amppz

Omówienie zadań

AMPPZ 2024

amppz

A – Akronim

Najszybsze rozwiązanie:

Jagiellonian 1 (0:04)

Autor zadania:

Kamil Dębowski

A – Akronim

Treść zadania

Wśród danych słów, należy znaleźć akronim, czyli takie słowo, że jego litery występują jako pierwsze litery innych słów (można powtarzać).

A – Akronim

Treść zadania

Wśród danych słów, należy znaleźć akronim, czyli takie słowo, że jego litery występują jako pierwsze litery innych słów (można powtarzać).

Na przykład, ANIA to akronim od „AKRONIM NA IMIENINY AGI”.

A – Akronim

Dla każdej literki A-Z, szukamy jakiegoś słowa na tę literkę. Tablica rozmiaru 26, albo map/dictionary.
Dla każdego słowa, sprawdzamy czy każda jego literka wystąpiła gdzieś jako pierwsza. Jeśli tak jest, wypisujemy słowa zapamiętane pod tymi literkami.

A – Akronim

Dla każdej literki A-Z, szukamy jakiegoś słowa na tę literkę. Tablica rozmiaru 26, albo map/dictionary.
Dla każdego słowa, sprawdzamy czy każda jego literka wystąpiła gdzieś jako pierwsza. Jeśli tak jest, wypisujemy słowa zapamiętane pod tymi literkami.

pseudokod

```
 for every s:
  map[s[0]] = s
for every s:
  flag = TRUE
  for every char in s:
    if char not in map:
      flag = FALSE
  if flag:
    ...
```

A – Akronim

Dla każdej literki A-Z, szukamy jakiegoś słowa na tę literkę. Tablica rozmiaru 26, albo map/dictionary.
Dla każdego słowa, sprawdzamy czy każda jego literka wystąpiła gdzieś jako pierwsza. Jeśli tak jest, wypisujemy słowa zapamiętane pod tymi literkami.

pseudokod

```
 for every s:
  map[s[0]] = s
for every s:
  flag = TRUE
  for every char in s:
    if char not in map:
      flag = FALSE
  if flag:
    ...
```

amppz

K – Kwadracik

Najszybsze rozwiązanie:
Jagiellonian 3 (0:08)

Autor zadania:
Kamil Dębowski

K – Kwadracik

Treść zadania

Nauczyciel wybrał liczby h i w, po czym wypisał liczby od 1 do $h \cdot w$ w prostokącie $h \times w$. Zmazał wszystko poza kwadracikiem 2×2. Dla danego kwadracika 4 liczb, znajdź możliwe h i w.

K – Kwadracik

Kwadracik musi wyglądać tak:

```
x x+1
y y+1
```

A różnica między dwoma wierszami to w.

K – Kwadracik

Kwadracik musi wyglądać tak:
`x x+1`
`y y+1`
A różnica między dwoma wierszami to w.

`a b`
`c d`
Musi być $w = c - a$.
Sprawdzamy: $b == a + 1$, $d == c + 1$, $w \geq 2$.
Ale kwadracik musi mieścić się w planszy!

K – Kwadracik

Pełne rozwiązanie

$w = c - a$

```
a + 1 == b and c + 1 == d and w >= 2 and a % w != 0
```

$h = d/w + 1$ albo po prostu $h = 100\,000$

amppz

G – Gra MPO

Najszybsze rozwiązanie:
UWr 9 (0:14)

Autor zadania:
Kamil Dębowski

G – Gra MPO

Treść zadania

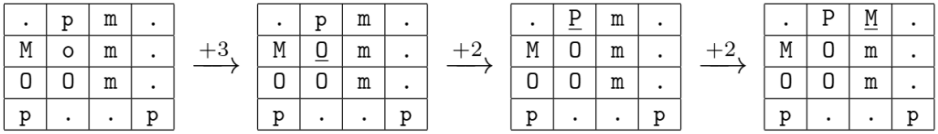
Dana jest mała kwadratowa plansza z polami kilku typów:

- Miasto
- Park
- Ocean

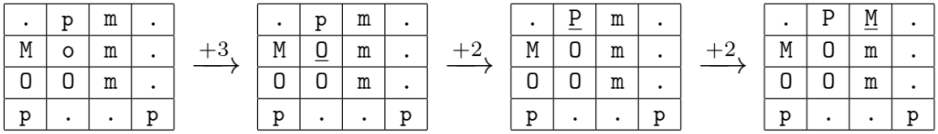
Każdy typ może być zbudowany (wielka litera) lub niezbudowany (mała litera).

Dostajemy 1 punkt za zbudowane pole oraz 1 punkt za parę sąsiadujących miasto-park oraz ocean-ocean. Mamy zasymulować wykonywanie ruchu z największą liczbą punktów. **Każdy ruch musi dawać co najmniej dwa punkty.** Wypisujemy końcową planszę i liczbę punktów.

G – Gra MPO



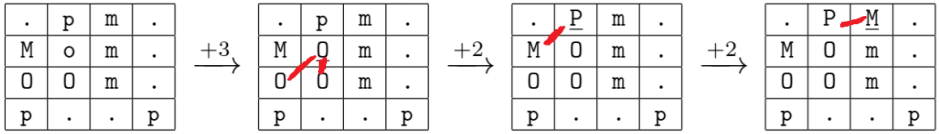
G – Gra MPO



Możliwe rozwiązanie

Bezpośrednia symulacja kolejką priorytetową (lub bez).

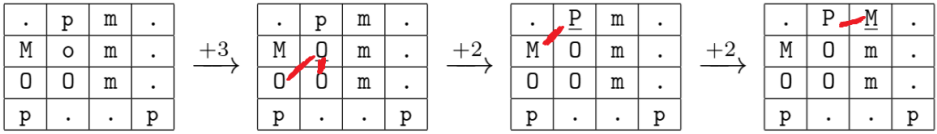
G – Gra MPO



Wygodne rozwiązanie

Pola „zarażają” sąsiadów, więc zadziała DFS. Jeśli pole jest wielką literą, to przeglądaj wszystkich sąsiadów i sprawdzaj pary M-p, P-m, O-o. $\mathcal{O}(n^2)$

G – Gra MPO



Wygodne rozwiązanie

Pola „zarażają” sąsiadów, więc zadziała DFS. Jeśli pole jest wielką literą, to przeglądaj wszystkich sąsiadów i sprawdzaj pary M-p, P-m, O-o. $\mathcal{O}(n^2)$

Przeglądanie sąsiadów

```
for dx = x-1..x+1:
 for dy = y-1..y+1:
   if dx != x or dy != y:
     ...
```

amppz

I – Intensywny trening

Najszybsze rozwiązanie:

UW10 (0:29), Huawei WarsawRC (0:28)

Autor zadania:

Kamil Dębowski

Treść zadania

Treść zadania

Dane:

- $a_1, a_2, \ldots a_n$ ciąg zdolności zawodników

Treść zadania

Dane:

- $a_1, a_2, \ldots a_n$ ciąg zdolności zawodników
- k wielkość drużyny

Treść zadania

Dane:

- $a_1, a_2, \ldots a_n$ ciąg zdolności zawodników
- k wielkość drużyny

Pary drużyn grają mecze.

Treść zadania

Dane:

- $a_1, a_2, \ldots a_n$ ciąg zdolności zawodników
- k wielkość drużyny

Pary drużyn grają mecze.
Szukamy:

Treść zadania

Dane:

- $a_1, a_2, \ldots a_n$ ciąg zdolności zawodników
- k wielkość drużyny

Pary drużyn grają mecze.
Szukamy:

- maksymalna różnica między sumarycznymi zdolnościami

Treść zadania

Dane:

- $a_1, a_2, \ldots a_n$ ciąg zdolności zawodników
- k wielkość drużyny

Pary drużyn grają mecze.
Szukamy:

- maksymalna różnica między sumarycznymi zdolnościami
- liczba par drużyn z taką różnicą

Maksymalna różnica

- Sortowanie – $a_1 \leq a_2 \leq \ldots \leq a_n$

Maksymalna różnica

- Sortowanie – $a_1 \leq a_2 \leq \ldots \leq a_n$
- Najsłabsza drużyna:

$$\sum_{i=1}^{i=k} a_i$$

Maksymalna różnica

- Sortowanie – $a_1 \leq a_2 \leq \ldots \leq a_n$
- Najsłabsza drużyna:

$$\sum_{i=1}^{i=k} a_i$$

- Najlepsza drużyna:

$$\sum_{i=n-k+1}^{i=n} a_i$$

Maksymalna różnica

- Sortowanie – $a_1 \leq a_2 \leq \ldots \leq a_n$
- Najsłabsza drużyna:

$$\sum_{i=1}^{i=k} a_i$$

- Najlepsza drużyna:

$$\sum_{i=n-k+1}^{i=n} a_i$$

- Wynik:

$$\sum_{i=n-k+1}^{i=n} a_i - \sum_{i=1}^{i=k} a_i$$

Liczba drużyn

Liczba drużyn – przypadek $a_k \neq a_{n-k+1}$

Liczba drużyn – przypadek $a_k \neq a_{n-k+1}$

Drużyny wybieramy niezależnie.

Liczba drużyn – przypadek $a_k \neq a_{n-k+1}$

Drużyny wybieramy niezależnie.

$$a_1 \leq a_2 \leq \ldots a_{x-1} < a_x = a_{x+1} = \ldots = a_k = \ldots = a_{y-1} = a_y < a_{y+1}$$

Liczba drużyn – przypadek $a_k \neq a_{n-k+1}$

Drużyny wybieramy niezależnie.

$$a_1 \leq a_2 \leq \ldots a_{x-1} < a_x = a_{x+1} = \ldots = a_k = \ldots = a_{y-1} = a_y < a_{y+1}$$

Wybieramy $a_1, a_2, \ldots, a_{x-1}$

Liczba drużyn – przypadek $a_k \neq a_{n-k+1}$

Drużyny wybieramy niezależnie.

$$a_1 \leq a_2 \leq \ldots a_{x-1} < a_x = a_{x+1} = \ldots = a_k = \ldots = a_{y-1} = a_y < a_{y+1}$$

Wybieramy $a_1, a_2, \ldots, a_{x-1}$ oraz $k - x + 1$ zawodników wśród $a_x, \ldots a_y$

Liczba drużyn – przypadek $a_k \neq a_{n-k+1}$

Drużyny wybieramy niezależnie.

$$a_1 \leq a_2 \leq \ldots a_{x-1} < a_x = a_{x+1} = \ldots = a_k = \ldots = a_{y-1} = a_y < a_{y+1}$$

Wybieramy $a_1, a_2, \ldots, a_{x-1}$ oraz $k - x + 1$ zawodników wśród $a_x, \ldots a_y$

$$\binom{y-x+1}{k-x+1}$$

Liczba drużyn – przypadek $a_k \neq a_{n-k+1}$

Drużyny wybieramy niezależnie.

$$a_1 \leq a_2 \leq \ldots a_{x-1} < a_x = a_{x+1} = \ldots = a_k = \ldots = a_{y-1} = a_y < a_{y+1}$$

Wybieramy $a_1, a_2, \ldots, a_{x-1}$ oraz $k - x + 1$ zawodników wśród $a_x, \ldots a_y$

$$\binom{y-x+1}{k-x+1}$$

Najlepszy team

Liczba drużyn – przypadek $a_k \neq a_{n-k+1}$

Drużyny wybieramy niezależnie.

$$a_1 \leq a_2 \leq \ldots a_{x-1} < a_x = a_{x+1} = \ldots = a_k = \ldots = a_{y-1} = a_y < a_{y+1}$$

Wybieramy $a_1, a_2, \ldots, a_{x-1}$ oraz $k - x + 1$ zawodników wśród $a_x, \ldots a_y$

$$\binom{y-x+1}{k-x+1}$$

Najlepszy team – analogicznie dwumian.

