Omówienie zadań

Mistrzostwa Polski Szkół Średnich
w Programowaniu Zespołowym 2025

8 czerwca 2025

Zadanie L – Zestawy do nurkowania

Zadanie L – Zestawy do nurkowania

Zaakceptowanych zgłoszeń: 55

Zadanie L – Zestawy do nurkowania

Problem

Dane jest odpowiednio M, P i B przedmiotów trzech typów. Powiedzieć ile maksymalnie w najgorszym przypadku pełnych zestawów można utworzyć, jeżeli K pewnych rzeczy jest niedostępnych.

Zadanie L – Zestawy do nurkowania

Problem

Dane jest odpowiednio M, P i B przedmiotów trzech typów. Powiedzieć ile minimalnie pełnych zestawów można utworzyć, jeżeli K pewnych rzeczy jest niedostępnych.

- W najgorszym przypadku wszystkie zamówienia są na ten sam element, w szczególności na ten, którego jest najmniej.
- Odpowiedzią jest więc najmniejszy element odjąć K, a jeżeli jest to ujemne, to 0.

Zadanie A – Aqua racer

Zadanie A – Aqua racer

Zaakceptowanych zgłoszeń: 54

Zadanie A – Aqua Racer

Problem

Dane są trzy niekoniecznie różne liczby naturalne. Wypisać wyrażenie arytmetyczne składające się z dwóch liczb naturalnych i operacji +, -, lub *, którego wynikiem jest dokładnie jedna z tych liczb (albo stwierdzić, że takie nie istnieje).

Zadanie A – Aqua Racer

Jeśli dane są trzy takie same liczby, to nie mamy sposobu, by je odróżnić. W przeciwnym przypadku istnieje taka liczba x, która pojawia się na wejściu dokładnie raz. Można wypisać $(x + 1) - 1$.

Zadanie K – Krzyżyki

Zadanie K – Krzyżyki

Zaakceptowanych zgłoszeń: 50

Zadanie K – Krzyżyki

Problem

Dana jest następująca gra na planszy 3×3. Gracze na zmianę stawiają zajmują pola krzyżykami. Ten kto zamknie któryś wiersz, kolumnę lub przekątną, wygrywa. Należy znaleźć strategię wygrywającą dla gracza rozpoczynającego.

Zadanie K – Krzyżyki

Problem

Dana jest następująca gra na planszy 3×3. Gracze na zmianę stawiają zajmują pola krzyżykami. Ten kto zamknie któryś wiersz, kolumnę lub przekątną, wygrywa. Należy znaleźć strategię wygrywającą dla gracza rozpoczynającego.

Gra jest na tyle prosta, że można ręcznie rozpisać wszystkie możliwości albo kazać je policzyć komputerowi.

Zadanie K – Krzyżyki

Problem

Dana jest następująca gra na planszy 3×3. Gracze na zmianę stawiają zajmują pola krzyżykami. Ten kto zamknie któryś wiersz, kolumnę lub przekątną, wygrywa. Należy znaleźć strategię wygrywającą dla gracza rozpoczynającego.

Gra jest na tyle prosta, że można ręcznie rozpisać wszystkie możliwości albo kazać je policzyć komputerowi.
Przeanalizujmy najprostszy możliwy pierwszy ruch - zagranie w środek.

Zadanie K – Krzyżyki

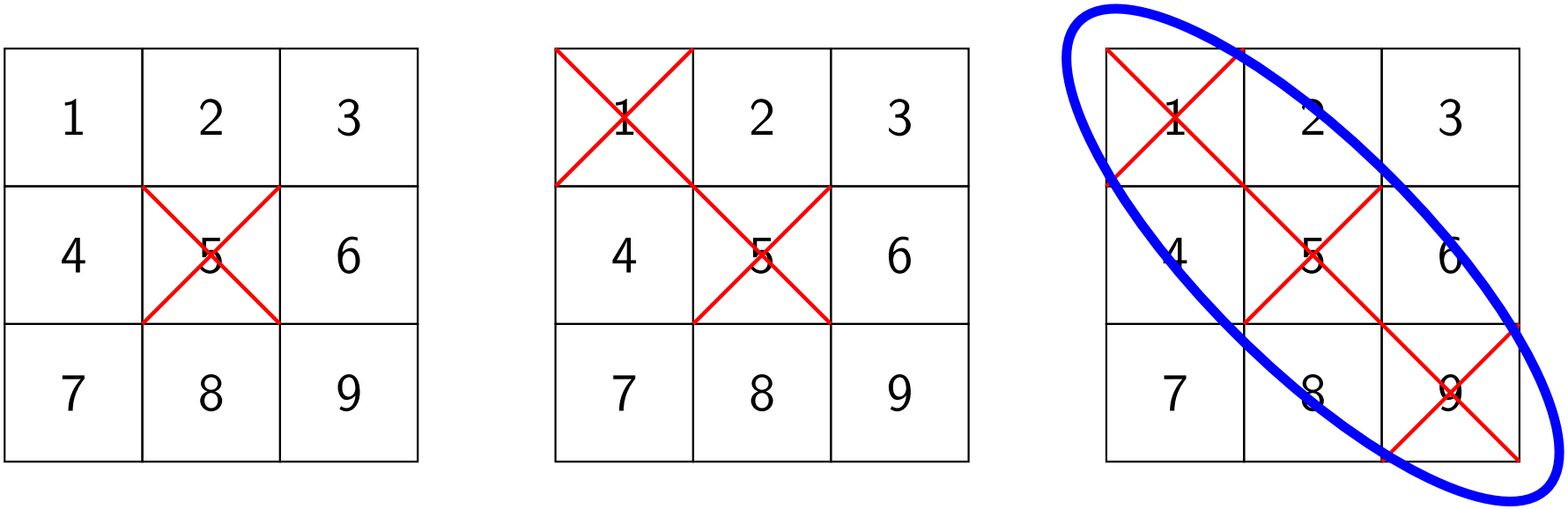
Ruch w środku

Możemy wtedy zawsze wygrać w drugim ruchu, odbijając ruch przeciwnika symetrycznie względem środka.

Zadanie K – Krzyżyki

Ruch w środku

Możemy wtedy zawsze wygrać w drugim ruchu, odbijając ruch przeciwnika symetrycznie względem środka.



Inne strategie

Podobnie jak w zwykłej grze w kółko i krzyżyk, można zrobić dowolny pierwszy ruch a potem dopasować do niego strategię.

Zadanie K – Krzyżyki

```
int num_of_games;
cin >> num_of_games;

for(int game = 1; game <= num_of_games; game++){
    cout << 5 << endl;
    int move;
    cin >> move;
    cout << 10 - move << endl;
}
```

Zadanie O – Powrót Atlantydy

Zadanie O – Powrót Atlantydy

Zaakceptowanych zgłoszeń: 46

Zadanie O – Powrót Atlantydy

Problem

Dana jest liczba N cyfrowa wyświetlana na wyświetlaczu siedmiosegmentowym. Określić jaką największą liczbę może on wyświetlać, jeżeli możemy zmienić stan co najwyżej K segmentów.

Zadanie O – Powrót Atlantydy

Problem

Dana jest liczba N cyfrowa wyświetlana na wyświetlaczu siedmiosegmentowym. Określić jaką największą liczbę może on wyświetlać, jeżeli możemy zmienić stan co najwyżej K segmentów.

- Przy porównywaniu która liczba jest większa, wpierw patrzymy na jej długość. Chcemy więc, by liczba wyświetlana była jak najdłuższa.
- Wydłużać liczbę z wejścia będziemy chcieli jak najmniejszym kosztem. Cyfra, która ma najmniej zapalonych segmentów to 1.
- Dokładamy więc na początek jak najwięcej jedynek, a jeżeli K jest nieparzyste, to możemy pierwszą 1 zamienić na 7.

Zadanie M – Kraby

Zadanie M – Kraby

Zaakceptowanych zgłoszeń: 42

Zadanie M – Kraby

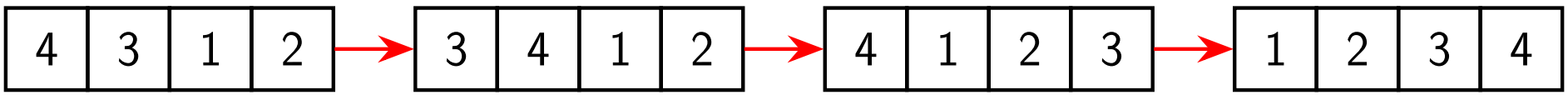
Problem

Dany jest proces, w którym przestawiamy pierwszy element permutacji na prawo od elementu o jeden od niego mniejszego, dopóki pierwszym elementem tej permutacji nie będzie wartość 1. Dla początkowej permutacji należy stwierdzić, czy powyższy proces kiedykolwiek się zakończy, a jeśli tak — podać permutację utworzoną w wyniku tego procesu.

Zadanie M – Kraby

Problem

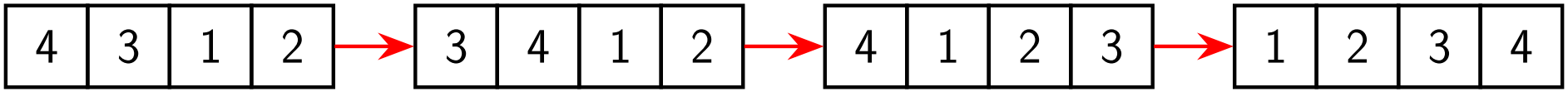
Dany jest proces, w którym przestawiamy pierwszy element permutacji na prawo od elementu o jeden od niego mniejszego, dopóki pierwszym elementem tej permutacji nie będzie wartość 1. Dla początkowej permutacji należy stwierdzić, czy powyższy proces kiedykolwiek się zakończy, a jeśli tak — podać permutację utworzoną w wyniku tego procesu.



Zadanie M – Kraby

Problem

Dany jest proces, w którym przestawiamy pierwszy element permutacji na prawo od elementu o jeden od niego mniejszego, dopóki pierwszym elementem tej permutacji nie będzie wartość 1. Dla początkowej permutacji należy stwierdzić, czy powyższy proces kiedykolwiek się zakończy, a jeśli tak — podać permutację utworzoną w wyniku tego procesu.



Obserwacja

Proces się zawsze zakończy.

Zadanie M – Kraby

Obserwacja

Proces się zawsze zakończy.

Zadanie M – Kraby

Obserwacja

Proces się zawsze zakończy.

Dowód

Pokolorujmy liczby w taki sposób, aby dwie sąsiednie liczby miały ten sam kolor wtedy, i tylko wtedy, gdy prawa jest o 1 większa od lewej. W ten sposób otrzymamy spójne jednokolorowe bloki liczb. Załóżmy, że skrajny lewy blok ma długość L. Wtedy po L krokach procesu, cały blok zostanie przeniesiony na prawo od pewnego innego bloku. Gdy pokolorujemy liczby w ten sam sposób co powyżej, to zauważymy, że przeniesiony blok został złączony z blokiem na lewo od niego, zatem mamy o co najmniej 1 blok mniej. Schematu tego nie możemy powtarzać bez końca, bo z każdą iteracją jest co raz mniej bloków.

Zadanie M – Kraby

Obserwacja

Proces się zawsze zakończy.

Dowód

Pokolorujmy liczby w taki sposób, aby dwie sąsiednie liczby miały ten sam kolor wtedy, i tylko wtedy, gdy prawa jest o 1 większa od lewej. W ten sposób otrzymamy spójne jednokolorowe bloki liczb. Załóżmy, że skrajny lewy blok ma długość L. Wtedy po L krokach procesu, cały blok zostanie przeniesiony na prawo od pewnego innego bloku. Gdy pokolorujemy liczby w ten sam sposób co powyżej, to zauważymy, że przeniesiony blok został złączony z blokiem na lewo od niego, zatem mamy o co najmniej 1 blok mniej. Schematu tego nie możemy powtarzać bez końca, bo z każdą iteracją jest co raz mniej bloków.

3	4	5	7	1	2	8	6

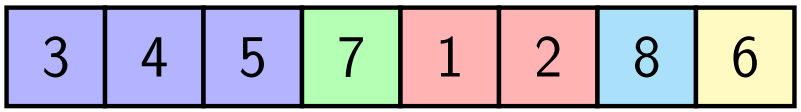
Zadanie M – Kraby

Obserwacja

Proces się zawsze zakończy.

Dowód

Pokolorujmy liczby w taki sposób, aby dwie sąsiednie liczby miały ten sam kolor wtedy, i tylko wtedy, gdy prawa jest o 1 większa od lewej. W ten sposób otrzymamy spójne jednokolorowe bloki liczb. Załóżmy, że skrajny lewy blok ma długość L. Wtedy po L krokach procesu, cały blok zostanie przeniesiony na prawo od pewnego innego bloku. Gdy pokolorujemy liczby w ten sam sposób co powyżej, to zauważymy, że przeniesiony blok został złączony z blokiem na lewo od niego, zatem mamy o co najmniej 1 blok mniej. Schematu tego nie możemy powtarzać bez końca, bo z każdą iteracją jest co raz mniej bloków.



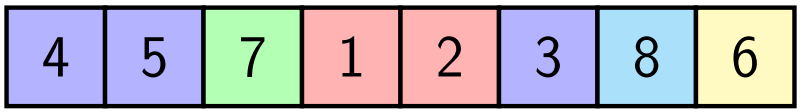
Zadanie M – Kraby

Obserwacja

Proces się zawsze zakończy.

Dowód

Pokolorujmy liczby w taki sposób, aby dwie sąsiednie liczby miały ten sam kolor wtedy, i tylko wtedy, gdy prawa jest o 1 większa od lewej. W ten sposób otrzymamy spójne jednokolorowe bloki liczb. Załóżmy, że skrajny lewy blok ma długość L. Wtedy po L krokach procesu, cały blok zostanie przeniesiony na prawo od pewnego innego bloku. Gdy pokolorujemy liczby w ten sam sposób co powyżej, to zauważymy, że przeniesiony blok został złączony z blokiem na lewo od niego, zatem mamy o co najmniej 1 blok mniej. Schematu tego nie możemy powtarzać bez końca, bo z każdą iteracją jest co raz mniej bloków.



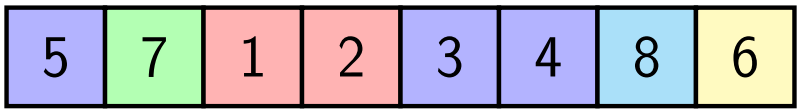
Zadanie M – Kraby

Obserwacja

Proces się zawsze zakończy.

Dowód

Pokolorujmy liczby w taki sposób, aby dwie sąsiednie liczby miały ten sam kolor wtedy, i tylko wtedy, gdy prawa jest o 1 większa od lewej. W ten sposób otrzymamy spójne jednokolorowe bloki liczb. Załóżmy, że skrajny lewy blok ma długość L. Wtedy po L krokach procesu, cały blok zostanie przeniesiony na prawo od pewnego innego bloku. Gdy pokolorujemy liczby w ten sam sposób co powyżej, to zauważymy, że przeniesiony blok został złączony z blokiem na lewo od niego, zatem mamy o co najmniej 1 blok mniej. Schematu tego nie możemy powtarzać bez końca, bo z każdą iteracją jest co raz mniej bloków.



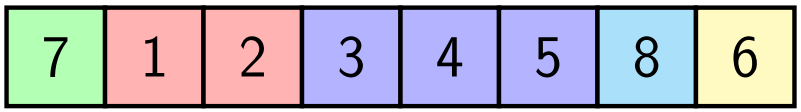
Zadanie M – Kraby

Obserwacja

Proces się zawsze zakończy.

Dowód

Pokolorujmy liczby w taki sposób, aby dwie sąsiednie liczby miały ten sam kolor wtedy, i tylko wtedy, gdy prawa jest o 1 większa od lewej. W ten sposób otrzymamy spójne jednokolorowe bloki liczb. Załóżmy, że skrajny lewy blok ma długość L. Wtedy po L krokach procesu, cały blok zostanie przeniesiony na prawo od pewnego innego bloku. Gdy pokolorujemy liczby w ten sam sposób co powyżej, to zauważymy, że przeniesiony blok został złączony z blokiem na lewo od niego, zatem mamy o co najmniej 1 blok mniej. Schematu tego nie możemy powtarzać bez końca, bo z każdą iteracją jest co raz mniej bloków.



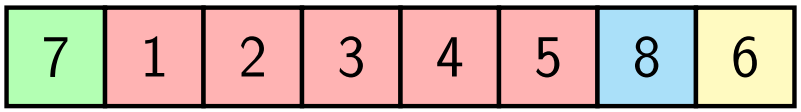
Zadanie M – Kraby

Obserwacja

Proces się zawsze zakończy.

Dowód

Pokolorujmy liczby w taki sposób, aby dwie sąsiednie liczby miały ten sam kolor wtedy, i tylko wtedy, gdy prawa jest o 1 większa od lewej. W ten sposób otrzymamy spójne jednokolorowe bloki liczb. Załóżmy, że skrajny lewy blok ma długość L. Wtedy po L krokach procesu, cały blok zostanie przeniesiony na prawo od pewnego innego bloku. Gdy pokolorujemy liczby w ten sam sposób co powyżej, to zauważymy, że przeniesiony blok został złączony z blokiem na lewo od niego, zatem mamy o co najmniej 1 blok mniej. Schematu tego nie możemy powtarzać bez końca, bo z każdą iteracją jest co raz mniej bloków.



Zadanie M – Kraby

Zatem proces się zawsze kończy, ale jak otrzymać końcową permutację?

Zadanie M – Kraby

Zatem proces się zawsze kończy, ale jak otrzymać końcową permutację?

Bezpośrednia symulacja wymaga w najgorszym przypadku $O(N^2)$ kroków (Zbyt wolno!).

Zadanie M – Kraby

Zatem proces się zawsze kończy, ale jak otrzymać końcową permutację?

Obserwacja

Jeżeli liczba X została co najmniej raz przeniesiona to:

- $X-1$ znajduje się bezpośrednio na lewo od niej lub
- $X-1$ zostało przeniesione w poprzednim kroku, więc X jest pierwszym elementem permutacji i zostanie ponownie przeniesione na prawo od $X-1$ w następnym kroku.

Zadanie M – Kraby

Zatem proces się zawsze kończy, ale jak otrzymać końcową permutację?

Obserwacja

Jeżeli liczba X została co najmniej raz przeniesiona to:

- $X-1$ znajduje się bezpośrednio na lewo od niej lub
- $X-1$ zostało przeniesione w poprzednim kroku, więc X jest pierwszym elementem permutacji i zostanie ponownie przeniesione na prawo od $X-1$ w następnym kroku.

Skoro po zakończonym procesie pierwszym elementem permutacji jest 1, to jeżeli X zostało chociaż raz przeniesione, to będzie ono stało na prawo od $X-1$. Wiemy zatem gdzie X będzie się znajdował w końcowej permutacji, więc nie ma on już dla nas większego znaczenia.

Zadanie M – Kraby

Zatem proces się zawsze kończy, ale jak otrzymać końcową permutację?

