z zapytań wyznaczyć optymalny koszt przewiezienia z użyciem odpowiedniej kombinacji taksówek (być może różnych) firm taksówkarskich.

E – Taksówki (Karol Pokorski)

- Dzielimy firmy na grupy względem pojemności. Każdą grupę rozpatrujemy osobno (grup jest $\leqslant 15$).

E – Taksówki (Karol Pokorski)

- Dzielimy firmy na grupy względem pojemności. Każdą grupę rozpatrujemy osobno (grup jest $\leqslant 15$).
- Koszt wynajęcia taksówki jest funkcją liniową odległości.
- Obliczamy otoczkę wypukłą tych prostych. Wyznacza to przedziały odległości, na których (w danej grupie) konkretna firma taksówkarska ma najniższą cenę.

E – Taksówki (Karol Pokorski)

- Dzielimy firmy na grupy względem pojemności. Każdą grupę rozpatrujemy osobno (grup jest $\leqslant 15$).
- Koszt wynajęcia taksówki jest funkcją liniową odległości.
- Obliczamy otoczkę wypukłą tych prostych. Wyznacza to przedziały odległości, na których (w danej grupie) konkretna firma taksówkarska ma najniższą cenę.
- Zapytania rozwiązujemy offline, sortując je według dystansu podróży.
- Przechodząc do coraz większych odległości, zamiatamy otoczki wypukłe przesuwając się do przodu na każdej z nich. To ogranicza problem do ustalenia wyniku z zadania, mając ograniczony zbiór firm taksówkarskich do $\leqslant 15$ firm.

E – Taksówki (Karol Pokorski)

Obserwacja

Zapytanie można teraz rozwiązać używając rozwiązania problemu pakowania plecaka o pojemności $\geqslant m_i$ przedmiotami (firmami taksówkarskimi) o wagach c_i i kosztach d_i.

E – Taksówki (Karol Pokorski)

Obserwacja

Zapytanie można teraz rozwiązać używając rozwiązania problemu pakowania plecaka o pojemności $\geqslant m_i$ przedmiotami (firmami taksówkarskimi) o wagach c_i i kosztach d_i.

Problem

Rozwiązanie problemu plecakowego działa w czasie $O(\max(c_i) \cdot m_i)$.

E – Taksówki (Karol Pokorski)

- Spośród firm taksówkarskich rozpatrzmy zachłannie tę, która oferuje najlepszy stosunek ceny do liczby przewiezionych osób. Załóżmy, że ta firma oferuje taksówki mieszczące C osób każda.
- Spośród pozostałych firm nie opłaca się nigdy użyć $\geqslant C$ taksówek. Gdyby użyć więcej, możnaby zamienić rozwiązanie na takie o mniejszym koszcie (użyć „zachłannej” taksówki więcej razy).

E – Taksówki (Karol Pokorski)

- Spośród firm taksówkarskich rozpatrzmy zachłannie tę, która oferuje najlepszy stosunek ceny do liczby przewiezionych osób. Załóżmy, że ta firma oferuje taksówki mieszczące C osób każda.
- Spośród pozostałych firm nie opłaca się nigdy użyć $\geqslant C$ taksówek. Gdyby użyć więcej, możnaby zamienić rozwiązanie na takie o mniejszym koszcie (użyć „zachłannej" taksówki więcej razy).

Rozwiązanie zagadki

Można zatem ograniczyć algorytm dynamiczny do pojemności C^2. Resztę plecaka opłaca się upchnąć przedmiotem o masie C. Wystarczy tylko odgadnąć (przetestować każdą możliwość) jaką masę przedmiotów bierzemy z rozwiązania opartego o programowanie dynamiczne.

F – Sieć antyspołecznościowa (Paweł Gawrychowski)

Zadanie

Dany jest graf nieskierowany na $n \leqslant 250\,000$ wierzchołkach i $m \leqslant 2\,000\,000$ krawędziach. Mamy wiele operacji typu: dla danych $v_1, v_2, \ldots, v_s$ flipnij wszystkie krawędzie (v_i, v_j) dla $1 \leqslant i < j \leqslant s$. Które wierzchołki są izolowane w finalnym grafie?

F – Sieć antyspołecznościowa (Paweł Gawrychowski)

- Wylosujmy dla każdego wierzchołka jedną liczbę $x_u \in \{0, 1\}$.

F – Sieć antyspołecznościowa (Paweł Gawrychowski)

- Wylosujmy dla każdego wierzchołka jedną liczbę $x_u \in \{0, 1\}$.
- Niech E' będzie zbiorem krawędzi finalnego grafu. Jeśli wierzchołek v jest w nim izolowany to $\sum_{(u,v)\in E'} x_u = 0 \bmod 2$.

F – Sieć antyspołecznościowa (Paweł Gawrychowski)

- Wylosujmy dla każdego wierzchołka jedną liczbę $x_u \in \{0, 1\}$.
- Niech E' będzie zbiorem krawędzi finalnego grafu. Jeśli wierzchołek v jest w nim izolowany to $\sum_{(u,v)\in E'} x_u = 0 \bmod 2$.
- ...a jeśli nie jest to $\sum_{(u,v)\in E'} x_u \neq 0 \bmod 2$ z prawdopodobieństwem $\frac{1}{2}$!

F – Sieć antyspołecznościowa (Paweł Gawrychowski)

- Wylosujmy dla każdego wierzchołka jedną liczbę $x_u \in \{0, 1\}$.
- Niech E' będzie zbiorem krawędzi finalnego grafu. Jeśli wierzchołek v jest w nim izolowany to $\sum_{(u,v)\in E'} x_u = 0 \bmod 2$.
- ...a jeśli nie jest to $\sum_{(u,v)\in E'} x_u \neq 0 \bmod 2$ z prawdopodobieństwem $\frac{1}{2}$!
- Nazwijmy tę sumę $s(v)$. Zaczynamy od policzenia wszystkich sum w pierwotnym grafie.

F – Sieć antyspołecznościowa (Paweł Gawrychowski)

- Wylosujmy dla każdego wierzchołka jedną liczbę $x_u \in \{0, 1\}$.
- Niech E' będzie zbiorem krawędzi finalnego grafu. Jeśli wierzchołek v jest w nim izolowany to $\sum_{(u,v)\in E'} x_u = 0 \bmod 2$.
- ...a jeśli nie jest to $\sum_{(u,v)\in E'} x_u \neq 0 \bmod 2$ z prawdopodobieństwem $\frac{1}{2}$!
- Nazwijmy tę sumę $s(v)$. Zaczynamy od policzenia wszystkich sum w pierwotnym grafie.
- Łatwo zaktualizować wszystkie sumy po jednej operacji flipnięcia w czasie $O(s)$, trzeba tylko policzyć $t = \sum_{i=1}^{s} x_{v_i} \bmod 2$, a następnie do każdego $s(v_i)$ dodać $t + x_{v_i} \bmod 2$.

