Omówienie zadań

26 czerwca 2022

Problem A - Era Imperium

Problem A - Era Imperium

Zadanie

Dla danej liczby N wypisać najmniejszą liczbę k, taką że N da się przedstawić jako sumę k liczb całkowitych z przedziału $[2, 3]$ lub stwierdzić, że nie jest to możliwe.

Problem A - Era Imperium

Zadanie

Dla danej liczby N wypisać najmniejszą liczbę k, taką że N da się przedstawić jako sumę k liczb całkowitych z przedziału $[2, 3]$ lub stwierdzić, że nie jest to możliwe.

Jeżeli reszta z dzielenia N przez 3 to:

- 0 – używamy tylko liczb 3.
- 1 – używamy dwóch liczb 2. Jeżeli $N = 1$, to odpowiedź nie istnieje.
- 2 – używamy jednej liczby 2.

Problem B - Zabawy z zapałkami

Problem B - Zabawy z zapałkami

Zadanie

Ile liczb podzielnych przez 111 da się ułożyć z co najwyżej N zapałek?

Problem B - Zabawy z zapałkami

Zadanie

Ile liczb podzielnych przez 111 da się ułożyć z co najwyżej N zapałek?

Obserwacja

Największa liczba, jaką możemy ułożyć z N zapałek, nie jest większa niż $(1.111\ldots) \cdot 10^{\lceil N/2 \rceil - 1}$. Dla $N = 20$ jest to $1\,111\,111\,111$.

Wystarczy sprawdzić $\frac{1\,111\,111\,111}{111} \sim 10^7$ liczb. Każdą liczbę sprawdzamy w czasie liniowym od liczby jej cyfr.

Problem C - Wiedźmak

Problem C - Wiedźmak

Zadanie

Odszyfruj 3 listy, które zostały zaszyfrowane poniższą funkcją.

```
unsigned int stan;
unsigned int los() {
    stan ^= stan << 13;
    stan ^= stan >> 17;
    stan ^= stan << 5;
    return stan;
}
char szyfruj_znak(char c) {
    if (c < 'a' || c > 'z') return c;
    int stary_numer_znaku = c - 'a'; // od 0 do 25
    int przesuniecie = los() % 26;
    int nowy_numer_znaku = (stary_numer_znaku + przesuniecie) % 26;
    return 'a' + nowy_numer_znaku;
}
std::string szyfruj_tekst(std::string s, unsigned int ziarno) {
    stan = ziarno;
    for (int i = 0; i < s.size(); ++i) {
        s[i] = szyfruj_znak(s[i]);
    }
    return s;
}
```

Problem C - Wiedźmak

Jeżeli znamy ziarno, to żeby odszyfrować tekst wystarczy podmienić w kodzie jedną linię w funkcji `szyfruj_znak`.

```
char szyfruj_znak(char c) {
    if (c < 'a' || c > 'z') return c;
    int stary_numer_znaku = c - 'a'; // od 0 do 25
    int przesuniecie = los() % 26;
    int nowy_numer_znaku = (stary_numer_znaku + przesuniecie) % 26;
    return 'a' + nowy_numer_znaku;
}
```

```
char deszyfruj_znak(char c) {
    if (c < 'a' || c > 'z') return c;
    int stary_numer_znaku = c - 'a'; // od 0 do 25
    int przesuniecie = los() % 26;
    int nowy_numer_znaku = (stary_numer_znaku - przesuniecie + 26) % 26;
    return 'a' + nowy_numer_znaku;
}
```

Problem C - Wiedźmak

Możliwych wartości ziarna jest zbyt wiele, żeby sprawdzać je wszystkie ręcznie. Zamiast tego możemy przefiltrować odszyfrowane teksty.

Sposób pierwszy

Korzystamy ze wskazówek zawartych w treści.

- Pierwszy list zawiera palindrom, składający się z co najmniej 6 liter.
- Drugi list zawiera słowo "kwadratowe" o długości co najmniej 5.
- Podpis autora jest taki sam we wszystkich listach (*dalibor gosciwuj*).

Problem C - Wiedźmak

Możliwych wartości ziarna jest zbyt wiele, żeby sprawdzać je wszystkie ręcznie. Zamiast tego możemy przefiltrować odszyfrowane teksty.

Sposób pierwszy

Korzystamy ze wskazówek zawartych w treści.

- Pierwszy list zawiera palindrom, składający się z co najmniej 6 liter.
- Drugi list zawiera słowo "kwadratowe" o długości co najmniej 5.
- Podpis autora jest taki sam we wszystkich listach (*dalibor gosciwuj*).

Sposób drugi

Heurystycznie oceniamy, czy dany tekst jest ciągiem losowych znaków, czy poprawnym tekstem w języku polskim. Na przykład:

- Sprawdzamy częstotliwość samogłosek.
- Zliczamy często pojawiające się słowa. Na przykład spójniki.

Problem D - Pradawne zwoje

Problem D - Pradawne zwoje

Zadanie

Rozważmy grę, w której otrzymujemy pewien ciąg binarny. Możemy wykonywać operację odwrócenia ostatniego bitu oraz oprację odwrócenia bitu znajdującego się bezpośrednio po lewej od pierwszej jedynki od prawej. Celem gry jest otrzymanie ciągu złożonego z samych zer.
Naszym zadaniem jest wypisanie takiego ciągu binarnego, że najmniejsza liczba operacji potrzebna do wygrania gry jest równa N lub stwierdzenie, że taki ciąg nie istnieje.

Problem D - Pradawne zwoje

Zadanie

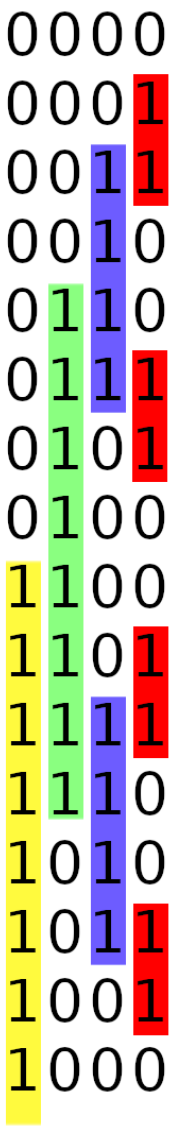
Rozważmy grę, w której otrzymujemy pewien ciąg binarny. Możemy wykonywać operację odwrócenia ostatniego bitu oraz oprację odwrócenia bitu znajdującego się bezpośrednio po lewej od pierwszej jedynki od prawej. Celem gry jest otrzymanie ciągu złożonego z samych zer.
Naszym zadaniem jest wypisanie takiego ciągu binarnego, że najmniejsza liczba operacji potrzebna do wygrania gry jest równa N lub stwierdzenie, że taki ciąg nie istnieje.

Obserwacja 1

Przy optymalnej strategii nie wykonamy tej samej operacji dwa razy z rzędu. Ponadto ostatnią operacją zawsze będzie odwrócenie ostatniego bitu.

Czyli równie dobrze możemy rozważać grę, w której zaczynamy od ciągu złożonego z samych zer, wykonujemy operacje naprzemiennie, zaczynając od operacji odwrócenia ostatniego bitu i chcemy otrzymać podany ciąg.

Problem D - Pradawne zwoje



Problem D - Pradawne zwoje

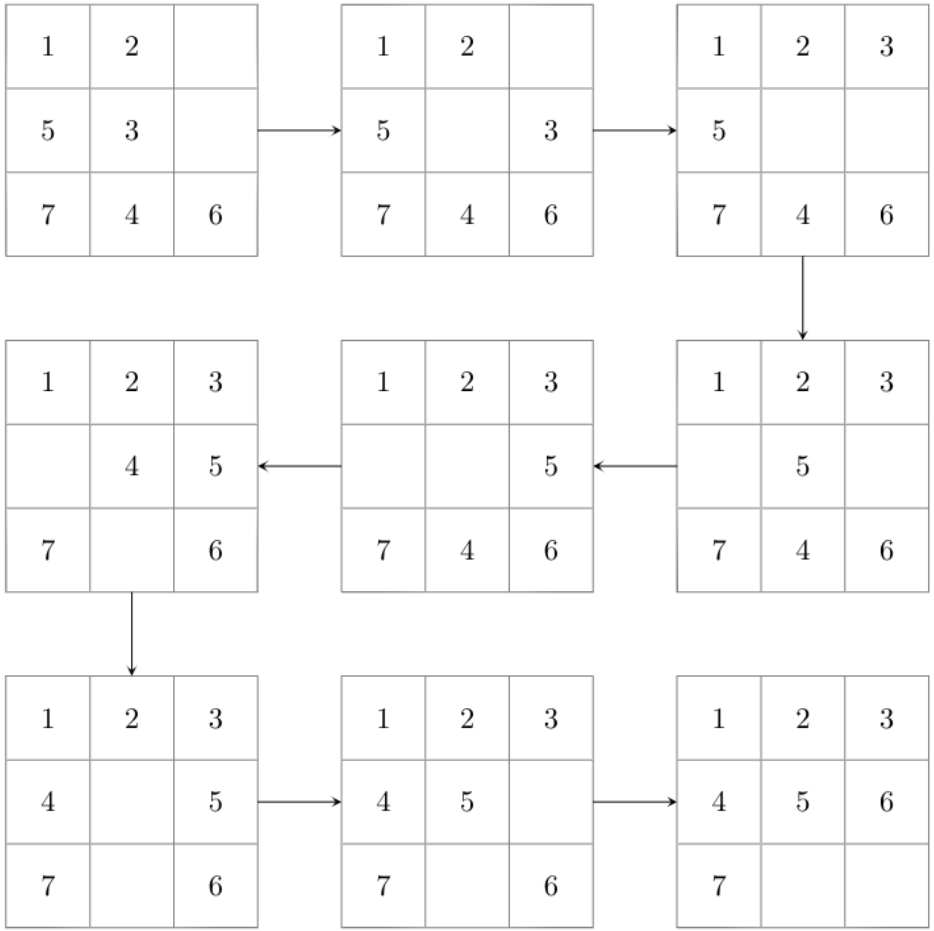
Po wykonaniu x operacji i-ty bit (numeracja od 0) jest równy

$$\left\lfloor \frac{x + 2^i}{2^{i+1}} \right\rfloor \mod 2 .$$

Czyli rozwiązanie zawsze istnieje i potrafimy je obliczyć w czasie $O(\log N)$.

Problem E - Świadek

Problem E - Świadek



Problem E - Świadek

Zadanie

Dana jest pomieszana układanka wymiaru $N \times M$, w której dwa pola są puste. Mamy podać sekwencję ruchów polegających na przesuwaniu kafelków na sąsiednie wolne pole, która doprowadzi do poprawnego ułożenia.

Problem E - Świadek

Zadanie

Dana jest pomieszana układanka wymiaru $N \times M$, w której dwa pola są puste. Mamy podać sekwencję ruchów polegających na przesuwaniu kafelków na sąsiednie wolne pole, która doprowadzi do poprawnego ułożenia.

Kafelki będziemy przesuwali na poprawne miejsce w kolejności zgodnej z numerem pozycji docelowej. Ruchy pozwalające na przesunięcie jednego kafelka (nazwijmy go X) możemy znaleźć przy pomocy algorytmu BFS, który zadba dodatkowo o to, żeby nie przesuwać kafelków już ułożonych poprawnie.

Problem E - Świadek

Zadanie

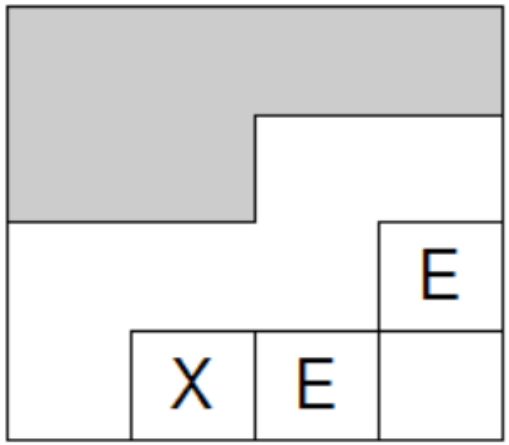
Dana jest pomieszana układanka wymiaru $N \times M$, w której dwa pola są puste. Mamy podać sekwencję ruchów polegających na przesuwaniu kafelków na sąsiednie wolne pole, która doprowadzi do poprawnego ułożenia.

Kafelki będziemy przesuwali na poprawne miejsce w kolejności zgodnej z numerem pozycji docelowej. Ruchy pozwalające na przesunięcie jednego kafelka (nazwijmy go X) możemy znaleźć przy pomocy algorytmu BFS, który zadba dodatkowo o to, żeby nie przesuwać kafelków już ułożonych poprawnie.