Obserwacja

Jeżeli liczba X została co najmniej raz przeniesiona to:

- $X-1$ znajduje się bezpośrednio na lewo od niej lub
- $X-1$ zostało przeniesione w poprzednim kroku, więc X jest pierwszym elementem permutacji i zostanie ponownie przeniesione na prawo od $X-1$ w następnym kroku.

Skoro po zakończonym procesie pierwszym elementem permutacji jest 1, to jeżeli X zostało chociaż raz przeniesione, to będzie ono stało na prawo od $X-1$. Wiemy zatem gdzie X będzie się znajdował w końcowej permutacji, więc nie ma on już dla nas większego znaczenia.

Rozwiązanie

Ignorowanie wszystkich elementów, które zostały już chociaż raz przeniesione.

Zadanie M – Kraby

Rozwiązanie

Ignorowanie wszystkich elementów, które zostały już chociaż raz przeniesione.

Przeiterujmy się po permutacji od lewej do prawej przekreślając elementy które zostaną chociaż raz przeniesione, do momentu gdy nie natrafimy na jedynkę.

Zadanie M – Kraby

Rozwiązanie

Ignorowanie wszystkich elementów, które zostały już chociaż raz przeniesione.

Przeiterujmy się po permutacji od lewej do prawej przekreślając elementy które zostaną chociaż raz przeniesione, do momentu gdy nie natrafimy na jedynkę.

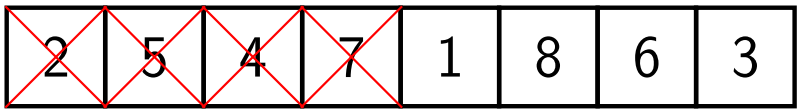
2	5	4	7	1	8	6	3

Zadanie M – Kraby

Rozwiązanie

Ignorowanie wszystkich elementów, które zostały już chociaż raz przeniesione.

Przeiterujmy się po permutacji od lewej do prawej przekreślając elementy które zostaną chociaż raz przeniesione, do momentu gdy nie natrafimy na jedynkę.



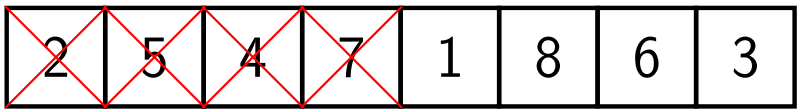
Gdzie należy przełożyć przekreślone liczby?

Zadanie M – Kraby

Rozwiązanie

Ignorowanie wszystkich elementów, które zostały już chociaż raz przeniesione.

Przeiterujmy się po permutacji od lewej do prawej przekreślając elementy które zostaną chociaż raz przeniesione, do momentu gdy nie natrafimy na jedynkę.



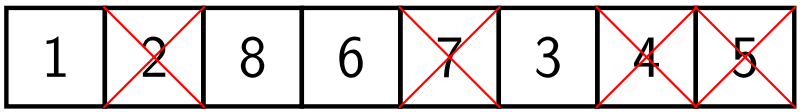
Gdzie należy przełożyć przekreślone liczby?
Zgodnie z wcześniejszą obserwacją, co najmniej raz przeniesiona liczba będzie się znajdować na prawo od liczby od siebie o jeden mniejszej.

Zadanie M – Kraby

Rozwiązanie

Ignorowanie wszystkich elementów, które zostały już chociaż raz przeniesione.

Przeiterujmy się po permutacji od lewej do prawej przekreślając elementy które zostaną chociaż raz przeniesione, do momentu gdy nie natrafimy na jedynkę.

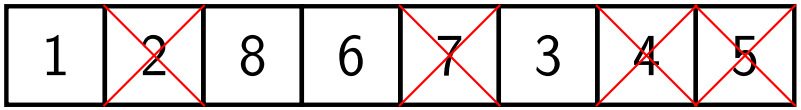


Gdzie należy przełożyć przekreślone liczby?
Zgodnie z wcześniejszą obserwacją, co najmniej raz przeniesiona liczba będzie się znajdować na prawo od liczby od siebie o jeden mniejszej.

Zadanie M – Kraby

Rozwiązanie

Ignorowanie wszystkich elementów, które zostały już chociaż raz przeniesione.

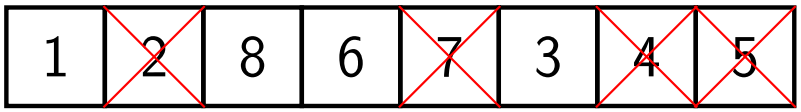


A jak to zaimplementować?

Zadanie M – Kraby

Rozwiązanie

Ignorowanie wszystkich elementów, które zostały już chociaż raz przeniesione.

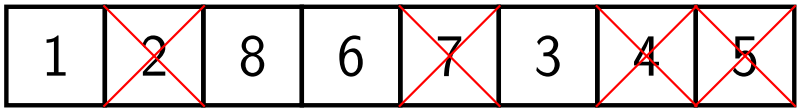


A jak to zaimplementować?
Można na przykład, dla każdej kolejnej nieprzekreślonej liczby od lewej do prawej, wypisać ją oraz kolejne większe od niej liczby, które są przekreślone.

Zadanie M – Kraby

Rozwiązanie

Ignorowanie wszystkich elementów, które zostały już chociaż raz przeniesione.



A jak to zaimplementować?
Można na przykład, dla każdej kolejnej nieprzekreślonej liczby od lewej do prawej, wypisać ją oraz kolejne większe od niej liczby, które są przekreślone.

Złożoność

Czas działania: $O(N)$

Zadanie N – Rozmowy wielorybów

Zadanie N – Rozmowy wielorybów

Zaakceptowanych zgłoszeń: 39

Zadanie N – Rozmowy wielorybów

Problem

W każdym z zapytań dany jest ciąg liczb. Określić czy dla każdych i, j różnych od siebie istnieje w ciągu liczba $a_i \cdot a_j$.

Zadanie N – Rozmowy wielorybów

Obserwacja

Jeśli odpowiedź dla danego ciągu brzmi *TAK*, to w ciągu może znajdować się co najwyżej jedna liczba, której wartość bezwzględna jest większa niż 1.

Dowód

Gdyby w ciągu występowały przynajmniej dwie takie liczby, wybierzmy dwie o największych wartościach bezwzględnych. Ich iloczyn będzie miał wartość bezwzględną większą niż którakolwiek z liczb w ciągu, zatem nie ma go w ciągu.

Zatem w ciągu będą maksymalnie cztery różne wartości. Możemy więc go zredukować, tak aby każda z tych wartości pojawiła się najwyżej dwa razy. W tak przekształconym ciągu sprawdzamy naiwnie, czy iloczyn każdych dwóch jego elementów w nim występuje.

Zadanie C – Kąpiel

Zadanie C – Kąpiel

Zaakceptowanych zgłoszeń: 33

Zadanie C – Kąpiel

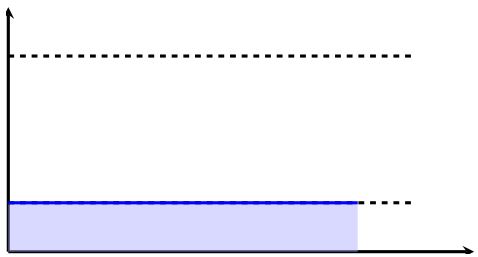
Problem

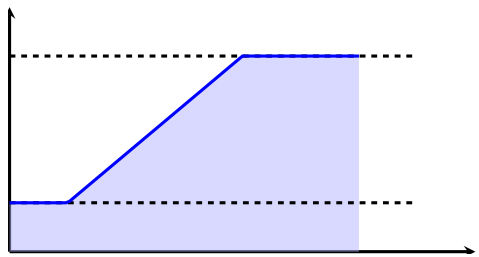
Dane są kran, wanna oraz boiler.
Zadanie polega na wlaniu do wanny co najmniej v litrów wody o temperaturze dokładnie t.
Z kranu początkowo leci woda o temperaturze z. Po włączeniu boilera, woda w kranie zaczyna się stopniowo nagrzewać, aż do osiągnięcia temperatury c, po czym zostaje stała. Po wyłączeniu boilera, temperatura wody w kranie stopniowo spada, aż do osiągnięcia temperatury z.

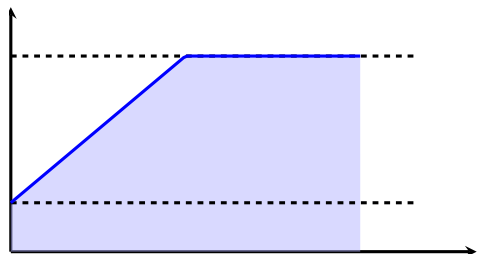
Zadanie C – Kąpiel

Najmniejsza temperatura jaką jesteśmy w stanie uzyskać to z. Co więcej, utrzymując włączony boiler jesteśmy w stanie uzyskać każdą temperaturę mniejszą niż c.

Policzmy jaka jest maksymalna temperatura, którą możemy otrzymać po czasie x.
Włączając boiler odpowiednią chwilę później, będziemy w stanie otrzymać każdą temperaturę niższą od niej (ale nie mniejszą niż z), również w czasie x.



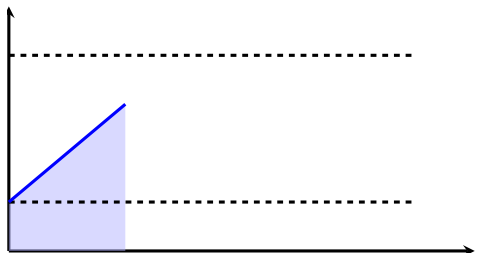


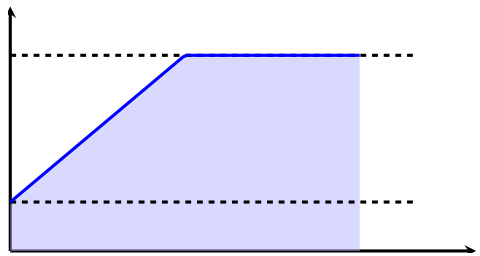


Zadanie C – Kąpiel

Jaka będzie temperatura wody, gdy będziemy ją lać x sekund?

- $t = z + \frac{x}{2}$,
  gdy $x \leqslant c - z$,
- $t = \frac{r^2/2 + (x-r)\cdot r}{x} + z$,
  gdzie $r = c - z$, gdy $x > c - z$.





Aby rozwiazać zadanie, wystarczy znaleźć odpowiednie x za pomocą wyszukiwania binarnego, bądź wyznaczając go z powyższego wzoru.

Zadanie J – Szyfr Atlantydów

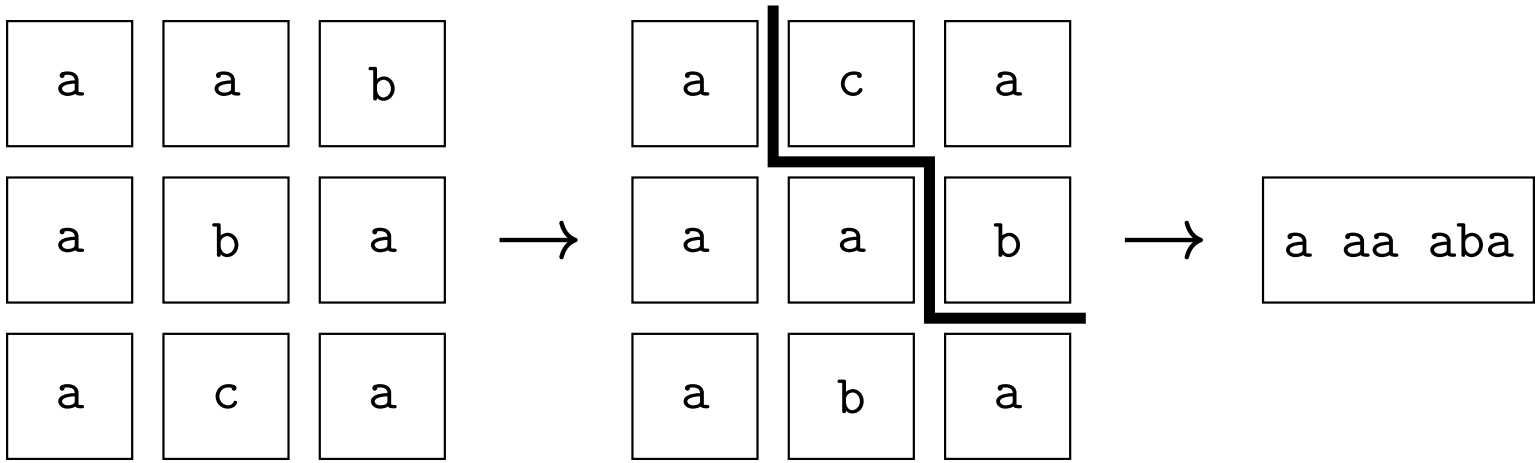
Zadanie J – Szyfr Atlantydów

Zaakceptowanych zgłoszeń: 28

Zadanie J – Szyfr Atlantydów

Problem

Danych N słow długości N. Ustawić je tak, aby konkatenacja i pierwszych liter z i-tego słowa była leksykograficznie najmniejsza.



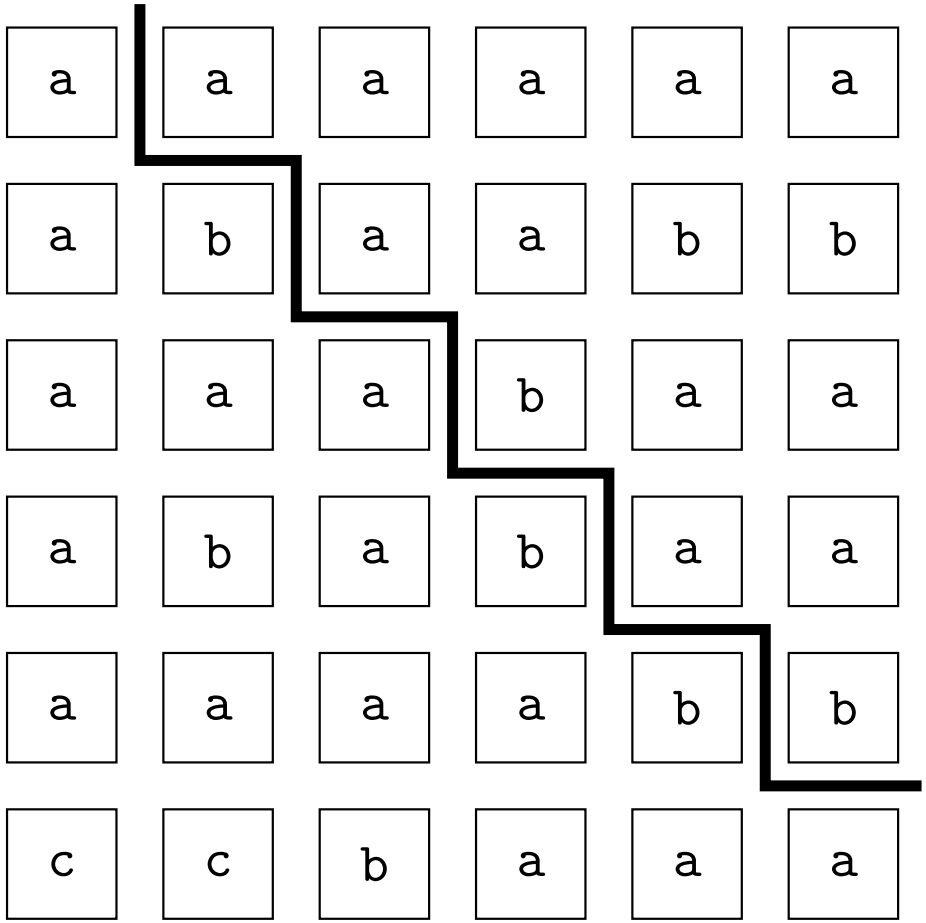
Zadanie J – Szyfr Atlantydów

Problem

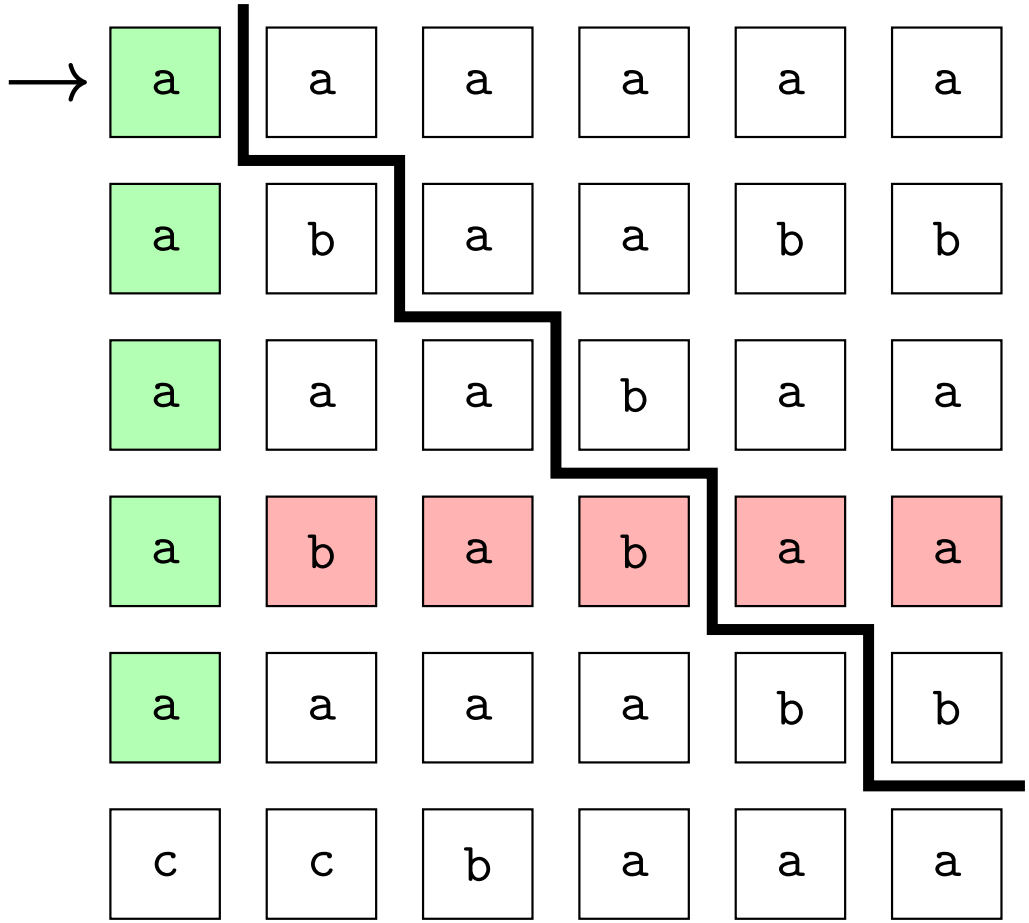
Danych N słow długości N. Ustawić je tak, aby konkatenacja i pierwszych liter z i-tego słowa była leksykograficznie najmniejsza.