Liczba drużyn

Liczba drużyn – przypadek $a_k = a_{n-k+1}$

Liczba drużyn – przypadek $a_k = a_{n-k+1}$

$$a_{x-1} < a_x = a_{x+1} = \ldots = a_k = \ldots = a_{n-k+1} = \ldots = a_{y-1} = a_y < a_{y+1}$$

Liczba drużyn – przypadek $a_k = a_{n-k+1}$

$$a_{x-1} < a_x = a_{x+1} = \ldots = a_k = \ldots = a_{n-k+1} = \ldots = a_{y-1} = a_y < a_{y+1}$$

$a_1, a_2, \ldots, a_{x-1}$ do słabszej

Liczba drużyn – przypadek $a_k = a_{n-k+1}$

$$a_{x-1} < a_x = a_{x+1} = \ldots = a_k = \ldots = a_{n-k+1} = \ldots = a_{y-1} = a_y < a_{y+1}$$

$a_1, a_2, \ldots, a_{x-1}$ do słabszej, $a_{y+1}, a_{y+2}, \ldots, a_n$ do lepszej

Liczba drużyn – przypadek $a_k = a_{n-k+1}$

$$a_{x-1} < a_x = a_{x+1} = \ldots = a_k = \ldots = a_{n-k+1} = \ldots = a_{y-1} = a_y < a_{y+1}$$

$a_1, a_2, \ldots, a_{x-1}$ do słabszej, $a_{y+1}, a_{y+2}, \ldots, a_n$ do lepszej
Dobieramy pozostałych dowolnie

Liczba drużyn – przypadek $a_k = a_{n-k+1}$

$$a_{x-1} < a_x = a_{x+1} = \ldots = a_k = \ldots = a_{n-k+1} = \ldots = a_{y-1} = a_y < a_{y+1}$$

$a_1, a_2, \ldots, a_{x-1}$ do słabszej, $a_{y+1}, a_{y+2}, \ldots, a_n$ do lepszej
Dobieramy pozostałych dowolnie

$$\binom{y-x+1}{k-x+1, k-n+y} = \frac{(y-x+1)!}{(k-x+1)!(k-n+y)!(n-2\cdot k)!}$$

Liczba drużyn – przypadek $a_k = a_{n-k+1}$

$$a_{x-1} < a_x = a_{x+1} = \ldots = a_k = \ldots = a_{n-k+1} = \ldots = a_{y-1} = a_y < a_{y+1}$$

$a_1, a_2, \ldots, a_{x-1}$ do słabszej, $a_{y+1}, a_{y+2}, \ldots, a_n$ do lepszej
Dobieramy pozostałych dowolnie

$$\binom{y-x+1}{k-x+1, k-n+y} = \frac{(y-x+1)!}{(k-x+1)!(k-n+y)!(n-2\cdot k)!}$$

Uwaga! Szczególny przypadek $a_1 = a_n$

Liczba drużyn – przypadek $a_k = a_{n-k+1}$

$$a_{x-1} < a_x = a_{x+1} = \ldots = a_k = \ldots = a_{n-k+1} = \ldots = a_{y-1} = a_y < a_{y+1}$$

$a_1, a_2, \ldots, a_{x-1}$ do słabszej, $a_{y+1}, a_{y+2}, \ldots, a_n$ do lepszej
Dobieramy pozostałych dowolnie

$$\binom{y-x+1}{k-x+1, k-n+y} = \frac{(y-x+1)!}{(k-x+1)!(k-n+y)!(n-2\cdot k)!}$$

Uwaga! Szczególny przypadek $a_1 = a_n$ – nie wiadomo, która lepsza

Liczba drużyn – przypadek $a_k = a_{n-k+1}$

$$a_{x-1} < a_x = a_{x+1} = \ldots = a_k = \ldots = a_{n-k+1} = \ldots = a_{y-1} = a_y < a_{y+1}$$

$a_1, a_2, \ldots, a_{x-1}$ do słabszej, $a_{y+1}, a_{y+2}, \ldots, a_n$ do lepszej
Dobieramy pozostałych dowolnie

$$\binom{y-x+1}{k-x+1, k-n+y} = \frac{(y-x+1)!}{(k-x+1)!(k-n+y)!(n-2\cdot k)!}$$

Uwaga! Szczególny przypadek $a_1 = a_n$ – nie wiadomo, która lepsza – wynik/=2

Implementacja

Implementacja

- Max różnica

Implementacja

- Max różnica – proste pętle

Implementacja

- Max różnica – proste pętle
- Liczba teamów

Implementacja

- Max różnica – proste pętle
- Liczba teamów – obliczenie indeksów

Implementacja

- Max różnica – proste pętle
- Liczba teamów – obliczenie indeksów ; dwumiany Newtona

Implementacja

- Max różnica – proste pętle
- Liczba teamów – obliczenie indeksów ; dwumiany Newtona lub silnie.

Implementacja

- Max różnica – proste pętle
- Liczba teamów – obliczenie indeksów ; dwumiany Newtona lub silnie.

Złożoność

dynamik dwumiany

Implementacja

- Max różnica – proste pętle
- Liczba teamów – obliczenie indeksów ; dwumiany Newtona lub silnie.

Złożoność

dynamik dwumiany – $O(n^2)$

Implementacja

- Max różnica – proste pętle
- Liczba teamów – obliczenie indeksów ; dwumiany Newtona lub silnie.

Złożoność

dynamik dwumiany – $O(n^2)$
lub
silnie i ich odwrotności

Implementacja

- Max różnica – proste pętle
- Liczba teamów – obliczenie indeksów ; dwumiany Newtona lub silnie.

Złożoność

dynamik dwumiany – $O(n^2)$
lub
silnie i ich odwrotności – $O(n \log P)$ lub $O(n)$

amppz

B – Bagaże

Najszybsze rozwiązanie:
Jagiellonian 1 (0:55)

Autor zadania:
Marcin Smulewicz

B – Bagaże

Treść zadania

Dany graf skierowany.

B – Bagaże

Treść zadania

Dany graf skierowany. Krawędzie (loty):

B – Bagaże

Treść zadania

Dany graf skierowany. Krawędzie (loty):

- koszt

B – Bagaże

Treść zadania

Dany graf skierowany. Krawędzie (loty):

- koszt
- limit przewiezionego bagażu

B – Bagaże

Treść zadania

Dany graf skierowany. Krawędzie (loty):

- koszt
- limit przewiezionego bagażu

Bartek lata z bagażami, może zostawiać je w wierzchołkach.

B – Bagaże

Treść zadania

Dany graf skierowany. Krawędzie (loty):

- koszt
- limit przewiezionego bagażu

Bartek lata z bagażami, może zostawiać je w wierzchołkach.
Dla każdych x, y – minimalny koszt przewiezienia dwóch bagaży $x \to y$.

B – Bagaże

Ważne momenty

Obie walizki i Bartek w tym samym miejscu

B – Bagaże

Ważne momenty

Obie walizki i Bartek w tym samym miejscu

Jak lecą walizki między ważnymi momentami?

- lecą razem – krawędź z limitem 2

B – Bagaże

Ważne momenty

Obie walizki i Bartek w tym samym miejscu

Jak lecą walizki między ważnymi momentami?

- lecą razem – krawędź z limitem 2
- lecą osobno

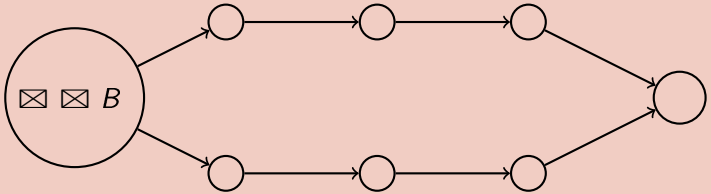
B – Bagaże

Ważne momenty

Obie walizki i Bartek w tym samym miejscu

Jak lecą walizki między ważnymi momentami?

- lecą razem – krawędź z limitem 2
- lecą osobno



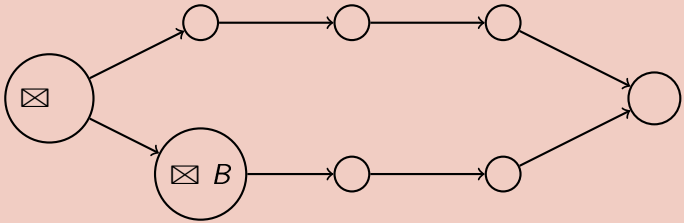
B – Bagaże

Ważne momenty

Obie walizki i Bartek w tym samym miejscu

Jak lecą walizki między ważnymi momentami?

- lecą razem – krawędź z limitem 2
- lecą osobno



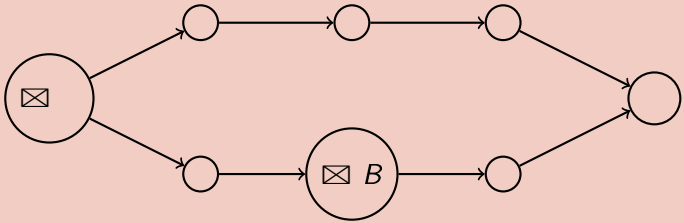
B – Bagaże

Ważne momenty

Obie walizki i Bartek w tym samym miejscu

Jak lecą walizki między ważnymi momentami?

- lecą razem – krawędź z limitem 2
- lecą osobno



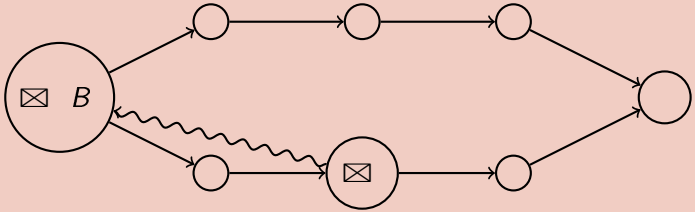
B – Bagaże

Ważne momenty

Obie walizki i Bartek w tym samym miejscu

Jak lecą walizki między ważnymi momentami?

- lecą razem – krawędź z limitem 2
- lecą osobno



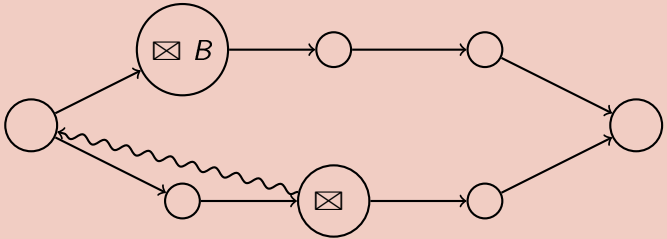
B – Bagaże

Ważne momenty

Obie walizki i Bartek w tym samym miejscu

Jak lecą walizki między ważnymi momentami?

- lecą razem – krawędź z limitem 2
- lecą osobno



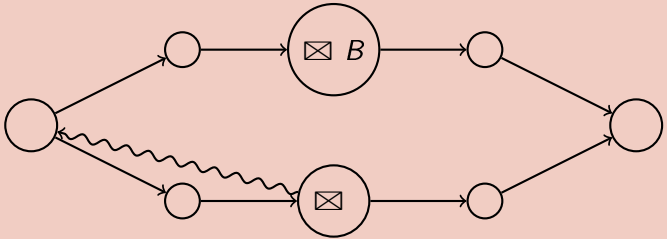
B – Bagaże

Ważne momenty

Obie walizki i Bartek w tym samym miejscu

Jak lecą walizki między ważnymi momentami?

- lecą razem – krawędź z limitem 2
- lecą osobno



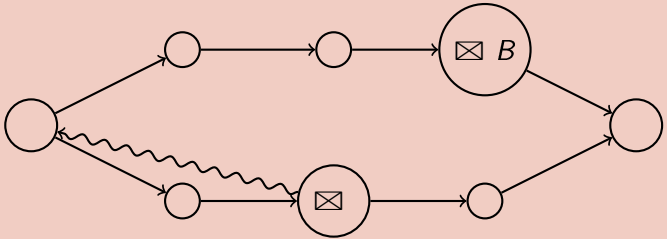
B – Bagaże

Ważne momenty

Obie walizki i Bartek w tym samym miejscu

Jak lecą walizki między ważnymi momentami?

- lecą razem – krawędź z limitem 2
- lecą osobno



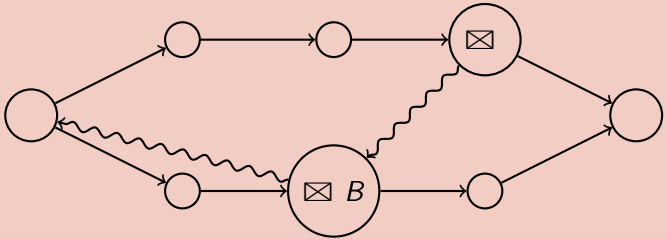
B – Bagaże

Ważne momenty

Obie walizki i Bartek w tym samym miejscu

Jak lecą walizki między ważnymi momentami?

- lecą razem – krawędź z limitem 2
- lecą osobno



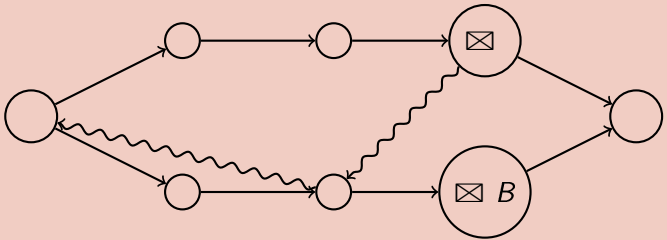
B – Bagaże

Ważne momenty

Obie walizki i Bartek w tym samym miejscu

Jak lecą walizki między ważnymi momentami?

- lecą razem – krawędź z limitem 2
- lecą osobno



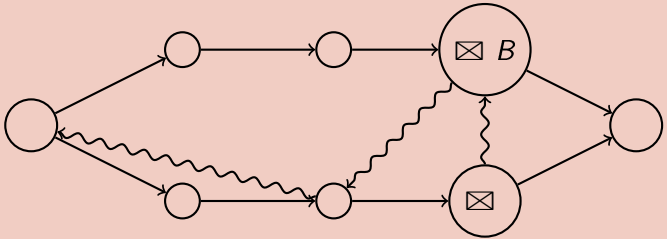
B – Bagaże

Ważne momenty

Obie walizki i Bartek w tym samym miejscu

Jak lecą walizki między ważnymi momentami?

- lecą razem – krawędź z limitem 2
- lecą osobno



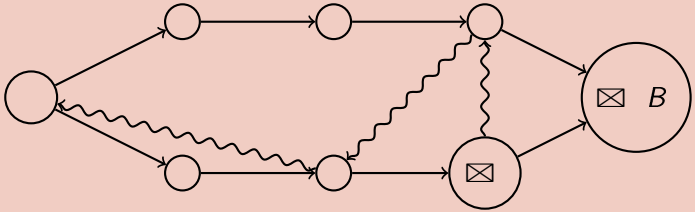
B – Bagaże

Ważne momenty

Obie walizki i Bartek w tym samym miejscu

Jak lecą walizki między ważnymi momentami?

- lecą razem – krawędź z limitem 2
- lecą osobno



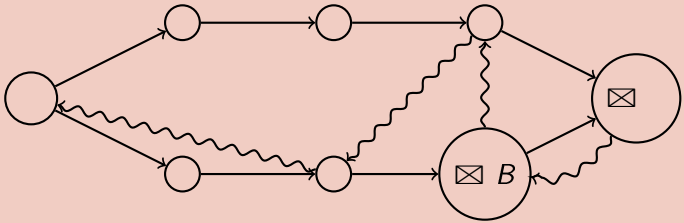
B – Bagaże

Ważne momenty

Obie walizki i Bartek w tym samym miejscu

Jak lecą walizki między ważnymi momentami?