F – Sieć antyspołecznościowa (Paweł Gawrychowski)

- Wylosujmy dla każdego wierzchołka jedną liczbę $x_u \in \{0, 1\}$.
- Niech E' będzie zbiorem krawędzi finalnego grafu. Jeśli wierzchołek v jest w nim izolowany to $\sum_{(u,v)\in E'} x_u = 0 \bmod 2$.
- ...a jeśli nie jest to $\sum_{(u,v)\in E'} x_u \neq 0 \bmod 2$ z prawdopodobieństwem $\frac{1}{2}$!
- Nazwijmy tę sumę $s(v)$. Zaczynamy od policzenia wszystkich sum w pierwotnym grafie.
- Łatwo zaktualizować wszystkie sumy po jednej operacji flipnięcia w czasie $O(s)$, trzeba tylko policzyć $t = \sum_{i=1}^{s} x_{v_i} \bmod 2$, a następnie do każdego $s(v_i)$ dodać $t + x_{v_i} \bmod 2$.
- Całą procedurę należy dla pewności powtórzyć 50 razy. Tak naprawdę można symulować taką amplifikację losując nie jeden bit a `long long`a i robiąc `xora`.

L – Trybunał konstytucyjny (Paweł Gawrychowski)

Zadanie

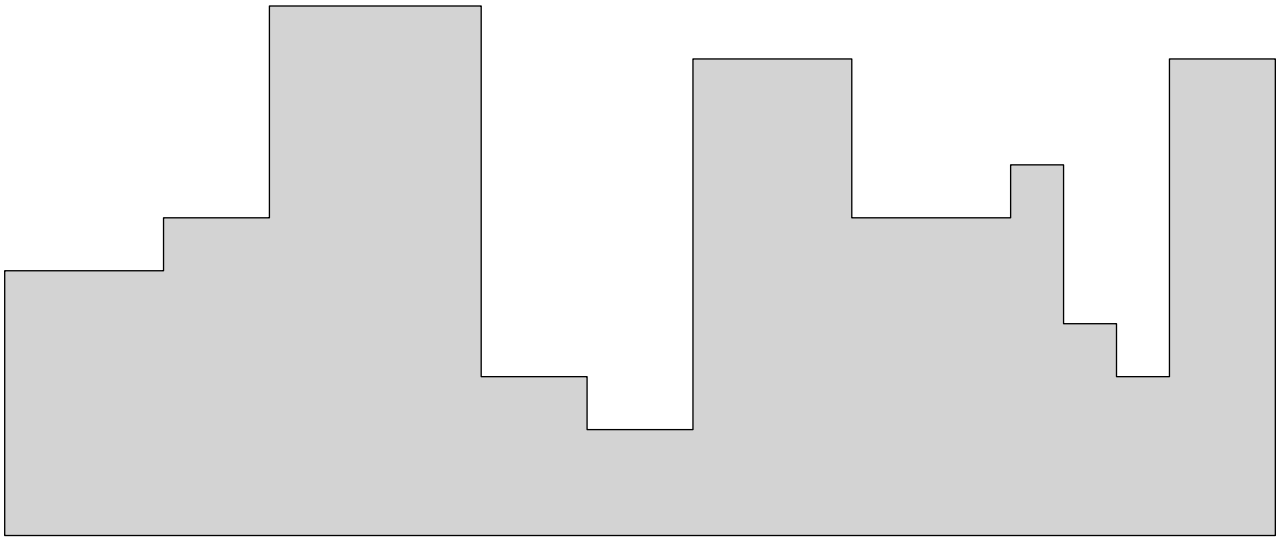
Dane jest drzewo ukorzenione na n wierzchołkach, $n \leqslant 5\,000$. Wierzchołek i ma przepustowość c_i oraz początkowo przechowuje s_i jednostek, które chciałby przesłać do korzenia. W każdej jednostce czasu wierzchołek może przesłać co najwyżej c_i ze znajdujących się w nim aktualnie jednostek do swojego ojca. Jak szybko można przesłać wszystkie jednostki do korzenia całego drzewa?

L – Trybunał konstytucyjny (Paweł Gawrychowski)

Dla każdego wierzchołka budujemy strukturę opisującą monety, które są przez niego przesyłane do ojca. Myślimy, że jest to po prostu wykres, który dla każdego momentu podaje liczbę przesyłanych monet.

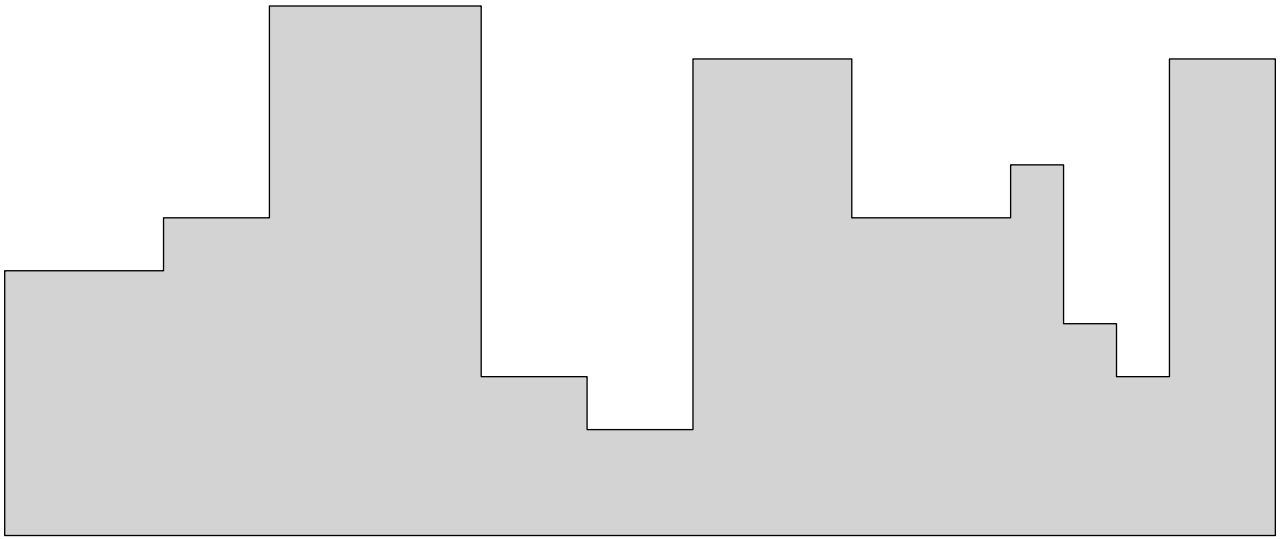
L – Trybunał konstytucyjny (Paweł Gawrychowski)

Dla każdego wierzchołka budujemy strukturę opisującą monety, które są przez niego przesyłane do ojca. Myślimy, że jest to po prostu wykres, który dla każdego momentu podaje liczbę przesyłanych monet.



L – Trybunał konstytucyjny (Paweł Gawrychowski)

Dla każdego wierzchołka budujemy strukturę opisującą monety, które są przez niego przesyłane do ojca. Myślimy, że jest to po prostu wykres, który dla każdego momentu podaje liczbę przesyłanych monet.