- Podejście zachłanne – rozwiążmy zadanie wiersz po wierszu.
- Na i-ty wiersz wybieramy wolne słowo, które posiada leksykograficznie najmniejszy prefix i pierwszych znaków.
- Takich słów może być wiele, więc wybierzmy z nich to, które jest najwieksze leksykograficznie na sufiksie $(N - i)$ znaków.

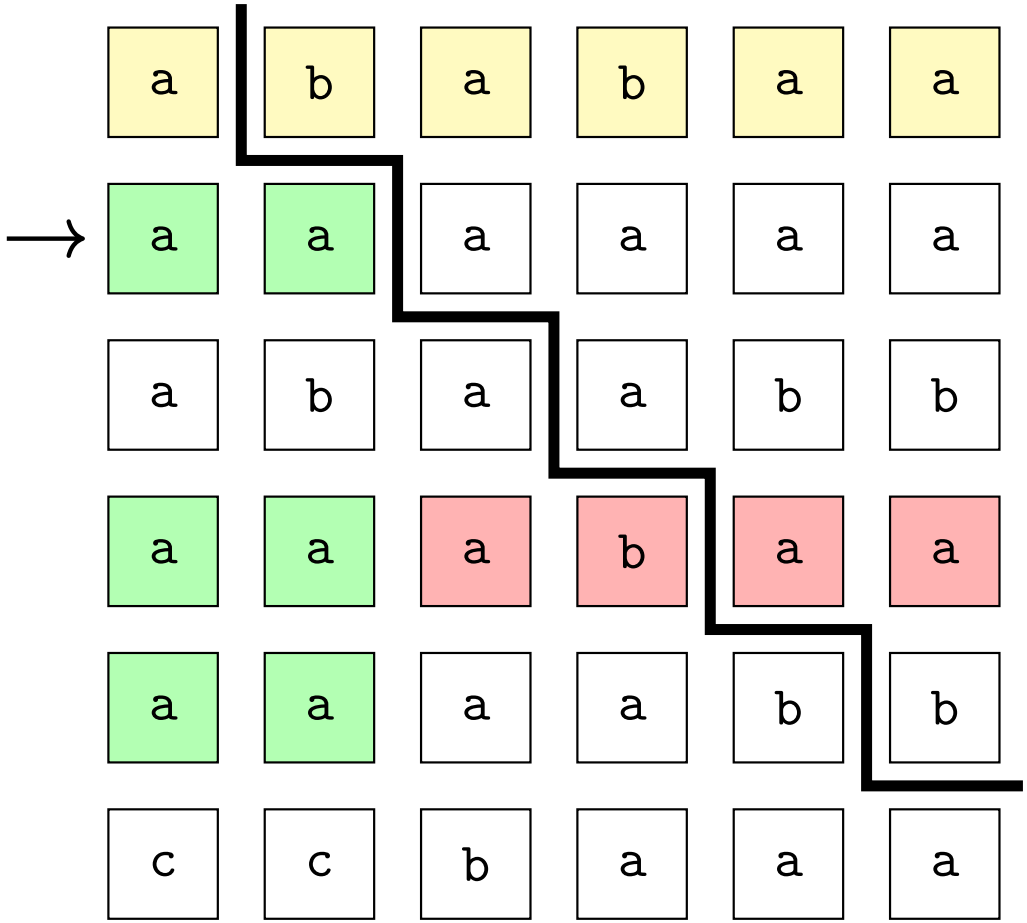
Zadanie J – Szyfr Atlantydów – Przykład



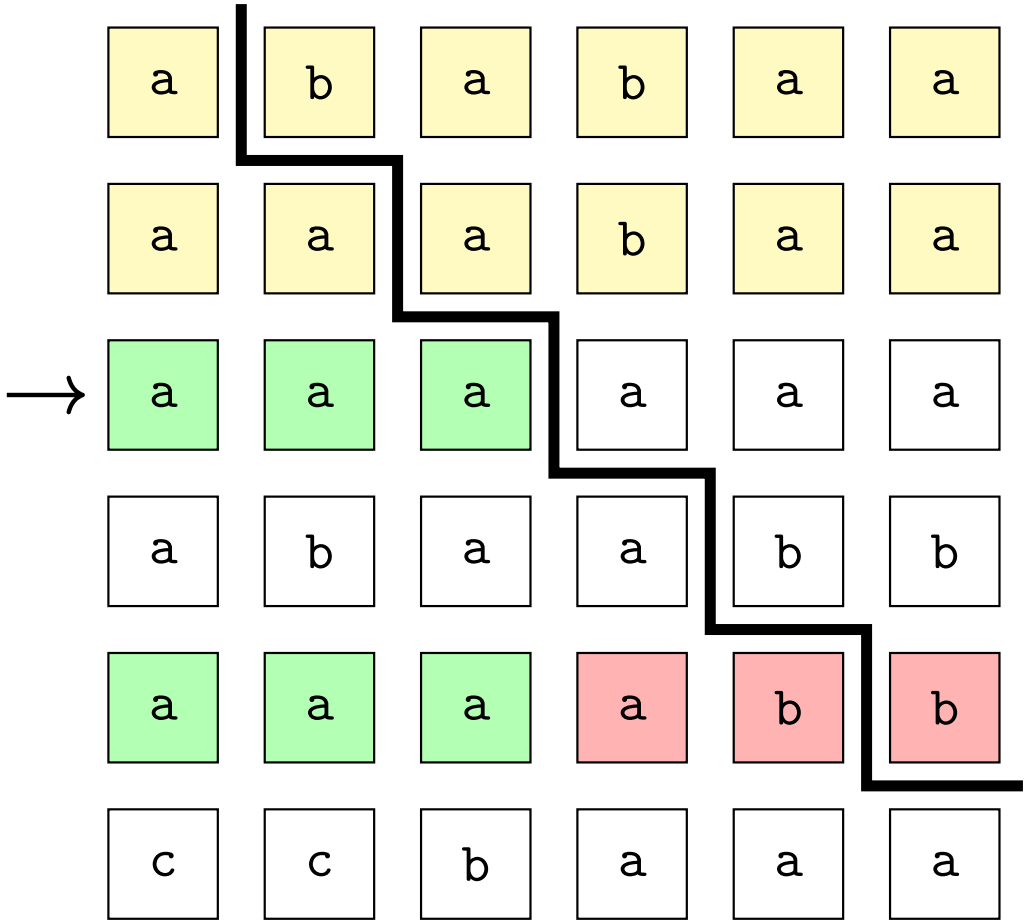
Zadanie J – Szyfr Atlantydów – Przykład



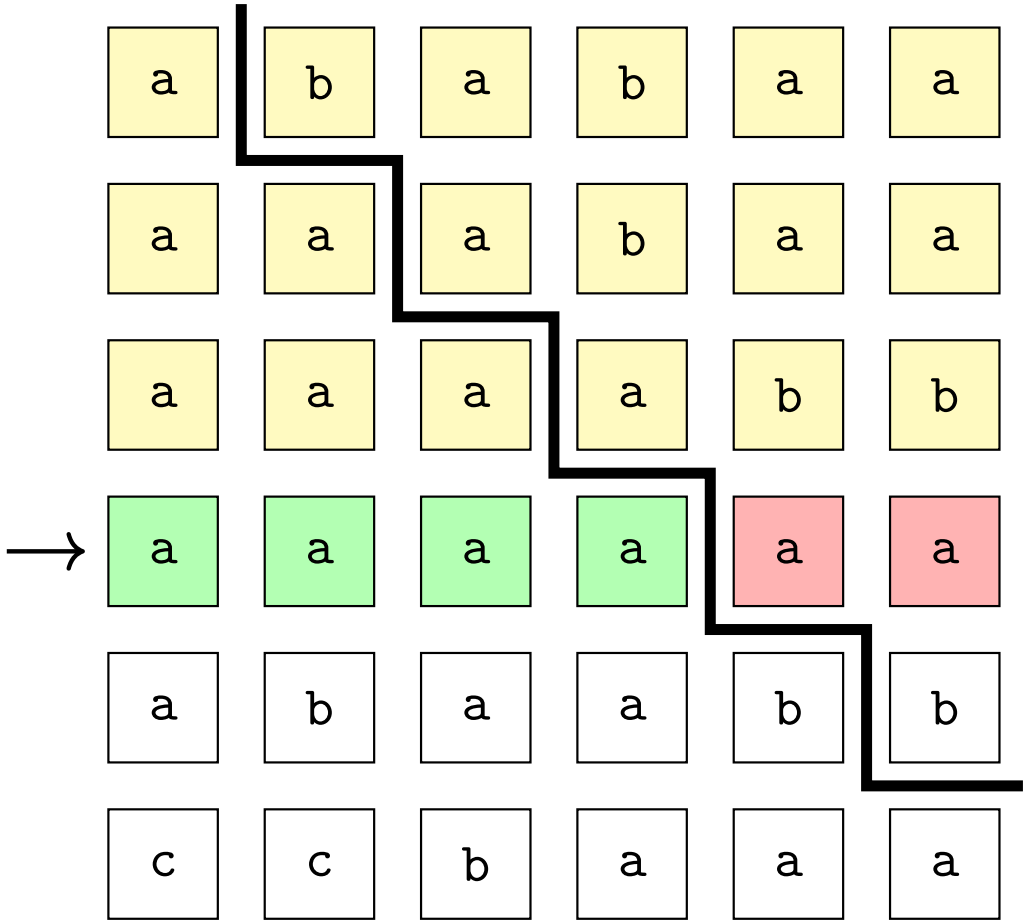
Zadanie J – Szyfr Atlantydów – Przykład



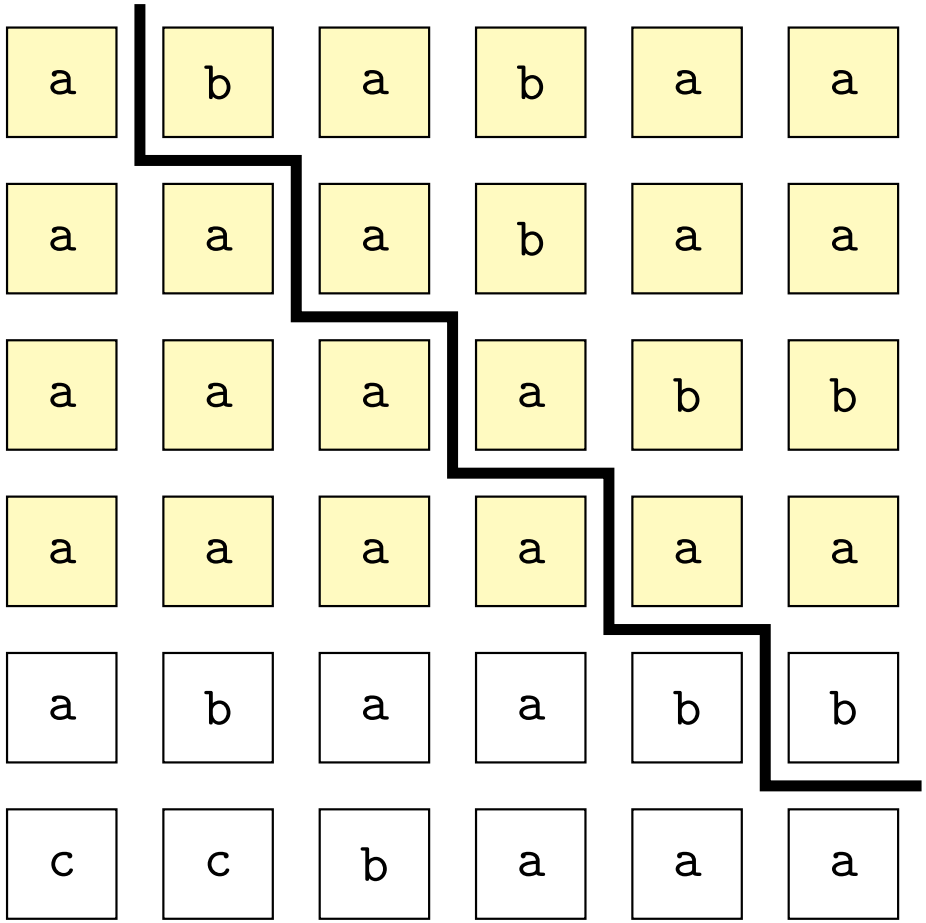
Zadanie J – Szyfr Atlantydów – Przykład



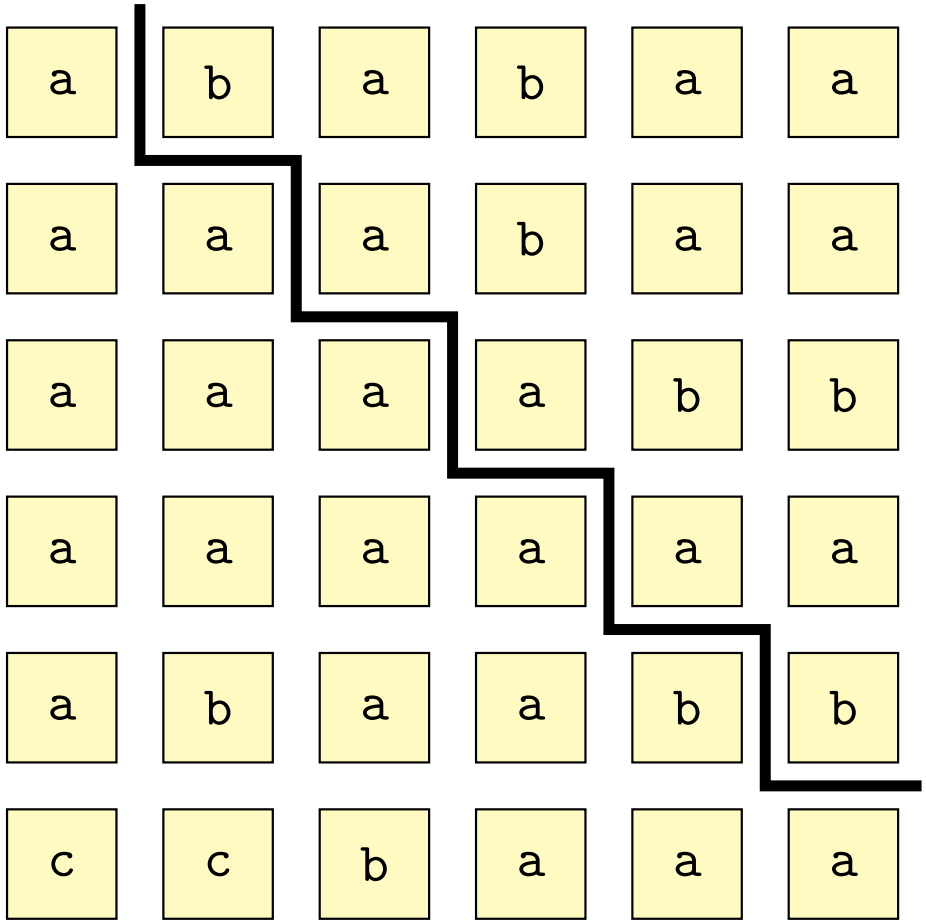
Zadanie J – Szyfr Atlantydów – Przykład



Zadanie J – Szyfr Atlantydów – Przykład



Zadanie J – Szyfr Atlantydów – Przykład



Zadanie J – Szyfr Atlantydów

- Urzymujemy zbiór wolnych słów w trie bądź `set<string>`.
- Finalna złożoność $O(N^2)$ lub $O(N^2 logN)$.

Zadanie E – Zatopione zadania

Zadanie E – Zatopione zadania

Zaakceptowanych zgłoszeń: 19

Zadanie E – Zatopione zadania

Problem

Dane są współrzędne dziur. Określić czy mogły one powstać poprzez zginanie kartki i wykonanie w niej jednego przebicia.

Zadanie E – Zatopione zadania

Problem

Dane są współrzędne dziur. Określić czy mogły one powstać poprzez zginanie kartki i wykonanie w niej jednego przebicia.

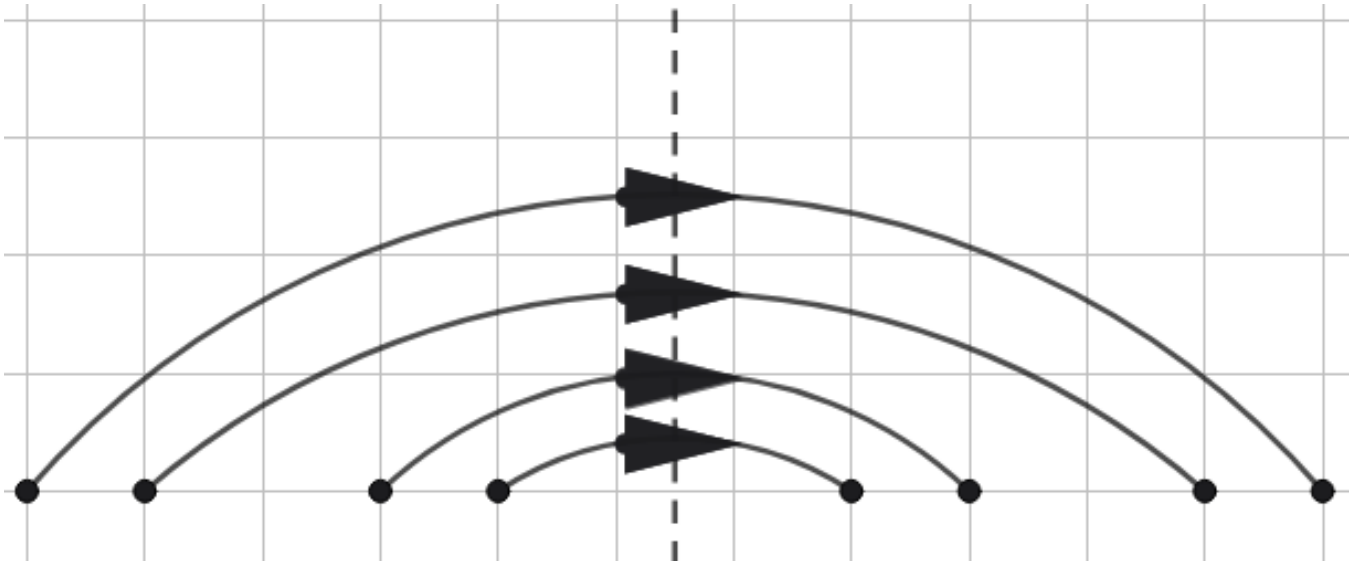
Obserwacja

Dziury powstałe w procesie opisanym w zadaniu są w 2^x wierszach i 2^y kolumnach tak, że na każdym przecięciu wiersza i kolumny jest dziura. Jest ich więc sumarycznie 2^{x+y}.

Obserwacja

Kolejność zgięć nie ma znaczenia, więc wykonajmy wpierw wszystkie pionowe. Sprowadziliśmy tak problem do sprawdzenia, czy możemy otrzymać dziury w jednym wymiarze. Analogicznie, gdy wykonamy wpierw poziome zgięcia.