- lecą razem – krawędź z limitem 2
- lecą osobno



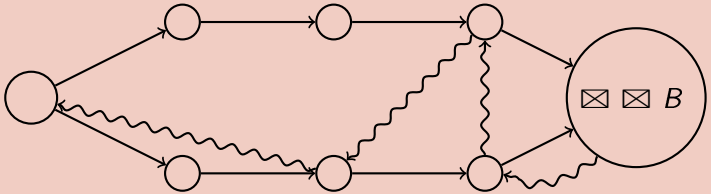
B – Bagaże

Ważne momenty

Obie walizki i Bartek w tym samym miejscu

Jak lecą walizki między ważnymi momentami?

- lecą razem – krawędź z limitem 2
- lecą osobno



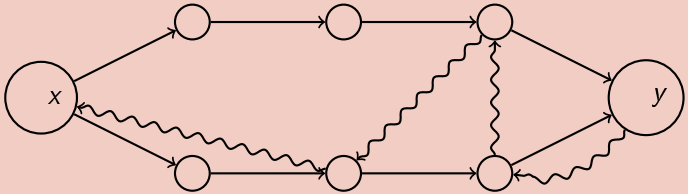
B – Bagaże

Ważne momenty

Obie walizki i Bartek w tym samym miejscu

Jak lecą walizki między ważnymi momentami?

- lecą razem – krawędź z limitem 2
- lecą osobno



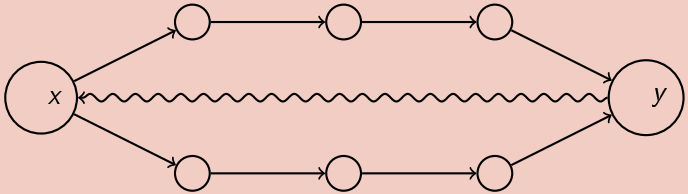
B – Bagaże

Ważne momenty

Obie walizki i Bartek w tym samym miejscu

Jak lecą walizki między ważnymi momentami?

- lecą razem – krawędź z limitem 2
- lecą osobno



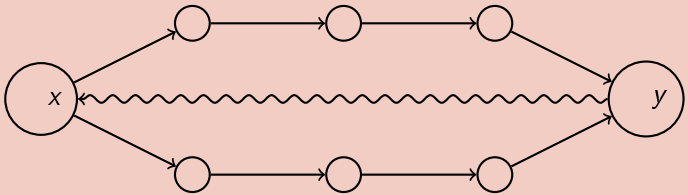
B – Bagaże

Ważne momenty

Obie walizki i Bartek w tym samym miejscu

Jak lecą walizki między ważnymi momentami?

- lecą razem – krawędź z limitem 2
- lecą osobno – b.s.o. $x \xrightarrow{1} y \xrightarrow{0} x \xrightarrow{1} y$



Implementacja

- najkrótsze ścieżki 0 bagaży – krawędzie $0, 1, 2$

Implementacja

- najkrótsze ścieżki 0 bagaży – krawędzie $0, 1, 2$
- najkrótsze ścieżki 1 bagaż – krawędzie $1, 2$

Implementacja

- najkrótsze ścieżki 0 bagaży – krawędzie $0, 1, 2$
- najkrótsze ścieżki 1 bagaż – krawędzie $1, 2$
- lecą osobno z x od y

Implementacja

- najkrótsze ścieżki 0 bagaży – krawędzie $0, 1, 2$
- najkrótsze ścieżki 1 bagaż – krawędzie $1, 2$
- lecą osobno z x od y – nowa krawędź $2 \cdot dist_1(x, y) + dist_0(y, x)$

Implementacja

- najkrótsze ścieżki 0 bagaży – krawędzie $0, 1, 2$
- najkrótsze ścieżki 1 bagaż – krawędzie $1, 2$
- lecą osobno z x od y – nowa krawędź $2 \cdot dist_1(x, y) + dist_0(y, x)$
- najkrótsze ścieżki 2 bagaże – krawędzie 2 (lecą razem) oraz nowe (lecą osobno).

Implementacja

- najkrótsze ścieżki 0 bagaży – krawędzie $0, 1, 2$
- najkrótsze ścieżki 1 bagaż – krawędzie $1, 2$
- lecą osobno z x od y – nowa krawędź $2 \cdot dist_1(x, y) + dist_0(y, x)$
- najkrótsze ścieżki 2 bagaże – krawędzie 2 (lecą razem) oraz nowe (lecą osobno).

Złożoność

Implementacja

- najkrótsze ścieżki 0 bagaży – krawędzie $0, 1, 2$
- najkrótsze ścieżki 1 bagaż – krawędzie $1, 2$
- lecą osobno z x od y – nowa krawędź $2 \cdot dist_1(x, y) + dist_0(y, x)$
- najkrótsze ścieżki 2 bagaże – krawędzie 2 (lecą razem) oraz nowe (lecą osobno).

Złożoność

Floyd-Warshall $O(n^3)$

amppz

J – Jedynki i zera

Najszybsze rozwiązanie:

UWr 9 (1:10), Huawei WarsawRC (0:59)

Autor zadania:

Yahor Dubovik

J – Jedynki i zera

Gra

0100101010010101
1010011101011011
1010010010100101

Plansza rozmiaru $h \times w$, $h \leq 8$, $w \leq 100\,000$.

Operacje:

- Zaneguj bit.
- Zaneguj kolumnę.
- Zaneguj wiersz.

Trudność planszy: liczba operacji potrzebnych do wyzerowania całej planszy.

J – Jedynki i zera

Gra

0100101010010101
1010011101011011
1010010010100101

Plansza rozmiaru $h \times w$, $h \leq 8$, $w \leq 100\,000$.

Operacje:

- Zaneguj bit.
- Zaneguj kolumnę.
- Zaneguj wiersz.

Trudność planszy: liczba operacji potrzebnych do wyzerowania całej planszy.

Zadanie

- Dana jest początkowa plansza i ciąg q operacji ($q \leq 100\,000$).
- Oblicz trudność początkową i po każdej operacji.

Pojedyncza kolumna

0
1
1
0

Reprezentujemy kolumnę przez zbiór (maskę bitową) $S \subseteq \{0, \ldots, h-1\}$.
Operacje:

- Zaneguj bit y: $S' = S \oplus \{y\} = S$ xor $\{y\}$.
- Zaneguj kolumnę: $S' = S \oplus \{0, \ldots, h-1\}$.

J – Jedynki i zera

Pojedyncza kolumna

0
1
1
0

Reprezentujemy kolumnę przez zbiór (maskę bitową) $S \subseteq \{0, \ldots, h-1\}$.

Operacje:

- Zaneguj bit y: $S' = S \oplus \{y\} = S$ xor $\{y\}$.
- Zaneguj kolumnę: $S' = S \oplus \{0, \ldots, h-1\}$.

Trudność kolumny

$$K(S) = \min(|S|, h - |S| + 1)$$

Początkowy stan

- Najpierw operacje na wierszach.
- Wybierzmy $W \subseteq \{0, \ldots, h-1\}$: zbiór operacji na całych wierszach.
- W_0 = zanegowane już wiersze
- $KS(W)$ = sumaryczna trudność kolumn po zanegowaniu wierszy W
- P = trudność całej planszy

Początkowy stan

- Najpierw operacje na wierszach.
- Wybierzmy $W \subseteq \{0, \ldots, h-1\}$: zbiór operacji na całych wierszach.
- W_0 = zanegowane już wiersze
- $KS(W)$ = sumaryczna trudność kolumn po zanegowaniu wierszy W
- P = trudność całej planszy

$$W_0 = \{\}$$
$$KS(W) = \sum_x K(S_x \oplus W)$$
$$P = \min_W(|W \oplus W_0| + KS(W))$$

Złożoność początkowego stanu

- Obliczenie $KS(W)$: $O(w2^h)$
- Obliczenie P: $O(2^h)$.

Zanegowanie bitu lub kolumny

$$
\begin{aligned}
S'_x &= S_x \oplus \{y\} \qquad \text{lub} \\
S'_x &= S_x \oplus \{0, \ldots, h-1\} \\
KS'(W) &= KS(W) - K(S_x \oplus W) + K(S'_x \oplus W) \\
P &= \min_W(|W \oplus W_0| + KS(W))
\end{aligned}
$$

Zanegowanie bitu lub kolumny

$$S'_x = S_x \oplus \{y\} \qquad \text{lub}$$
$$S'_x = S_x \oplus \{0, \ldots, h-1\}$$
$$KS'(W) = KS(W) - K(S_x \oplus W) + K(S'_x \oplus W)$$
$$P = \min_W(|W \oplus W_0| + KS(W))$$

Złożoność

- Poprawienie S_x: $O(1)$
- Poprawienie $KS(W)$: $O(2^h)$
- Poprawienie P: $O(2^h)$

Zanegowanie wiersza

$$W_0' = W_0 \oplus \{y\}$$
$$P = \min_W(|W \oplus W_0| + KS(W))$$

Zanegowanie wiersza

$$W_0' = W_0 \oplus \{y\}$$
$$P = \min_W(|W \oplus W_0| + KS(W))$$

Złożoność

- Poprawienie W_0: $O(1)$
- Poprawienie P: $O(2^h)$

Złożoność całego rozwiązania

- Początkowa plansza: $O(w2^h)$
- Każda operacja: $O(2^h)$
- Razem: $O((w+q)2^h)$

amppz

H – Hop

Najszybsze rozwiązanie:

UW10 (0:48)

Autor zadania:

Marcin Smulewicz

H – Hop

Treść zadania

Możemy ustawiać wysokość poprzeczki (w skoku wzwyż) na liczbę całkowitą od 1 do n. P-stwo przeskoczenia wysokości i to p_i. Znajdź maksymalne EV najwyższej przeskoczonej poprzeczki.

Brutalne rozwiązanie

$$dp_i = max(p_j \cdot dp_j + (1 - p_j) \cdot i)$$

H – Hop

Treść zadania

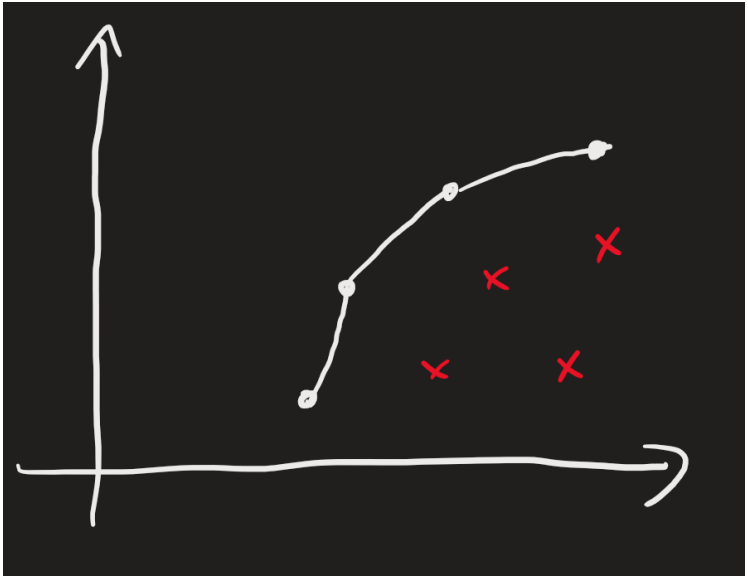
Możemy ustawiać wysokość poprzeczki (w skoku wzwyż) na liczbę całkowitą od 1 do n. P-stwo przeskoczenia wysokości i to p_i. Znajdź maksymalne EV najwyższej przeskoczonej poprzeczki.

Brutalne rozwiązanie

$$dp_i = max(p_j \cdot dp_j + (1 - p_j) \cdot i)$$

$$dp_a = max(Y_j + X_j \cdot a)$$

H – Hop



amppz

D – Dodawanie ułamków

Najszybsze rozwiązanie:
Jagiellonian 2 (2:08)

Autor zadania:
Marek Sokołowski

D – Dodawanie ułamków

Treść zadania

Znaleźć najmniejszą dodatnią sumę liczb wymiernych o mianownikach $\leq n$:

$$\sum \frac{a_i}{b_i} > 0$$
$$|a_i|, b_i \leq n$$

D – Dodawanie ułamków

Treść zadania

Znaleźć najmniejszą dodatnią sumę liczb wymiernych o mianownikach $\leq n$:

$$\sum \frac{a_i}{b_i} > 0$$
$$|a_i|, b_i \leq n$$

Limity

t testów, $n_i \leq 50\,000$, $\sum n_i \leq 4\,000\,000$

D – Dodawanie ułamków

W jaką sumę celujemy?

Niech $M = \mathrm{NWW}(1, 2, \ldots, n)$.

$$M \cdot \frac{a_i}{b_i} \in \mathbb{Z}$$

$$M \sum \frac{a_i}{b_i} \geq 1$$

$$\sum \frac{a_i}{b_i} \geq \frac{1}{M}$$

D – Dodawanie ułamków

W jaką sumę celujemy?