Stanem w przeszukiwaniu jest trójka: pozycja kafelka X, pozycja pierwszego pustego pola, pozycja drugiego pustego pola. Chcemy znaleźć ścieżkę od stanu reprezentującego aktualną pozycję na planszy do stanu, w którym X jest na docelowym miejscu.

Problem E - Świadek

1	2	3	4
5	6	9	10
12	11	8	
14	7		13



Problem E - Świadek

Algorytm BFS znajdzie poprawną ścieżkę, o ile wciąż zostały co najmniej dwa nieułożone wiersze – w takiej sytuacji zawsze możemy "ominąć" wierzchołek X, gdy przesuwamy puste pola, a tym samym każdy poprawny stan jest osiągalny.

Problem E - Świadek

Algorytm BFS znajdzie poprawną ścieżkę, o ile wciąż zostały co najmniej dwa nieułożone wiersze – w takiej sytuacji zawsze możemy "ominąć" wierzchołek X, gdy przesuwamy puste pola, a tym samym każdy poprawny stan jest osiągalny.

Gdy do ułożenia zostaną dwa ostatnie wiersze, to zmieniamy kolejność kafelków. Tym razem będziemy je ustawiać kolumnami od lewej do prawej, najpierw ustawiając kafelek w wierszu przedostatnim, a następnie w ostatnim.

Podobnie jak w pierwszej fazie, algorytm BFS za każdym razem znajdzie ścieżkę.

Problem E - Świadek

Liczba wykonanych ruchów

Wstępnie możemy oszacować, że będziemy potrzebowali $(M \cdot N - 2)$ (liczba kafelków) $\cdot (N \cdot M)^3$ (liczba stanów w BFS) ruchów.

Problem E - Świadek

Liczba wykonanych ruchów

Wstępnie możemy oszacować, że będziemy potrzebowali $(M \cdot N - 2)$ (liczba kafelków) $\cdot (N \cdot M)^3$ (liczba stanów w BFS) ruchów.

To szacowanie jest jednak mocno zawyżone, BFS znajduje przecież najkrótszą ścieżkę.

Problem E - Świadek

Liczba wykonanych ruchów

Wstępnie możemy oszacować, że będziemy potrzebowali $(M \cdot N - 2)$ (liczba kafelków) $\cdot (N \cdot M)^3$ (liczba stanów w BFS) ruchów.

To szacowanie jest jednak mocno zawyżone, BFS znajduje przecież najkrótszą ścieżkę.

Pokażemy przykładową sekwencję ruchów poprawiającą kafelek X, która jest krótka. A skoro BFS znajduje optymalną ścieżkę, to będzie ona nie dłuższa, niż ta pokazana przez nas.

Problem E - Świadek

1. Najpierw musimy przesunąć puste pola obok X, to zajmie $2 \cdot (N + M)$ ruchów.

Problem E - Świadek

1. Najpierw musimy przesunąć puste pola obok X, to zajmie $2 \cdot (N + M)$ ruchów.
2. Następnie przesuniemy kafelek X na docelowe miejsce. Skoro puste pole jest obok X, to poruszenie X w jakimś kierunku wymaga co najwyżej 5 przesunięć kafelków dookoła, a następnie przesunięcia X. Zatem przestawienie X na poprawną pozycję zajmie co najwyżej $6 \cdot (N + M)$ ruchów.

Problem E - Świadek

1. Najpierw musimy przesunąć puste pola obok X, to zajmie $2 \cdot (N + M)$ ruchów.
2. Następnie przesuniemy kafelek X na docelowe miejsce. Skoro puste pole jest obok X, to poruszenie X w jakimś kierunku wymaga co najwyżej 5 przesunięć kafelków dookoła, a następnie przesunięcia X. Zatem przestawienie X na poprawną pozycję zajmie co najwyżej $6 \cdot (N + M)$ ruchów.

Problem E - Świadek

1. Najpierw musimy przesunąć puste pola obok X, to zajmie $2 \cdot (N + M)$ ruchów.
2. Następnie przesuniemy kafelek X na docelowe miejsce. Skoro puste pole jest obok X, to poruszenie X w jakimś kierunku wymaga co najwyżej 5 przesunięć kafelków dookoła, a następnie przesunięcia X. Zatem przestawienie X na poprawną pozycję zajmie co najwyżej $6 \cdot (N + M)$ ruchów.

Poprawione oszacowanie

$$(N \cdot M - 2) \cdot (2 \cdot (N + M) + 6 \cdot (N + M)) \leqslant 100\,000$$

Problem F - Turniej szachowy

Problem F - Turniej szachowy

Zadanie

Dane jest drzewo o N wierzchołkach z przypisanymi losowymi wagami w_v. Aktywujemy krawędzie w podanej kolejności. Po każdej aktywacji mamy wypisać ile jest wierzchołków v, którym zwiększyła się wartość $\max_{u \in X}(w_v \oplus w_u)$, gdzie X to zbiór wierzchołków, do których można dojść z v korzystając jedynie z aktywowanych krawędzi.

Problem F - Turniej szachowy

Zadanie

Dane jest drzewo o N wierzchołkach z przypisanymi losowymi wagami w_v. Aktywujemy krawędzie w podanej kolejności. Po każdej aktywacji mamy wypisać ile jest wierzchołków v, którym zwiększyła się wartość $\max_{u \in X}(w_v \oplus w_u)$, gdzie X to zbiór wierzchołków, do których można dojść z v korzystając jedynie z aktywowanych krawędzi.

Dodajemy do grafu krawędzie w podanej kolejności, utrzymując aktualne spójne składowe w strukturze zbiorów rozłącznych. Dla każdej spójnej utrzymujemy również drzewo trie zawierające wagi jej wierzchołków.

Po dodaniu krawędzi łączymy spójne (w tym drzewa trie) metodą *mniejszy do większego*.

Problem F - Turniej szachowy

Łącząc dwie spójne A i B, musimy zliczyć liczbę wierzchołków $v \in A$, takich że $\max_{b \in B}(w_v \oplus w_b) > \max_{a \in A}(w_v \oplus w_a)$ i analogicznie dla $v \in B$.

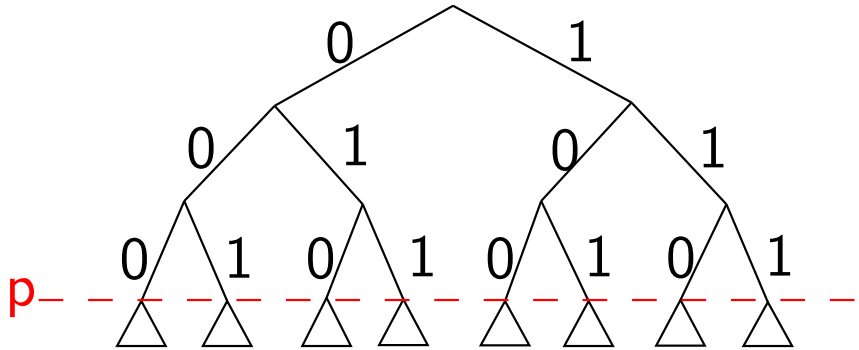
Załóżmy, że $|A| < |B|$.

Zgodnie z zasadą *mniejszy do większego* możemy się przeiterować po wszystkich wierzchołkach z $v \in A$ i w trie B znaleźć wagę x, która maksymalizuje $(w_v \oplus x)$.

Po wierzchołkach $v \in B$ nie możemy się przeiterować, więc spróbujmy dla każdego wierzchołka $a \in A$ szybko znaleźć wszystkie $v \in B$, którym zwiększa się wartość. Okazuje się, że nie będzie ich wiele.

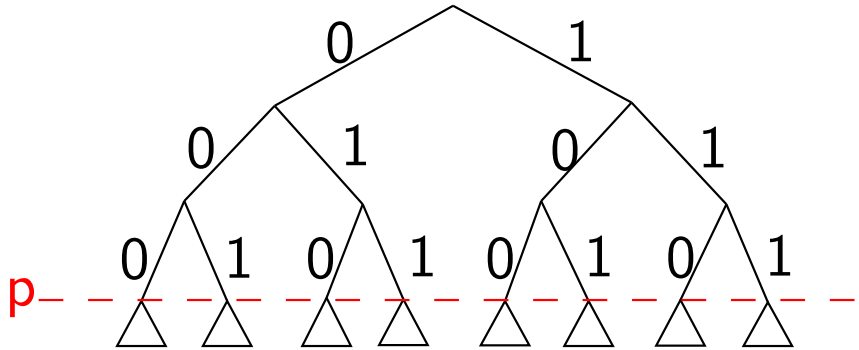
Problem F - Turniej szachowy

Dla drzewa trie B, przez p oznaczmy maksymalny poziom, na którym mamy wszystkie wierzchołki.



Problem F - Turniej szachowy

Dla drzewa trie B, przez p oznaczmy maksymalny poziom, na którym mamy wszystkie wierzchołki.

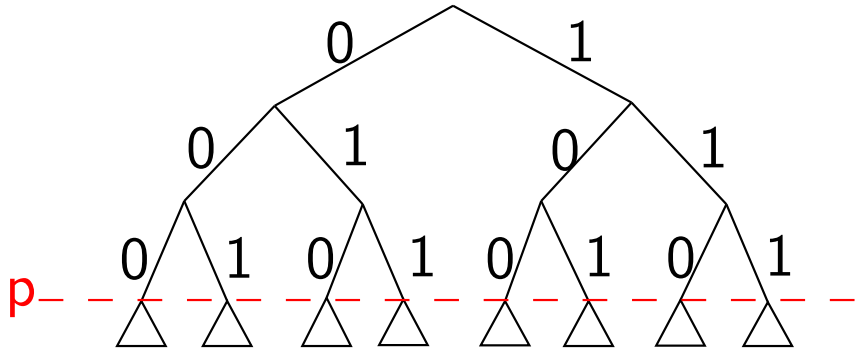


Obserwacja 1

Ponieważ wagi są losowe, w poddrzewie każdego wierzchołka na poziomie p oczekiwana liczba elementów wynosi $O(\log |B|)$.

Problem F - Turniej szachowy

Dla drzewa trie B, przez p oznaczmy maksymalny poziom, na którym mamy wszystkie wierzchołki.



Obserwacja 1

Ponieważ wagi są losowe, w poddrzewie każdego wierzchołka na poziomie p oczekiwana liczba elementów wynosi $O(\log |B|)$.

Obserwacja 2

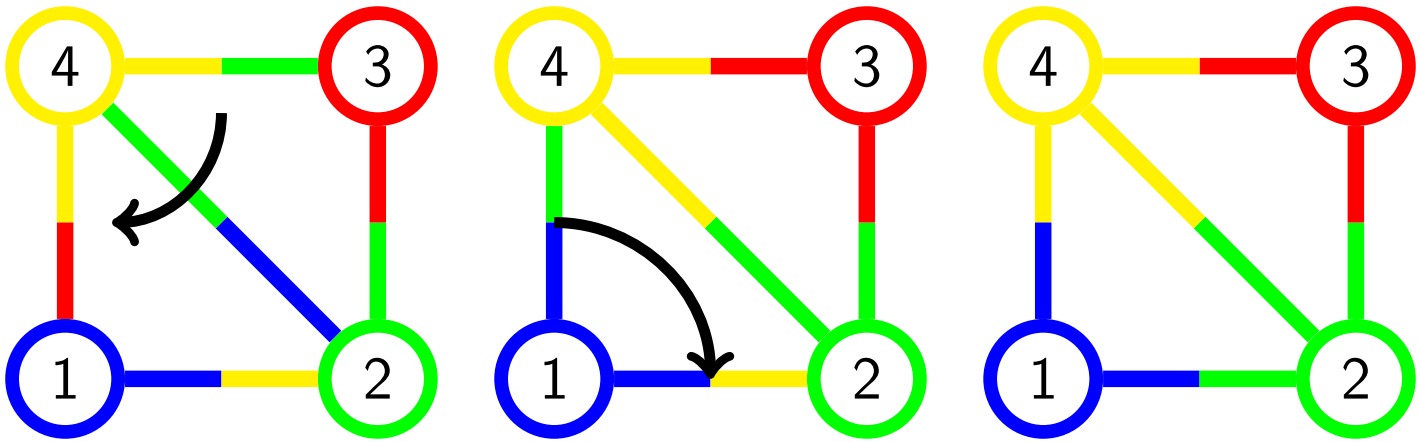
Każdy wierzchołek $a \in A$ może zwiększyć wartości wyłącznie w jednym (losowym) spośród tych poddrzew. Łatwo je wyznaczyć negując w_a.