- By sprawdzić czy dziury mogły powstać w jednym wymiarze, popatrzmy na odległości pomiędzy nimi.
- Wyobraźmy sobie pojedyncze zgięcie:



- Zauważmy, że pierwszy punkt przechodzi na ostatni, drugi na przedostatni i tak dalej, zatem ciąg odległości musi być palindromem. Po zgięciu zostaje tylko połowa punktów, odległości, pomiędzy którymi też muszą być palindromem. Analogicznie, przy kolejnych zgięciach.
- Sprawdzamy więc, czy cały ciąg, jego prawa połowa, prawa 1/4 itd. są palindromami.

Zadanie G – Podmorska świątynia

Zadanie G – Podmorska świątynia

Zaakceptowanych zgłoszeń: 15

Zadanie G – Podmorska świątynia

Problem

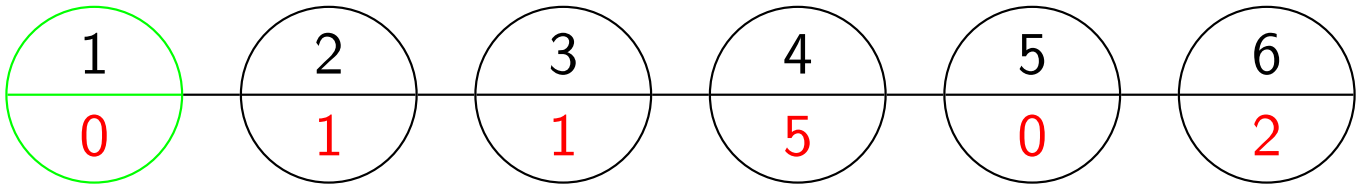
Dany jest graf z wagami na wierzchołkach. Po grafie tym będziemy chodzić. Naszym poziomem będziemy nazywać liczbę odwiedzonych przez nas wierzchołków. Do wierzchołka v można wejść tylko wtedy, gdy nasz poziom wynosi co najmniej $w(v)$. Wędrówkę zaczynamy w wierzchołku o numerze 1. Wyznaczyć najmniejszy poziom, z którym możemy wejść do wierzchołka N albo stwierdzić, że się nie da.

Zadanie G – Podmorska świątynia

W tym zadaniu będziemy szukać ścieżki w pewnym sensie najtańszej, a później pokażemy że wystarczy nią iść kiedy tylko możemy.

Zadanie G – Podmorska świątynia

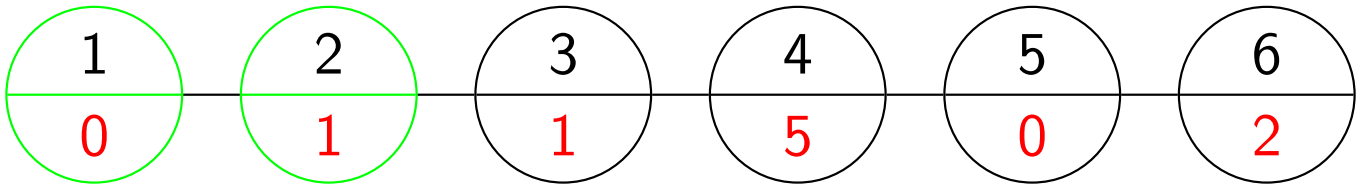
Koszt ścieżki (def. obrazkowa):



Koszt aktualnego prefiksu: 1

Zadanie G – Podmorska świątynia

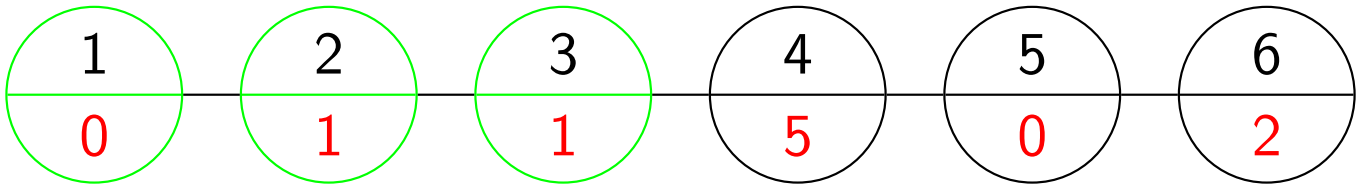
Koszt ścieżki (def. obrazkowa):



Koszt aktualnego prefiksu: 2

Zadanie G – Podmorska świątynia

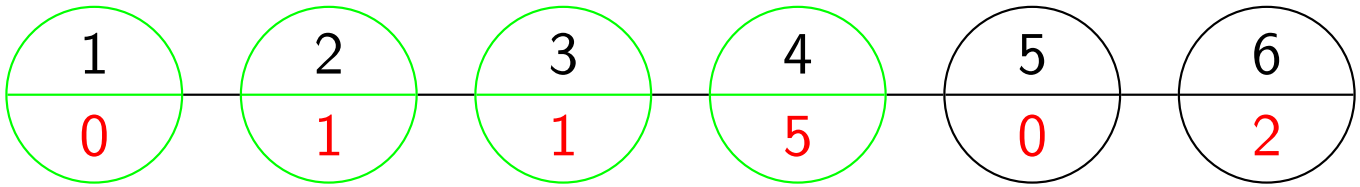
Koszt ścieżki (def. obrazkowa):



Koszt aktualnego prefiksu: 3

Zadanie G – Podmorska świątynia

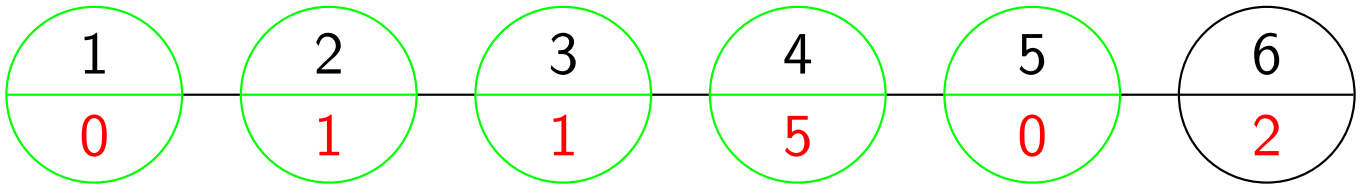
Koszt ścieżki (def. obrazkowa):



Koszt aktualnego prefiksu: 6

Zadanie G – Podmorska świątynia

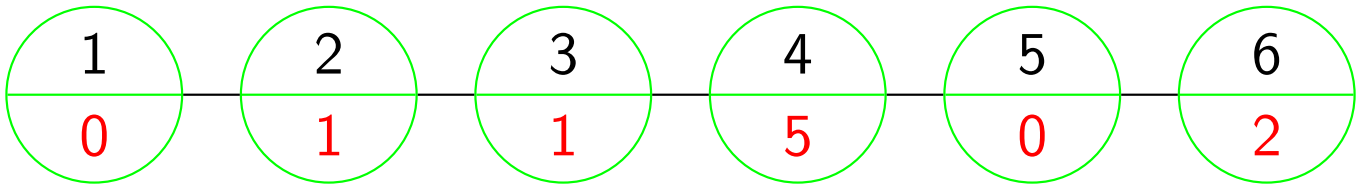
Koszt ścieżki (def. obrazkowa):



Koszt aktualnego prefiksu: 6

Zadanie G – Podmorska świątynia

Koszt ścieżki (def. obrazkowa):



Koszt aktualnego prefiksu: 7

Zadanie G – Podmorska świątynia

Obserwacja: Niech C będzie najmniejszym kosztem, spośród kosztów wszystkich ścieżek od wierzchołka o numerze 1 do wierzchołka o numerze N. Jeśli do wierzchołka N da się jakkolwiek dojść, to odpowiedzią jest C.

Zadanie G – Podmorska świątynia

Obserwacja: Niech C będzie najmniejszym kosztem, spośród kosztów wszystkich ścieżek od wierzchołka o numerze 1 do wierzchołka o numerze N. Jeśli do wierzchołka N da się jakkolwiek dojść, to odpowiedzią jest C.

Dowód minimalności C:
Po ścieżce o koszcie K nie można przejść kończąc z niższym poziomem niż K. Wszystkie rozwiązania przechodzą po jakiejś ścieżce od wierzchołka o numerze 1 do wierzchołka o numerze N, stąd odpowiedź nie może być mniejsza niż C.

Zadanie G – Podmorska świątynia

Dowód tego, że C dobre:

Zadanie G – Podmorska świątynia

Dowód tego, że C dobre:
Intuicyjnie: Bierzemy dowolny ciąg ruchów, który doprowadza nas do N. Będziemy chcieli iść po optymalnej ścieżce jeśli tylko możemy. Jeśli nie możemy, bo mamy zbyt niski poziom, to papugujemy to gorsze rozwiązanie, do momentu w którym osiągniemy wystarczająco duży poziom.

Zadanie G – Podmorska świątynia

Dowód tego, że C dobre:
Bardziej ściśle:
Niech $a_1, \ldots, a_{K_1}$ ścieżką prostą od wierzchołka o numerze 1 do wierzchołka o numerze N minimaluzjącą koszt. Niech $b_1, \ldots, b_{K_2}$ będzie dowolną ścieżką od 1 do N, którą można przejść.

Zadanie G – Podmorska świątynia

Dowód tego, że C dobre:
Bardziej ściśle:
Niech $a_1, \ldots, a_{K_1}$ ścieżką prostą od wierzchołka o numerze 1 do wierzchołka o numerze N minimaluzjącą koszt. Niech $b_1, \ldots, b_{K_2}$ będzie dowolną ścieżką od 1 do N, którą można przejść.
Skonstruujemy teraz optymalną ścieżkę, zaczynając od wierzchołka $1 = a_1 = b_1$.

Zadanie G – Podmorska świątynia

Dowód tego, że C dobre:
Bardziej ściśle:
Niech $a_1, \ldots, a_{K_1}$ ścieżką prostą od wierzchołka o numerze 1 do wierzchołka o numerze N minimaluzjącą koszt. Niech $b_1, \ldots, b_{K_2}$ będzie dowolną ścieżką od 1 do N, którą można przejść.
Skonstruujemy teraz optymalną ścieżkę, zaczynając od wierzchołka $1 = a_1 = b_1$.
Załóżmy, że nasza ścieżka odwiedziła już wierzchołki $a_1, \ldots a_i$, oraz $b_1, \ldots b_j$. W takim razie bez wzrostu poziomu możemy dojść do wierzchołka sąsiadującego z a_{i+1} oraz do wierzchołka sąsiadującego z b_{j+1}.

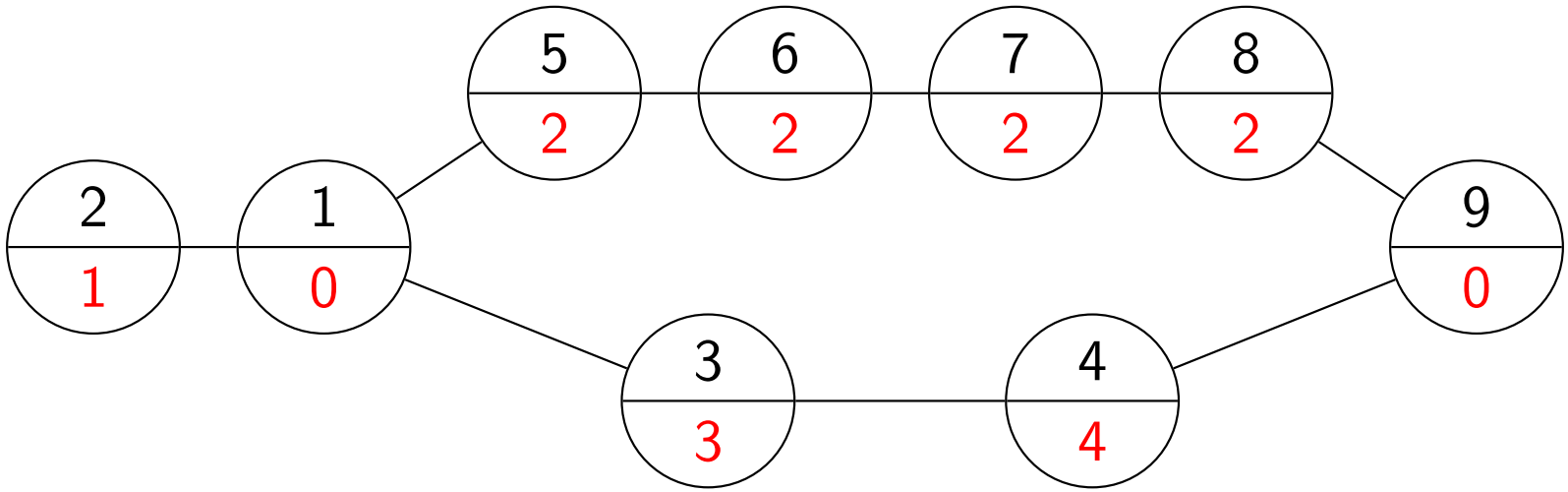
Zadanie G – Podmorska świątynia

Dowód tego, że C dobre:
Bardziej ściśle:
Niech $a_1, \ldots, a_{K_1}$ ścieżką prostą od wierzchołka o numerze 1 do wierzchołka o numerze N minimaluzjącą koszt. Niech $b_1, \ldots, b_{K_2}$ będzie dowolną ścieżką od 1 do N, którą można przejść.
Skonstruujemy teraz optymalną ścieżkę, zaczynając od wierzchołka $1 = a_1 = b_1$.
Załóżmy, że nasza ścieżka odwiedziła już wierzchołki $a_1, \ldots a_i$, oraz $b_1, \ldots b_j$. W takim razie bez wzrostu poziomu możemy dojść do wierzchołka sąsiadującego z a_{i+1} oraz do wierzchołka sąsiadującego z b_{j+1}.
Jeśli nasz poziom pozwala nam wejść do wierzchołka a_{i+1}, to wchodzimy.
Jeśli nie, to wchodzimy do b_{j+1}.

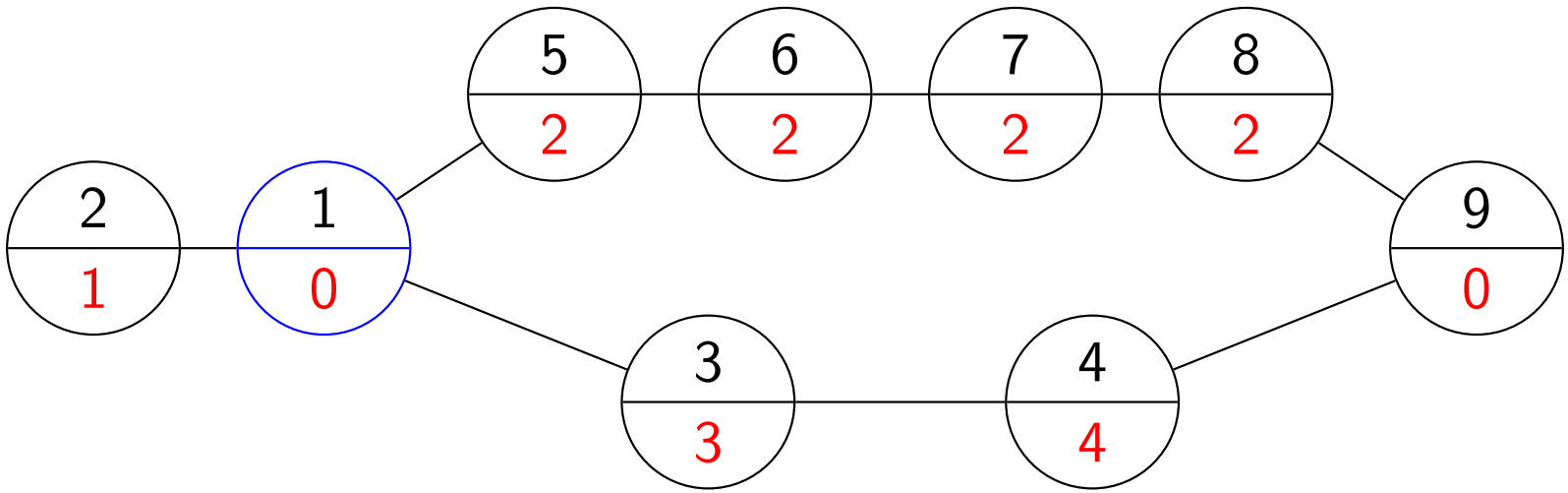
Zadanie G – Podmorska świątynia

Dowód tego, że C dobre:
Bardziej ściśle:
Niech $a_1, \ldots, a_{K_1}$ ścieżką prostą od wierzchołka o numerze 1 do wierzchołka o numerze N minimaluzjącą koszt. Niech $b_1, \ldots, b_{K_2}$ będzie dowolną ścieżką od 1 do N, którą można przejść.
Skonstruujemy teraz optymalną ścieżkę, zaczynając od wierzchołka $1 = a_1 = b_1$.
Załóżmy, że nasza ścieżka odwiedziła już wierzchołki $a_1, \ldots a_i$, oraz $b_1, \ldots b_j$. W takim razie bez wzrostu poziomu możemy dojść do wierzchołka sąsiadującego z a_{i+1} oraz do wierzchołka sąsiadującego z b_{j+1}.
Jeśli nasz poziom pozwala nam wejść do wierzchołka a_{i+1}, to wchodzimy.
Jeśli nie, to wchodzimy do b_{j+1}.
Łatwo zauważyc, że przechodząc po tak skonstrukowanej ścieżce dojdziemy do N mając poziom C.