Niech $M = \mathrm{NWW}(1, 2, \ldots, n)$.

$$M \cdot \frac{a_i}{b_i} \in \mathbb{Z}$$

$$M \sum \frac{a_i}{b_i} \geq 1$$

$$\sum \frac{a_i}{b_i} \geq \frac{1}{M}$$

Cel

$$\sum \frac{a_i}{b_i} = \frac{1}{M}$$

Mianowniki: potęgi liczb pierwszych

$$b_i = p_i^{\alpha_i} \leq n$$

$$\mathrm{NWW}(p_1^{\alpha_1}, p_2^{\alpha_2}, \ldots, p_k^{\alpha_k}) = p_1^{\alpha_1} p_2^{\alpha_2} \ldots p_k^{\alpha_k} = M$$

$$\frac{a_0}{1} + \frac{a_1}{p_1^{\alpha_1}} + \frac{a_2}{p_2^{\alpha_2}} + \ldots + \frac{a_k}{p_k^{\alpha_k}} = \frac{1}{M}$$

$$Ma_0 + \frac{M}{p_1^{\alpha_1}} a_1 + \frac{M}{p_2^{\alpha_2}} a_2 + \ldots + \frac{M}{p_k^{\alpha_k}} a_k = 1$$

$$\frac{M}{p_1^{\alpha_1}} a_1 + \frac{M}{p_2^{\alpha_2}} a_2 + \ldots + \frac{M}{p_k^{\alpha_k}} a_k \equiv 1 \pmod{M}$$

Chińskie twierdzenie o resztach

$$\begin{aligned}
\frac{M}{p_1^{\alpha_1}} a_1 &\equiv 1 \pmod{p_1^{\alpha_1}} \\
\frac{M}{p_2^{\alpha_2}} a_2 &\equiv 1 \pmod{p_2^{\alpha_2}} \\
&\dots \\
\frac{M}{p_k^{\alpha_k}} a_k &\equiv 1 \pmod{p_k^{\alpha_k}}
\end{aligned}$$

Rozpiszmy iloczyny

$$
\begin{aligned}
(p_2^{\alpha_2} p_3^{\alpha_3} \dots p_k^{\alpha_k}) a_1 &\equiv 1 \pmod{p_1^{\alpha_1}} \\
(p_1^{\alpha_1} p_3^{\alpha_3} \dots p_k^{\alpha_k}) a_2 &\equiv 1 \pmod{p_2^{\alpha_2}} \\
&\dots \\
(p_1^{\alpha_1} p_2^{\alpha_2} \dots p_{k-1}^{\alpha_{k-1}}) a_k &\equiv 1 \pmod{p_k^{\alpha_k}}
\end{aligned}
$$

Rozwiązanie

$$a_1 = (p_2^{\alpha_2} p_3^{\alpha_3} \dots p_k^{\alpha_k})^{-1} \bmod p_1^{\alpha_1}$$
$$a_2 = (p_1^{\alpha_1} p_3^{\alpha_3} \dots p_k^{\alpha_k})^{-1} \bmod p_2^{\alpha_2}$$
$$\dots$$
$$a_k = (p_1^{\alpha_1} p_2^{\alpha_2} \dots p_{k-1}^{\alpha_{k-1}})^{-1} \bmod p_k^{\alpha_k}$$
$$a_0 = \frac{1}{M} - \frac{a_1}{p_1^{\alpha_1}} - \frac{a_2}{p_2^{\alpha_2}} - \dots - \frac{a_k}{p_k^{\alpha_k}}$$

D – Dodawanie ułamków

Rozwiązanie

$$a_1 = (p_2^{\alpha_2} p_3^{\alpha_3} \dots p_k^{\alpha_k})^{-1} \bmod p_1^{\alpha_1}$$
$$a_2 = (p_1^{\alpha_1} p_3^{\alpha_3} \dots p_k^{\alpha_k})^{-1} \bmod p_2^{\alpha_2}$$
$$\dots$$
$$a_k = (p_1^{\alpha_1} p_2^{\alpha_2} \dots p_{k-1}^{\alpha_{k-1}})^{-1} \bmod p_k^{\alpha_k}$$
$$a_0 = \frac{1}{M} - \frac{a_1}{p_1^{\alpha_1}} - \frac{a_2}{p_2^{\alpha_2}} - \dots - \frac{a_k}{p_k^{\alpha_k}}$$

Złożoność

- Liczba potęg liczb pierwszych: $O(n/\log n)$.
- Złożoność $O(\sum n_i^2 / \log^2 n_i)$.

D – Dodawanie ułamków

Rozwiązanie

$$a_1 = (p_2^{\alpha_2} p_3^{\alpha_3} \dots p_k^{\alpha_k})^{-1} \bmod p_1^{\alpha_1}$$
$$a_2 = (p_1^{\alpha_1} p_3^{\alpha_3} \dots p_k^{\alpha_k})^{-1} \bmod p_2^{\alpha_2}$$
$$\dots$$
$$a_k = (p_1^{\alpha_1} p_2^{\alpha_2} \dots p_{k-1}^{\alpha_{k-1}})^{-1} \bmod p_k^{\alpha_k}$$
$$a_0 = \frac{1}{M} - \frac{a_1}{p_1^{\alpha_1}} - \frac{a_2}{p_2^{\alpha_2}} - \dots - \frac{a_k}{p_k^{\alpha_k}}$$

Złożoność

- Liczba potęg liczb pierwszych: $O(n/\log n)$.
- Złożoność $O(\sum n_i^2 / \log^2 n_i)$.
- Za wolno!

Funkcja pomocnicza

$$F(q^\beta, n) = \left(\prod_{p_i^{\alpha_i} \le n, p_i \neq q} p_i^{\alpha_i} \right) \bmod q^\beta$$

$$F(q^\beta, p^\alpha) = F(q^\beta, p^\alpha - 1) \cdot p \bmod q^\beta$$

$$F(q^\beta, n) = F(q^\beta, n-1) \qquad n \notin \{p^\alpha, p \neq q\}$$

Funkcja pomocnicza

$$F(q^\beta, n) = \left(\prod_{p_i^{\alpha_i} \le n, p_i \neq q} p_i^{\alpha_i} \right) \bmod q^\beta$$

$$F(q^\beta, p^\alpha) = F(q^\beta, p^\alpha - 1) \cdot p \bmod q^\beta$$

$$F(q^\beta, n) = F(q^\beta, n-1) \qquad n \notin \{p^\alpha, p \neq q\}$$

Liczymy tylko $F(q^\beta, p^\alpha)$.

D – Dodawanie ułamków

Algorytm

- Posortuj testy po n.
- Dla każdego n w kolejności rosnącej:
  - Oblicz $F(p_i^{\alpha_i}, n)$.
  - Oblicz $a_i = F(p_i^{\alpha_i}, n)^{-1} \bmod p_i^{\alpha_i}$.
  - Oblicz $a_0 = \text{round}\left(\frac{1}{M} - \sum \frac{a_i}{p_i^{\alpha_i}}\right)$.
  - Wynik: $\frac{a_0}{1} + \frac{a_1}{p_1^{\alpha_1}} + \frac{a_2}{p_2^{\alpha_2}} + \ldots + \frac{a_k}{p_k^{\alpha_k}} = \frac{1}{M}$.

D – Dodawanie ułamków

Algorytm

- Posortuj testy po n.
- Dla każdego n w kolejności rosnącej:
  - Oblicz $F(p_i^{\alpha_i}, n)$.
  - Oblicz $a_i = F(p_i^{\alpha_i}, n)^{-1} \bmod p_i^{\alpha_i}$.
  - Oblicz $a_0 = \text{round}\left(\frac{1}{M} - \sum \frac{a_i}{p_i^{\alpha_i}}\right)$.
  - Wynik: $\frac{a_0}{1} + \frac{a_1}{p_1^{\alpha_1}} + \frac{a_2}{p_2^{\alpha_2}} + \ldots + \frac{a_k}{p_k^{\alpha_k}} = \frac{1}{M}$.

Złożoność

- Obliczanie funkcji F: $O((N/\log N)^2)$, gdzie $N = \max n_i$.
- Każdy test: $O(\frac{n}{\log n} \cdot \log n)$.
- Razem: $O(N^2/log^2 N + \sum n_i)$.

amppz

L – Licz trójki PKN

Najszybsze rozwiązanie:

UWr 1 (2:35)

Autor zadania:

Jakub Onufry Wojtaszczyk, Kamil Dębowski

L – Licz trójki PKN

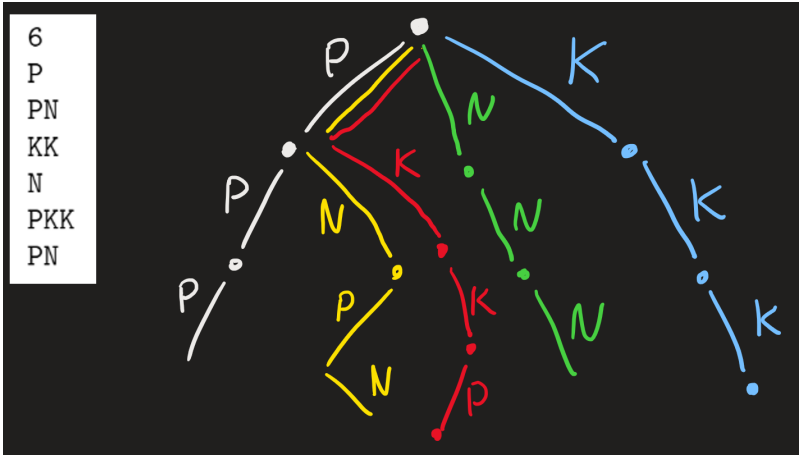
Treść zadania

Dane jest $n \leq 10^5$ słów nad alfabetem `PKN`, suma długości $L \leq 10^6$. Rozważamy ich nieskończone rozwinięcia, np. $PK \to PKPKPKPK$.
Słowo reprezentuje kolejne ruchy zawodnika w papier-kamień-nożyce. Zlicz trójki zawodników, że A bije B, B bije C, C bije A.

L – Licz trójki PKN

Rozwiązanie brutalne

Zbuduj TRIE, w każdym wierzchołku dodaj iloczyn rozmiaru dzieci.



L – Licz trójki PKN

Trójki

Szukamy trójek, które mają ten sam prefiks i potem trzy różne znaki.
Na przykład `P`, `PKK`, `PN` mają prefiks `P` i różnią się na drugiej pozycji.

Pierwsza różnica

Nieskończone rozwinięcia słów długości X i Y różnią się najpóźniej na pozycji $X + Y$.

Szukanie różnicy

Hashe, lcp, brutalnie.

Sortowanie

Można posortować słowa kompatarotem szybszym niż $A + B < B + A$.

Skierowane trójki w turnieju

Wystarczy policzyć stopnie wierzchołków indeg, outdeg, remisy.

L – Licz trójki PKN

Zbicie do najmniejszego okresu

PKPKPK $\rightarrow$ PK

```
for d = 1..len:
  if len % d == 0:
    if s[0..len-d-1] == s[d..len-1]:
      ...
```

L – Licz trójki PKN

Zbicie do najmniejszego okresu

PKPKPK $\rightarrow$ PK

```
for d = 1..len:
  if len % d == 0:
    if s[0..len-d-1] == s[d..len-1]:
      ...
```

Piękne rozwiązanie

Kompresujemy słowa do okresów, po czym budujemy TRIE. $\mathcal{O}(\text{okresy} + n)$

Gratulujemy drużynie Jagiellonian 2 za submita w 3:58, max czas 0.15/4s.

L – Licz trójki PKN

```cpp
// Onufry, O(N+L)
#include <bits/stdc++.h>

std::vector<std::string> strs;
std::vector<long long> reps(1000000, 1);

long long solve(const std::vector<int> &qset, int p) {
  int cn(0), sho(-1); std::vector<int> sub[3]; std::vector<long long> siz(3, 0);
  for (int i : qset) {
    siz[(strs[i][p % strs[i].size()] - 'I') / 3] += reps[i];
    if ((int) strs[i].size() * 2 < p) {
      if (sho != -1) { reps[sho] += reps[i]; continue; } else sho = i;
    }
    cn += 1;
    sub[(strs[i][p % strs[i].size()] - 'I') / 3].push_back(i);
  }
  if (cn < 3) return 0;
  long long res = siz[0] * siz[1] * siz[2];
  for (int i = 0; i < 3; ++i) res += solve(sub[i], p + 1);
  return res;
}

int main() {
  std::ios::sync_with_stdio(false); std::cin.tie(nullptr);
  int N; std::cin >> N;
  for (int i = 0; i < N; ++i) {
    std::string S; std::cin >> S; strs.push_back(S);
  }
  std::vector<int> a(strs.size()); std::iota(a.begin(), a.end(), 0);
  std::cout << solve(a, 0) << std::endl;
}
```

Dowód złożoności

Dwa słowa o długościach X i Y różnią się najpóźniej na pozycji $X + Y$.
Jeśli nieskończone rozwinięcia słów o długościach X i Y wciąż są takie same na pozycji $X + Y$, to są takie same i skompresowaliśmy je wcześniej do jednego słowa. Więc w TRIE w wierzchołku na głębokości d mamy co najwyżej jedno słowo o długości co najwyżej $d/2$.

Dowód złożoności

Dwa słowa o długościach X i Y różnią się najpóźniej na pozycji $X + Y$.
Jeśli nieskończone rozwinięcia słów o długościach X i Y wciąż są takie same na pozycji $X + Y$, to są takie same i skompresowaliśmy je wcześniej do jednego słowa. Więc w TRIE w wierzchołku na głębokości d mamy co najwyżej jedno słowo o długości co najwyżej $d/2$.
Zatem w wierzchołku na głębokości d interesują nas słowa trzech typów:

- Jedno krótkie słowo.
- Słowa o długościach $[d/2, d]$.
- Dłuższe słowa.

Suma po takich licznościach to co najwyżej $2 \cdot L$.

amppz

C – Cierpliwe krowy

Najszybsze rozwiązanie:
Jagiellonian 2 (2:42)

Autor zadania:
Kamil Dębowski

C – Cierpliwe krowy

Treść zadania

Dany ciąg h_i – wysokości trawy. W jednym kroku krowa i:

- jeśli $h_i > 0$, je trawę z i
- jeśli $h_i = 0$, je trawę z $i - 1$ lub $i + 1$.