Problem G - Magnetyczna zabawka

Problem G - Magnetyczna zabawka

Zadanie

Dany jest graf z kolorowymi wierzchołkami i krawędziami o końcach w różnych kolorach.
Możemy wykonywać operację *obrotu wierzchołka, razem z przylegającymi do niego krawędziami*.
Zadanie: określić czy da się za pomocą tych operacji doprowadzić graf do pozycji *ułożonej*.



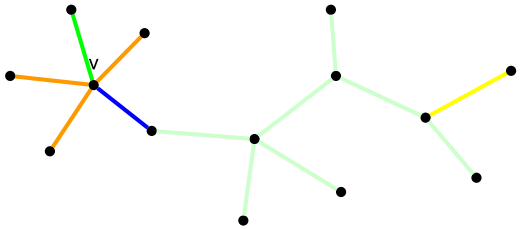
Problem G - Magnetyczna zabawka

Nie wszystkie ustawienia i skierowania krawędzi da się otrzymać za pomocą powyższych ruchów.

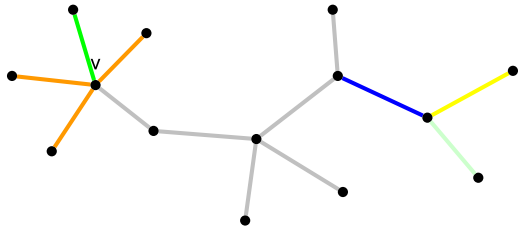
1. Sprawdzanie skierowania krawędzi
   1. Graf dwudzielny
      Jeżeli któraś z krawędzi jest źle obrócona, należy wypisać `NIE`.
   2. Graf niedwudzielny
      Trzeba sprawdzić "parzystość zbioru źle obróconych krawędzi"
2. Sprawdzanie położenia krawędzi
   1. Graf jest gwiazdą (drzewem o $n-1$ liściach)
   2. Graf zawierający tylko wierzchołki o nieparzystych stopniach
      Jeśli układ krawędzi jest permutacją nieparzystą, wypisz `NIE`.
   3. Graf zawierający wierzchołek o parzystym stopniu
      Wszystkie ustawienia są możliwe.

Problem G - Magnetyczna zabawka

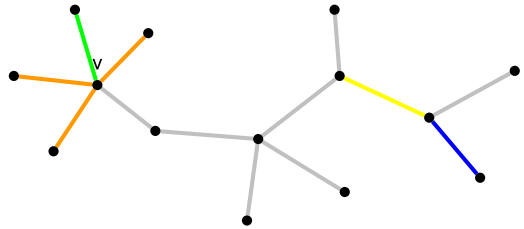
Algorytm zamieniający miejscami 3 krawędzie (potrzebny tylko do dowodu poprawności):



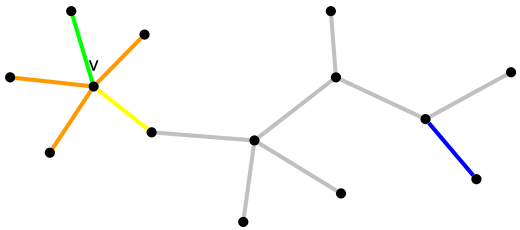
(a)



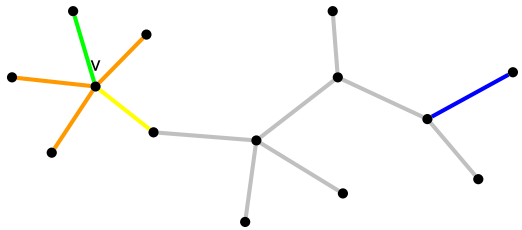
(b)



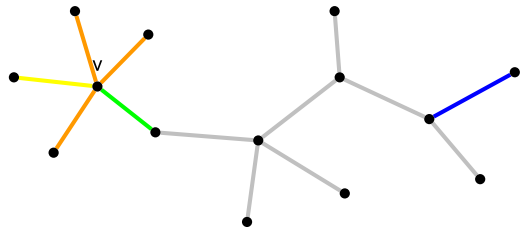
(c)



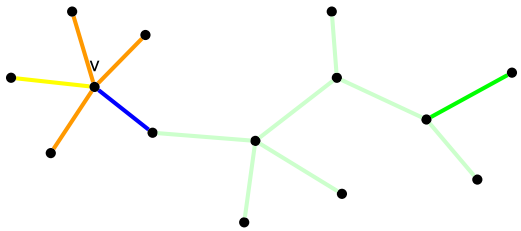
(d)



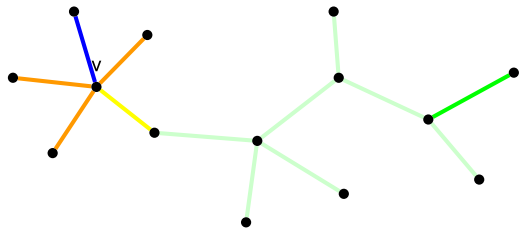
(e)



(f)



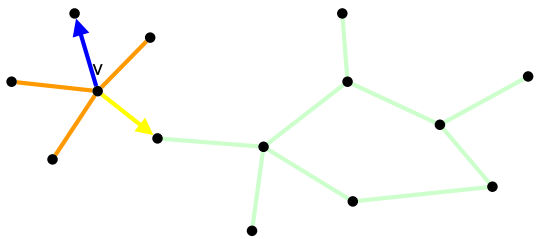
(g)



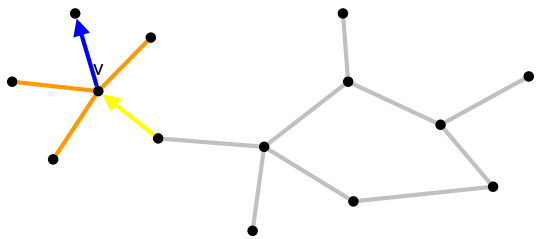
(h)

Problem G - Magnetyczna zabawka

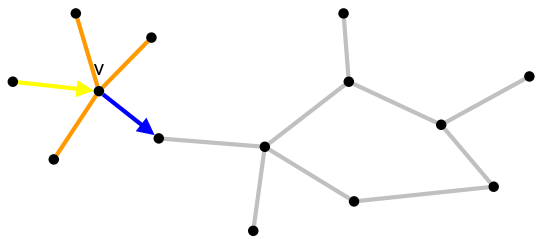
Algorytm obracający 2 krawędzie (potrzebny tylko do dowodu poprawności):



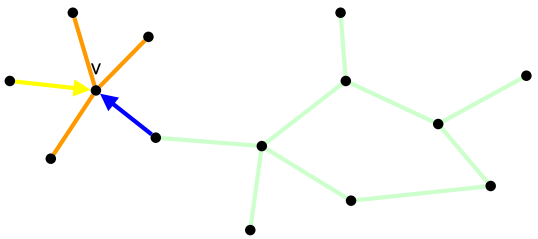
(a)



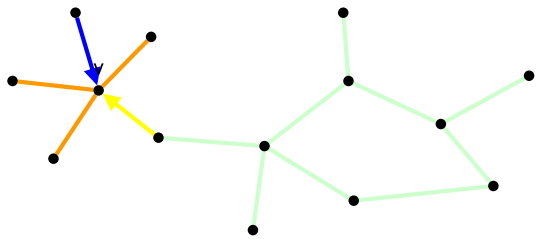
(b)



(c)



(d)



(e)

Problem G - Magnetyczna zabawka

Rozwiązanie wymagało:

- sprawdzenia czy graf jest gwiazdą (oraz sprawdzenia czy dwie gwiazdy mają krawędzie w tej samej kolejności),
- sprawdzenia dwudzielności grafu (ze sprawdzeniem czy konkretne wierzchołki leżą po tej samej stronie),
- sprawdzenia czy liczba krawędzi o niepokrywającym się skierowaniu jest nieparzysta,
- sprawdzenia czy graf zawiera wierzchołek o stopniu parzystym,
- sprawdzenia czy permutacja krawędzi grafu jest parzysta.

Problem H - (Nie)uczciwy remik

Problem H - (Nie)uczciwy remik

Zadanie

Dany jest ciąg liczb $A_1, A_2, \ldots, A_n$. Mamy wypisać najmniejszą możliwą sumę elementów ciągu po wykonaniu pewnej liczby operacji zamiany dwóch sąsiednich elementów A_i, A_{i+1} na jeden element równy $A_i \oplus A_{i+1}$.

Problem H - (Nie)uczciwy remik

Zadanie

Dany jest ciąg liczb $A_1, A_2, \ldots, A_n$. Mamy wypisać najmniejszą możliwą sumę elementów ciągu po wykonaniu pewnej liczby operacji zamiany dwóch sąsiednich elementów A_i, A_{i+1} na jeden element równy $A_i \oplus A_{i+1}$.

Obserwacja

Dla dowolnych nieujemnych x, y zachodzi $x \oplus y \leqslant x + y$.

Problem H - (Nie)uczciwy remik

Zadanie

Dany jest ciąg liczb $A_1, A_2, \ldots, A_n$. Mamy wypisać najmniejszą możliwą sumę elementów ciągu po wykonaniu pewnej liczby operacji zamiany dwóch sąsiednich elementów A_i, A_{i+1} na jeden element równy $A_i \oplus A_{i+1}$.

Obserwacja

Dla dowolnych nieujemnych x, y zachodzi $x \oplus y \leqslant x + y$.

Zamieniając dwie karty nigdy nie pogorszymy wyniku. Wobec tego istnieje optymalne rozwiązanie, w którym Jaś zostaje z tylko jedną kartą.

Ponieważ działanie $\oplus$ jest łączne, wartość ostatniej karty jest równa $A_1 \oplus A_2 \oplus \cdots \oplus A_n$, niezależnie od operacji, które wykonamy.

Problem I - Akrobatyka

Problem I - Akrobatyka

Zadanie

Należy wybrać kolejność odwiedzania punktów $p_1, p_2, \ldots, p_N$ znajdujących się na prostej tak, aby sumarycznie przebyć jak najdłuższą drogę.

Problem I - Akrobatyka

Zadanie

Należy wybrać kolejność odwiedzania punktów $p_1, p_2, \ldots, p_N$ znajdujących się na prostej tak, aby sumarycznie przebyć jak najdłuższą drogę.

Ile co najwyżej razy można przejść odcinek pomiędzy punktami p_i oraz p_{i+1}?

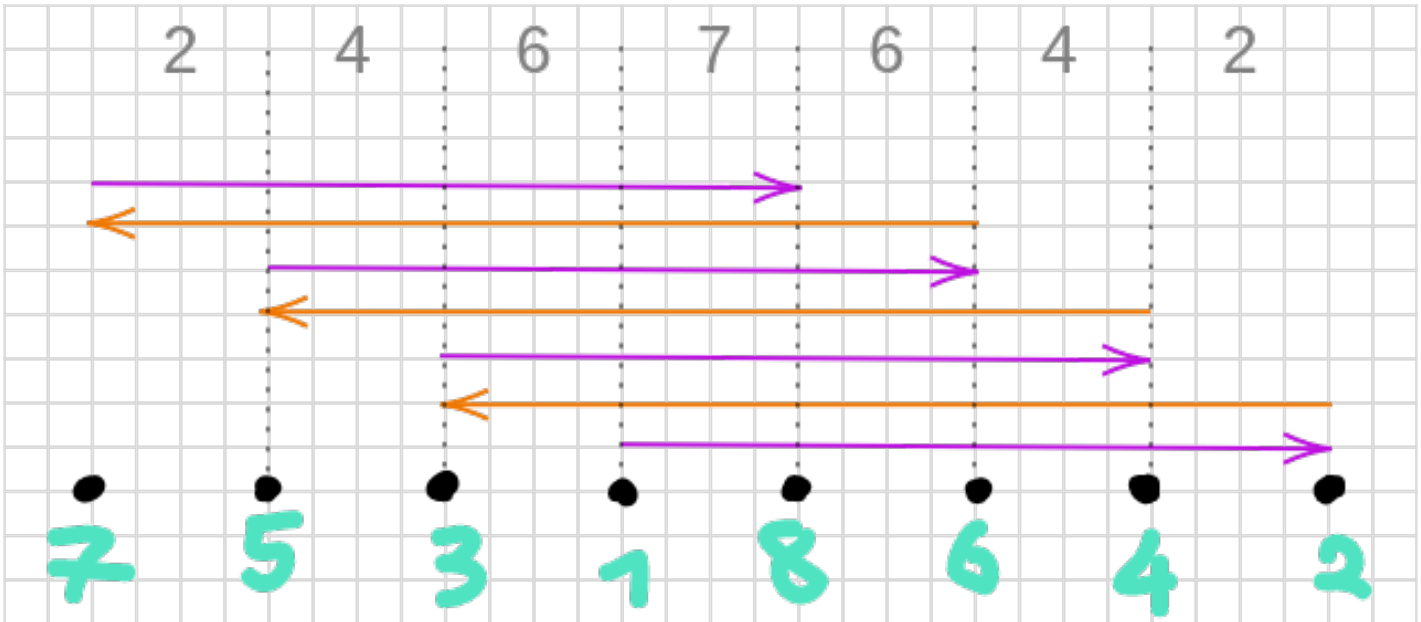
Obserwacja

Po lewej stronie od odcinka znajduje się i punktów, a po prawej $N - i$. Możemy więc przez niego przejść maksymalnie $2 \min(i, N - i)$ razy.