Zadanie G – Podmorska świątynia

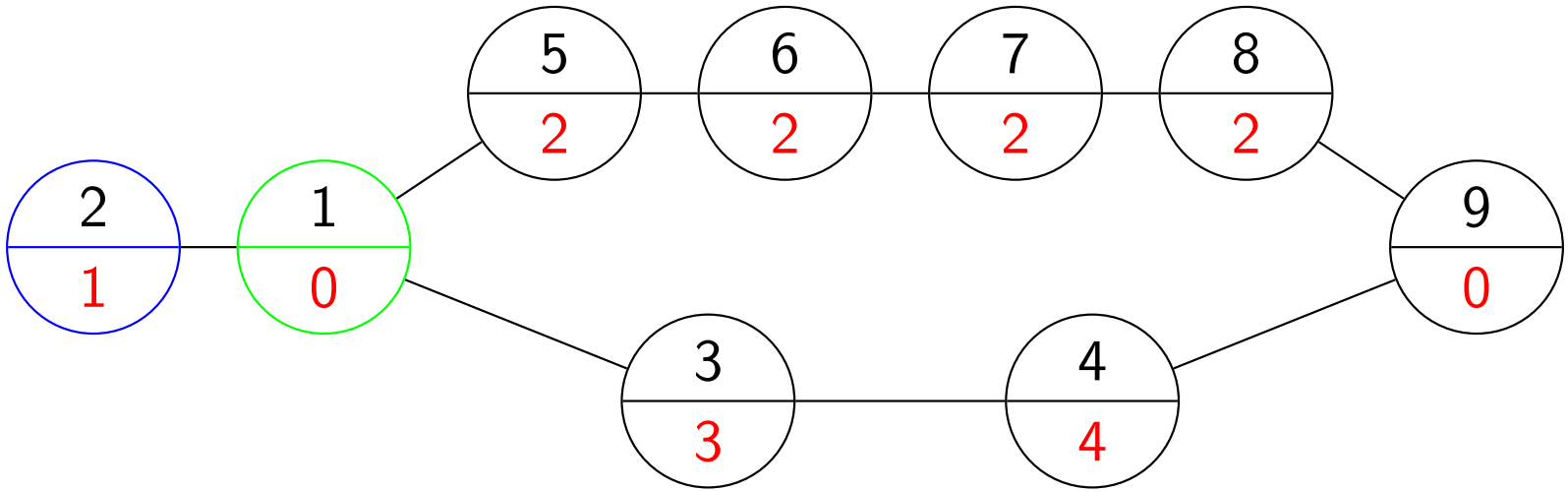


Zadanie G – Podmorska świątynia



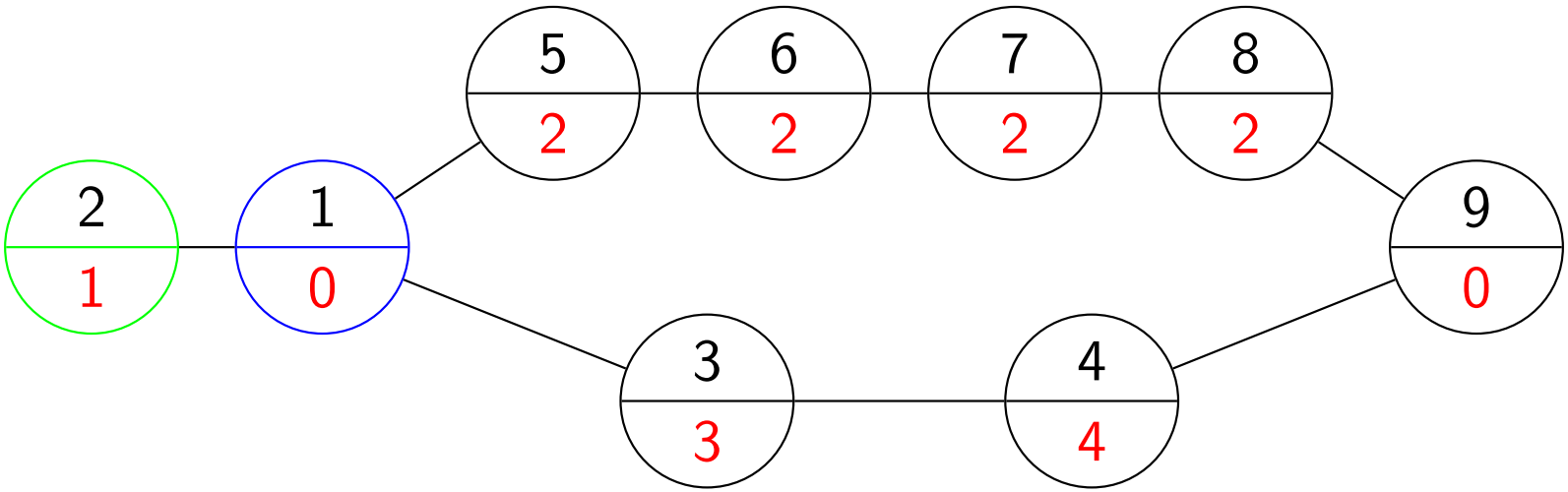
Poziom: 1

Zadanie G – Podmorska świątynia



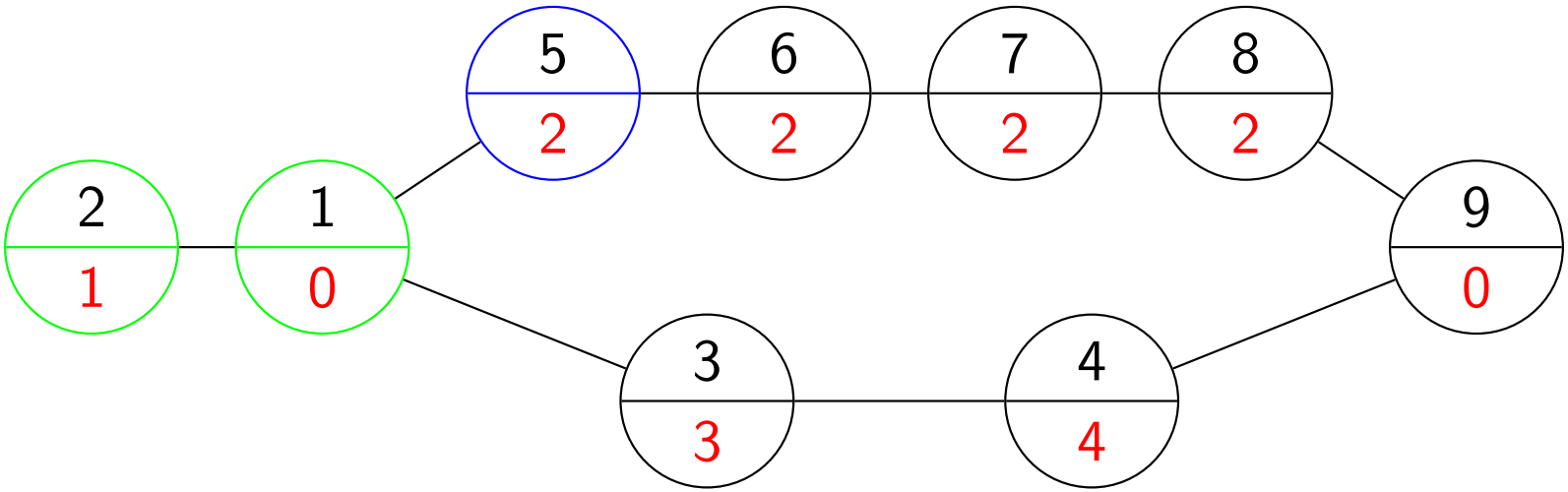
Poziom: 2

Zadanie G – Podmorska świątynia



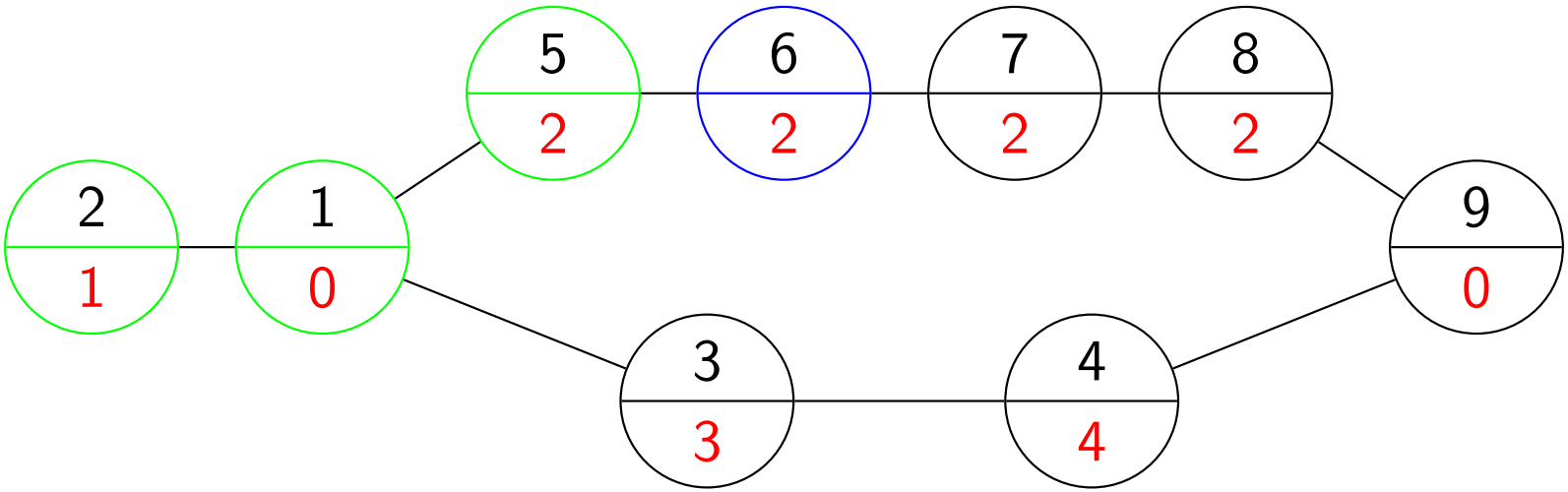
Poziom: 2

Zadanie G – Podmorska świątynia



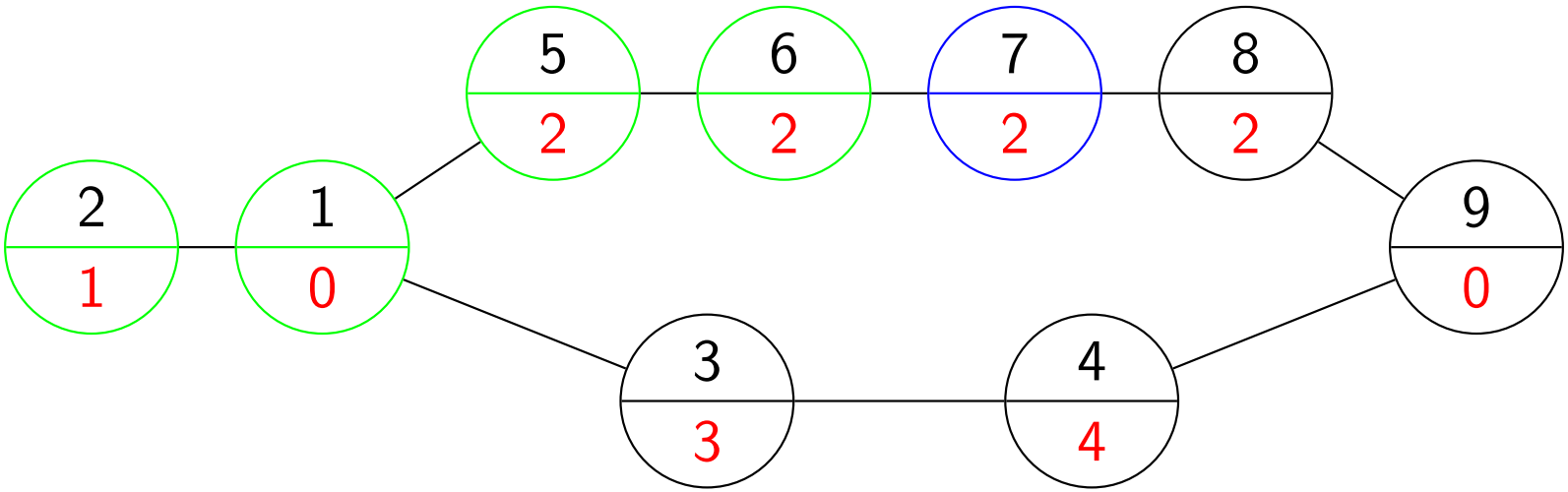
Poziom: 3

Zadanie G – Podmorska świątynia



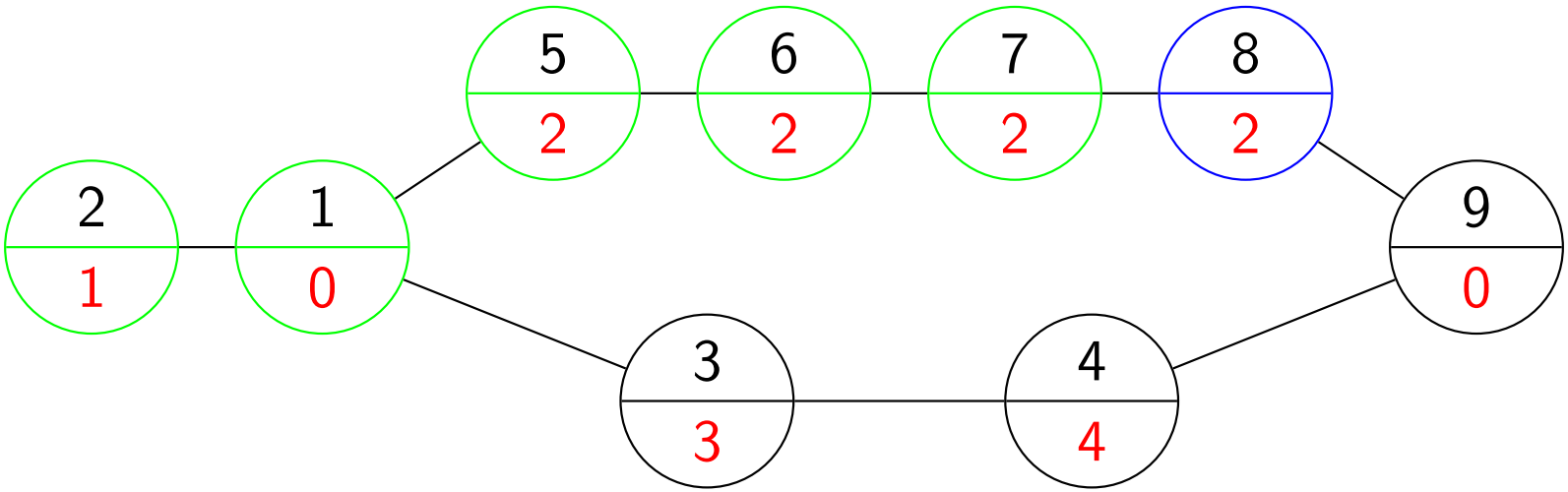
Poziom: 4

Zadanie G – Podmorska świątynia



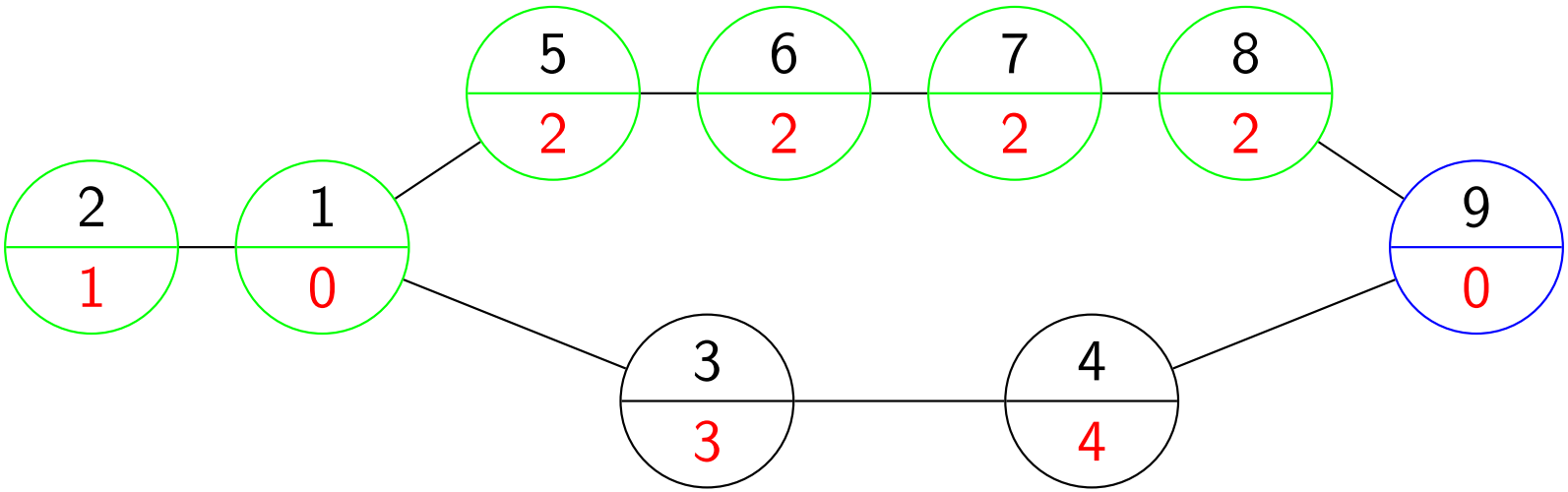
Poziom: 5

Zadanie G – Podmorska świątynia



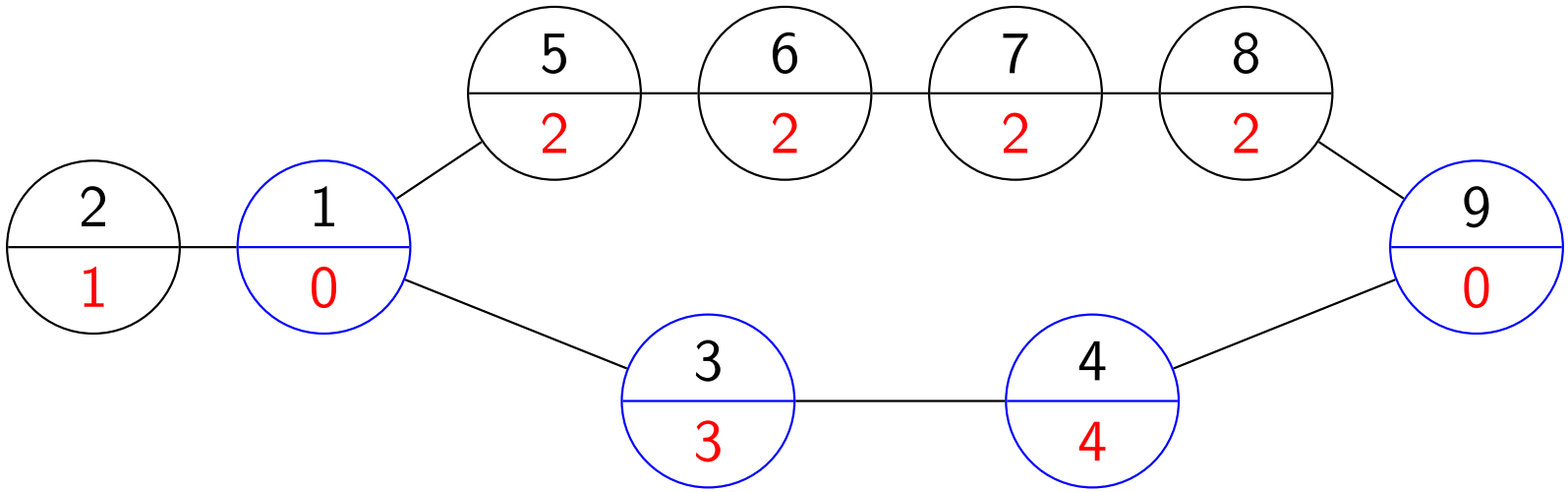
Poziom: 6

Zadanie G – Podmorska świątynia



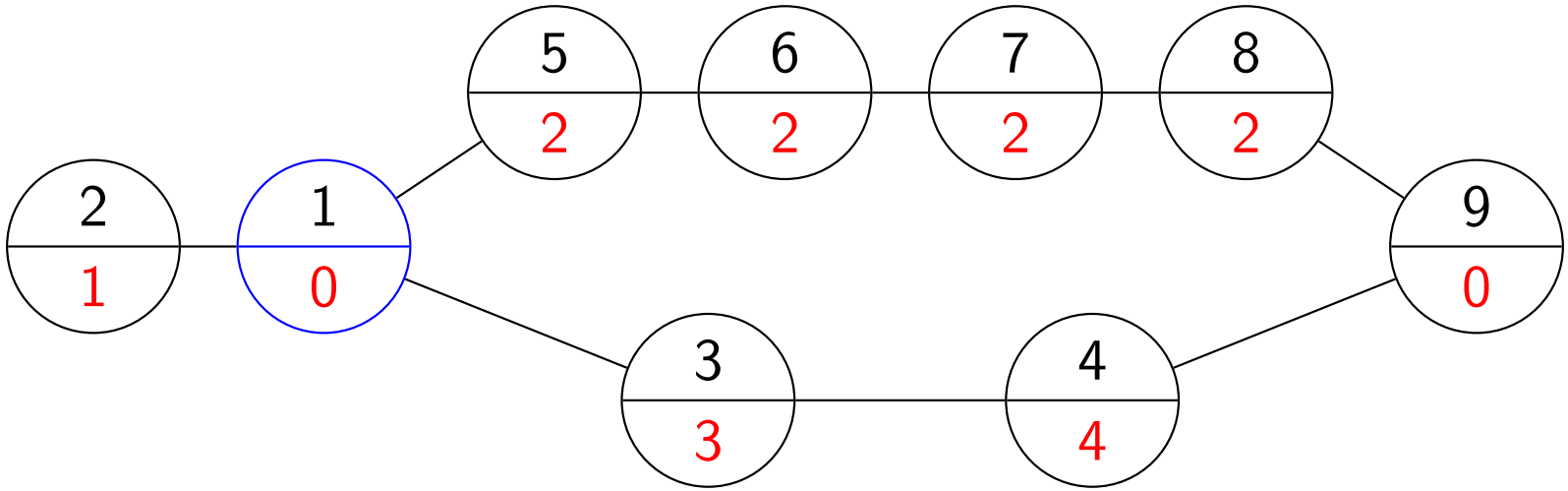
Poziom: 6

Zadanie G – Podmorska świątynia



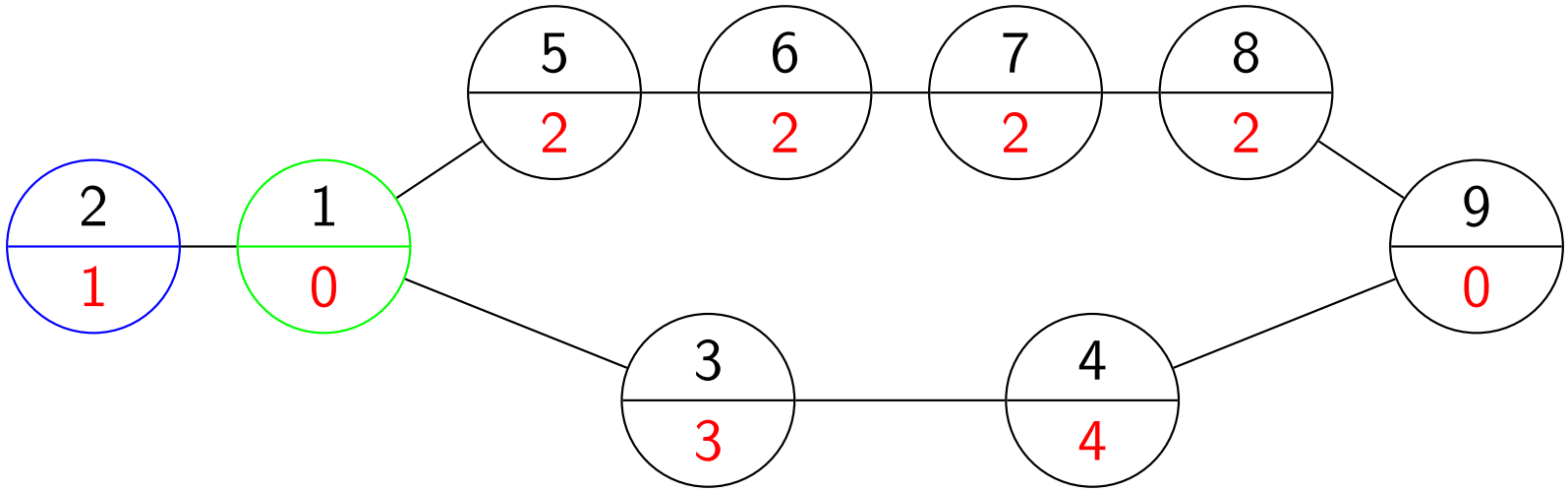
Ścieżka prosta minimalizująca koszt (jej koszt to 5)

Zadanie G – Podmorska świątynia



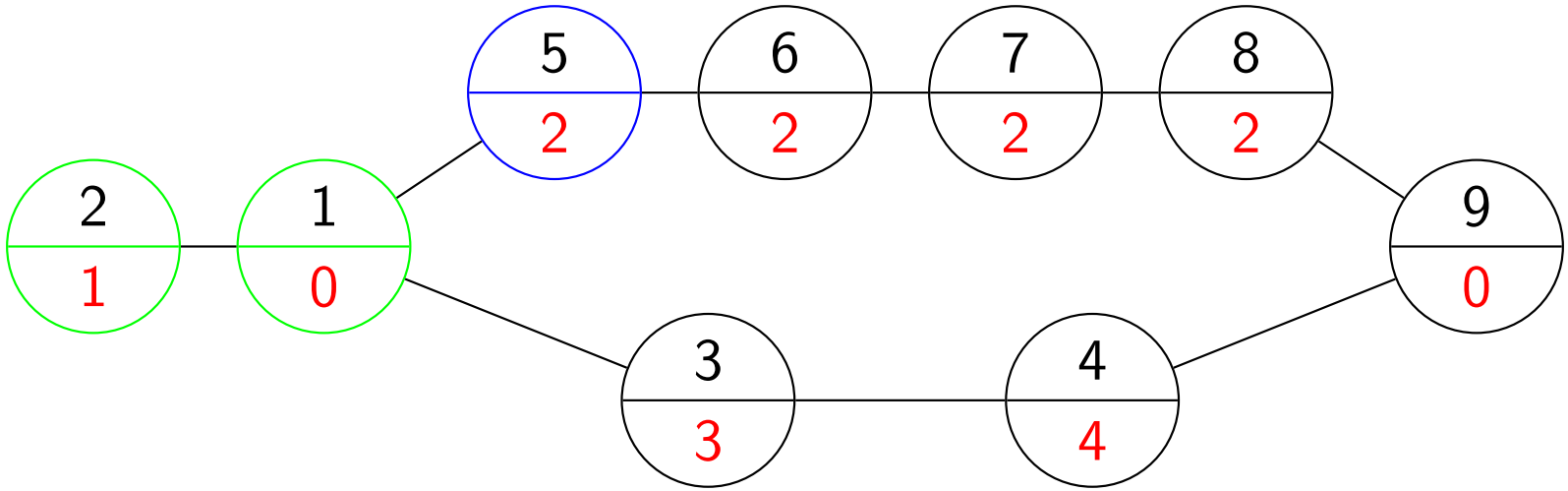
Poziom: 1 - nie możemy przejść po najtańszej ścieżce

Zadanie G – Podmorska świątynia



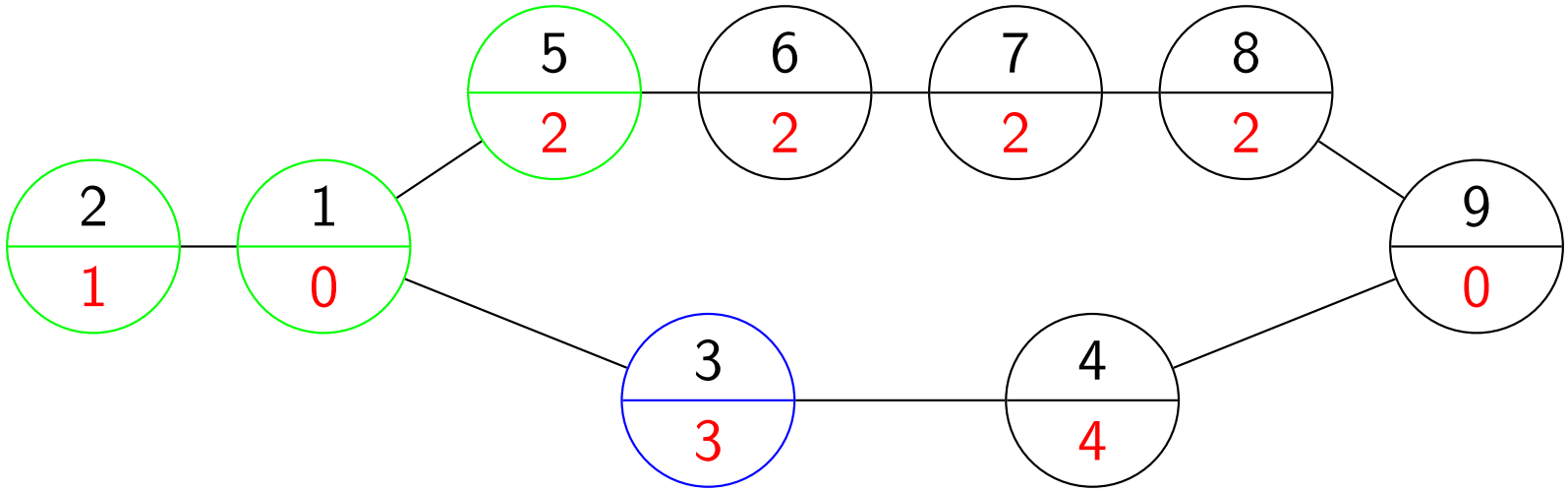
Poziom: 2 - nie możemy przejść po najtańszej ścieżce

Zadanie G – Podmorska świątynia



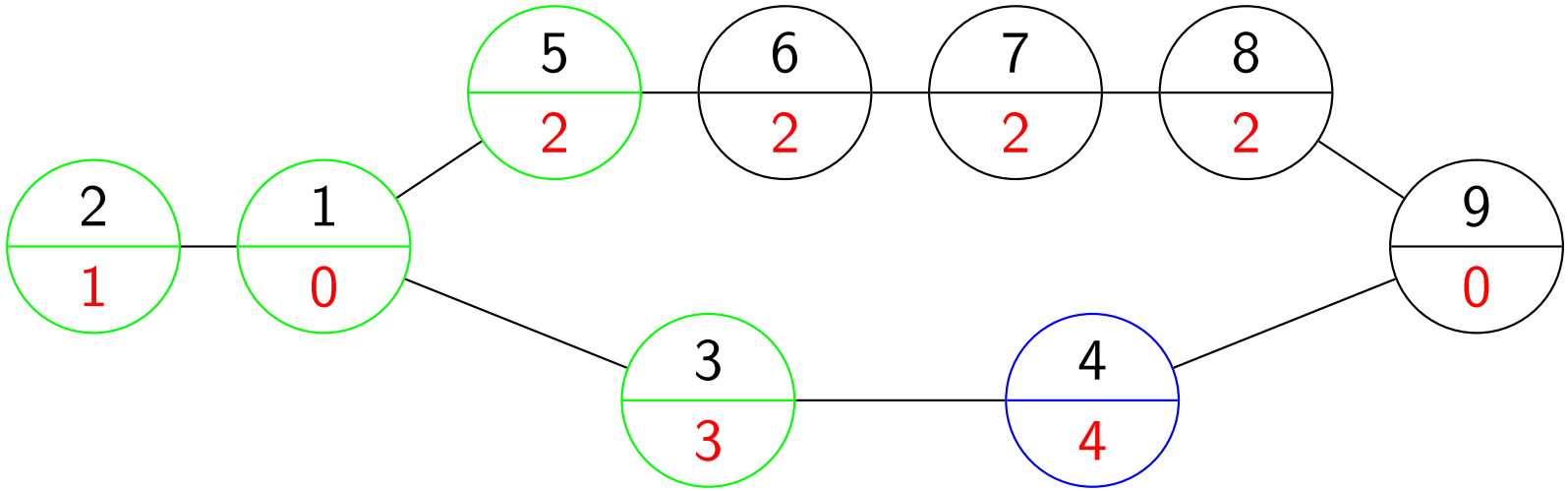
Poziom: 3 - możemy przejść po najtańszej ścieżce

Zadanie G – Podmorska świątynia



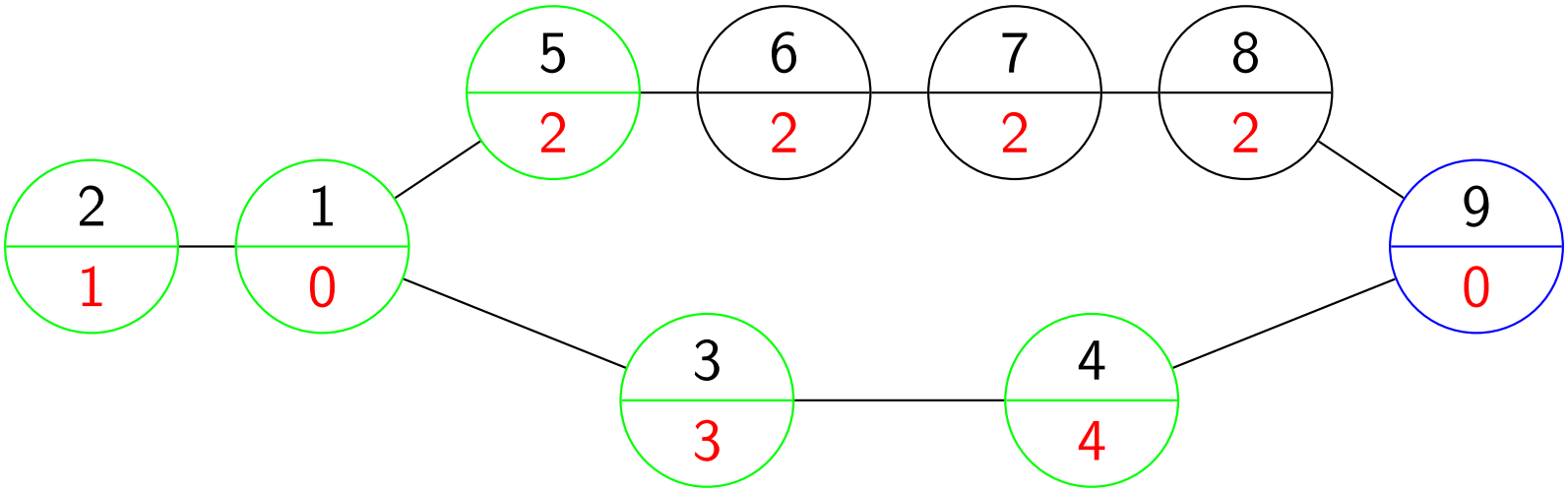
Poziom: 4 - możemy przejść po najtańszej ścieżce

Zadanie G – Podmorska świątynia



Poziom: 5 - możemy przejść po najtańszej ścieżce

Zadanie G – Podmorska świątynia



Poziom: 5 - doszliśmy

Zadanie G – Podmorska świątynia

Algorytm:

- Sprawdź, czy do wierzchołka N da się dojść.
- Jeśli tak, to oblicz i wypisz wartość C.
- Jeśli nie, to wypisz -1.

Zadanie G – Podmorska świątynia

Wartość C znajdujemy zmodyfikowanym algorytmem Dijkstry.

```
int dijkstra()
{
    ustaw dystans do wszystkich wierzcholkow na $\infty$
    ustaw dystans do wierzcholka o numerze 1 na 1
    S $\leftarrow$ zbior wszystkich wierzcholkow
    while (S niepuste)
    {
        v $\leftarrow$ wierzcholek z S o najnizszym dystansie
        usun v z S
        for (u : sasiedzi v ktorzy sa w S)
        {
            ustaw dystans do u na min(dystans do u, max(dystans do v, waga u) + 1)
        }
    }