Zjedzenie całej trawy

C – Cierpliwe krowy

Treść zadania

Dany ciąg h_i – wysokości trawy. W jednym kroku krowa i:

- jeśli $h_i > 0$, je trawę z i
- jeśli $h_i = 0$, je trawę z $i - 1$ lub $i + 1$.

Zjedzenie całej trawy – minimalna liczba kroków.

Wyszukiwanie binarne

Czy zjedzą trawę w x czasu?

Wyszukiwanie binarne

Czy zjedzą trawę w x czasu?

Strategia zachłanna

Wyszukiwanie binarne

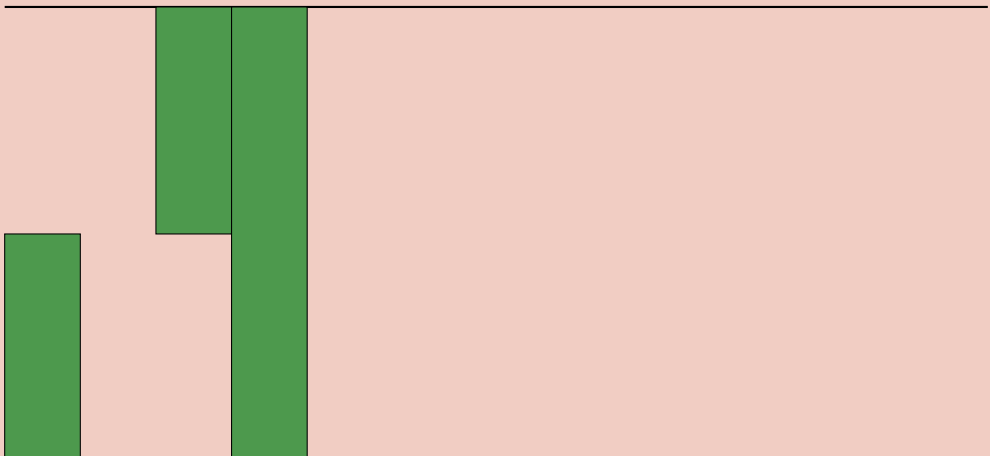
Czy zjedzą trawę w x czasu?

Strategia zachłanna

Wyszukiwanie binarne

Czy zjedzą trawę w x czasu?

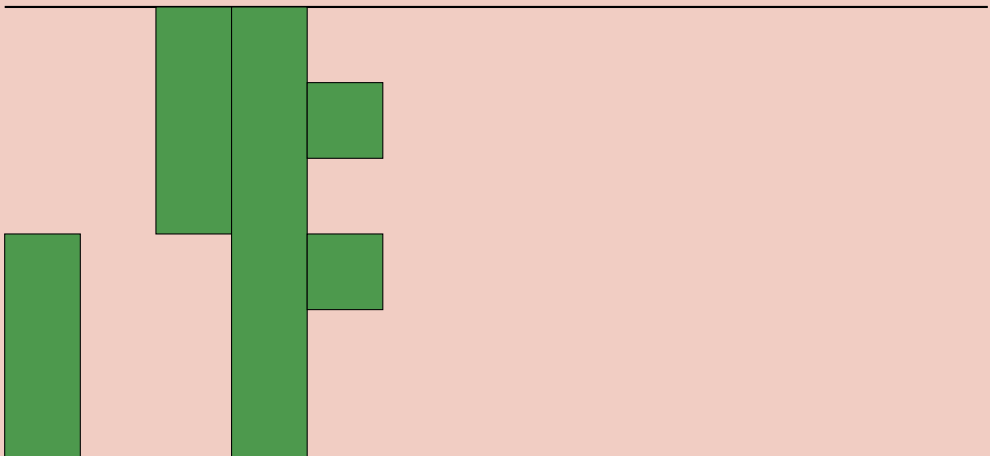
Strategia zachłanna



Wyszukiwanie binarne

Czy zjedzą trawę w x czasu?

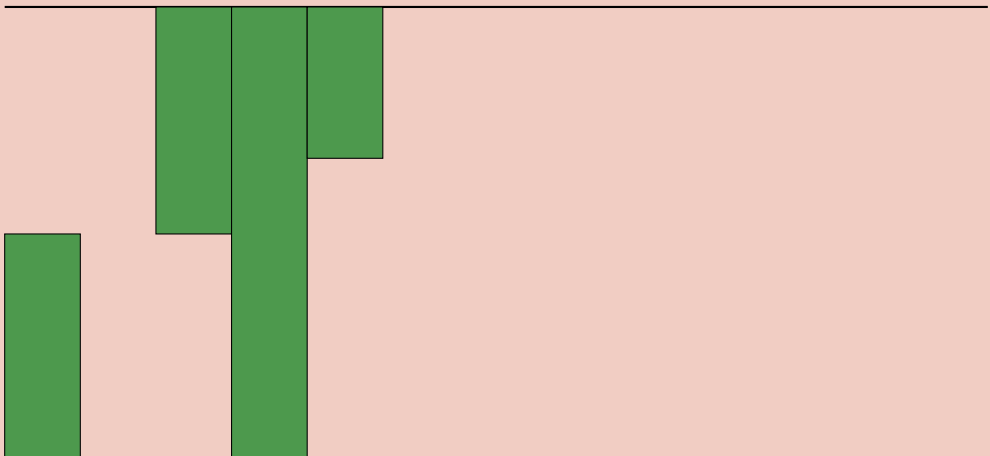
Strategia zachłanna



Wyszukiwanie binarne

Czy zjedzą trawę w x czasu?

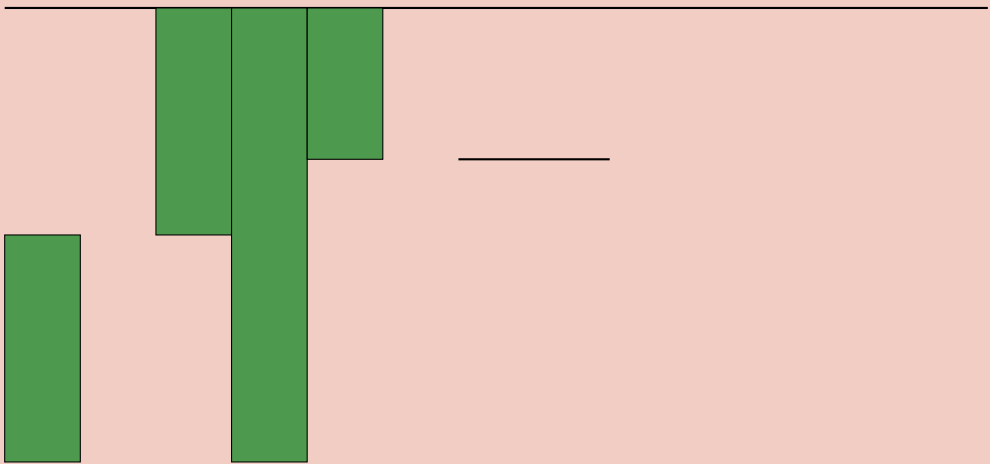
Strategia zachłanna



Wyszukiwanie binarne

Czy zjedzą trawę w x czasu?

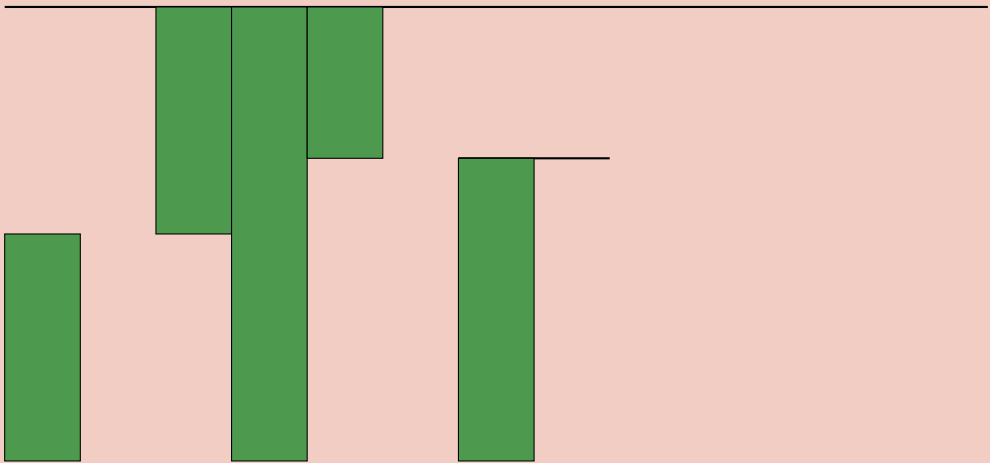
Strategia zachłanna



Wyszukiwanie binarne

Czy zjedzą trawę w x czasu?

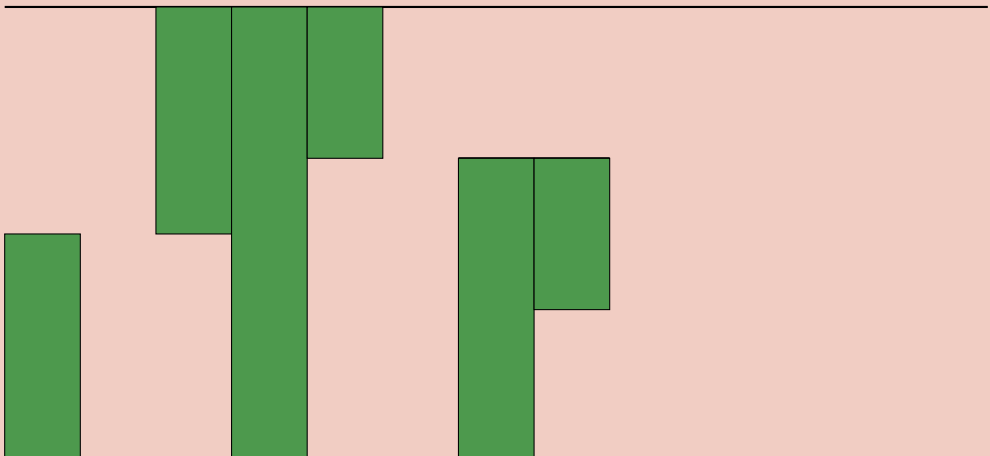
Strategia zachłanna



Wyszukiwanie binarne

Czy zjedzą trawę w x czasu?

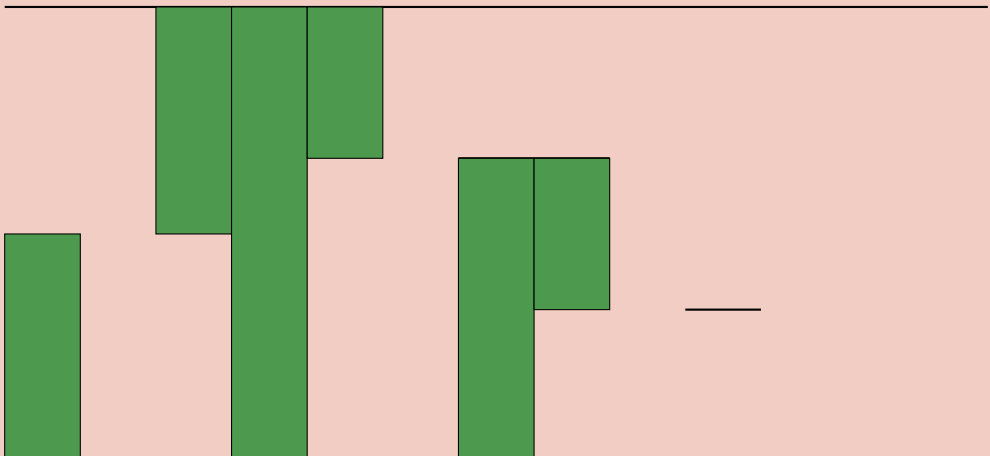
Strategia zachłanna



Wyszukiwanie binarne

Czy zjedzą trawę w x czasu?

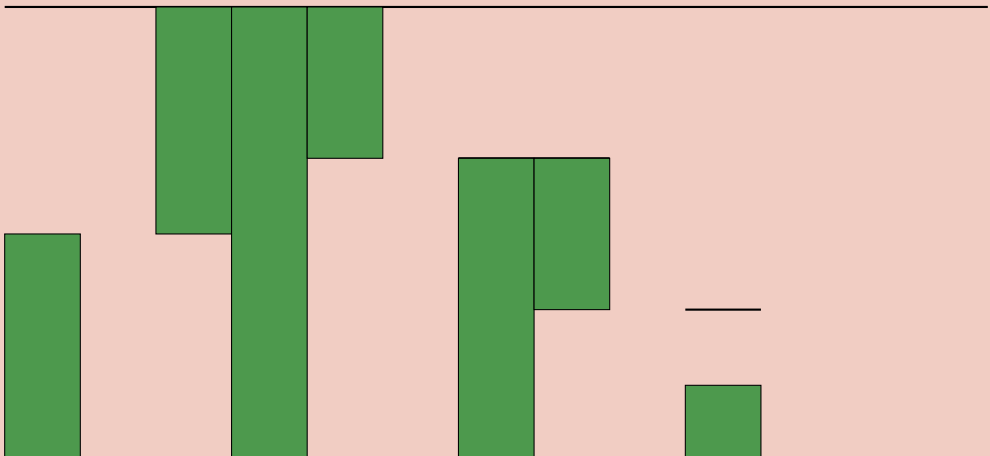
Strategia zachłanna



Wyszukiwanie binarne

Czy zjedzą trawę w x czasu?