Problem I - Akrobatyka

Rozważmy dwa przypadki:

- N jest parzyste – wtedy środkową krawędź przejdziemy maksymalnie $N-1$ razy (tyle skoków wykonamy). Optymalną ścieżką, która przejdzie każdy odcinek maksymalną możliwą liczbę razy, jest: $p_{N/2},\ p_N,\ p_{N/2-1},\ p_{N-1},\ \ldots,\ p_1, p_{N/2+1}$.



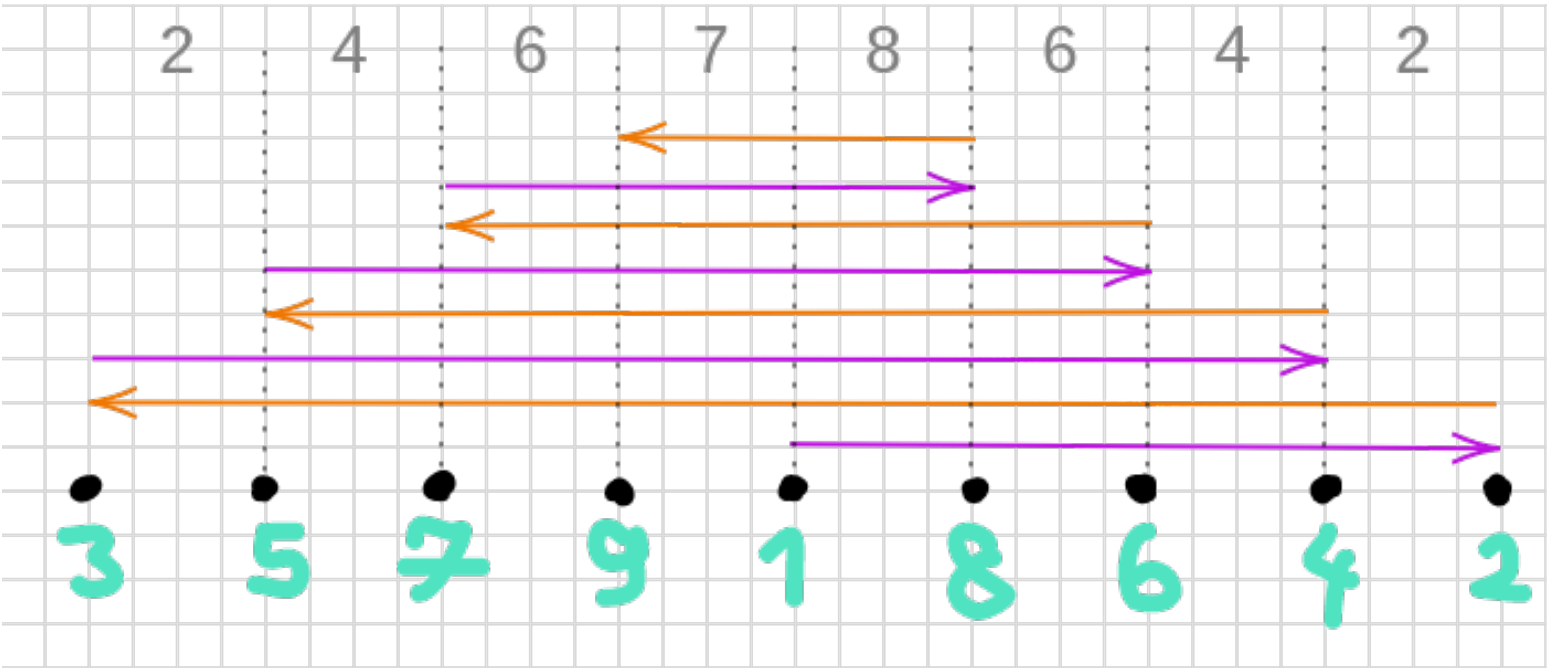
Problem I - Akrobatyka

Rozważmy dwa przypadki:

- N jest nieparzyste – wtedy jedną ze środkowych krawędzi przejdziemy maksymalnie $N-2$ razy. Optymalną ścieżką jest wtedy

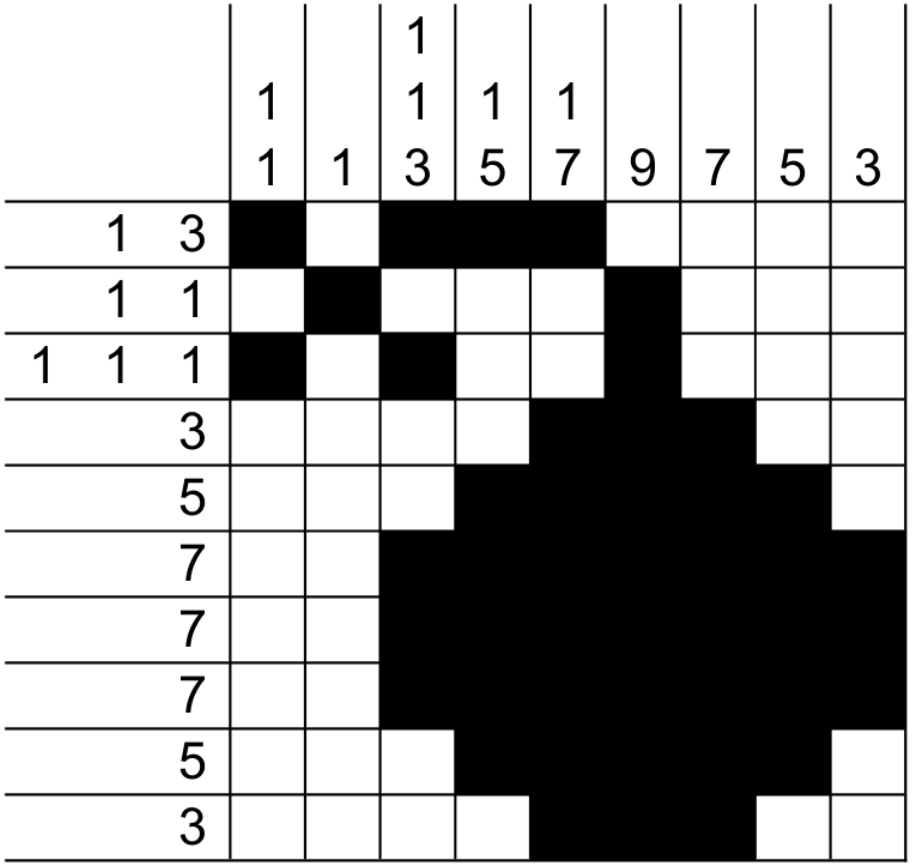
$$p_{(N+1)/2},\ p_N,\ p_1,\ p_{N-1},\ p_2,\ \ldots,\ ,p_{(N-1)/2}$$

lub jej lustrzane odbicie (w zależności, który z odcinków środkowych jest dłuższy).



Problem J - Obrazki logiczne

Problem J - Obrazki logiczne

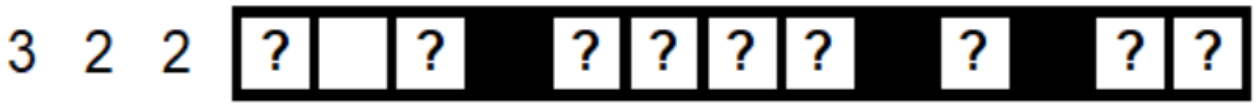


Problem J - Obrazki logiczne

Zadanie

Dla danego jednego, częściowo rozwiązanego, wiersza obrazka logicznego, uzupełnij wszystkie możliwe pola.

Przykład:

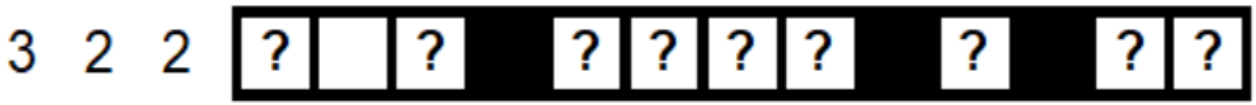


Problem J - Obrazki logiczne

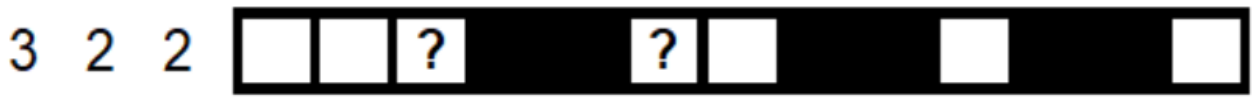
Zadanie

Dla danego jednego, częściowo rozwiązanego, wiersza obrazka logicznego, uzupełnij wszystkie możliwe pola.

Przykład:



Odpowiedź:



Problem J - Obrazki logiczne

Pomysł: sprawdźmy kolejno dla każdego pola czy może ono być zamalowane, albo pozostać niezamalowane.

Problem J - Obrazki logiczne

Pomysł: sprawdźmy kolejno dla każdego pola czy może ono być zamalowane, albo pozostać niezamalowane.

Jeżeli któraś z opcji nie będzie możliwa, to musimy wybrać drugą opcję.

Problem J - Obrazki logiczne

Pomysł: sprawdźmy kolejno dla każdego pola czy może ono być zamalowane, albo pozostać niezamalowane.

Jeżeli któraś z opcji nie będzie możliwa, to musimy wybrać drugą opcję.

Tylko jak sprawdzić czy dane zamalowanie jest jeszcze poprawne...

Problem J - Obrazki logiczne

Pomysł: sprawdźmy kolejno dla każdego pola czy może ono być zamalowane, albo pozostać niezamalowane.

Jeżeli któraś z opcji nie będzie możliwa, to musimy wybrać drugą opcję.

Tylko jak sprawdzić czy dane zamalowanie jest jeszcze poprawne... programowaniem dynamicznym!

Problem J - Obrazki logiczne

Niech stan $DP[i][j]$ oznacza czy jesteśmy w stanie zamalować pola od 1 do i tak, aby na obrazku zamalowane były dokładnie odcinki $d_1, \ldots, d_j$, w tejże kolejności.

Problem J - Obrazki logiczne

Niech stan $DP[i][j]$ oznacza czy jesteśmy w stanie zamalować pola od 1 do i tak, aby na obrazku zamalowane były dokładnie odcinki $d_1, \ldots, d_j$, w tejże kolejności.

Jeśli chcemy zostawić pole i niezamalowane, to:

- musi ono mieć możliwość bycia niezamalowanym,
- musi być spełnione $DP[i-1][j]$.

Problem J - Obrazki logiczne

Niech stan $DP[i][j]$ oznacza czy jesteśmy w stanie zamalować pola od 1 do i tak, aby na obrazku zamalowane były dokładnie odcinki $d_1, \ldots, d_j$, w tejże kolejności.

Jeśli chcemy zostawić pole i niezamalowane, to:

- musi ono mieć możliwość bycia niezamalowanym,
- musi być spełnione $DP[i-1][j]$.

Jeśli chcemy aby pole i zostało zamalowane, to :

- końcowy fragment o długości d_j (pola od $i - d_j + 1$ do i) musi mieć możliwość bycia zamalowanym,
- $i - d_j$ musi zostać puste,
- musi być spełnione $DP[i - d_j - 1][j - 1]$.

Problem J - Obrazki logiczne

Niech stan $DP[i][j]$ oznacza czy jesteśmy w stanie zamalować pola od 1 do i tak, aby na obrazku zamalowane były dokładnie odcinki $d_1, \ldots, d_j$, w tejże kolejności.

Jeśli chcemy zostawić pole i niezamalowane, to:

- musi ono mieć możliwość bycia niezamalowanym,
- musi być spełnione $DP[i-1][j]$.

Jeśli chcemy aby pole i zostało zamalowane, to :

- końcowy fragment o długości d_j (pola od $i - d_j + 1$ do i) musi mieć możliwość bycia zamalowanym,
- $i - d_j$ musi zostać puste,
- musi być spełnione $DP[i - d_j - 1][j-1]$.