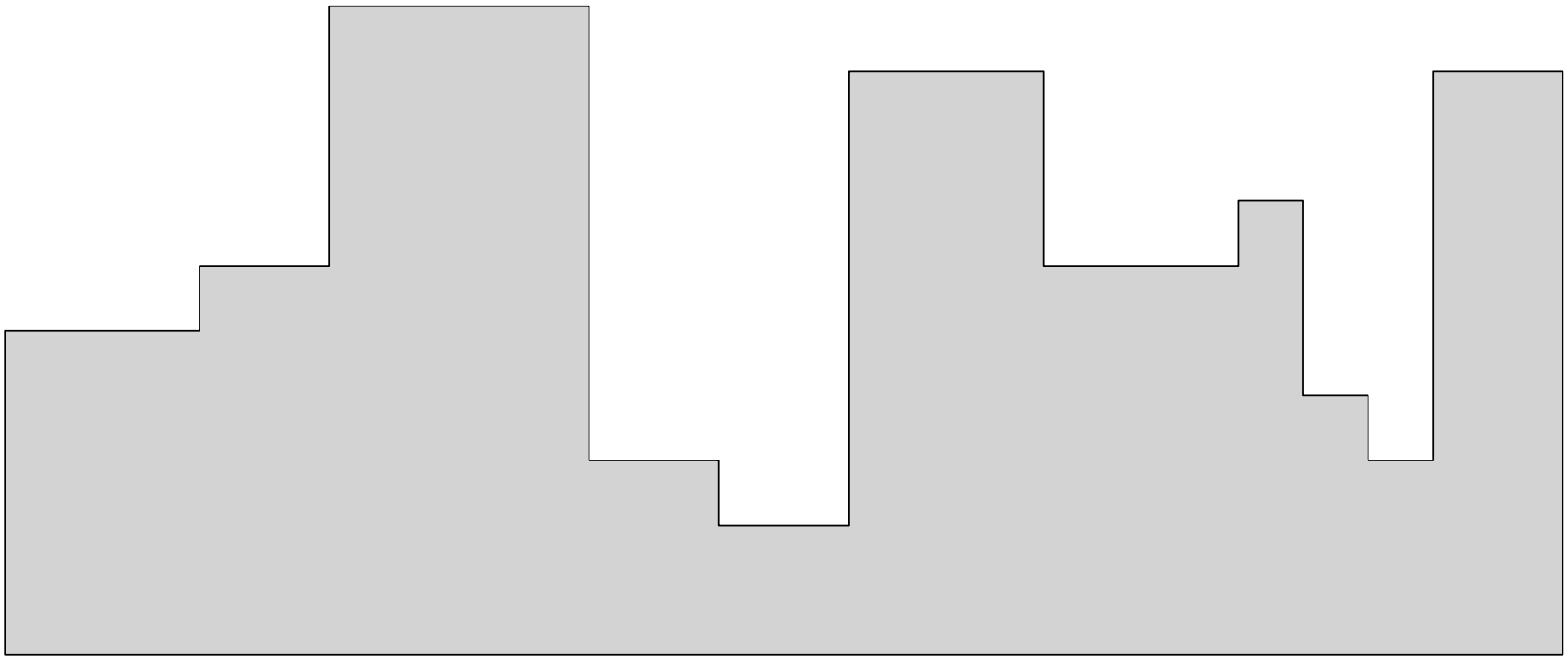


Kluczowa obserwacja

Taki wykres składa się tylko z $O(n)$ schodków.

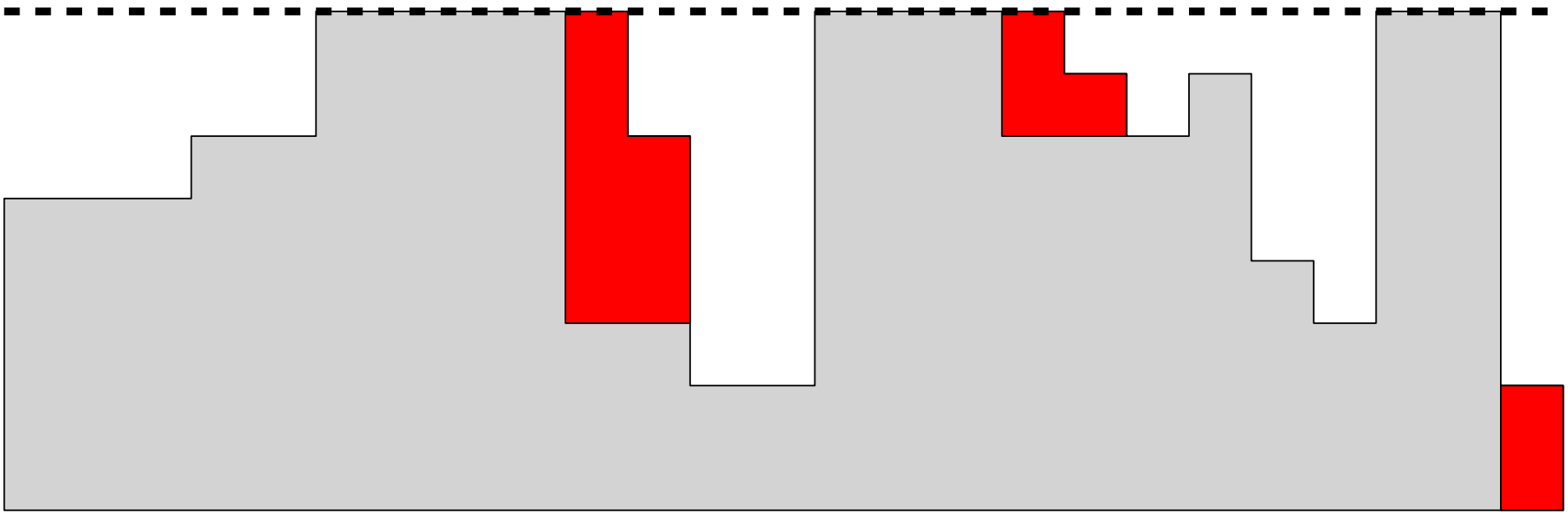
L – Trybunał konstytucyjny (Paweł Gawrychowski)

Trzeba tylko umieć skonstruować strukturę dla wierzchołka u mając struktury dla jego dzieci. W tym celu najpierw sumujemy struktury dzieci. Następnie trzeba uwzględnić limit na liczbę monet przesyłanych w każdej jednostce czasu przez u do ojca.



L – Trybunał konstytucyjny (Paweł Gawrychowski)

Trzeba tylko umieć skonstruować strukturę dla wierzchołka u mając struktury dla jego dzieci. W tym celu najpierw sumujemy struktury dzieci. Następnie trzeba uwzględnić limit na liczbę monet przesyłanych w każdej jednostce czasu przez u do ojca.



L – Trybunał konstytucyjny (Paweł Gawrychowski)

- Takie przycięcie może być wykonane w czasie $O(n)$ skanując schodki od lewej do prawej.

L – Trybunał konstytucyjny (Paweł Gawrychowski)

- Takie przycięcie może być wykonane w czasie $O(n)$ skanując schodki od lewej do prawej.
- Podobnie, zsumowanie dwóch struktur to właściwie scalanie dwóch posortowanych list, a więc również $O(n)$.

L – Trybunał konstytucyjny (Paweł Gawrychowski)

- Takie przycięcie może być wykonane w czasie $O(n)$ skanując schodki od lewej do prawej.
- Podobnie, zsumowanie dwóch struktur to właściwie scalanie dwóch posortowanych list, a więc również $O(n)$.
- Takich operacji jest $O(n)$, całe rozwiązanie działa więc w czasie $O(n)$.

I – Górska wędrówka (Jakub Tarnawski)

Zadanie

Dany jest graf acykliczny na n wierzchołkach i m ważonych krawędziach, $n \leqslant 1\,000$, $m \leqslant 10\,000$, oraz wyróżnione wierzchołki s_1, t_1, s_2, t_2. Znajdź dwie rozłączne wierzchołkowo ścieżki $s_1 \longrightarrow t_1$ oraz $s_2 \longrightarrow t_2$ o najmniejszej możliwej sumie kosztów krawędzi.