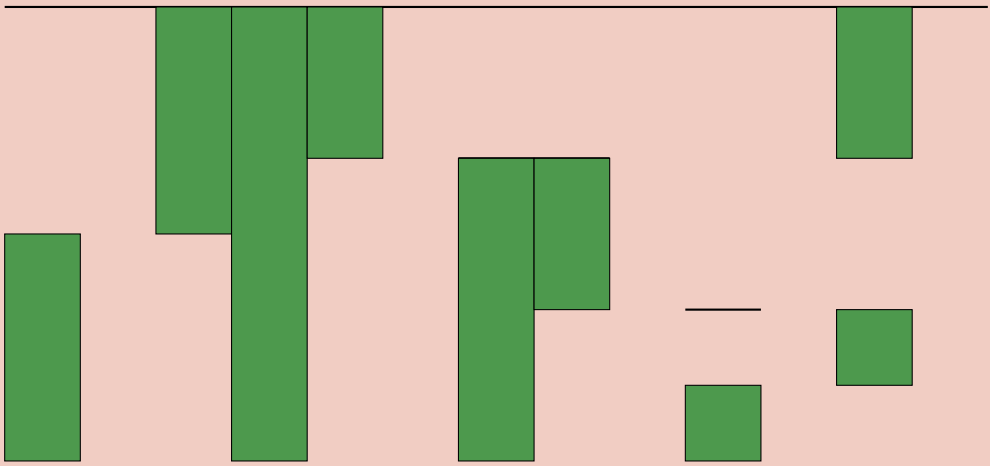
Strategia zachłanna



Wyszukiwanie binarne

Czy zjedzą trawę w x czasu?

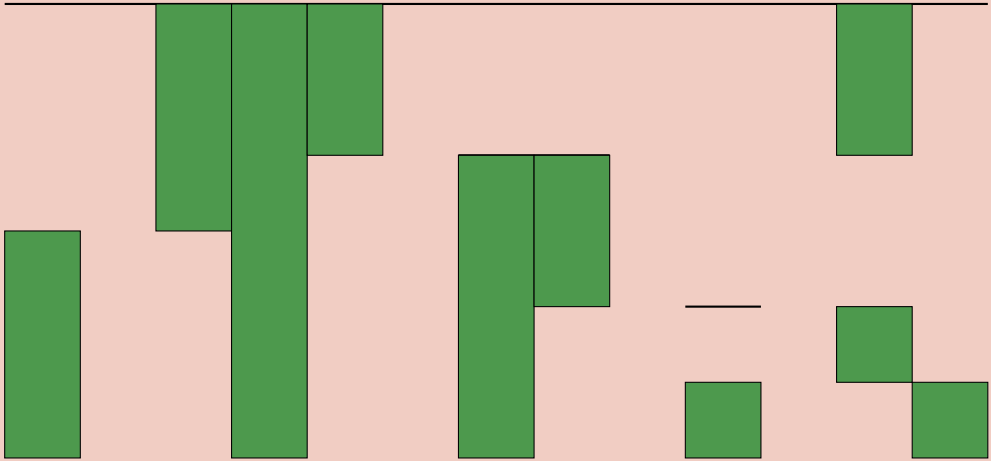
Strategia zachłanna



Wyszukiwanie binarne

Czy zjedzą trawę w x czasu?

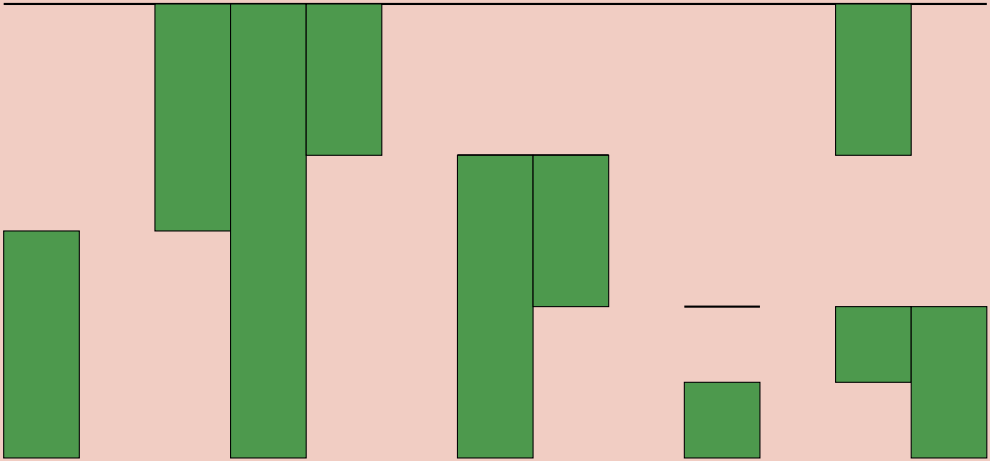
Strategia zachłanna



Wyszukiwanie binarne

Czy zjedzą trawę w x czasu?

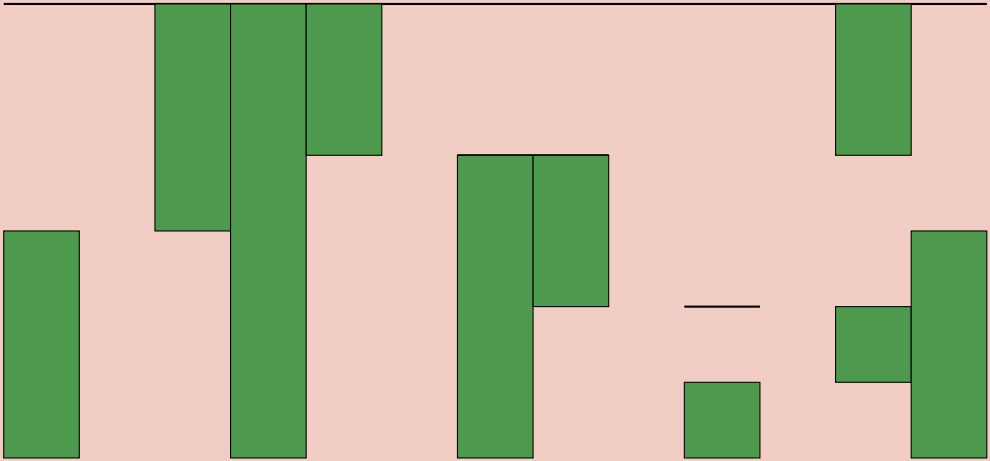
Strategia zachłanna



Wyszukiwanie binarne

Czy zjedzą trawę w x czasu?

Strategia zachłanna



Podsumowanie strategii

Podsumowanie strategii

- trzeba pomóc poprzedniej na przedziale

Podsumowanie strategii

- trzeba pomóc poprzedniej na przedziale
  - nie zdąży przed deadlinem – następna musi pomóc (1 przedział)

Podsumowanie strategii

- trzeba pomóc poprzedniej na przedziale
  - nie zdąży przed deadlinem – następna musi pomóc (1 przedział)
  - zdążymy – pomagamy następnej (≤ 2 przedziały)

Podsumowanie strategii

- trzeba pomóc poprzedniej na przedziale
  - nie zdąży przed deadlinem – następna musi pomóc (1 przedział)
  - zdążymy – pomagamy następnej (≤ 2 przedziały)
- poprzednia pomaga (≤ 2 przedziały)

Podsumowanie strategii

- trzeba pomóc poprzedniej na przedziale
  - nie zdąży przed deadlinem – następna musi pomóc (1 przedział)
  - zdążymy – pomagamy następnej (≤ 2 przedziały)
- poprzednia pomaga (≤ 2 przedziały)
  - nie zdąży przed limitem – następna musi pomóc (1 przedział)

Podsumowanie strategii

- trzeba pomóc poprzedniej na przedziale
  - nie zdąży przed deadlinem – następna musi pomóc (1 przedział)
  - zdążymy – pomagamy następnej (≤ 2 przedziały)
- poprzednia pomaga (≤ 2 przedziały)
  - nie zdąży przed limitem – następna musi pomóc (1 przedział)
  - zdążymy – pomagamy następnej (1 przedział)

Podsumowanie strategii

- trzeba pomóc poprzedniej na przedziale
  - nie zdąży przed deadlinem – następna musi pomóc (1 przedział)
  - zdążymy – pomagamy następnej (≤ 2 przedziały)
- poprzednia pomaga (≤ 2 przedziały)
  - nie zdąży przed limitem – następna musi pomóc (1 przedział)
  - zdążymy – pomagamy następnej (1 przedział)

Czas działania

Podsumowanie strategii

- trzeba pomóc poprzedniej na przedziale
  - nie zdąży przed deadlinem – następna musi pomóc (1 przedział)
  - zdążymy – pomagamy następnej (≤ 2 przedziały)
- poprzednia pomaga (≤ 2 przedziały)
  - nie zdąży przed limitem – następna musi pomóc (1 przedział)
  - zdążymy – pomagamy następnej (1 przedział)

Czas działania

$O(n \log maxH)$

amppz

F – Fikające żaby

Najszybsze rozwiązanie:

UWr 1 (2:11)

Autor zadania:

Marcin Smulewicz

F – Fikające żaby

Treść zadania

Dane:

- początkowe pozycje żab – $a_1, a_2, \ldots, a_n$
- długość skoku – k

F – Fikające żaby

Treść zadania

Dane:

- początkowe pozycje żab – $a_1, a_2, \ldots, a_n$
- długość skoku – k

Dla prefiksów $a_1, a_2, \ldots, a_p$.

F – Fikające żaby

Treść zadania

Dane:

- początkowe pozycje żab – $a_1, a_2, \ldots, a_n$
- długość skoku – k

Dla prefiksów $a_1, a_2, \ldots, a_p$. Żaba i gania żabę $i\%p + 1$.

F – Fikające żaby

Treść zadania

Dane:

- początkowe pozycje żab – $a_1, a_2, \ldots, a_n$
- długość skoku – k

Dla prefiksów $a_1, a_2, \ldots, a_p$. Żaba i gania żabę $i\%p + 1$.
Jeden krok, żaby na raz skaczą:

- w lewo (a_i-=k), dla $a_i > a_{i\%p+1}$
- w prawo (a_i+=k), dla $a_i \leq a_{i\%p+1}$

F – Fikające żaby

Treść zadania

Dane:

- początkowe pozycje żab – $a_1, a_2, \ldots, a_n$
- długość skoku – k

Dla prefiksów $a_1, a_2, \ldots, a_p$. Żaba i gania żabę $i\%p + 1$.
Jeden krok, żaby na raz skaczą:

- w lewo (a_i-=k), dla $a_i > a_{i\%p+1}$
- w prawo (a_i+=k), dla $a_i \leq a_{i\%p+1}$

Czy kiedyś wyskoczą poza staw $[-99^{(99^{99})}, 99^{(99^{99})}]$ wykonując kroki?

Definicja

Żaby i, $i+1$ są blisko:

Definicja

Żaby i, $i+1$ są blisko:

$$a_{i+1} - 2 \cdot k < a_i \leq a_{i+1} + 2 \cdot k.$$

Definicja

Żaby i, $i+1$ są blisko:

$$a_{i+1} - 2 \cdot k < a_i \leq a_{i+1} + 2 \cdot k.$$

Lemat 1 – Jeśli $i, i+1$ są blisko to zawsze będą blisko.

Definicja

Żaby i, $i+1$ są blisko:

$$a_{i+1} - 2 \cdot k < a_i \leq a_{i+1} + 2 \cdot k.$$

Lemat 1 – Jeśli $i, i+1$ są blisko to zawsze będą blisko.

- skaczą w tę samą stronę

Definicja

Żaby i, $i+1$ są blisko:

$$a_{i+1} - 2 \cdot k < a_i \leq a_{i+1} + 2 \cdot k.$$

Lemat 1 – Jeśli $i, i+1$ są blisko to zawsze będą blisko.

- skaczą w tę samą stronę – oczywiste

Definicja

Żaby i, $i+1$ są blisko:

$$a_{i+1} - 2 \cdot k < a_i \leq a_{i+1} + 2 \cdot k.$$

Lemat 1 – Jeśli $i, i+1$ są blisko to zawsze będą blisko.

- skaczą w tę samą stronę – oczywiste
- a_i skacze w lewo, a_{i+1} w prawo:

Definicja

Żaby i, $i+1$ są blisko:

$$a_{i+1} - 2 \cdot k < a_i \leq a_{i+1} + 2 \cdot k.$$

Lemat 1 – Jeśli $i, i+1$ są blisko to zawsze będą blisko.

- skaczą w tę samą stronę – oczywiste
- a_i skacze w lewo, a_{i+1} w prawo:

$$a_{i+1} < a_i \leq a_{i+1} + 2 \cdot k$$

Definicja

Żaby i, $i+1$ są blisko:

$$a_{i+1} - 2 \cdot k < a_i \leq a_{i+1} + 2 \cdot k.$$

Lemat 1 – Jeśli $i, i+1$ są blisko to zawsze będą blisko.

- skaczą w tę samą stronę – oczywiste
- a_i skacze w lewo, a_{i+1} w prawo:

$$a_{i+1} < a_i \leq a_{i+1} + 2 \cdot k \Rightarrow (a_{i+1} + k) - 2 \cdot k \leq (a_i - k) < (a_{i+1} + k)$$

Definicja

Żaby i, $i+1$ są blisko:

$$a_{i+1} - 2 \cdot k < a_i \leq a_{i+1} + 2 \cdot k.$$

Lemat 1 – Jeśli $i, i+1$ są blisko to zawsze będą blisko.

- skaczą w tę samą stronę – oczywiste
- a_i skacze w lewo, a_{i+1} w prawo:

$$a_{i+1} < a_i \leq a_{i+1} + 2 \cdot k \Rightarrow (a_{i+1} + k) - 2 \cdot k \leq (a_i - k) < (a_{i+1} + k)$$

Nowe pozycje: $a_{i+1} - 2 \cdot k < a_i \leq a_{i+1}$

Definicja

Żaby i, $i+1$ są blisko:

$$a_{i+1} - 2 \cdot k < a_i \leq a_{i+1} + 2 \cdot k.$$

Lemat 1 – Jeśli $i, i+1$ są blisko to zawsze będą blisko.