To rozwiązanie dobrze zaimplementowane ma złożoność $O(N^2 \cdot K)$, co było akceptowane, ale można je zoptymalizować do $O(N \cdot K)$.

Problem K - Drewno, glina i owce

Problem K - Drewno, glina i owce

Zadanie

Dostajemy graf meduzę – graf z co najwyżej jednym cyklem. Każdy wierzchołek grafu jest pokolorowany na jeden z 3 kolorów. Pytamy się, na ile sposobów można podzielić graf na spójne fragmenty, tak żeby każdy fragment zawierał wierzchołek każdego z 3 kolorów.

Problem K - Drewno, glina i owce

Zadanie

Dostajemy graf meduzę – graf z co najwyżej jednym cyklem. Każdy wierzchołek grafu jest pokolorowany na jeden z 3 kolorów. Pytamy się, na ile sposobów można podzielić graf na spójne fragmenty, tak żeby każdy fragment zawierał wierzchołek każdego z 3 kolorów.

Wersja prostsza – drzewo

W takiej wersji problem można rozwiązać standardowym programowaniem dynamicznym: dla każdego wierzchołka v i maski kolorów m obliczamy, na ile sposobów można podzielić poddrzewo v tak, że wszystkie fragmenty oprócz najwyższego zawierają wszystkie kolory, a najwyższy fragment (ten z v), zawiera kolory z m.

Problem K - Drewno, glina i owce

1. Znajdujemy cykl w grafie. Niech c oznacza jego długość.

Problem K - Drewno, glina i owce

1. Znajdujemy cykl w grafie. Niech c oznacza jego długość.
2. Dla każdego wierzchołka na cyklu v i dla każdej maski kolorów m liczymy wynik $\mathrm{dp}_{\mathrm{tree}}[v][m]$ w grafie ograniczonym do poddrzewa v.

Problem K - Drewno, glina i owce

1. Znajdujemy cykl w grafie. Niech c oznacza jego długość.
2. Dla każdego wierzchołka na cyklu v i dla każdej maski kolorów m liczymy wynik $\mathrm{dp}_{\mathrm{tree}}[v][m]$ w grafie ograniczonym do poddrzewa v.
3. Wybierz dowolny wierzchołek na cyklu, nazwijmy go u, a u_p to jego poprzednik na cyklu

Problem K - Drewno, glina i owce

1. Znajdujemy cykl w grafie. Niech c oznacza jego długość.
2. Dla każdego wierzchołka na cyklu v i dla każdej maski kolorów m liczymy wynik $\mathrm{dp}_{\mathrm{tree}}[v][m]$ w grafie ograniczonym do poddrzewa v.
3. Wybierz dowolny wierzchołek na cyklu, nazwijmy go u, a u_p to jego poprzednik na cyklu
4. Dla każdego kolejnego wierzchołka na cyklu v liczymy $\mathrm{dp}_{\mathrm{cycle}}[v][m_p][m_o][k]$, które jest równe liczbie podziałów fragmentu cyklu od u do v na k fragmentów, z czego wewnętrzne fragmenty zawierają wszystkie kolory, pierwszy zawiera kolory z maski m_p, a ostatni z maski m_o.

Problem K - Drewno, glina i owce

1. Znajdujemy cykl w grafie. Niech c oznacza jego długość.
2. Dla każdego wierzchołka na cyklu v i dla każdej maski kolorów m liczymy wynik $\mathrm{dp}_{\mathrm{tree}}[v][m]$ w grafie ograniczonym do poddrzewa v.
3. Wybierz dowolny wierzchołek na cyklu, nazwijmy go u, a u_p to jego poprzednik na cyklu
4. Dla każdego kolejnego wierzchołka na cyklu v liczymy $\mathrm{dp}_{\mathrm{cycle}}[v][m_p][m_o][k]$, które jest równe liczbie podziałów fragmentu cyklu od u do v na k fragmentów, z czego wewnętrzne fragmenty zawierają wszystkie kolory, pierwszy zawiera kolory z maski m_p, a ostatni z maski m_o.
5. Dla $k = 1$ wartość m_o nie ma znaczenia, możemy uznać, że wtedy $m_o = 000_2$

Problem K - Drewno, glina i owce

Wynik to suma:

- bez podziału: $\mathrm{dp}_{\mathrm{cycle}}[u_p][111_2][000_2][1]$
- dwa fragmenty: $\mathrm{dp}_{\mathrm{cycle}}[u_p][111_2][111_2][2]$
- podziału na co najmniej 3 fragmenty, w którym łączymy pierwszy i ostatni fragment: $\sum_{k=3}^{c} \sum_{m_p | m_k = 111_2} \mathrm{dp}_{\mathrm{cycle}}[u_p][m_p][m_o][k]$
- podziału na co najmniej 3 fragmenty, w którym pierwszy i ostatni fragment pozostają rozdzielone: $\sum_{k=3}^{c} \mathrm{dp}_{\mathrm{cycle}}[u_p][111_2][111_2][k]$

Problem K - Drewno, glina i owce

Wynik to suma:

- bez podziału: $\mathrm{dp}_{\mathrm{cycle}}[u_p][111_2][000_2][1]$
- dwa fragmenty: $\mathrm{dp}_{\mathrm{cycle}}[u_p][111_2][111_2][2]$
- podziału na co najmniej 3 fragmenty, w którym łączymy pierwszy i ostatni fragment: $\sum_{k=3}^{c} \sum_{m_p | m_k = 111_2} \mathrm{dp}_{\mathrm{cycle}}[u_p][m_p][m_o][k]$
- podziału na co najmniej 3 fragmenty, w którym pierwszy i ostatni fragment pozostają rozdzielone: $\sum_{k=3}^{c} \mathrm{dp}_{\mathrm{cycle}}[u_p][111_2][111_2][k]$

Takie rozwiązanie jest jednak za wolne. Wystarczy za to liczyć $\mathrm{dp}_{\mathrm{cycle}}$ dla trzech możliwych typów wartości k: $\{1, 2, \geqslant 3\}$.

Problem L - Pora roku

Problem L - Pora roku

Zadanie

Dla podanej daty należy określić jaka pora roku obowiązuje tego dnia.

Problem L - Pora roku

Zadanie

Dla podanej daty należy określić jaka pora roku obowiązuje tego dnia.

Najłatwiej to zrobić pisząc jedną pomocniczą funkcję, która porównuje dwie daty i stwierdza, która z nich jest wcześniej. Taka funkcja może operować na napisach, ale jeszcze prościej jest, gdy operuje ona na liczbach. Należy tylko pamiętać o zamianie wczytanych dat na liczby.

Następnie za pomocą tak napisanej funkcji i instrukcji warunkowych wystarczy wypisać nazwę odpowiedniej pory roku.

Problem M - Zaminowane kładki

Problem M - Zaminowane kładki

Zadanie

Na wejściu dostajemy permutację powstałą w wyniku powtórzenia specjalnej operacji zamian miejscami dwóch elementów na zadanych pozycjach. Musimy wrócić do permutacji identycznościowej również korzystając ze specjalnej operacji, jednak nie możemy powtarzać dwukrotnie zamian elementów na tych samych pozycjach (liczą się także zamiany wykonane do uzyskania sytuacji z wejścia). Dodatkowo mamy zapewnione, że dwa elementy nigdy nie zostały ruszone.

Problem M - Zaminowane kładki

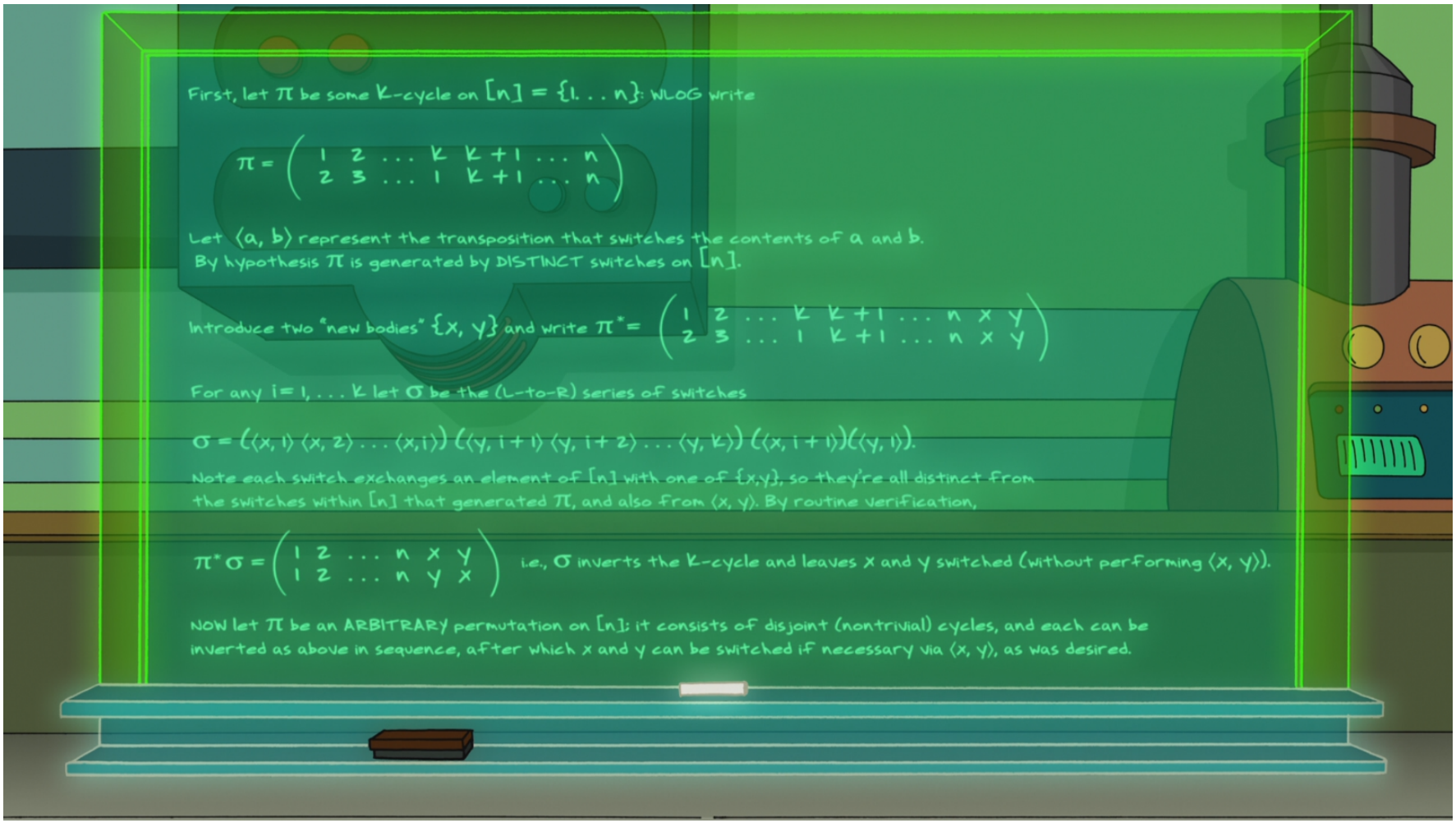
Zadanie

Na wejściu dostajemy permutację powstałą w wyniku powtórzenia specjalnej operacji zamian miejscami dwóch elementów na zadanych pozycjach. Musimy wrócić do permutacji identycznościowej również korzystając ze specjalnej operacji, jednak nie możemy powtarzać dwukrotnie zamian elementów na tych samych pozycjach (liczą się także zamiany wykonane do uzyskania sytuacji z wejścia). Dodatkowo mamy zapewnione, że dwa elementy nigdy nie zostały ruszone.

Przy założeniu, że dwa elementy były wcześniej nieruszone, **zawsze** da się znaleźć ciąg zamian, który przywraca początkowe ułożenie.

Tzw. „Twierdzenie Futuramy” autorstwa Kena Keelera, odcinek „Prisoner of Benda”.