I – Górska wędrówka (Jakub Tarnawski)

Zadanie

Dany jest graf acykliczny na n wierzchołkach i m ważonych krawędziach, $n \leqslant 1\,000$, $m \leqslant 10\,000$, oraz wyróżnione wierzchołki s_1, t_1, s_2, t_2. Znajdź dwie rozłączne wierzchołkowo ścieżki $s_1 \to t_1$ oraz $s_2 \to t_2$ o najmniejszej możliwej sumie kosztów krawędzi.

- Przenumerujmy wierzchołki tak, żeby po posortowaniu według wysokości nad poziomem morza nazywały się $1, 2, \ldots, n$.

I – Górska wędrówka (Jakub Tarnawski)

Zadanie

Dany jest graf acykliczny na n wierzchołkach i m ważonych krawędziach, $n \leqslant 1\,000$, $m \leqslant 10\,000$, oraz wyróżnione wierzchołki s_1, t_1, s_2, t_2. Znajdź dwie rozłączne wierzchołkowo ścieżki $s_1 \to t_1$ oraz $s_2 \to t_2$ o najmniejszej możliwej sumie kosztów krawędzi.

- Przenumerujmy wierzchołki tak, żeby po posortowaniu według wysokości nad poziomem morza nazywały się $1, 2, \ldots, n$.
- Sprawdźmy osobno brzegowe przypadki, w których $|\{s_1, t_1, s_2, t_2\}| < 4$.

I – Górska wędrówka (Jakub Tarnawski)

Zadanie

Dany jest graf acykliczny na n wierzchołkach i m ważonych krawędziach, $n \leqslant 1\,000$, $m \leqslant 10\,000$, oraz wyróżnione wierzchołki s_1, t_1, s_2, t_2. Znajdź dwie rozłączne wierzchołkowo ścieżki $s_1 \to t_1$ oraz $s_2 \to t_2$ o najmniejszej możliwej sumie kosztów krawędzi.

- Przenumerujmy wierzchołki tak, żeby po posortowaniu według wysokości nad poziomem morza nazywały się $1, 2, \ldots, n$.
- Sprawdźmy osobno brzegowe przypadki, w których $|\{s_1, t_1, s_2, t_2\}| < 4$.
- Teraz konstruujemy nowy graf, w którym wierzchołkami są (uporządkowane) pary (u, v), $u \neq v$. Krawędzie i ich wagi w nowym grafie powinny być tak dobrane, aby ścieżka z (s_1, s_2) do (u, v) odpowiadała dwóm rozłącznym wierzchołkowo ścieżkom $s_1 \to u$ oraz $s_2 \to v$.

I – Górska wędrówka (Jakub Tarnawski)

Zadanie

Dany jest graf acykliczny na n wierzchołkach i m ważonych krawędziach, $n \leqslant 1\,000$, $m \leqslant 10\,000$, oraz wyróżnione wierzchołki s_1, t_1, s_2, t_2. Znajdź dwie rozłączne wierzchołkowo ścieżki $s_1 \to t_1$ oraz $s_2 \to t_2$ o najmniejszej możliwej sumie kosztów krawędzi.

- Przenumerujmy wierzchołki tak, żeby po posortowaniu według wysokości nad poziomem morza nazywały się $1, 2, \ldots, n$.
- Sprawdźmy osobno brzegowe przypadki, w których $|\{s_1, t_1, s_2, t_2\}| < 4$.
- Teraz konstruujemy nowy graf, w którym wierzchołkami są (uporządkowane) pary (u, v), $u \neq v$. Krawędzie i ich wagi w nowym grafie powinny być tak dobrane, aby ścieżka z (s_1, s_2) do (u, v) odpowiadała dwóm rozłącznym wierzchołkowo ścieżkom $s_1 \to u$ oraz $s_2 \to v$.
- Można myśleć, że nowy graf „symuluje” przesuwanie dwóch palców po tym oryginalnym.

I – Górska wędrówka (Jakub Tarnawski)

- Mamy dwa przypadki. Jeśli $u < v$ lub $v = t_2$ to próbujemy zgadnąć następnik u na odpowiedniej ścieżce, to jest przesunąć lewy palec. W tym celu trzeba przejrzeć wszystkie krawędzie (u, u') takie, że $u' \neq v$. Dla każdej takiej krawędzi w oryginalnym grafie dodajemy krawędź łączącą (u, v) z (u', v) o takim samym koszcie w nowym grafie.

I – Górska wędrówka (Jakub Tarnawski)

- Mamy dwa przypadki. Jeśli $u < v$ lub $v = t_2$ to próbujemy zgadnąć następnik u na odpowiedniej ścieżce, to jest przesunąć lewy palec. W tym celu trzeba przejrzeć wszystkie krawędzie (u, u') takie, że $u' \neq v$. Dla każdej takiej krawędzi w oryginalnym grafie dodajemy krawędź łączącą (u, v) z (u', v) o takim samym koszcie w nowym grafie.
- ...jeśli $v < u$ lub $u = t_1$ to próbujemy zgadnąć następnik v, czyli przeglądamy krawędzie (v, v') takie, że $v' \neq u$.

I – Górska wędrówka (Jakub Tarnawski)

- Mamy dwa przypadki. Jeśli $u < v$ lub $v = t_2$ to próbujemy zgadnąć następnik u na odpowiedniej ścieżce, to jest przesunąć lewy palec. W tym celu trzeba przejrzeć wszystkie krawędzie (u, u') takie, że $u' \neq v$. Dla każdej takiej krawędzi w oryginalnym grafie dodajemy krawędź łączącą (u, v) z (u', v) o takim samym koszcie w nowym grafie.
- ...jeśli $v < u$ lub $u = t_1$ to próbujemy zgadnąć następnik v, czyli przeglądamy krawędzie (v, v') takie, że $v' \neq u$.
- Teraz zauważamy, że tak naprawdę wystarczy znaleźć najkrótszą ścieżkę z (s_1, s_2) do (t_1, t_2) w nowym grafie!

I – Górska wędrówka (Jakub Tarnawski)

- Mamy dwa przypadki. Jeśli $u < v$ lub $v = t_2$ to próbujemy zgadnąć następnik u na odpowiedniej ścieżce, to jest przesunąć lewy palec. W tym celu trzeba przejrzeć wszystkie krawędzie (u, u') takie, że $u' \neq v$. Dla każdej takiej krawędzi w oryginalnym grafie dodajemy krawędź łączącą (u, v) z (u', v) o takim samym koszcie w nowym grafie.
- ... jeśli $v < u$ lub $u = t_1$ to próbujemy zgadnąć następnik v, czyli przeglądamy krawędzie (v, v') takie, że $v' \neq u$.
- Teraz zauważamy, że tak naprawdę wystarczy znaleźć najkrótszą ścieżkę z (s_1, s_2) do (t_1, t_2) w nowym grafie!
- Ponieważ nowy graf także jest acykliczny, nie trzeba implementować Dijkstry: wystarczy przejrzeć graf w kolejności topologicznej wyliczając odległości od źródła (s_1, s_2) w czasie liniowym od rozmiaru nowego grafu.