}
```

Po zakończeniu algorytmu C jest równe dystansowi do wierzchołka o numerze N. Dowód taki sam jak zwykłego algorytmu Dijkstry.

Zadanie G – Podmorska świątynia

Sprawdzenie, czy do wierzchołka N da się dojść, też można wykonać algorytmem podobnym do Dijkstry. Wystarczy wchodzić do dostępnych wierzchołków o najniższej wadze dopóki nie jesteśmy zablokowani, bądź nie doszliśmy do wierzchołka o numerze N.

Zadanie G – Podmorska świątynia

Algorytm:

- Sprawdź, czy do wierzchołka N da się dojść.
- Jeśli tak, to oblicz i wypisz wartość C.
- Jeśli nie, to wypisz -1.

Zadanie G – Podmorska świątynia

Algorytm:

- Sprawdź, czy do wierzchołka N da się dojść.
- Jeśli tak, to oblicz i wypisz wartość C.
- Jeśli nie, to wypisz -1.

Złożoność:
Zależna od implementacji algorytmu Dijkstry. Jeśli używamy standardowej kolejki priorytetowej, to złożoność wynosi $\mathcal{O}((N+M)\log N)$. Ponieważ koszt ścieżki nie przekracza N, to algorytm Dijkstry można zaimplementować za pomocą kubełków w $\mathcal{O}(N+M)$ (to się wtedy nazywa algorytm Diala)
Ostateczna złożoność wynosi więc $\mathcal{O}(N+M)$

Zadanie B – Kraken

Zadanie B – Kraken

Zaakceptowanych zgłoszeń: 9

Zadanie B – Kraken

Problem

Mamy Q operacji wstawiania i usuwania liczb ze zbioru. Po każdej z nich stwierdzić czy da się usunąć co najwyżej jedną liczbę by *AND* i *XOR* pozostałych był sobie równy.

Zadanie B – Kraken

Problem

Mamy Q operacji wstawiania i usuwania liczb ze zbioru. Po każdej z nich stwierdzić czy da się usunąć co najwyżej jedną liczbę by *AND* i *XOR* pozostałych był sobie równy.

- Aby sprawdzić, czy *AND* i *XOR* zbioru są równe, możemy analizować każdy bit osobno. Dla każdego z nich przechowujemy liczbę elementów w zbiorze, które mają go zapalony. Wtedy, jeżeli dla każdego bitu zachodzi:
  - wszystkie liczby mają zapalony ten bit oraz liczb jest nieparzyście wiele lub
  - nie wszystkie liczby mają zapalony ten bit oraz tych, które mają go zapalony, jest parzyście wiele

  to *AND* i *XOR* zbioru są równe.

Zadanie B – Kraken

1. Usunięcie liczby zmienia wartość *AND* zbioru.

Obserwacja

Liczba, której usunięcie zmienia *AND* zbioru, musi być jedyną liczbą, która na pewnym bicie miała 0. Takich liczb może być co najwyżej tyle ile bitów.