- skaczą w tę samą stronę – oczywiste
- a_i skacze w lewo, a_{i+1} w prawo:

$$a_{i+1} < a_i \leq a_{i+1} + 2 \cdot k \Rightarrow (a_{i+1} + k) - 2 \cdot k \leq (a_i - k) < (a_{i+1} + k)$$

  Nowe pozycje: $a_{i+1} - 2 \cdot k < a_i \leq a_{i+1}$
- a_i skacze w prawo, a_{i+1} w lewo:

$$a_{i+1} - 2 \cdot k < a_i \leq a_{i+1}$$

Definicja

Żaby i, $i+1$ są blisko:

$$a_{i+1} - 2 \cdot k < a_i \leq a_{i+1} + 2 \cdot k.$$

Lemat 1 – Jeśli $i, i+1$ są blisko to zawsze będą blisko.

- skaczą w tę samą stronę – oczywiste
- a_i skacze w lewo, a_{i+1} w prawo:

$$a_{i+1} < a_i \leq a_{i+1} + 2 \cdot k \Rightarrow (a_{i+1} + k) - 2 \cdot k \leq (a_i - k) < (a_{i+1} + k)$$

  Nowe pozycje: $a_{i+1} - 2 \cdot k < a_i \leq a_{i+1}$
- a_i skacze w prawo, a_{i+1} w lewo:

$$a_{i+1} - 2 \cdot k < a_i \leq a_{i+1} \Rightarrow (a_{i+1} - k) < (a_i + k) \leq (a_{i+1} - k) + 2 \cdot k$$

Definicja

Żaby i, $i+1$ są blisko:

$$a_{i+1} - 2 \cdot k < a_i \leq a_{i+1} + 2 \cdot k.$$

Lemat 1 – Jeśli $i, i+1$ są blisko to zawsze będą blisko.

- skaczą w tę samą stronę – oczywiste
- a_i skacze w lewo, a_{i+1} w prawo:

$$a_{i+1} < a_i \leq a_{i+1} + 2 \cdot k \Rightarrow (a_{i+1} + k) - 2 \cdot k \leq (a_i - k) < (a_{i+1} + k)$$

  Nowe pozycje: $a_{i+1} - 2 \cdot k < a_i \leq a_{i+1}$
- a_i skacze w prawo, a_{i+1} w lewo:

$$a_{i+1} - 2 \cdot k < a_i \leq a_{i+1} \Rightarrow (a_{i+1} - k) < (a_i + k) \leq (a_{i+1} - k) + 2 \cdot k$$

  Nowe pozycje: $a_{i+1} < a_i \leq a_{i+1} + 2 \cdot k$

Definicja

Żaby i, $i+1$ są blisko:

$$a_{i+1} - 2 \cdot k < a_i \leq a_{i+1} + 2 \cdot k.$$

Lemat 1 – Jeśli $i, i+1$ są blisko to zawsze będą blisko.

- skaczą w tę samą stronę – oczywiste
- a_i skacze w lewo, a_{i+1} w prawo:

$$a_{i+1} < a_i \leq a_{i+1} + 2 \cdot k \Rightarrow (a_{i+1} + k) - 2 \cdot k \leq (a_i - k) < (a_{i+1} + k)$$

  Nowe pozycje: $a_{i+1} - 2 \cdot k < a_i \leq a_{i+1}$ (a_i skoczy w prawo!)
- a_i skacze w prawo, a_{i+1} w lewo:

$$a_{i+1} - 2 \cdot k < a_i \leq a_{i+1} \Rightarrow (a_{i+1} - k) < (a_i + k) \leq (a_{i+1} - k) + 2 \cdot k$$

  Nowe pozycje: $a_{i+1} < a_i \leq a_{i+1} + 2 \cdot k$ (a_i skoczy w lewo!)

Kolejne żaby blisko – rozwiązanie

Kolejne żaby blisko – rozwiązanie

Kierunek skoku i w następnym kroku = Kierunek skoku $i+1$ w poprzednim.

Kolejne żaby blisko – rozwiązanie

Kierunek skoku i w następnym kroku = Kierunek skoku $i+1$ w poprzednim.

Krok1: LLLRLRR

Kolejne żaby blisko – rozwiązanie

Kierunek skoku i w następnym kroku = Kierunek skoku $i+1$ w poprzednim.

Krok1: LLLRLRR

Krok2: LLRLRRL

Kolejne żaby blisko – rozwiązanie

Kierunek skoku i w następnym kroku = Kierunek skoku $i+1$ w poprzednim.

Krok1: LLLRLRR
Krok2: LLRLRRL
Krok3: LRLRRLL
Krok4: RLRRLLL
...

Kolejne żaby blisko – rozwiązanie

Kierunek skoku i w następnym kroku = Kierunek skoku $i+1$ w poprzednim.

Krok1: LLLRLRR

Krok2: LLRLRRL

Krok3: LRLRRLL

Krok4: RLRRLLL

...

Po n krokach – każda żaba każdy skok wykona raz.

Kolejne żaby blisko – rozwiązanie

Kierunek skoku i w następnym kroku = Kierunek skoku $i+1$ w poprzednim.

Krok1: LLLRLRR

Krok2: LLRLRRL

Krok3: LRLRRLL

Krok4: RLRRLLL

...

Po n krokach – każda żaba każdy skok wykona raz.

Przesunięcie o wektor $d := \#R \cdot k - \#L \cdot k$.

Kolejne żaby blisko – rozwiązanie

Kierunek skoku i w następnym kroku = Kierunek skoku $i+1$ w poprzednim.

Krok1: LLLRLRR

Krok2: LLRLRRL

Krok3: LRLRRLL

Krok4: RLRRLLL

...

Po n krokach – każda żaba każdy skok wykona raz.

Przesunięcie o wektor $d := \#R \cdot k - \#L \cdot k$.

- $d = 0$, znowu początkowa pozycja

Kolejne żaby blisko – rozwiązanie

Kierunek skoku i w następnym kroku = Kierunek skoku $i+1$ w poprzednim.

Krok1: LLLRLRR

Krok2: LLRLRRL

Krok3: LRLRRLL

Krok4: RLRRLLL

...

Po n krokach – każda żaba każdy skok wykona raz.

Przesunięcie o wektor $d := \#R \cdot k - \#L \cdot k$.

- $d = 0$, znowu początkowa pozycja – odpowiedź 0.

Kolejne żaby blisko – rozwiązanie

Kierunek skoku i w następnym kroku = Kierunek skoku $i+1$ w poprzednim.

Krok1: LLLRLRR

Krok2: LLRLRRL

Krok3: LRLRRLL

Krok4: RLRRLLL

...

Po n krokach – każda żaba każdy skok wykona raz.

Przesunięcie o wektor $d := \#R \cdot k - \#L \cdot k$.

- $d = 0$, znowu początkowa pozycja – odpowiedź 0.
- Wpp – odpowiedź 1

Kolejne żaby blisko – rozwiązanie

Kierunek skoku i w następnym kroku = Kierunek skoku $i+1$ w poprzednim.

Krok1: LLLRLRR

Krok2: LLRLRRL

Krok3: LRLRRLL

Krok4: RLRRLLL

...

Po n krokach – każda żaba każdy skok wykona raz.

Przesunięcie o wektor $d := \#R \cdot k - \#L \cdot k$.

- $d = 0$, znowu początkowa pozycja – odpowiedź 0.
- Wpp – odpowiedź 1, bo dddk $|d| \cdot k > 99^{(99^{99})}$

i dogoni $i+1$ – kiedyś będą blisko

i dogoni $i+1$ – kiedyś będą blisko

Przypadek $a_{i+1} < a_i$.

i dogoni $i+1$ – kiedyś będą blisko

Przypadek $a_{i+1} < a_i$. Nie wprost.

i dogoni $i+1$ – kiedyś będą blisko

Przypadek $a_{i+1} < a_i$. Nie wprost.
i zawsze skacze w lewo

i dogoni $i+1$ – kiedyś będą blisko

Przypadek $a_{i+1} < a_i$. Nie wprost.
i zawsze skacze w lewo $\Rightarrow$
$i+1$ skończenie wiele razy skoczy w prawo

i dogoni $i+1$ – kiedyś będą blisko

Przypadek $a_{i+1} < a_i$. Nie wprost.
i zawsze skacze w lewo $\Rightarrow$
$i+1$ skończenie wiele razy skoczy w prawo $\Rightarrow$
$i+1$ od pewnego momentu skacze tylko w lewo

i dogoni $i+1$ – kiedyś będą blisko

Przypadek $a_{i+1} < a_i$. Nie wprost.
i zawsze skacze w lewo $\Rightarrow$
$i+1$ skończenie wiele razy skoczy w prawo $\Rightarrow$
$i+1$ od pewnego momentu skacze tylko w lewo $\Rightarrow$
$i+2$ skończenie wiele razy skoczy w prawo

i dogoni $i+1$ – kiedyś będą blisko

Przypadek $a_{i+1} < a_i$. Nie wprost.
i zawsze skacze w lewo $\Rightarrow$
$i+1$ skończenie wiele razy skoczy w prawo $\Rightarrow$
$i+1$ od pewnego momentu skacze tylko w lewo $\Rightarrow$
$i+2$ skończenie wiele razy skoczy w prawo $\Rightarrow$
$\ldots$
$i-1$ od pewnego momentu skacze tylko w lewo

i dogoni $i+1$ – kiedyś będą blisko

Przypadek $a_{i+1} < a_i$. Nie wprost.
i zawsze skacze w lewo $\Rightarrow$
$i+1$ skończenie wiele razy skoczy w prawo $\Rightarrow$
$i+1$ od pewnego momentu skacze tylko w lewo $\Rightarrow$
$i+2$ skończenie wiele razy skoczy w prawo $\Rightarrow$
$\ldots$
$i-1$ od pewnego momentu skacze tylko w lewo $\Rightarrow$
Od pewnego momentu wszyscy zawsze skaczą w lewo

i dogoni $i+1$ – kiedyś będą blisko

Przypadek $a_{i+1} < a_i$. Nie wprost.
i zawsze skacze w lewo $\Rightarrow$
$i+1$ skończenie wiele razy skoczy w prawo $\Rightarrow$
$i+1$ od pewnego momentu skacze tylko w lewo $\Rightarrow$
$i+2$ skończenie wiele razy skoczy w prawo $\Rightarrow$
$\ldots$
$i-1$ od pewnego momentu skacze tylko w lewo $\Rightarrow$
Od pewnego momentu wszyscy zawsze skaczą w lewo $\Rightarrow$ Sprzeczność!

i dogoni $i+1$ – kiedyś będą blisko

Przypadek $a_{i+1} < a_i$. Nie wprost.
i zawsze skacze w lewo $\Rightarrow$
$i+1$ skończenie wiele razy skoczy w prawo $\Rightarrow$
$i+1$ od pewnego momentu skacze tylko w lewo $\Rightarrow$
$i+2$ skończenie wiele razy skoczy w prawo $\Rightarrow$
$\ldots$
$i-1$ od pewnego momentu skacze tylko w lewo $\Rightarrow$
Od pewnego momentu wszyscy zawsze skaczą w lewo $\Rightarrow$ Sprzeczność!
Przypadek $a_i \leq a_{i+1}$, podobnie.

Żaby daleko – rozwiązanie

Żaby daleko – rozwiązanie

- $a_i > a_{i+1} + 2 \cdot k$, czyli a_i skacze w lewo

Żaby daleko – rozwiązanie

- $a_i > a_{i+1} + 2 \cdot k$, czyli a_i skacze w lewo
  - a_{i+1} skacze w prawo, $(a_i - k) > (a_{i+1} + k)$.

Żaby daleko – rozwiązanie

- $a_i > a_{i+1} + 2 \cdot k$, czyli a_i skacze w lewo
  - a_{i+1} skacze w prawo, $(a_i - k) > (a_{i+1} + k)$.
    Nowe pozycje: $a_i > a_{i+1}$

Żaby daleko – rozwiązanie

- $a_i > a_{i+1} + 2 \cdot k$, czyli a_i skacze w lewo
  - a_{i+1} skacze w prawo, $(a_i - k) > (a_{i+1} + k)$.
    Nowe pozycje: $a_i > a_{i+1}$, czyli a_i skoczy w lewo (Zły kierunek)

Żaby daleko – rozwiązanie

- $a_i > a_{i+1} + 2 \cdot k$, czyli a_i skacze w lewo
  - a_{i+1} skacze w prawo, $(a_i - k) > (a_{i+1} + k)$.
    Nowe pozycje: $a_i > a_{i+1}$, czyli a_i skoczy w lewo (Zły kierunek)
    $a_i - a_{i+1}$ zmalało o $2 \cdot k$.

Żaby daleko – rozwiązanie

- $a_i > a_{i+1} + 2 \cdot k$, czyli a_i skacze w lewo
  - a_{i+1} skacze w prawo, $(a_i - k) > (a_{i+1} + k)$.
    Nowe pozycje: $a_i > a_{i+1}$, czyli a_i skoczy w lewo (Zły kierunek)
    $a_i - a_{i+1}$ zmalało o $2 \cdot k$.
  - a_{i+1} skacze w lewo.

Żaby daleko – rozwiązanie

- $a_i > a_{i+1} + 2 \cdot k$, czyli a_i skacze w lewo
  - a_{i+1} skacze w prawo, $(a_i - k) > (a_{i+1} + k)$.
    Nowe pozycje: $a_i > a_{i+1}$, czyli a_i skoczy w lewo (Zły kierunek)
    $a_i - a_{i+1}$ zmalało o $2 \cdot k$.
  - a_{i+1} skacze w lewo.