I – Górska wędrówka (Jakub Tarnawski)

- Mamy dwa przypadki. Jeśli $u < v$ lub $v = t_2$ to próbujemy zgadnąć następnik u na odpowiedniej ścieżce, to jest przesunąć lewy palec. W tym celu trzeba przejrzeć wszystkie krawędzie (u, u') takie, że $u' \neq v$. Dla każdej takiej krawędzi w oryginalnym grafie dodajemy krawędź łączącą (u, v) z (u', v) o takim samym koszcie w nowym grafie.
- ...jeśli $v < u$ lub $u = t_1$ to próbujemy zgadnąć następnik v, czyli przeglądamy krawędzie (v, v') takie, że $v' \neq u$.
- Teraz zauważamy, że tak naprawdę wystarczy znaleźć najkrótszą ścieżkę z (s_1, s_2) do (t_1, t_2) w nowym grafie!
- Ponieważ nowy graf także jest acykliczny, nie trzeba implementować Dijkstry: wystarczy przejrzeć graf w kolejności topologicznej wyliczając odległości od źródła (s_1, s_2) w czasie liniowym od rozmiaru nowego grafu.
- Nowy graf ma tylko $O(n \cdot m)$ krawędzi, tyle działa więc całe rozwiązanie.

B – Jaś sam w Nowym Jorku (Paweł Gawrychowski)

Zadanie

Znajdź ścieżkę Hamiltona z (s_x, s_y) do (t_x, t_y) na kratce $n \times m$, $4 \leqslant n, m \leqslant 1\,000$.

B – Jaś sam w Nowym Jorku (Paweł Gawrychowski)

Zadanie

Znajdź ścieżkę Hamiltona z (s_x, s_y) do (t_x, t_y) na kratce $n \times m$, $4 \leqslant n, m \leqslant 1\,000$.

Obserwacja

Sąsiednie wierzchołki na ścieżce mają różne kolory. A więc kolory s oraz t powinny być różne dokładnie wtedy gdy $n \cdot m$ jest parzyste.

B – Jaś sam w Nowym Jorku (Paweł Gawrychowski)

Zadanie

Znajdź ścieżkę Hamiltona z (s_x, s_y) do (t_x, t_y) na kratce $n \times m$, $4 \leqslant n, m \leqslant 1\,000$.

Obserwacja

Sąsiednie wierzchołki na ścieżce mają różne kolory. A więc kolory s oraz t powinny być różne dokładnie wtedy gdy $n \cdot m$ jest parzyste.

$n, m \geqslant 4$

Okazuje się, że dla takich krat parzystość jest warunkiem koniecznym i wystarczającym.

B – Jaś sam w Nowym Jorku (Paweł Gawrychowski)

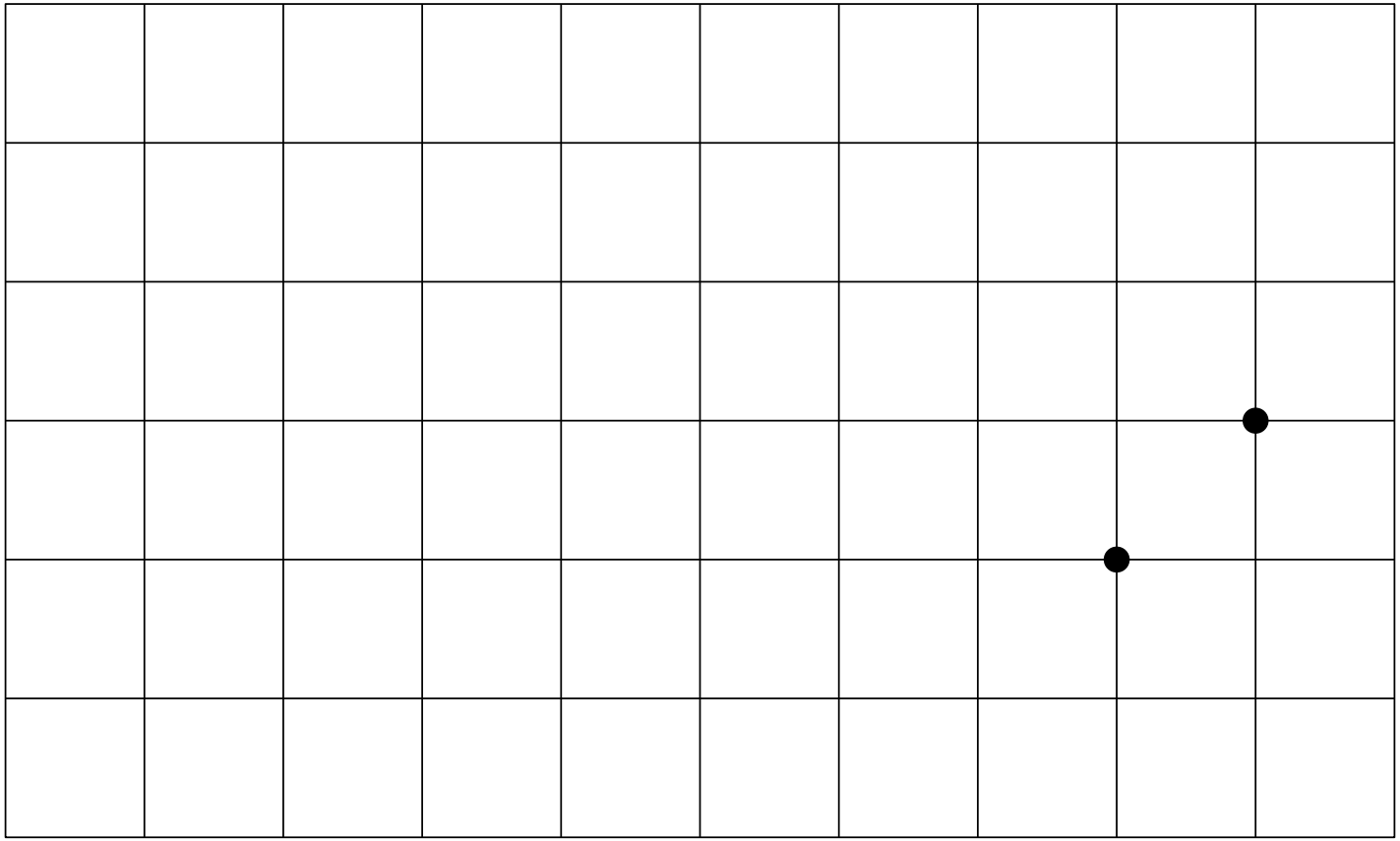
- . . . Można odpalić bruta dla $n, m \in \{4, 5, 6\}$. Ale jak wygląda dowód? I jak skonstruować szukaną ścieżkę?

B – Jaś sam w Nowym Jorku (Paweł Gawrychowski)

- . . . Można odpalić bruta dla $n, m \in \{4, 5, 6\}$. Ale jak wygląda dowód? I jak skonstruować szukaną ścieżkę?
- Spróbujmy pokazać przez indukcję względem $n \cdot m$, że rzeczywiście tak jest.