Problem M - Zaminowane kładki



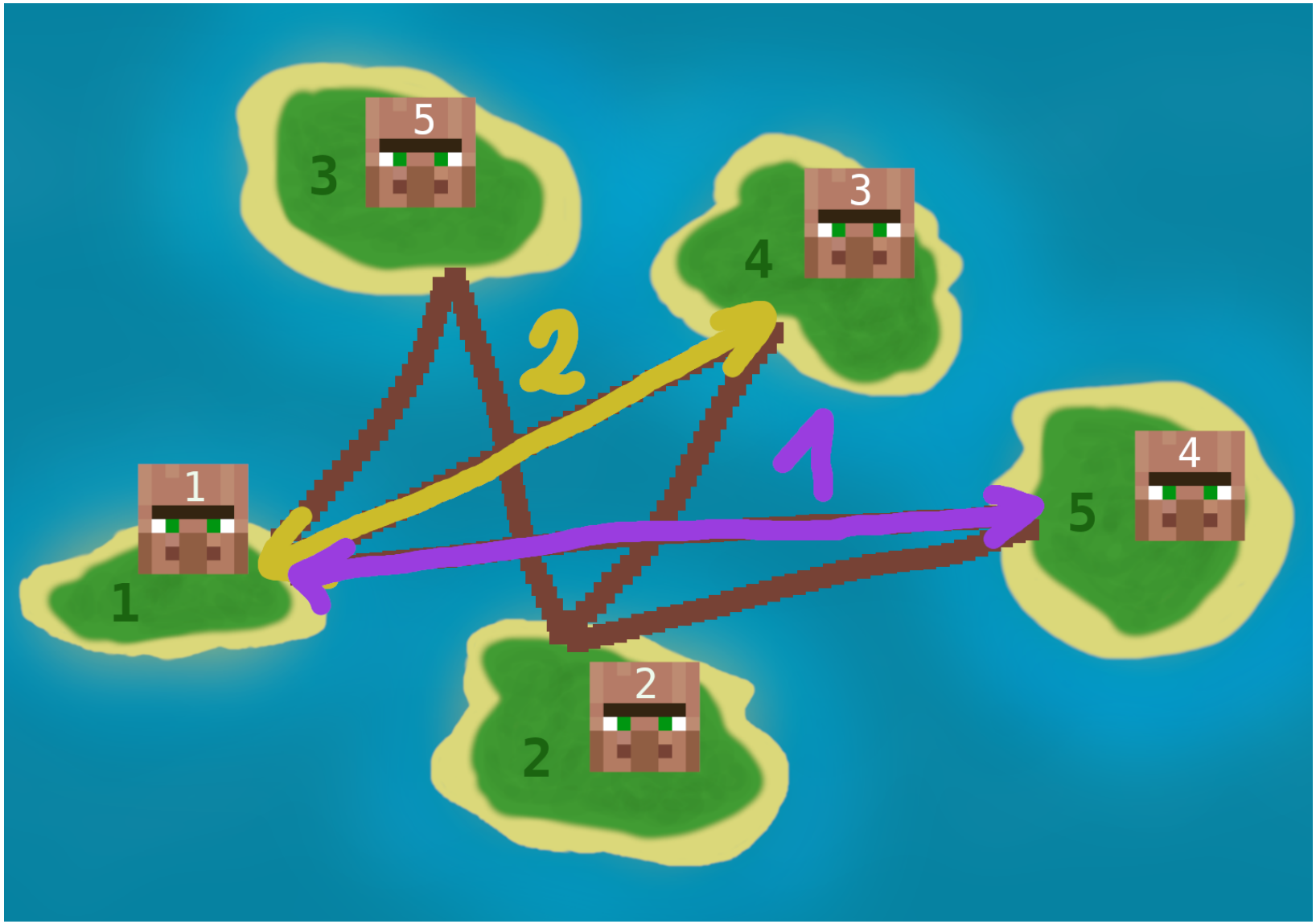
Problem M - Zaminowane kładki

Zastanówmy się nad prostą instancją tego problemu.

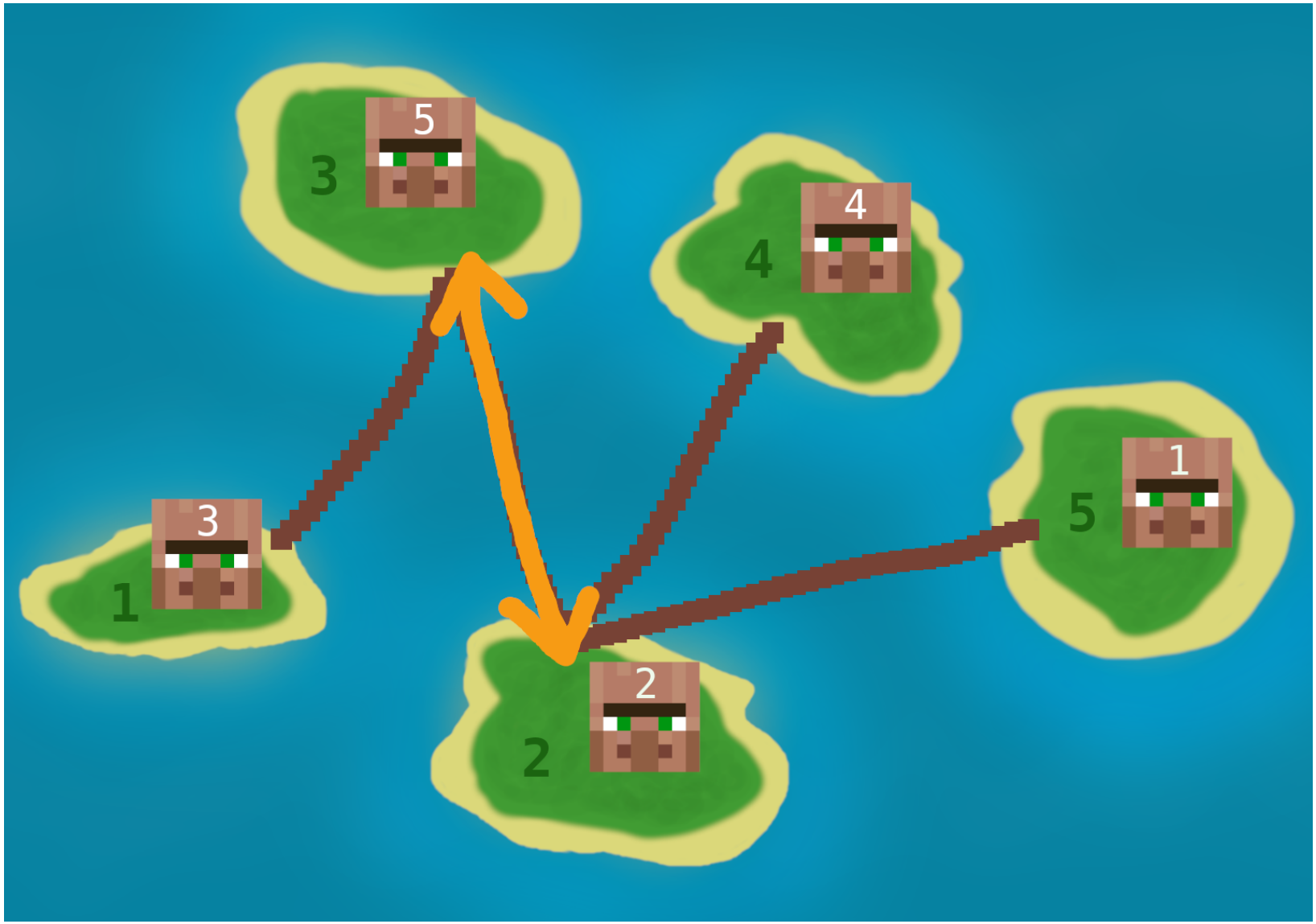
Wieśniacy o numerach 1 i 2 nigdy nie zostali ruszeni, a 3, 4 i 5 tworzą cykl (czyli 3 powinien być na wyspie z 4, 4 na wyspie z 5, a 5 na wyspie z 3.

Dla uproszczenia, na rysunkach pomijamy istniejący most pomiędzy ich wyspami. Jako pierwsze wykonujemy zamiany oznaczone na rysunku numerami 1 i 2, w takiej kolejności.