  Każdym z $\lfloor \frac{(a_{i+1} - a_i - 1)}{2 \cdot k} \rfloor$ złych kierunków

Żaby daleko – rozwiązanie

- $a_i > a_{i+1} + 2 \cdot k$, czyli a_i skacze w lewo
  - a_{i+1} skacze w prawo, $(a_i - k) > (a_{i+1} + k)$.
    Nowe pozycje: $a_i > a_{i+1}$, czyli a_i skoczy w lewo (Zły kierunek)
    $a_i - a_{i+1}$ zmalało o $2 \cdot k$.
  - a_{i+1} skacze w lewo.

  Każdym z $\lfloor \frac{(a_{i+1} - a_i - 1)}{2 \cdot k} \rfloor$ złych kierunków zmniejszy bilans $(\#R - \#L)$ o 2.

Żaby daleko – rozwiązanie

- $a_i > a_{i+1} + 2 \cdot k$, czyli a_i skacze w lewo
  - a_{i+1} skacze w prawo, $(a_i - k) > (a_{i+1} + k)$.
    Nowe pozycje: $a_i > a_{i+1}$, czyli a_i skoczy w lewo (Zły kierunek)
    $a_i - a_{i+1}$ zmalało o $2 \cdot k$.
  - a_{i+1} skacze w lewo.

  Każdym z $\lfloor \frac{(a_{i+1} - a_i - 1)}{2 \cdot k} \rfloor$ złych kierunków zmniejszy bilans $(\#R - \#L)$ o 2.
- $a_{i+1} - 2 \cdot k \leq a_i$, czyli a_i skacze w lewo

Żaby daleko – rozwiązanie

- $a_i > a_{i+1} + 2 \cdot k$, czyli a_i skacze w lewo
  - a_{i+1} skacze w prawo, $(a_i - k) > (a_{i+1} + k)$.
    Nowe pozycje: $a_i > a_{i+1}$, czyli a_i skoczy w lewo (Zły kierunek)
    $a_i - a_{i+1}$ zmalało o $2 \cdot k$.
  - a_{i+1} skacze w lewo.

  Każdym z $\lfloor \frac{(a_{i+1} - a_i - 1)}{2 \cdot k} \rfloor$ złych kierunków zmniejszy bilans $(\#R - \#L)$ o 2.
- $a_{i+1} - 2 \cdot k \leq a_i$, czyli a_i skacze w lewo
  Analogicznie, każdy z $\lfloor \frac{(a_i - a_{i+1})}{2 \cdot k} \rfloor$ złych kierunków zwiększy bilans $(\#R - \#L)$ o 2.

Żaby daleko – rozwiązanie

- $a_i > a_{i+1} + 2 \cdot k$, czyli a_i skacze w lewo
  - a_{i+1} skacze w prawo, $(a_i - k) > (a_{i+1} + k)$.
    Nowe pozycje: $a_i > a_{i+1}$, czyli a_i skoczy w lewo (Zły kierunek)
    $a_i - a_{i+1}$ zmalało o $2 \cdot k$.
  - a_{i+1} skacze w lewo.

  Każdym z $\lfloor \frac{(a_{i+1} - a_i - 1)}{2 \cdot k} \rfloor$ złych kierunków zmniejszy bilans $(\#R - \#L)$ o 2.
- $a_{i+1} - 2 \cdot k \leq a_i$, czyli a_i skacze w lewo
  Analogicznie, każdy z $\lfloor \frac{(a_i - a_{i+1})}{2 \cdot k} \rfloor$ złych kierunków zwiększy bilans $(\#R - \#L)$ o 2.

Implementacja

Żaby daleko – rozwiązanie

- $a_i > a_{i+1} + 2 \cdot k$, czyli a_i skacze w lewo
  - a_{i+1} skacze w prawo, $(a_i - k) > (a_{i+1} + k)$.
    Nowe pozycje: $a_i > a_{i+1}$, czyli a_i skoczy w lewo (Zły kierunek)
    $a_i - a_{i+1}$ zmalało o $2 \cdot k$.
  - a_{i+1} skacze w lewo.

  Każdym z $\lfloor \frac{(a_{i+1} - a_i - 1)}{2 \cdot k} \rfloor$ złych kierunków zmniejszy bilans $(\#R - \#L)$ o 2.
- $a_{i+1} - 2 \cdot k \leq a_i$, czyli a_i skacze w lewo
  Analogicznie, każdy z $\lfloor \frac{(a_i - a_{i+1})}{2 \cdot k} \rfloor$ złych kierunków zwiększy bilans $(\#R - \#L)$ o 2.

Implementacja

Suma prefiksowa bilansów (a_i, a_{i+1}) + bilans (a_p, a_1)

Żaby daleko – rozwiązanie

- $a_i > a_{i+1} + 2 \cdot k$, czyli a_i skacze w lewo
  - a_{i+1} skacze w prawo, $(a_i - k) > (a_{i+1} + k)$.
    Nowe pozycje: $a_i > a_{i+1}$, czyli a_i skoczy w lewo (Zły kierunek)
    $a_i - a_{i+1}$ zmalało o $2 \cdot k$.
  - a_{i+1} skacze w lewo.

  Każdym z $\lfloor \frac{(a_{i+1} - a_i - 1)}{2 \cdot k} \rfloor$ złych kierunków zmniejszy bilans $(\#R - \#L)$ o 2.
- $a_{i+1} - 2 \cdot k \leq a_i$, czyli a_i skacze w lewo
  Analogicznie, każdy z $\lfloor \frac{(a_i - a_{i+1})}{2 \cdot k} \rfloor$ złych kierunków zwiększy bilans $(\#R - \#L)$ o 2.

Implementacja

Suma prefiksowa bilansów (a_i, a_{i+1}) + bilans (a_p, a_1)
Czas $O(n)$

amppz

E – Ekspresowe rotacje

Najszybsze rozwiązanie:

UW11 (2:33)

Autor zadania:

Jakub Onufry Wojtaszczyk

E – Ekspresowe rotacje

Treść zadania

Dany jest ciąg liczb a_i: 6 10 6 5 4 5.
Operacje:

- Przestawienie liczby a_1 z początku na koniec: koszt a_1.
- Przestawienie liczby a_n z końca na początek: koszt a_n.
- Usunięcie największej liczby a_1 z początku: koszt 0.

Cel: usunąć wszystkie liczby minimalnym kosztem.

E – Ekspresowe rotacje

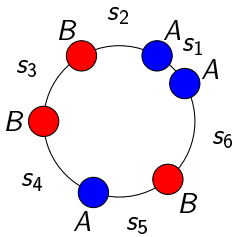
Programowanie dynamiczne

- k_i: koszt usunięcia wszystkich liczb $\geq a_i$, kończąc na pozycji i.
- Dodajmy dwie liczby dla uproszczenia: start $a_0 = \infty$, meta $a_1 = 0$.
- ∞ 0 6 10 6 5 4 5
- Obliczymy k_i od największych a_i.
- $k_0 = 0$.
- Wynik końcowy: k_1.

E – Ekspresowe rotacje

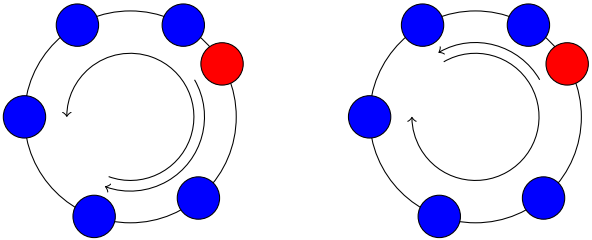
Faza obliczeń

- Obliczyliśmy k_i dla wszystkich $a_i = B$.
- A = kolejna wartość, $A < B$.
- Chcemy obliczyć k_i dla wszystkich $a_i = A$ (niebieskie wierzchołki).
- s_i = suma elementów mniejszych od A na przedziałach pomiędzy $a_i \in \{A, B\}$
- s_i obliczamy w $O(\log n)$ na drzewie potęgowym



Dwa sposoby na dojście z B do A

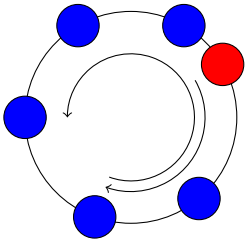
- W tył do następnika A, w przód do A.
- W przód do poprzedniego A, w tył do A.



E – Ekspresowe rotacje

Najpierw w tył, potem w przód

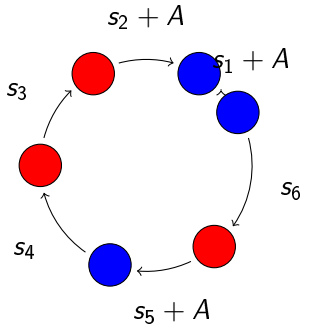
- Koszt przejścia w tył do elementu B: s_i.
- Koszt przejścia w tył do elementu A: $s_i + A$.
- Koszt przejścia w przód: s_i.



E – Ekspresowe rotacje

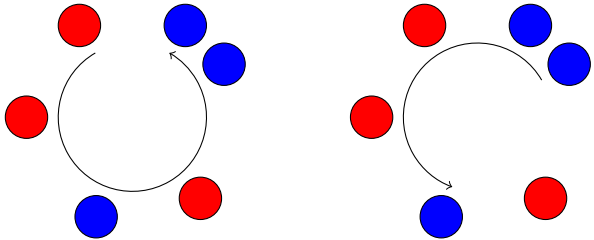
Tył–przód: najpierw przejścia w tył

- Najkrótsze ścieżki, koszt krawędzi s_i lub $s_i + A$.
- DAG jeśli usuniemy krawędź w tył od najmniejszego kosztu k_i.



Tył–przód: potem przejścia w przód

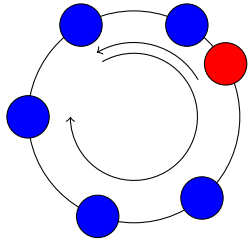
- Koszt: $\sum s_i$.
- Każde takie przejście w przód rozważamy w $O(1)$.



E – Ekspresowe rotacje

Najpierw w przód, potem w tył

- Koszt przejścia w przód: s_i.
- Koszt przejścia w tył do A jeśli usunęliśmy idąc w przód: s_i.
- Koszt przejścia w tył do A jeśli nie usunęliśmy idąc w przód: $s_i + A$.
- Koszt przejścia w tył do B: s_i.



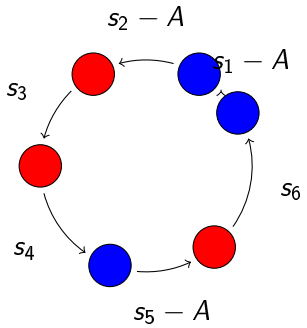
Najpierw w przód, potem w tył

Uproszczenie:

- Koszt przejścia w przód z B: s_i.
- Koszt przejścia w przód z A: $s_i - A$.
- Koszt przejścia w tył do A: $s_i + A$.
- Koszt przejścia w tył do B: s_i.

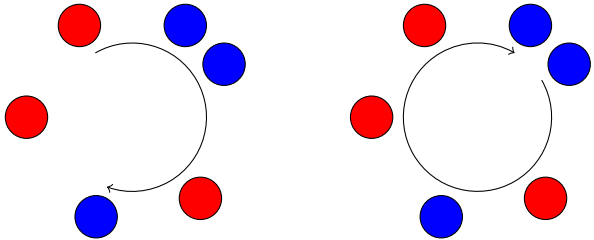
Przód–tył: przejścia w przód

- Koszt krawędzi s_i lub $s_i - A$.



Przód–tył: potem przejścia w tył

- Koszt: $\sum s_i$ plus $n_A A$ lub $(n_A - 1)A$.
- Każde takie przejście w tył rozważamy w $O(1)$.



Problem: ujemne krawędzie

- Co jeśli $s_i - A < 0$? Ujemne krawędzie przy przejściach w przód.
- Możliwy ujemny cykl!
- Nie chcemy przejść w przód po całym kółku, nawet jeśli koszt cyklu jest ujemny.

Rozwiązanie problemu ujemnych cykli

- Dla każdego wierzchołka liczymy cały ciąg możliwych kosztów oraz liczby użytych krawędzi.
- $(d_1, l_1), (d_2, l_2), \ldots, (d_m, l_m)$
- $d_1 < d_2 < \ldots < d_m$
- $n_{i-1} + n_i > l_1 > l_2 > \ldots > l_m$
- Kosztem dojścia do wierzchołka jest najlepszy koszt z listy: d_1

E – Ekspresowe rotacje

Relaksacja krawędzi o koszcie s

- s może być ujemne.
- Wszystkie koszty dla poprzedniego wierzchołka zwiększamy o s, liczbę krawędzi zwiększamy o 1: $(d_1 + s, l_1 + 1), (d_2 + s, l_2 + 1), \ldots, (d_m + s, l_m + 1)$
- Powyższe robimy w $O(1)$ (trzymamy offset d i l dla całej listy).
- Usuwamy krawędzie z początku listy jeśli $l_i \geq n_{i-1} + n_i$.
- Jeśli $a_j = B$ (czerwony wierzchołek), to:
  - Usuwamy krawędzie z końca listy jeśli koszt gorszy niż mamy z poprzedniej fazy: $d_i \geq k_j$.
  - Dodajemy koszt z poprzedniej fazy $(k_j, 0)$ na koniec listy.

E – Ekspresowe rotacje

Najkrótsze ścieżki z ujemnymi krawędziami

- Zaczynamy od dowolnego wierzchołka.
- Relaksujemy kolejne krawędzie, pamiętając listę możliwości o różnych długościach.
- Przechodzimy cały cykl 2 razy!
- W ten sposób rozważymy ścieżki z każdego do każdego wierzchołka, unikając ścieżek przechodzących po pełnym cyklu.

E – Ekspresowe rotacje

Złożoność

- Faza i: $O((n_{i-1} + n_i) \log n)$.
- Razem: $O(n \log n)$.