- Znajdźmy więc dla każdego bitu jakąś wartość, która jest aktualnie w zbiorze i ma na tej pozycji 0. Następnie wystarczy zasymulować usunięcie jej i sprawdzić wcześniej opisany warunek na równość *AND* i *XOR* zbioru.
- Trzymajmy dla każdego bitu listę liczb ze zbioru, które mają na tej pozycji 0. Przy dodawaniu liczby dopisujemy ją do odpowiednich list. Podczas szukania dla danego bitu wartości, która nadal jest w zbiorze, możemy leniwie usuwać z listy wartości, których nie ma już w zbiorze, aż trafimy na wciąż w nim obecną.

Zadanie B – Kraken

2. Usunięcie liczby nie zmienia wartości *AND* zbioru.

Chcemy więc zmienić *XOR* tak, by był równy obecnemu *AND*. Z własności *xor* wynika, że potrzebujemy usunąć liczbę równą:

$$XOR_{obecny} \oplus AND_{obceny}$$

Wystarczy więc sprawdzić, czy taka liczba istnieje w zbiorze i jej usunięcie nie zmieni *AND*.

Zadanie H – Podwodne tankowanie

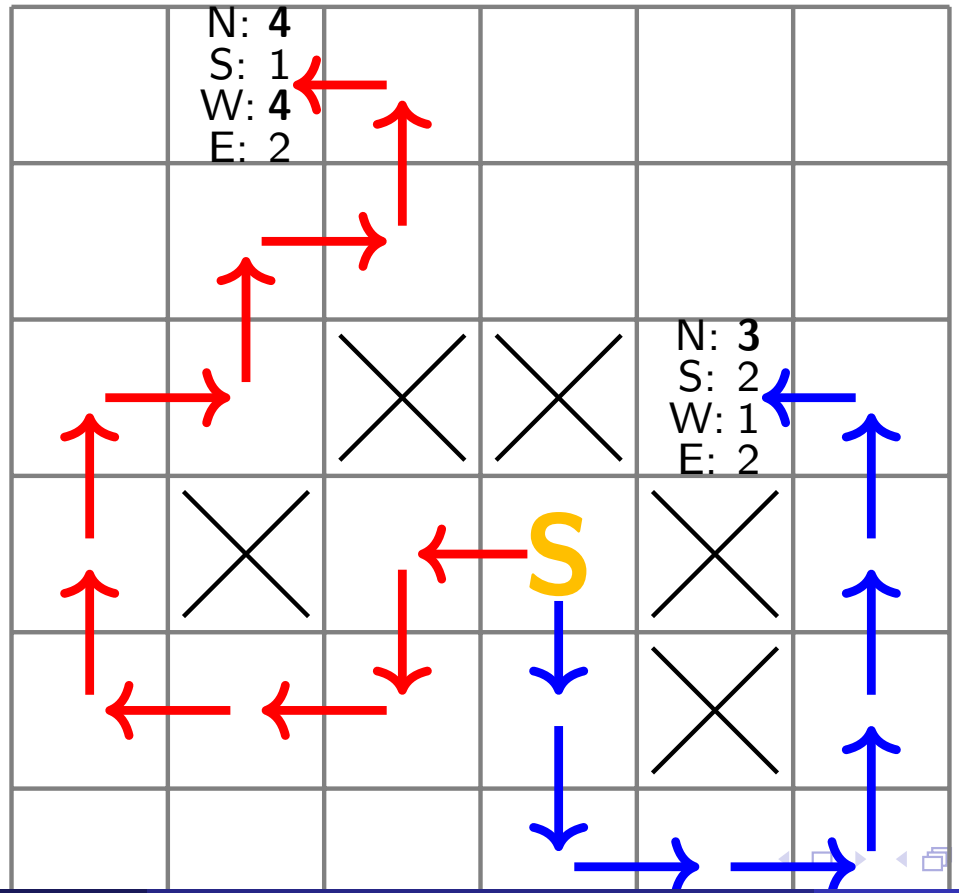
Zadanie H – Podwodne tankowanie

Zaakceptowanych zgłoszeń: 4

Zadanie H – Podwodne tankowanie

Problem

Dana plansza $N \times N$. Policzyć dla każdego pola minimalną liczbę K, taką że istnieje sekwencja ruchów, która wykonuje sumarycznie nie więcej niż K ruchów w jednym kierunków.



Zadanie H – Podwodne tankowanie

Problem

Dana plansza $N \times N$. Policzyć dla każdego pola minimalną liczbę K, taką że istnieje sekwencja ruchów, która wykonuje sumarycznie nie więcej niż K ruchów w jednym kierunków.

- Gdy zaczynamy w (x_S, y_S) i kończymy w (x_K, y_K) to różnica pomiedzy ruchami w prawo i lewo jest stała i wynosi $(x_S - x_K)$.
- Dzięki temu minimalizując liczbę ruchów w lewo minimalizujemy ruchy w prawo.
- To samo się tyczy ruchów do góry i w dół.
- Można zaaplikować $DP[x][y][l]$ = minimalna liczba ruchów w górę do pozycji (x, y), poruszając się dokładnie l razy w lewo.

Zadanie H – Podwodne tankowanie

- $DP[x][y][l]$ — minimalna liczba ruchów w górę do pozycji (x, y), poruszając się dokładnie l razy w lewo.
- Odpowiedź dla pozycji (x, y) to minimum z:

$$\max(l,\ DP[x_K][y_K][l],\ l + (x_S - x_K),\ DP[x_K][y_K][l] + (y_S - y_K))$$

- Tablicę DP obliczamy za pomocą algorytmu BFS 0-1.
- Finalna złożoność obliczeniowa wynosi $O(N^4)$.

Zadanie D – Misja Namazu

Zadanie D – Misja Namazu

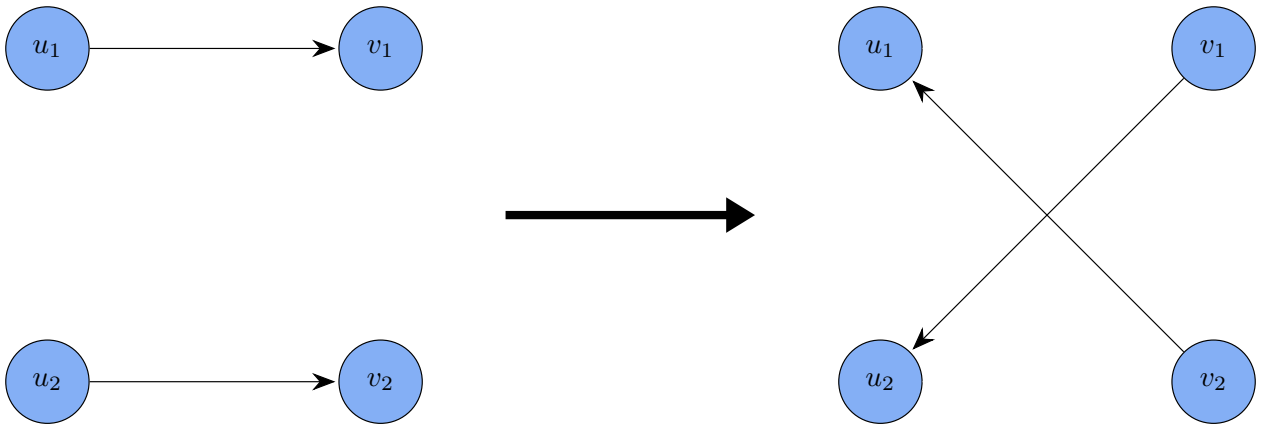
Zaakceptowanych zgłoszeń: 1

Zadanie D – Misja Namazu

Problem

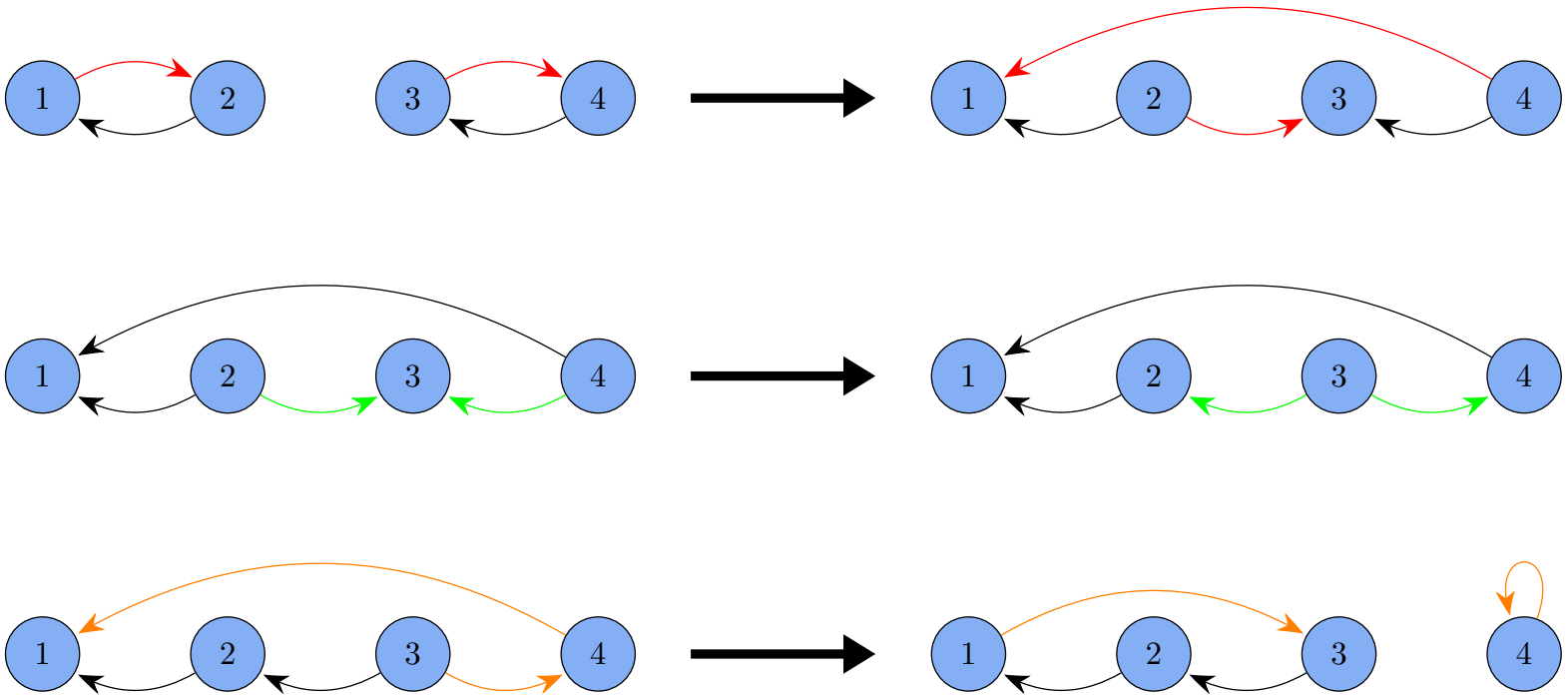
Dane są dwa grafy skierowane, w których każdy wierzchołek ma dokładnie jedną krawędź wychodzącą.
Zamień pierwszy z nich w drugi, używając wyłącznie operacji:

- wybierz dwie krawędzie (u_1, v_1) oraz (u_2, v_2),
- zamień je na krawędzie (v_1, u_2) oraz (v_2, u_1).



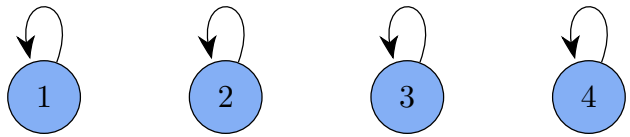
Zadanie D – Misja Namazu

Przykład:



Zadanie D – Misja Namazu

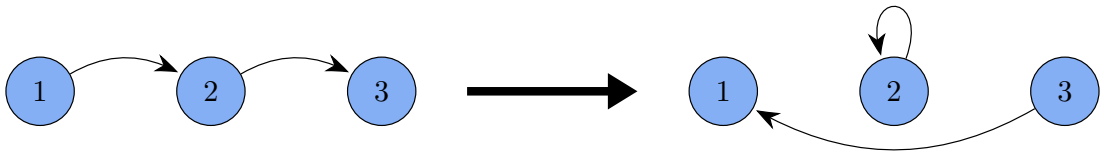
Cel to doprowadzić oba grafy do stanu, w którym każdy wierzchołek wskazuje na siebie.



Operacje na drugim grafie będziemy wykonywać w odwrotnej kolejności.

Zadanie D – Misja Namazu

Mając dane dwie krawędzie idące *jedna za drugą*, możemy łatwo stworzyć pętelkę.



Mając dane dwie krawędzie będące *na przeciwko siebie*, możemy je obrócić.

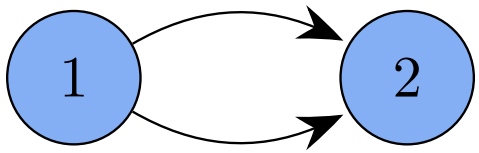
Zadanie D – Misja Namazu

Na początku rozwiązania połącz wszystki cykle w jeden wielki cykl (przyjmując, że krawędzie nie są skierowane).

Następnie zastosuj algorytm:

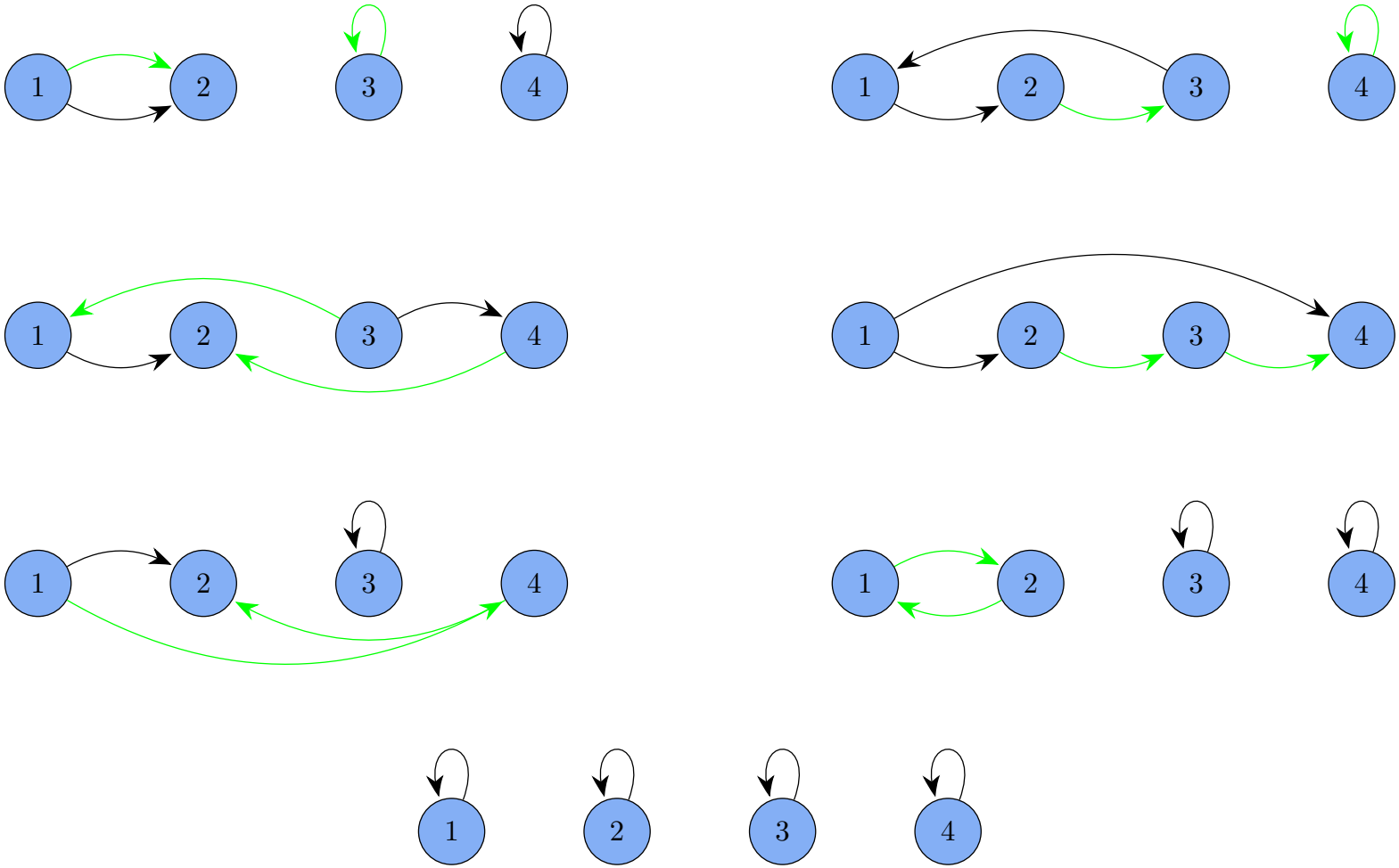
- jeśli istnieją dwie *kolejne* krawędzie, odłącz z nich pętelkę,
- w przeciwnym razie weź dowolne dwie *przeciwne* krawędzie i obróć je.

Po wykonaniu algorytmu graf albo będzie się składał z pojedynczych pętelek, albo będzie zawierał jedną parę krawędzi skierowanych w tę samą stronę.



Zadanie D – Misja Namazu

Konstrukcja rozwiązująca problem pary krawędzi:



Zadanie D – Misja Namazu

Podana konstrukcja nie działa dla trzech wierzchołków, wiec w tym przypadku odpowiedzią może być `NIE`.

BONUS: Zadanie da się rozwiązać również dla ogólnych grafów skierowanych.

Zadanie I – Naszyjnik z pereł

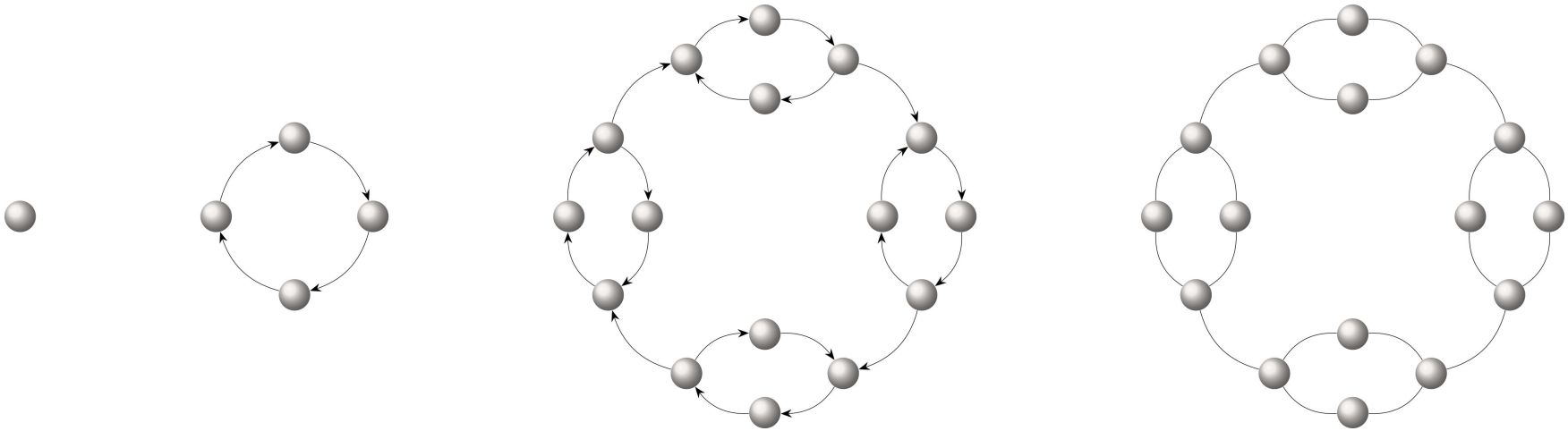
Zadanie I – Naszyjnik z pereł

Zaakceptowanych zgłoszeń: 0

Zadanie I – Naszyjnik z pereł

Problem

Według pewnych reguł konstruowany jest graf skierowany. Początkowo graf jest jednym wierzchołkiem. Wykonywanych jest n iteracji, w każdej z nich każdy wierzchołek zamienia się w A_i (parzyste) wierzchołków połączonych w cykl. Do zerowego wierzchołka na cyklu wchodzą stare wchodzące krawędzi, natomiast z $\frac{A_i}{2}$-tego wychodzą wszystkie wychodzące.