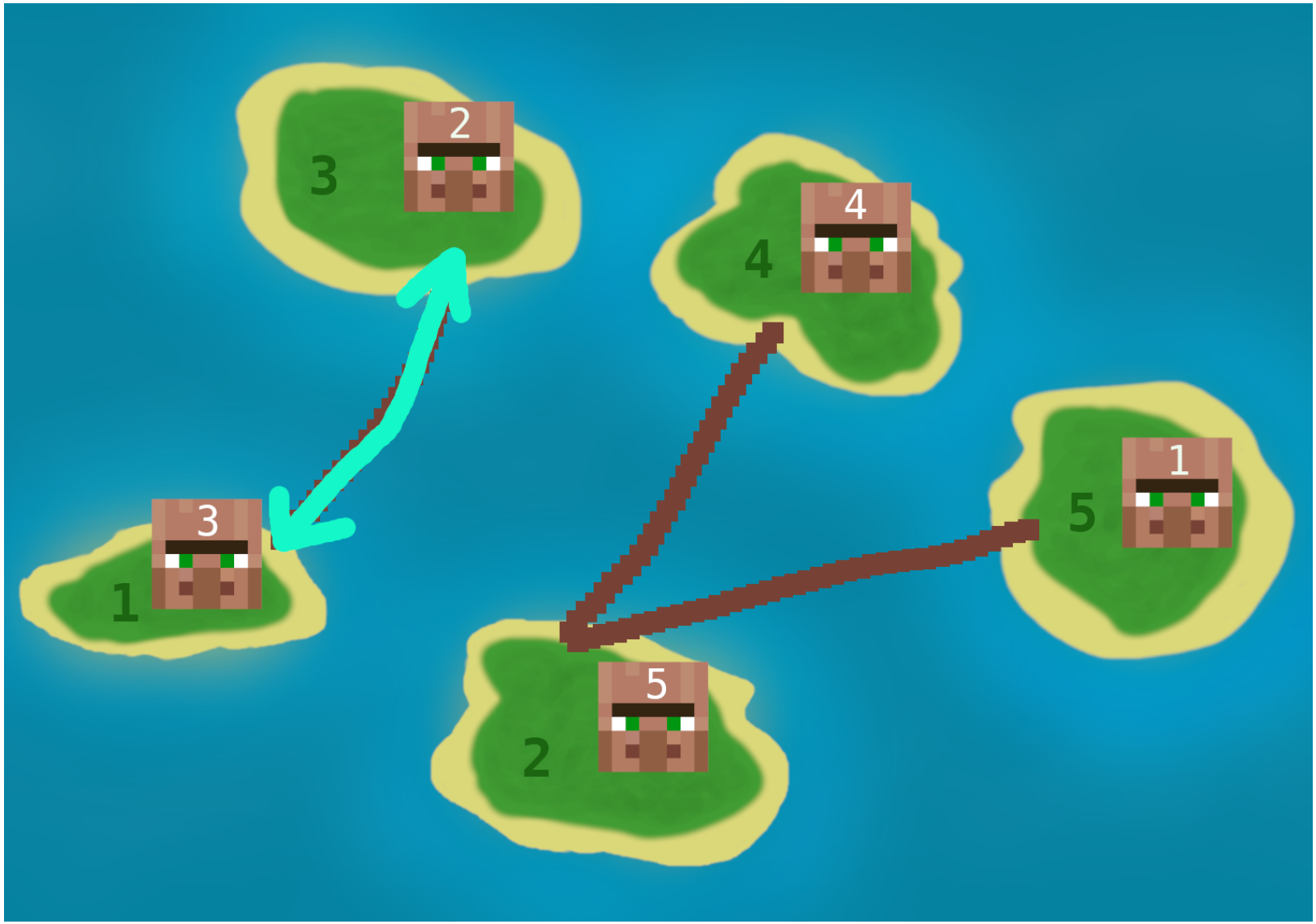
Problem M - Zaminowane kładki



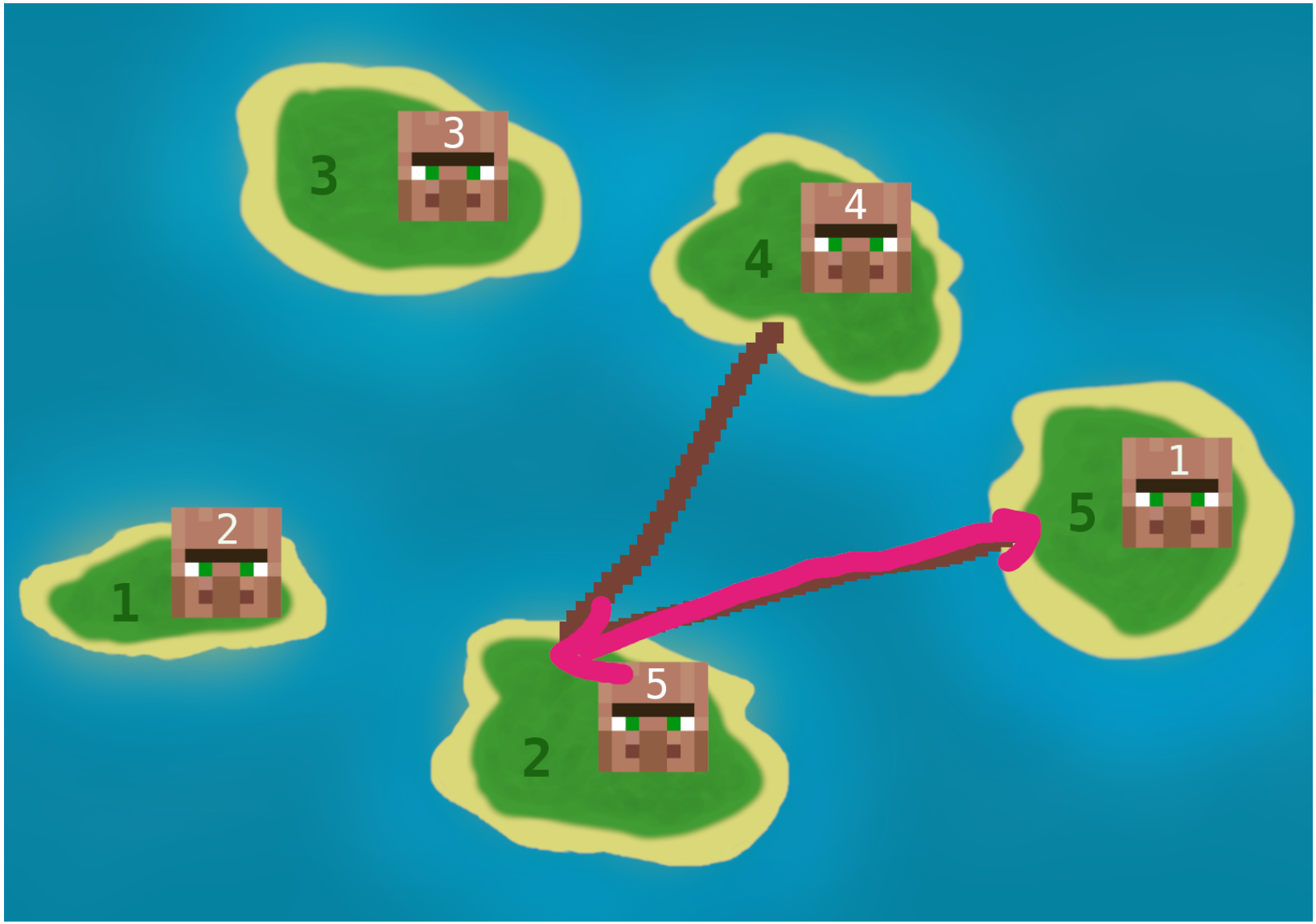
Problem M - Zaminowane kładki



Problem M - Zaminowane kładki



Problem M - Zaminowane kładki



Problem M - Zaminowane kładki

Udało nam się poprawić cykl trzyelementowy używając jedynie zamian z nieruszonymi wcześniej wyspami. Wieśniacy o numerach 1 i 2 są co prawda zamienieni miejscami, ale nigdy nie korzystaliśmy z mostu pomiędzy ich wyspami.

Dłuższy cykl

Zauważmy, że powyższe rozwiązanie da się naturalnie rozszerzyć dla dłuższych cykli. Wystarczy jedynie zamieniać po kolei elementy z cyklu z elementem na pozycji 1. Jeśli zostaną tylko dwa elementy do naprawy, wykonujemy zamiany z rysunków.

Więcej cykli

Permutacje składające się z większej liczby cykli również nie są problemem! Wystarczy poprawić osobno każdy z cykli, a na koniec, jeśli jest taka potrzeba, zamienić ze sobą wieśniaków z wysp 1 i 2.

Problem N - Era Mitologii

Problem N - Era Mitologii

Zadanie

Gramy w grę, w której zaczynamy z P punktami. Naszym celem jest zdobycie K punktów. W każdej sekundzie możemy zatrudnić robotników. Jeden robotnik kosztuje nas C punktów i po każdej sekundzie od momentu zatrudnienia przynosi nam V punktów. Naszym zadaniem jest wypisanie minimalnej liczby sekund, po których jesteśmy w stanie wygrać.

Problem N - Era Mitologii

Zadanie

Gramy w grę, w której zaczynamy z P punktami. Naszym celem jest zdobycie K punktów. W każdej sekundzie możemy zatrudnić robotników. Jeden robotnik kosztuje nas C punktów i po każdej sekundzie od momentu zatrudnienia przynosi nam V punktów. Naszym zadaniem jest wypisanie minimalnej liczby sekund, po których jesteśmy w stanie wygrać.

Obserwacja 1

Jeśli planujemy kupić jakiegoś robotnika, to opłaca nam się to zrobić jak najwcześniej, gdyż zawsze kosztuje on tyle samo, a im wcześniej go kupimy, tym więcej zysku może on przynieść.

Obserwacja 2

Czas zbierania pożywienia nie będzie większy niż 1000.

Problem N - Era Mitologii

Obserwacja 3

Nigdy nie będziemy potrzebować wiecej niż 1000 robotników.

Rozwiązanie zadania polega na sprawdzeniu po jakim czasie jesteśmy w stanie zebrać odpowiednią ilość pożywienia, zakładając, że zatrudnimy dokładnie $x \in [1, 1000]$ robotników, a następnie wypisaniu najmniejszego z tych czasów.

Dla ustalonego x iterujemy się po kolejnych sekundach od 1 do 1000, utrzymując licznik punktów L_p oraz licznik robotników L_r. W każdej sekundzie najpierw zwiększamy L_p o $L_r \cdot V$, a następnie zatrudniamy tylu robotników ilu możemy, nie przekraczając przy tym x.

Jeżeli nie zdobędziemy K punktów do 1000 sekundy, to nie musimy kontynuować symulacji, ponieważ i tak dla innego x uda nam się uzyskać lepszy czas.

Problem N - Era Mitologii

Zadanie da się też rozwiązać w złożoności liniowej od wielkości liczb na wejściu.

Problem O - Log Roll

Problem O - Log Roll

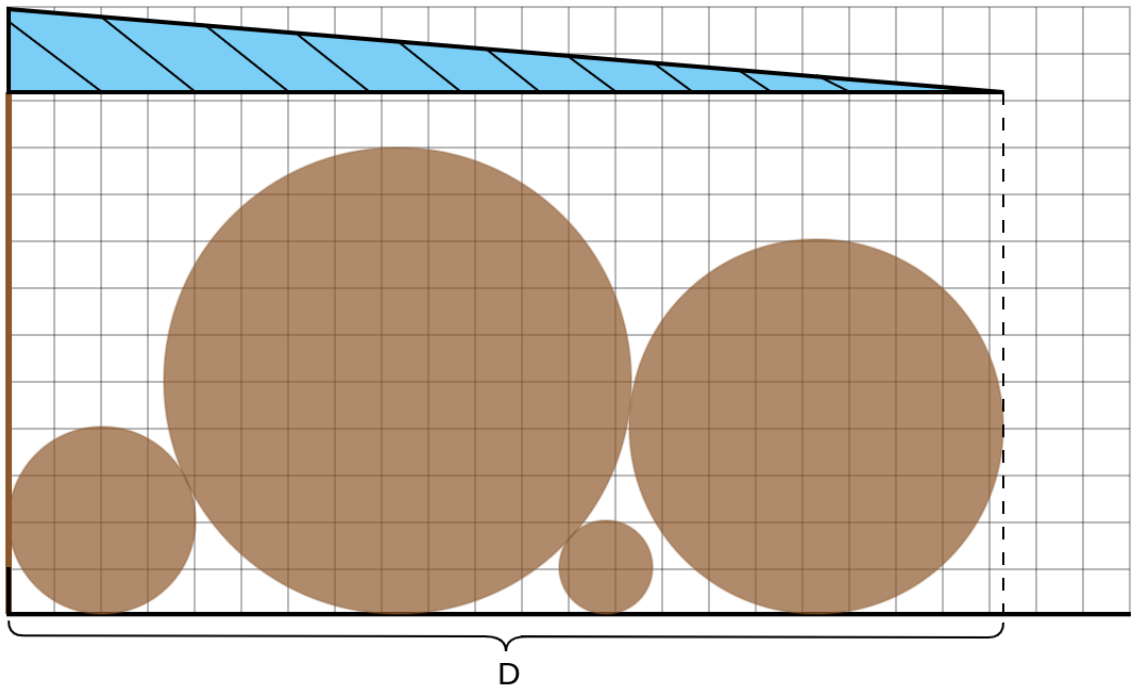
Zadanie

Chcemy ułożyć na prostej N kół (o różnych promieniach) w danej kolejności, tak by każde kolejne koło przylegało ściśle do jednego z poprzednich kół. Podaj szerokość figury, która jest na końcu sumą wszystkich kół.

Problem O - Log Roll

Zadanie

Chcemy ułożyć na prostej N kół (o różnych promieniach) w danej kolejności, tak by każde kolejne koło przylegało ściśle do jednego z poprzednich kół. Podaj szerokość figury, która jest na końcu sumą wszystkich kół.



Obserwacja

Ponieważ limity w zadaniu nie są za duże ($N \leqslant 3000$), będziemy szukali rozwiązania działającego w złożoności $\mathcal{O}(N^2)$.

Obserwacja

Ponieważ limity w zadaniu nie są za duże ($N \leqslant 3000$), będziemy szukali rozwiązania działającego w złożoności $\mathcal{O}(N^2)$.

Rozwiązanie

Będziemy kolejno dodawać każde kolejne koło i sprawdzać, w którym miejscu się zatrzyma jeżeli dosuniemy je od prawej strony.

Obserwacja

Ponieważ limity w zadaniu nie są za duże ($N \leqslant 3000$), będziemy szukali rozwiązania działającego w złożoności $\mathcal{O}(N^2)$.

Rozwiązanie

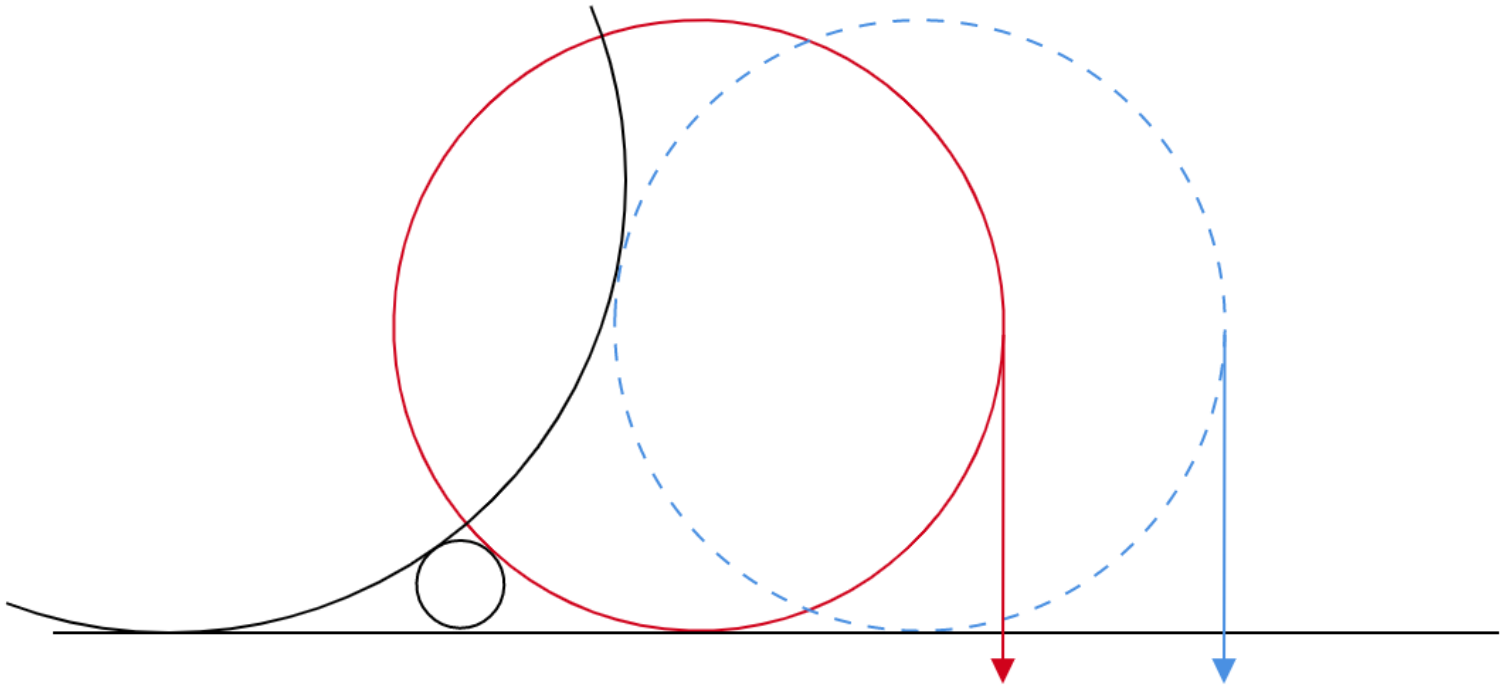
Będziemy kolejno dodawać każde kolejne koło i sprawdzać, w którym miejscu się zatrzyma jeżeli dosuniemy je od prawej strony.
Rozważymy, w którym miejscu zatrzymałoby się koło, jeżeli miałoby stykać się z jakimś z kół, które były wcześniej i mają już jednoznacznie ustaloną pozycję. Pozycją nowego koła, jest maksimum z rozważonych możliwości.