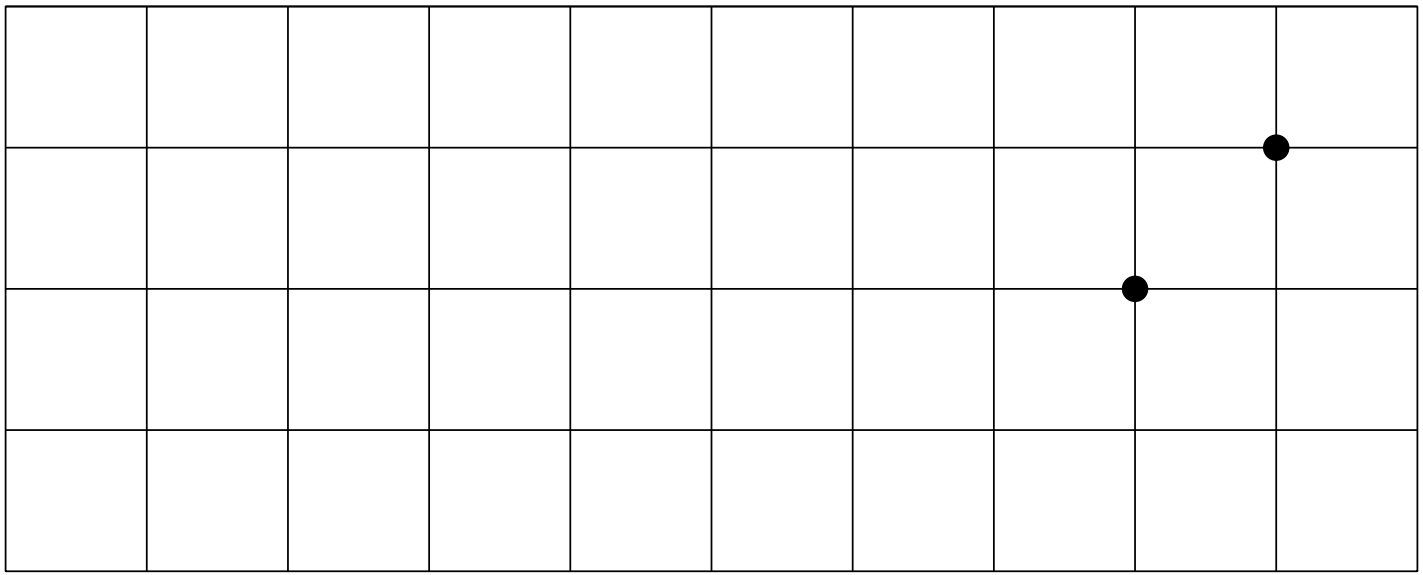
B – Jaś sam w Nowym Jorku (Paweł Gawrychowski)

Załóżmy, że $n \geqslant 7$ oraz dwa pierwsze wiersze kratki nie zawierają punktu startowego ani końcowego.



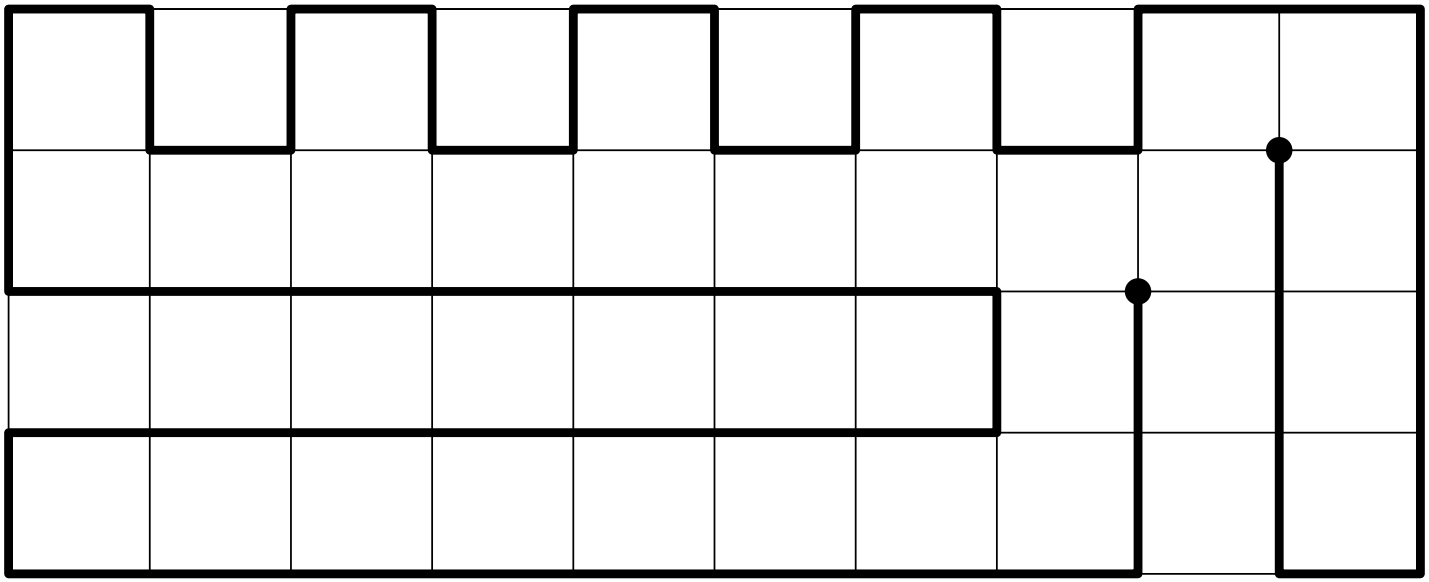
B – Jaś sam w Nowym Jorku (Paweł Gawrychowski)

Załóżmy, że $n \geqslant 7$ oraz dwa pierwsze wiersze kratki nie zawierają punktu startowego ani końcowego.