Na końcu usuwane jest skierowanie krawędzi. Należy policzyć ile w powstałym grafie jest drzew spinających.

Zadanie I – Naszyjnik z pereł

Powstawanie drzewa spinającego

Skupmy się na grafie powstałym na końcu. Ustalmy konkretne drzewo spinające, rozważmy jego krawędzi, które nie istniały jeszcze w czasie i. Tworzą one pewien las.

Zadanie I – Naszyjnik z pereł

Powstawanie drzewa spinającego

Skupmy się na grafie powstałym na końcu. Ustalmy konkretne drzewo spinające, rozważmy jego krawędzi, które nie istniały jeszcze w czasie i. Tworzą one pewien las.

Wierzchołki z jednej spójnej w tym lesie powstały z jednego wierzchołka istniejącego w czasie i.

Zadanie I – Naszyjnik z pereł

Powstawanie drzewa spinającego

Skupmy się na grafie powstałym na końcu. Ustalmy konkretne drzewo spinające, rozważmy jego krawędzi, które nie istniały jeszcze w czasie i. Tworzą one pewien las.

Wierzchołki z jednej spójnej w tym lesie powstały z jednego wierzchołka istniejącego w czasie i.

Dla ustalonego wierzchołka v istniejącego w czasie i mamy co najwyżej dwie składowe składające się z wierzchołków z niego powstałych.

Zadanie I – Naszyjnik z pereł

Powstawanie drzewa spinającego

Skupmy się na grafie powstałym na końcu. Ustalmy konkretne drzewo spinające, rozważmy jego krawędzi, które nie istniały jeszcze w czasie i. Tworzą one pewien las.

Wierzchołki z jednej spójnej w tym lesie powstały z jednego wierzchołka istniejącego w czasie i.

Dla ustalonego wierzchołka v istniejącego w czasie i mamy co najwyżej dwie składowe składające się z wierzchołków z niego powstałych.

Wynika to z tego, że do podgrafu utworzony z takich wierzchołków ma co najwyżej dwa wierzchołki mające krawędzi na zewnątrz (po przeciwnych stronach cyklu).

Zadanie I – Naszyjnik z pereł

Programowanie dynamiczne

Dla każdego wierzchołka w czasie i mamy dwa przypadki, może mu odpowiadać jedna składowa albo dwie składowe. Możemy policzyć ile jest różnych możliwości (wyglądu tej części grafu) w obu tych przypadkach. Musimy umieć wykonać przejście z $i + 1$ do i.

Zadanie I – Naszyjnik z pereł

Programowanie dynamiczne

Dla każdego wierzchołka w czasie i mamy dwa przypadki, może mu odpowiadać jedna składowa albo dwie składowe. Możemy policzyć ile jest różnych możliwości (wyglądu tej części grafu) w obu tych przypadkach. Musimy umieć wykonać przejście z $i + 1$ do i.

Początkowe wartości są proste:

- $dp[n]$[jedna składowa] $= 1$
- $dp[n]$[dwie składowe] $= 0$

Zadanie I – Naszyjnik z pereł

Programowanie dynamiczne

Dla każdego wierzchołka w czasie i mamy dwa przypadki, może mu odpowiadać jedna składowa albo dwie składowe. Możemy policzyć ile jest różnych możliwości (wyglądu tej części grafu) w obu tych przypadkach. Musimy umieć wykonać przejście z $i + 1$ do i.

Początkowe wartości są proste:

- $dp[n][\text{jedna składowa}] = 1$
- $dp[n][\text{dwie składowe}] = 0$

Wynikiem będzie $dp[0][1]$

Zadanie I – Naszyjnik z pereł

Grupa wierzchołków powstałych z v składa się z A_i grup złożonych z potomków każdego z dzieci v. Dzieci te są ustawione w cykl.
Mamy dwie ścieżki pomiędzy oboma przeciwległymi wierzchołkami cyklu. W każdej z nich co najwyżej jedno dziecko ma grupę mającą dwie składowe. Inaczej wierzchołki pomiędzy nimi nie byłyby połączone z resztą grafu.

Zadanie I – Naszyjnik z pereł

Grupa wierzchołków powstałych z v składa się z A_i grup złożonych z potomków każdego z dzieci v. Dzieci te są ustawione w cykl.
Mamy dwie ścieżki pomiędzy oboma przeciwległymi wierzchołkami cyklu. W każdej z nich co najwyżej jedno dziecko ma grupę mającą dwie składowe. Inaczej wierzchołki pomiędzy nimi nie byłyby połączone z resztą grafu.

W $dp[i][1]$ chcemy wybrać jedną z dwóch możliwości:

- dokładnie jedno dziecko miało grupę składającą się z dwóch składowych, bierzemy wówczas do drzewa wszystkie krawędzi powstałe w czasie i
- każde dziecko miało grupę składającą się z dokładnie jednej, ale wtedy nie bierzemy jednej z krawędzi powstałych w czasie i

Zadanie I – Naszyjnik z pereł

Grupa wierzchołków powstałych z v składa się z A_i grup złożonych z potomków każdego z dzieci v. Dzieci te są ustawione w cykl.
Mamy dwie ścieżki pomiędzy oboma przeciwległymi wierzchołkami cyklu. W każdej z nich co najwyżej jedno dziecko ma grupę mającą dwie składowe. Inaczej wierzchołki pomiędzy nimi nie byłyby połączone z resztą grafu.

W $dp[i][1]$ chcemy wybrać jedną z dwóch możliwości:

- dokładnie jedno dziecko miało grupę składającą się z dwóch składowych, bierzemy wówczas do drzewa wszystkie krawędzi powstałe w czasie i
- każde dziecko miało grupę składającą się z dokładnie jednej, ale wtedy nie bierzemy jednej z krawędzi powstałych w czasie i

Daje to $dp[i][1] = dp[i+1][1]^{A_i-1} \cdot dp[i+1][2] + dp[i+1][1]^{A_i} \cdot A_i$

Zadanie I – Naszyjnik z pereł

Żeby zliczyć przypadek dwóch drzew, spójrzmy na ścieżki od wierzchołka wchodzącego do wychodzącego. Będziemy chcieli je obie przeciąć. Zwróćmy uwagę na to, że mamy asymetrię. Ścieżka idąca z wierzchołka wchodzącego do wychodzącego ma w sobie $\frac{A_i}{2} + 1$ grup wierzchołków i łączących je $\frac{A_i}{2}$ krawędzi. Natomiast ścieżka w drugą stronę ma $\frac{A_i}{2} - 1$ grup i $\frac{A_i}{2}$ krawędzi.

Zadanie I – Naszyjnik z pereł

Żeby zliczyć przypadek dwóch drzew, spójrzmy na ścieżki od wierzchołka wchodzącego do wychodzącego. Będziemy chcieli je obie przeciąć. Zwróćmy uwagę na to, że mamy asymetrię. Ścieżka idąca z wierzchołka wchodzącego do wychodzącego ma w sobie $\frac{A_i}{2} + 1$ grup wierzchołków i łączących je $\frac{A_i}{2}$ krawędzi. Natomiast ścieżka w drugą stronę ma $\frac{A_i}{2} - 1$ grup i $\frac{A_i}{2}$ krawędzi.

Przecięcie ścieżki to albo wybranie którejś z grup i podzielenie jej na dwie składowe albo usunięcie którejś z krawędzi między grupami.

Zadanie I – Naszyjnik z pereł

Żeby zliczyć przypadek dwóch drzew, spójrzmy na ścieżki od wierzchołka wchodzącego do wychodzącego. Będziemy chcieli je obie przeciąć. Zwróćmy uwagę na to, że mamy asymetrię. Ścieżka idąca z wierzchołka wchodzącego do wychodzącego ma w sobie $\frac{A_i}{2} + 1$ grup wierzchołków i łączących je $\frac{A_i}{2}$ krawędzi. Natomiast ścieżka w drugą stronę ma $\frac{A_i}{2} - 1$ grup i $\frac{A_i}{2}$ krawędzi.

Przecięcie ścieżki to albo wybranie którejś z grup i podzielenie jej na dwie składowe albo usunięcie którejś z krawędzi między grupami.

Można to zapisać wzorem $dp[i][2] = L \cdot R \cdot dp[i+1][1]^{A_i - 2}$, gdzie:

$$L = (dp[i+1][2] \cdot \frac{A_i}{2} + dp[i+1][2] \cdot (\frac{A_i}{2} + 1))$$

$$R = (dp[i+1][2] \cdot \frac{A_i}{2} + dp[i+1][2] \cdot (\frac{A_i}{2} - 1))$$

Zadanie F – Zadanie od Jasia

Zadanie F – Zadanie od Jasia

Zaakceptowanych zgłoszeń: 0

Zadanie F – Zadanie od Jasia

Problem

Dla ciągu K liczb $B_1, \ldots, B_K$ rozważmy następujący problem: mamy do dyspozycji operację polegającą na wybraniu dowolnego przedziału (L, R) $(1 \leqslant L \leqslant R \leqslant K)$, oraz dla każdego i $(L \leqslant i \leqslant R)$ należącego do tego przedziału zamieniamy B_i na $|B_i - 1|$ (wartość bezwzględną z $B_i - 1$). Przez $f(B_1, \ldots, B_K)$ oznaczmy minimalną liczbę takich operacji potrzebną do zmienienia wszystkich B_i na zera.
Dany jest ciąg, należy podać wartość funkcji f dla pewnych jego przedziałów. Dodatkowo, ciąg ten może się zmieniać (jedna zmiana to modyfikacja wartości w punkcie).

Zadanie F – Zadanie od Jasia

Obserwacja

Przy liczeniu funkcji f, jeśli wartość w punkcie to X, musimy wykonać co najmniej X operacji zmniejszania zawierającego ten punkt. Dodatkowo, parzystość tej liczbe operacji jest taka sama jak parzystość X.
Kolejność operacji zmniejszania nie ma znaczenia.

Zadanie F – Zadanie od Jasia

Obserwacja

Przy liczeniu funkcji f, jeśli wartość w punkcie to X, musimy wykonać co najmniej X operacji zmniejszania zawierającego ten punkt. Dodatkowo, parzystość tej liczbe operacji jest taka sama jak parzystość X.
Kolejność operacji zmniejszania nie ma znaczenia.

Ważniejsza obserwacja

Nie opłaca się wykonać operacji zmniejszania na dwóch rozłącznych przedziałach.

Zadanie F – Zadanie od Jasia

Obserwacja

Przy liczeniu funkcji f, jeśli wartość w punkcie to X, musimy wykonać co najmniej X operacji zmniejszania zawierającego ten punkt. Dodatkowo, parzystość tej liczbe operacji jest taka sama jak parzystość X.
Kolejność operacji zmniejszania nie ma znaczenia.

Ważniejsza obserwacja

Nie opłaca się wykonać operacji zmniejszania na dwóch rozłącznych przedziałach.

Wykonanie operacji na rozłącznych przedziałach $[a,\ b]$, $[c,\ d]$ dla $a \leqslant b < c \leqslant d$ można zastąpić wykonaniem ich na $[a,\ c-1]$, $[b+1,\ d]$ (chyba, że $b+1=c$, ale wtedy można dwa przedziały złączyć w jeden). W ten sposób na punktach z $[a,\ b]$ oraz $[c,\ d]$ wykonamy tyle samo operacji, natomiast na $[b+1,\ c-1]$ o dwie więcej.

Zadanie F – Zadanie od Jasia

Z poprzedniej obserwacji można przyjąć dodatkowe założenie: spośród wszystkich najkrótszych ciągów operacji zmniejszania można wybrać takie, w którym wszystkie przedziały mają punkt wspólny.

Zadanie F – Zadanie od Jasia

Z poprzedniej obserwacji można przyjąć dodatkowe założenie: spośród wszystkich najkrótszych ciągów operacji zmniejszania można wybrać takie, w którym wszystkie przedziały mają punkt wspólny.

Można zmieniać rozłączne przedziały, przy każdej takiej zamianie zwiększa się ich suma długości.

Zadanie F – Zadanie od Jasia

Z poprzedniej obserwacji można przyjąć dodatkowe założenie: spośród wszystkich najkrótszych ciągów operacji zmniejszania można wybrać takie, w którym wszystkie przedziały mają punkt wspólny.

Można zmieniać rozłączne przedziały, przy każdej takiej zamianie zwiększa się ich suma długości. Wystarczy zatem teraz skupić się na tym, gdzie otwierają się i zamykają przedziały.

Liczenie odpowiedzi dla konkretnego punktu przecięcia

Powiedzmy, że liczymy odpowiedź dla przedziału $[l,\ r]$ i ustalonym punktem przecięcia przedziałów jest m. Odpowiedź to maksimum z potrzebnej liczby otworzeń i liczby zamknięć przedziałów. Minimalna liczba otworzeń przedziałów to:

$$\min_{v \in [l,\ m]} (A_v + \text{liczba zmian parzystości w ciągu } A_v,\ A_{v+1},\ \ldots\ A_m).$$

Zadanie F – Zadanie od Jasia

Wyszukiwanie binarne

Odpowiedź to maksimum z funkcji rosnącej i malejącej. Możemy zatem ten punkt znaleźć wyszukiwaniem binarnym.

Zadanie F – Zadanie od Jasia

Wyszukiwanie binarne

Odpowiedź to maksimum z funkcji rosnącej i malejącej. Możemy zatem ten punkt znaleźć wyszukiwaniem binarnym.

Z kolei konkretną wartość w punkcie można policzyć drzewem przedziałowym $(+, \min)$, umożliwia ono modyfikacje ciągu. Da to złożoność $\mathcal{O}(N + Q \log^2 N)$

Zadanie F – Zadanie od Jasia

Wyszukiwanie binarne

Odpowiedź to maksimum z funkcji rosnącej i malejącej. Możemy zatem ten punkt znaleźć wyszukiwaniem binarnym.

Z kolei konkretną wartość w punkcie można policzyć drzewem przedziałowym $(+, \min)$, umożliwia ono modyfikacje ciągu. Da to złożoność $\mathcal{O}(N + Q \log^2 N)$

Bonus

Można zamienić modyfikacje w punkcie na przykład na dodawanie na przedziale.

Statystyki z zawodów

Statystyki z zawodów

Łączna liczba zgłoszeń: 1326

Statystyki z zawodów

Łączna liczba zgłoszeń: 1326
Użyte języki programowania:

- C/C++: 1145+117+1
- Python3: 61
- Java: 2

Statystyki z zawodów

Łączna liczba zgłoszeń: 1326
Użyte języki programowania:

- C/C++: 1145+117+1
- Python3: 61
- Java: 2

Największa sumaryczna liczba zgłoszeń na jednym zadaniu: 184, Kraken

Statystyki z zawodów

Łączna liczba zgłoszeń: 1326
Użyte języki programowania:

- C/C++: 1145+117+1
- Python3: 61
- Java: 2

Największa sumaryczna liczba zgłoszeń na jednym zadaniu: 184, Kraken
Największa liczba zgłoszeń na jednym zadaniu z AC: 14, Aqua Racer

Statystyki z zawodów

Łączna liczba zgłoszeń: 1326
Użyte języki programowania:

- C/C++: 1145+117+1
- Python3: 61
- Java: 2

Największa sumaryczna liczba zgłoszeń na jednym zadaniu: 184, Kraken
Największa liczba zgłoszeń na jednym zadaniu z AC: 14, Aqua Racer
Największa liczba zgłoszeń na jednym zadaniu bez AC: 38, Aqua Racer

Statystyki z zawodów

Łączna liczba zgłoszeń: 1326
Użyte języki programowania:

- C/C++: 1145+117+1
- Python3: 61
- Java: 2

Największa sumaryczna liczba zgłoszeń na jednym zadaniu: 184, Kraken
Największa liczba zgłoszeń na jednym zadaniu z AC: 14, Aqua Racer
Największa liczba zgłoszeń na jednym zadaniu bez AC: 38, Aqua Racer
Sumaryczna liczba wykonanych testów: 16918

Statystyki z zawodów

Łączna liczba zgłoszeń: 1326
Użyte języki programowania:

- C/C++: 1145+117+1
- Python3: 61
- Java: 2

Największa sumaryczna liczba zgłoszeń na jednym zadaniu: 184, Kraken
Największa liczba zgłoszeń na jednym zadaniu z AC: 14, Aqua Racer
Największa liczba zgłoszeń na jednym zadaniu bez AC: 38, Aqua Racer
Sumaryczna liczba wykonanych testów: 16918
Sumaryczny rozmiar zgłoszeń: 1.7472 MB

Statystyki z zawodów

Łączna liczba zgłoszeń: 1326
Użyte języki programowania:

- C/C++: 1145+117+1
- Python3: 61
- Java: 2

Największa sumaryczna liczba zgłoszeń na jednym zadaniu: 184, Kraken
Największa liczba zgłoszeń na jednym zadaniu z AC: 14, Aqua Racer
Największa liczba zgłoszeń na jednym zadaniu bez AC: 38, Aqua Racer
Sumaryczna liczba wykonanych testów: 16918
Sumaryczny rozmiar zgłoszeń: 1.7472 MB
Najmniejszy rozmiar zaakceptowanego zgłoszenia: 88 bajtów, Zestawy do nurkowania

Statystyki z zawodów

Łączna liczba zgłoszeń: 1326
Użyte języki programowania:

- C/C++: 1145+117+1
- Python3: 61
- Java: 2

Największa sumaryczna liczba zgłoszeń na jednym zadaniu: 184, Kraken
Największa liczba zgłoszeń na jednym zadaniu z AC: 14, Aqua Racer
Największa liczba zgłoszeń na jednym zadaniu bez AC: 38, Aqua Racer
Sumaryczna liczba wykonanych testów: 16918
Sumaryczny rozmiar zgłoszeń: 1.7472 MB
Najmniejszy rozmiar zaakceptowanego zgłoszenia: 88 bajtów, Zestawy do nurkowania
Największy rozmiar zaakceptowanego zgłoszenia: 6.00 kilobajtów

Dziękujemy!