Czy to wystarcza?

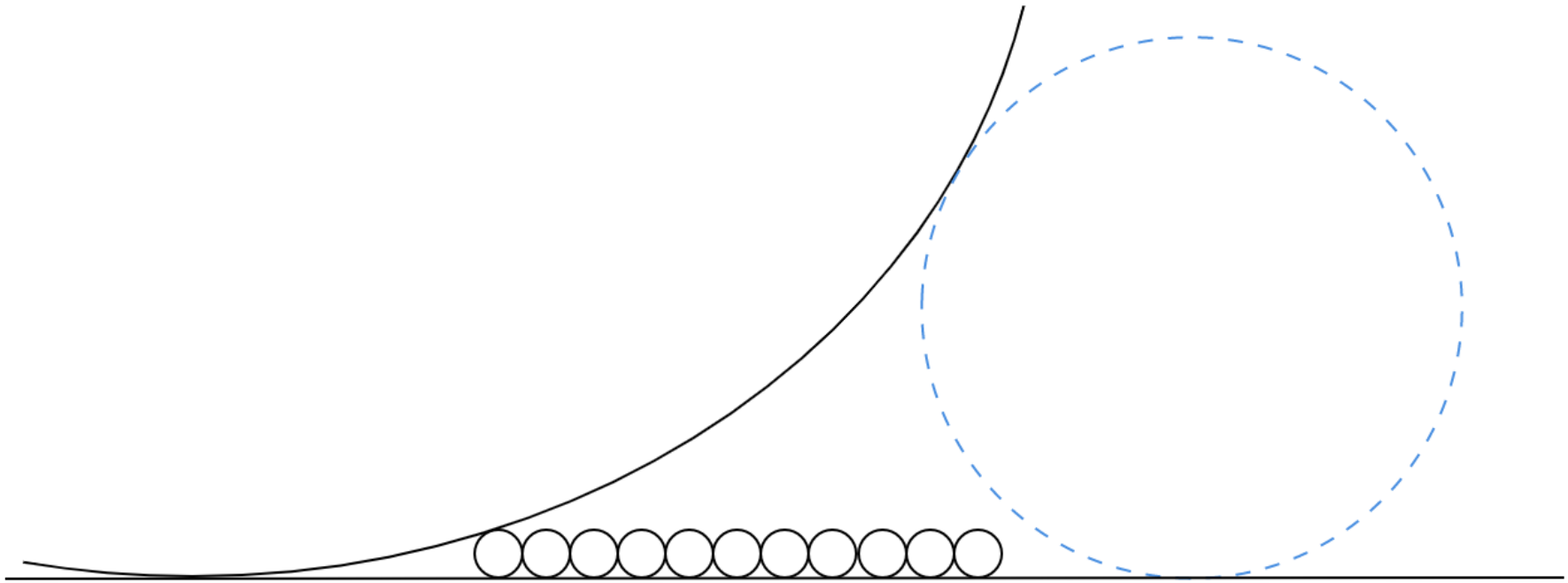
Symulując proces przesuwania koła w lewo w pewnym momencie natrafi ono na przeszkodę (inne koło, albo ścianę magazynu) i się tam zatrzyma. Rozważając wszystkie poprzednie koła, na pewno rozważymy też to, na którym rzeczywiście się zatrzyma. Na pewno też nie postawimy nowego koła za bardzo na prawo.

Czy to wystarcza?

Symulując proces przesuwania koła w lewo w pewnym momencie natrafi ono na przeszkodę (inne koło, albo ścianę magazynu) i się tam zatrzyma. Rozważając wszystkie poprzednie koła, na pewno rozważymy też to, na którym rzeczywiście się zatrzyma. Na pewno też nie postawimy nowego koła za bardzo na prawo.

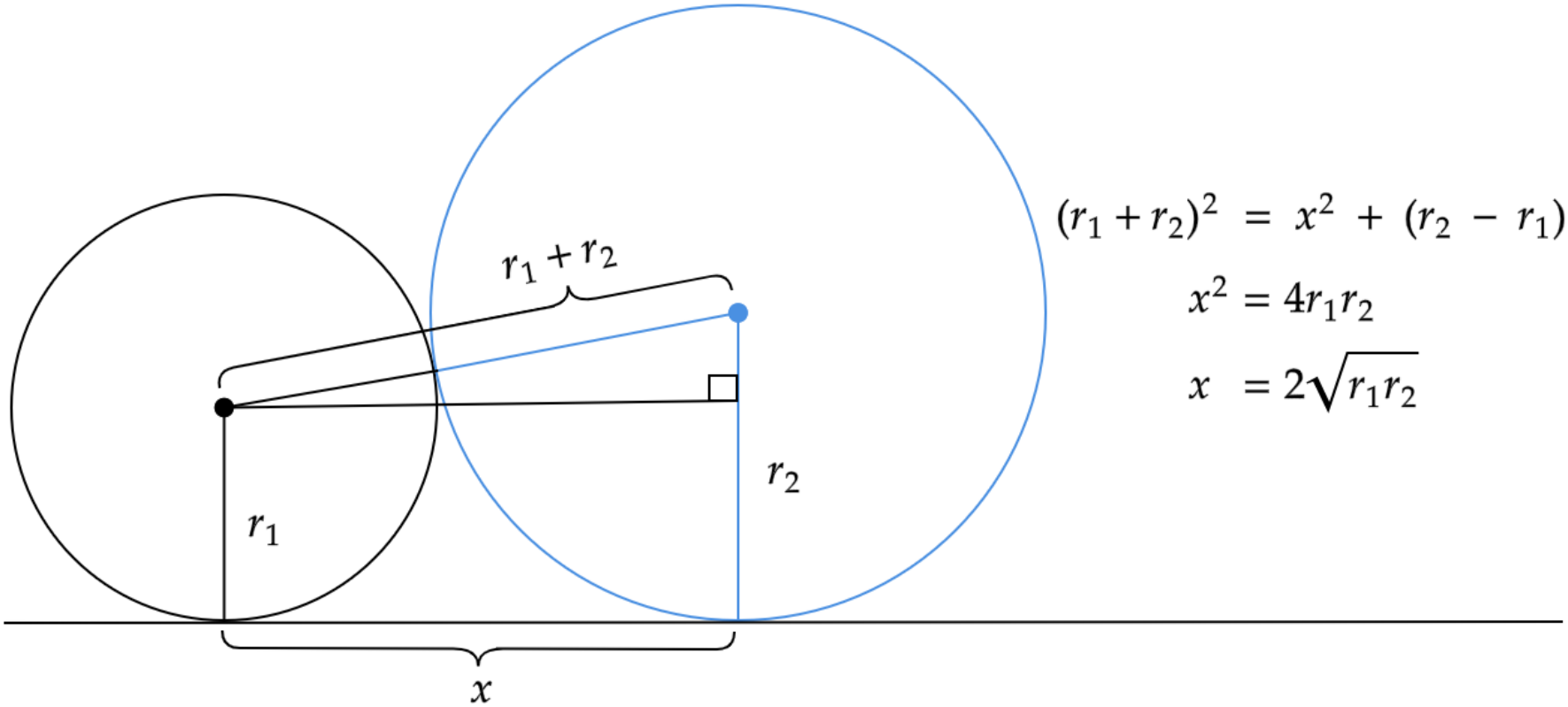


Zwróćmy uwagę na to, że w skrajnym przypadku musimy rozważyć okrąg, który pojawił się dużo wcześniej.



Wzór na poziomą odległość między środkami kół

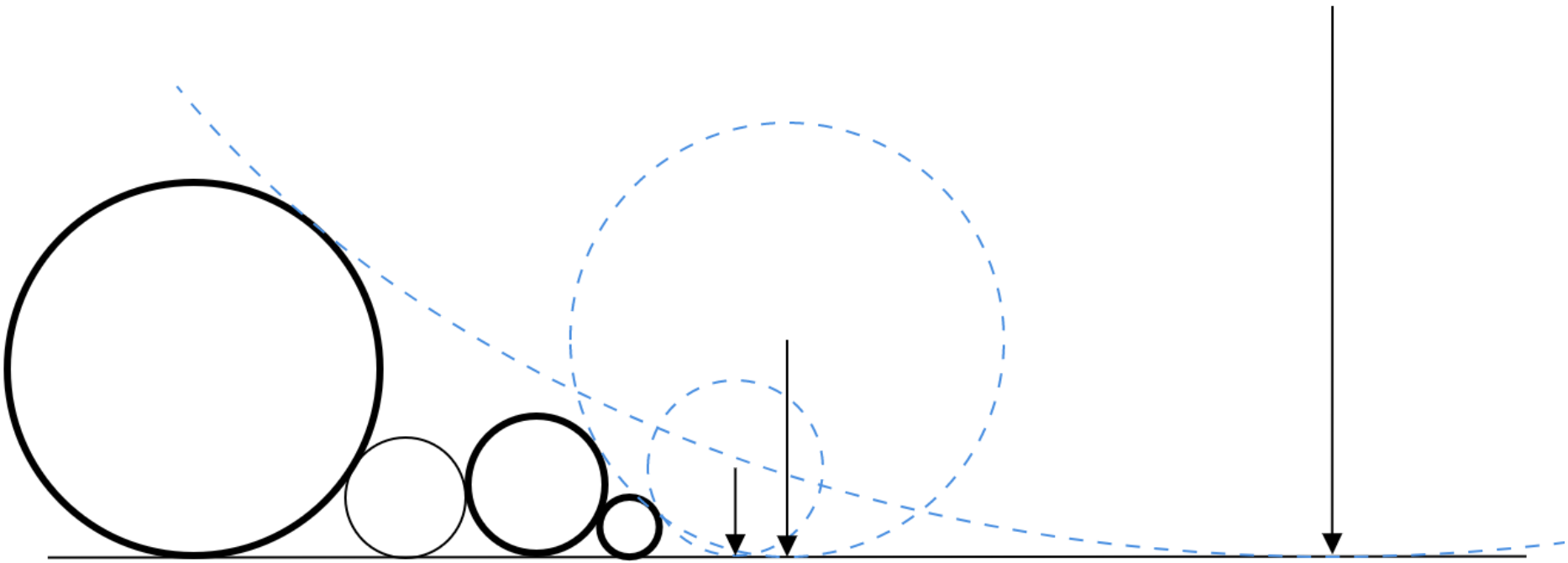
Za pomocą prostych rachunków i Twierdzenia Pitagorasa wyznaczamy następujący wzór:



$$(r_1 + r_2)^2 = x^2 + (r_2 - r_1)^2$$
$$x^2 = 4r_1r_2$$
$$x = 2\sqrt{r_1r_2}$$

Rozwiązania szybsze

Okazuje się, że istnieją rozwiązania działające w czasie $\mathcal{O}(N \log N)$ oraz $\mathcal{O}(N)$, korzystające z obserwacji, że okręgi mogące w przyszłości być jeszcze kandydatami na te kolidujące, mają malejące rozmiary. Jeżeli więc zwiększamy rozmiar dodawanego okręgu, to będzie się on zatrzymywał, na coraz wcześniejszych okręgach.



Lista zadań

1. Problem A - Era Imperium
2. Problem B - Zabawy z zapałkami
3. Problem C - Wiedźmak
4. Problem D - Pradawne zwoje
5. Problem E - Świadek
6. Problem F - Turniej szachowy
7. Problem G - Magnetyczna zabawka
8. Problem H - (Nie)uczciwy remik
9. Problem I - Akrobatyka
10. Problem J - Obrazki logiczne
11. Problem K - Drewno, glina i owce
12. Problem L - Pora roku
13. Problem M - Zaminowane kładki
14. Problem N - Era Mitologii
15. Problem O - Log Roll