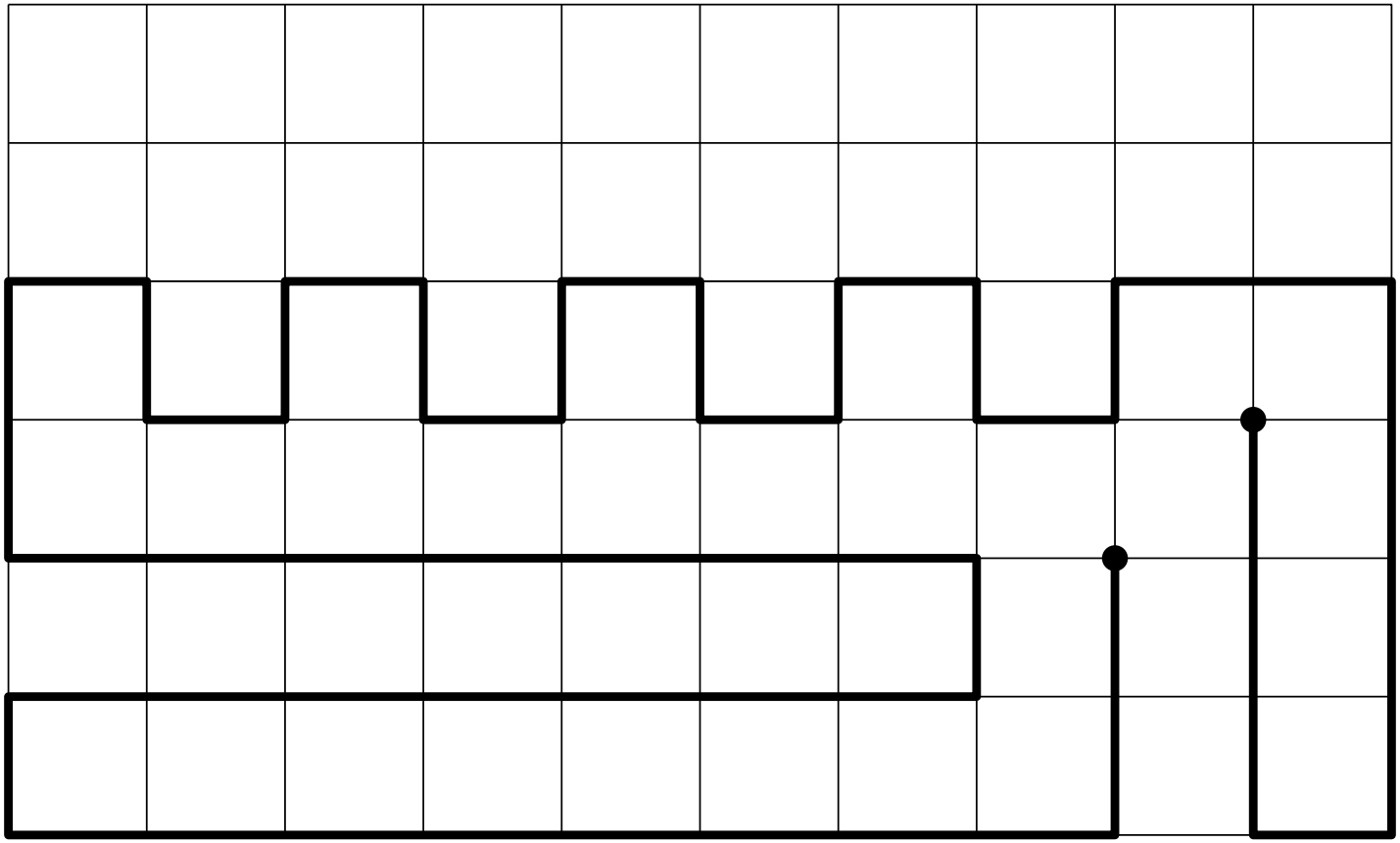
B – Jaś sam w Nowym Jorku (Paweł Gawrychowski)

Załóżmy, że $n \geqslant 7$ oraz dwa pierwsze wiersze kratki nie zawierają punktu startowego ani końcowego.



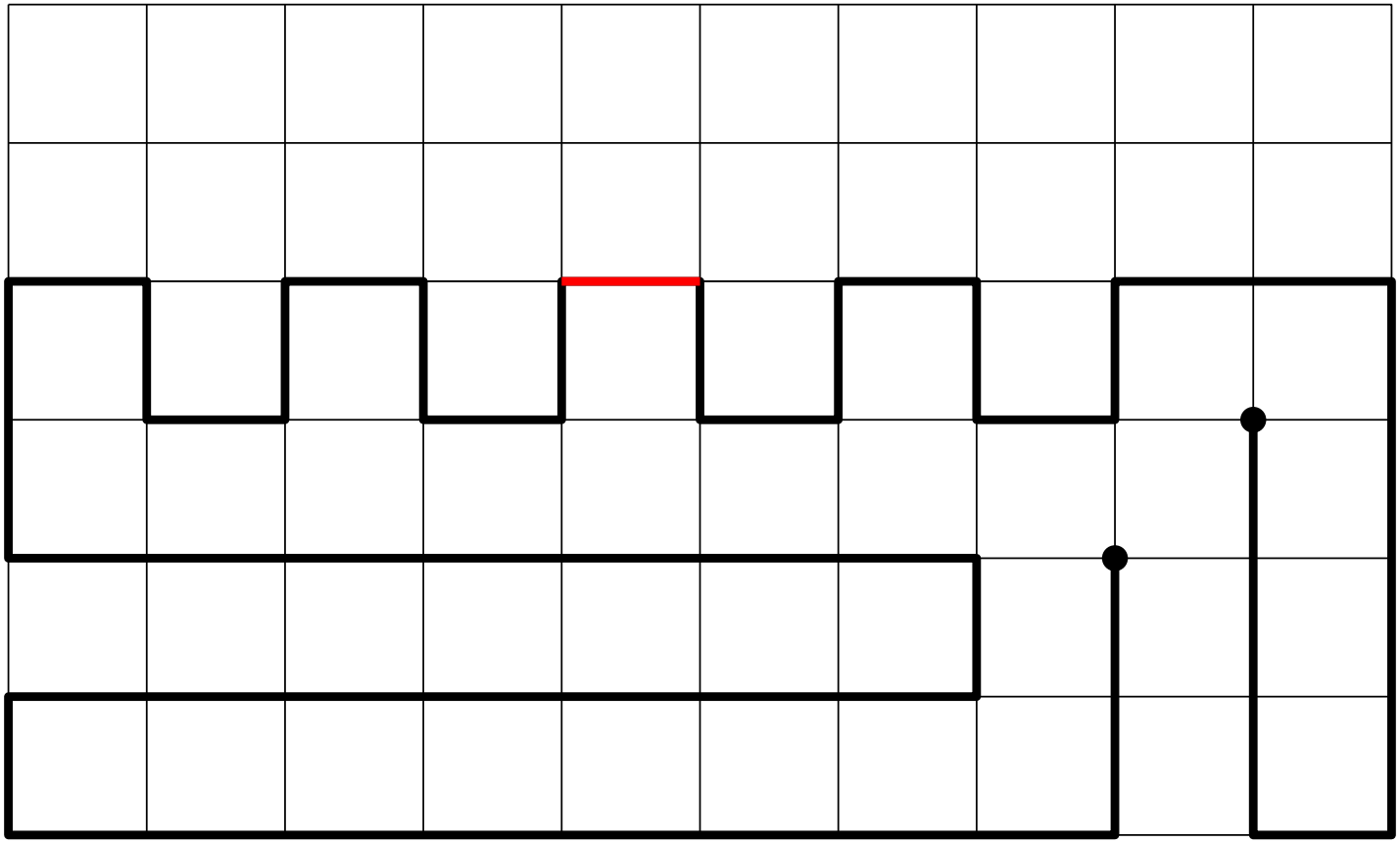
B – Jaś sam w Nowym Jorku (Paweł Gawrychowski)

Załóżmy, że $n \geqslant 7$ oraz dwa pierwsze wiersze kratki nie zawierają punktu startowego ani końcowego.



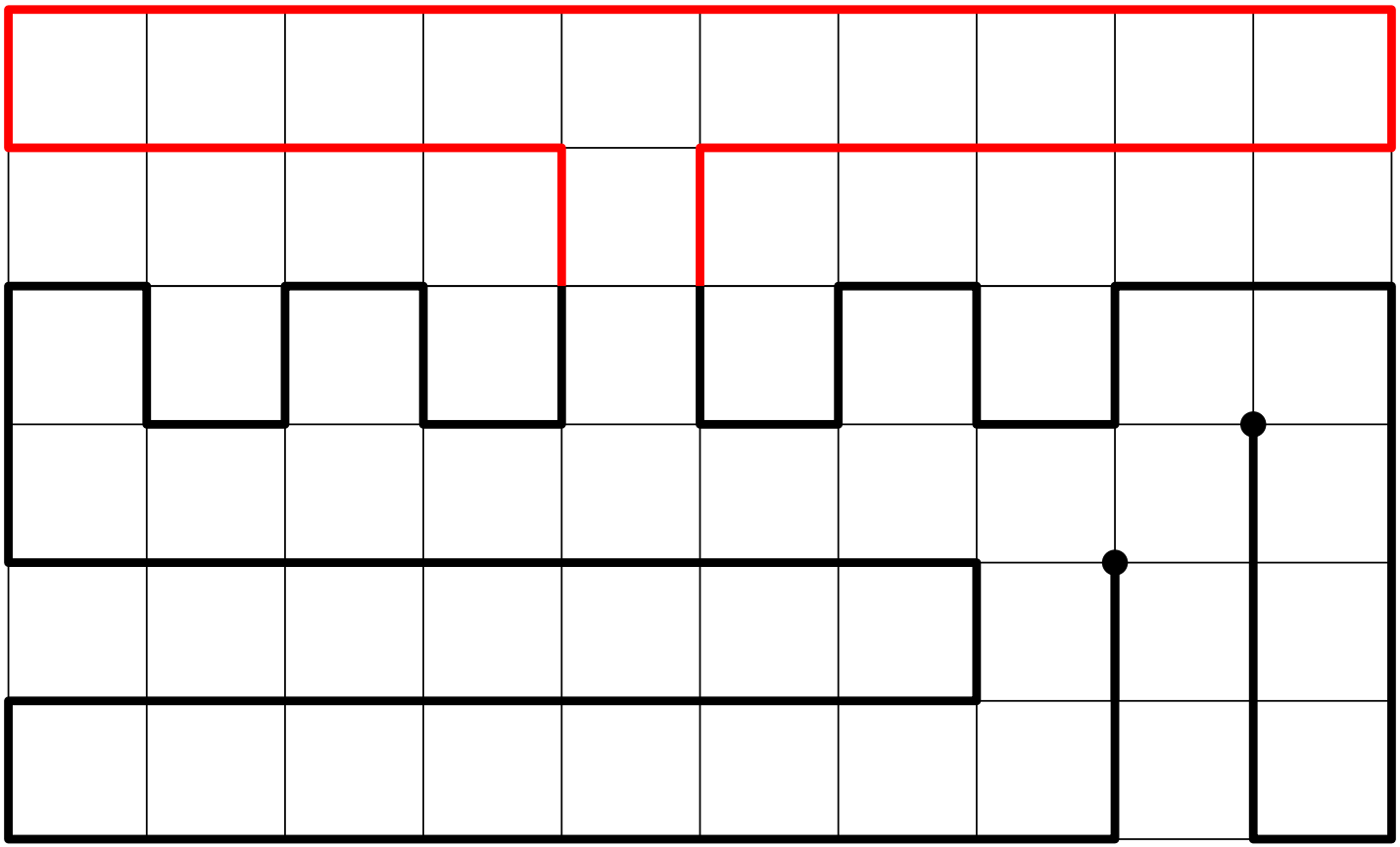
B – Jaś sam w Nowym Jorku (Paweł Gawrychowski)

Załóżmy, że $n \geqslant 7$ oraz dwa pierwsze wiersze kratki nie zawierają punktu startowego ani końcowego.



B – Jaś sam w Nowym Jorku (Paweł Gawrychowski)

Załóżmy, że $n \geqslant 7$ oraz dwa pierwsze wiersze kratki nie zawierają punktu startowego ani końcowego.



B – Jaś sam w Nowym Jorku (Paweł Gawrychowski)

- Taki trik działa także jeśli dwa ostatnie wiersze kratki nie zawierają punktu startowego ani końcowego.

B – Jaś sam w Nowym Jorku (Paweł Gawrychowski)

- Taki trik działa także jeśli dwa ostatnie wiersze kratki nie zawierają punktu startowego ani końcowego.
- Trzeba więc tylko poradzić sobie z sytuacją, w której punkt startowy leży w jednym z dwóch pierwszych wierszy, a punkt końcowy nie leży w żadnym z trzech pierwszy wierszy (skoro n nie jest bardzo małe).

B – Jaś sam w Nowym Jorku (Paweł Gawrychowski)

- Taki trik działa także jeśli dwa ostatnie wiersze kratki nie zawierają punktu startowego ani końcowego.
- Trzeba więc tylko poradzić sobie z sytuacją, w której punkt startowy leży w jednym z dwóch pierwszych wierszy, a punkt końcowy nie leży w żadnym z trzech pierwszy wierszy (skoro n nie jest bardzo małe).
- Podobnie jak w poprzednim przypadku można odkroić dwa pierwsze wiersze i rekurencyjnie znaleźć ścieżkę na kratce $(n-2) \times m$, przy czym punkt startowy zastępujemy odpowiednio dobranym punktem w trzecim wierszu.

Taka konstrukcja działa także gdy $m \geqslant 7$. Skoro dla $n, m \in \{4, 5, 6\}$ sprawdziliśmy prawdziwość tezy brutem, całe rozumowanie pokazuje, że warunek jest wystarczający dla każdych $n, m \geqslant 4$.

B – Jaś sam w Nowym Jorku (Paweł Gawrychowski)

Złożoność

Powyższą konstrukcję łatwo przerobić na algorytm działający w czasie $O((n+m)^3)$ przechowując już skonstruowane rozwiązanie na liście i „wklejając" w odpowiednim miejscu nowy kawałek.

B – Jaś sam w Nowym Jorku (Paweł Gawrychowski)

Złożoność

Powyższą konstrukcję łatwo przerobić na algorytm działający w czasie $O((n+m)^3)$ przechowując już skonstruowane rozwiązanie na liście i „wklejając" w odpowiednim miejscu nowy kawałek.

Ale chcemy szybciej! Najwolniejsze w powyższej implementacji jest odwracanie oraz „odbijanie" aktualnie skonstruowanego rozwiązania.

B – Jaś sam w Nowym Jorku (Paweł Gawrychowski)

Złożoność

Powyższą konstrukcję łatwo przerobić na algorytm działający w czasie $O((n+m)^3)$ przechowując już skonstruowane rozwiązanie na liście i „wklejając" w odpowiednim miejscu nowy kawałek.

Ale chcemy szybciej! Najwolniejsze w powyższej implementacji jest odwracanie oraz „odbijanie" aktualnie skonstruowanego rozwiązania.

Sprytniejsza struktura

Nietrudno wzbogacić listę tak, aby można było odwracać oraz „odbijać" rozwiązanie względem osi OX lub OY w czasie stałym. Można też zauważyć, że wklejanie zawsze następuje niedaleko jednego z końców, więc może być wykonane szybciej.

B – Jaś sam w Nowym Jorku (Paweł Gawrychowski)

Złożoność

Powyższą konstrukcję łatwo przerobić na algorytm działający w czasie $O((n+m)^3)$ przechowując już skonstruowane rozwiązanie na liście i „wklejając" w odpowiednim miejscu nowy kawałek.

Ale chcemy szybciej! Najwolniejsze w powyższej implementacji jest odwracanie oraz „odbijanie" aktualnie skonstruowanego rozwiązania.

Sprytniejsza struktura

Nietrudno wzbogacić listę tak, aby można było odwracać oraz „odbijać" rozwiązanie względem osi OX lub OY w czasie stałym. Można też zauważyć, że wklejanie zawsze następuje niedaleko jednego z końców, więc może być wykonane szybciej.

Prowadzi to do rozwiązania, które działa w czasie $O(n^2)$.