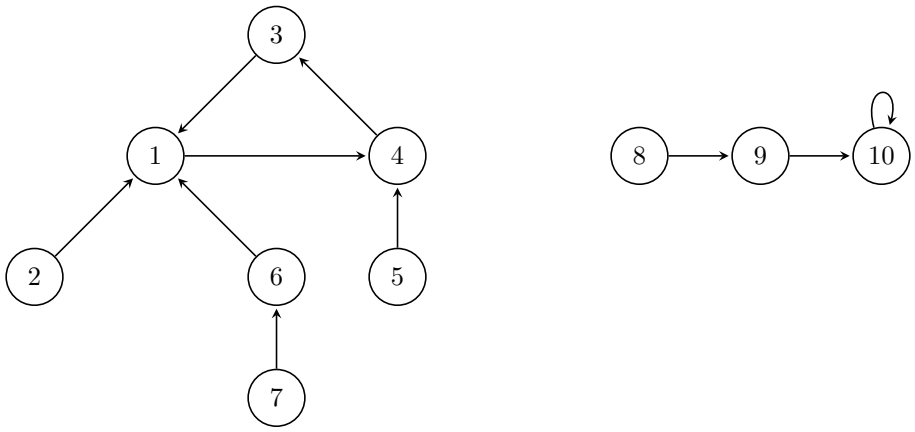
Speedrun (rozwiązanie)

Autor zadania: **Karol Pokorski**
Opracowanie: **Mateusz Lewko, Michał Niedziółka, Jakub Wasilewski**
Opis rozwiązania: **Lech Duraj, Alicja Kluczek, Bartosz Kostka, Mateusz Lewko**

Olimpiada Informatyczna Juniorów

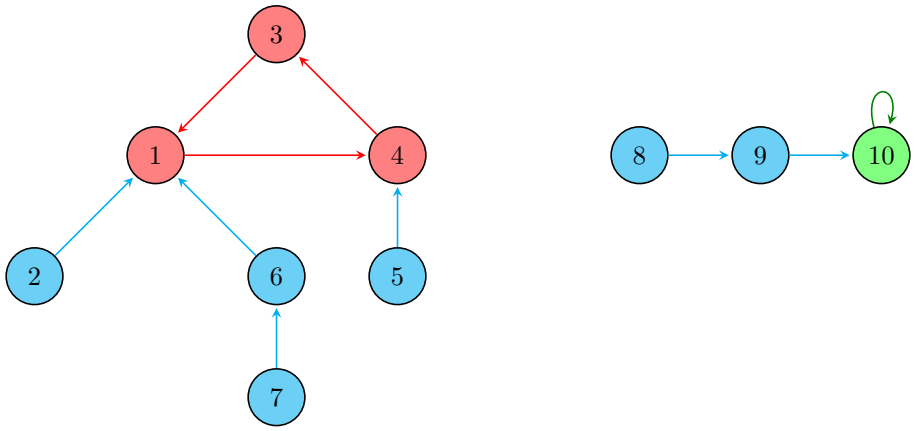
Spróbujmy rozrysować sobie jak wygląda sytuacja z pierwszego testu przykładowego. Z każdego poziomu poprowadzimy strzałkę do poziomu, który po nim następuje, to jest z poziomu i prowadzimy strzałkę do poziomu T_i.



Formalnie mówiąc, poziomy w grze Bajtosi to *wierzchołki*, a przejścia między są *krawędziami grafu skierowanego*. Bajtosia kończy grę w momencie, w którym miałaby przejść do poziomu, w którym już wcześniej była. W języku grafów powiemy, że Bajtosia kończy grę, gdy obejdzie w całości pewien *cykl* – ciąg kolejnych wierzchołków i przejść po krawędziach, który wraca do punktu początkowego. W powyższym grafie może to być na przykład ciąg $(1, 3, 4)$. Zadanie sprowadza się zatem do znalezienia cyklu o najmniejszej sumie numerów wierzchołków.

Rysując kilka przykładowych grafów na kartce, można zauważyć, że graf dany w zadaniu ma bardzo szczególną strukturę. Ze względu na to, że z każdego poziomu możliwe jest tylko jedno wyjście na pewien inny, zawsze ten sam poziom, mamy w grafie dokładnie jedną strzałkę wychodzącą z każdego wierzchołka. Po chwili zastanowienia dochodzimy do wniosku, że taki graf może składać się z kilku kawałków, ale każdy kawałek musi być zawsze jednym cyklem z dochodzącymi do niego „odgałęzieniami", jak na rysunkach. Dzieje się tak dlatego, że nasza wędrówka po poziomach musi się w końcu zapętlić – nieodkryte poziomy muszą się kiedyś skończyć. Zatem wychodząc z każdego wierzchołka musimy dojść do jakiegoś cyklu.

W przedstawionym na rysunkach grafie (z testu przykładowego) widzimy dwa cykle – cykl trzyelementowy złożony z wierzchołków 1, 3 i 4 (czerwony) oraz cykl jednoelementowy złożony z wierzchołka 10 (zielony). Do odgałęzień należą za to wierzchołki 2, 5, 6, 7, 8 i 9 (oznaczone kolorem niebieskim).



Aby zatem zakończyć grę możliwie szybko, nie opłaca nam się zaczynać od wierzchołka niebieskiego (z odgałęzienia) – powinniśmy zacząć od wierzchołka na którymś z cykli, obejść cykl i wrócić do punktu startowego. Musimy więc po prostu przeglądnąć wszystkie cykle, policzyć dla każdego sumę liczb na wierzchołkach, po czym wybrać cykl z najmniejszą sumą.

Olimpiada finansowana jest ze środków Ministerstwa Edukacji Narodowej w ramach zadania publicznego „Organizacja i przeprowadzenie olimpiad i turniejów w latach szkolnych 2019/2020, 2020/2021, 2021/2022".

Nie wiemy jednak na początku, które wierzchołki leżą na cyklach, a które w odgałęzieniach. Zaczniemy zatem od dowolnego wierzchołka (na przykład od 1) i będziemy szli tak długo, aż pewien wierzchołek (nazwijmy go x) odwiedzimy po raz drugi. O wierzchołku x będziemy już wiedzieć , że leży na cyklu, możemy więc zacząć od niego jeszcze raz nasze przejście, sumując liczby na wierzchołkach, aż wrócimy do x. Powstałą sumę zapamiętujemy, jako potencjalny wynik gry.

Łatwo jest wykryć, czy do wierzchołka wróciliśmy po raz drugi: trzymamy w pamięci tablicę `zaznaczone`, gdzie pamiętamy dla każdego wierzchołka informację, czy już w nim byliśmy. Na początku wszystkie jej wartości ustawiamy na *0*, a po wejściu do wierzchołka zmieniamy jego wartość na *1*. Jeśli wejdziemy do wierzchołka, który już miał tę wartość ustawioną na *1*, będą to nasze drugie odwiedziny w tym miejscu.

Przedstawiona powyżej procedura wykryje jednak tylko jeden cykl – ten, do którego da się dojść z wierzchołka 1. Aby policzyć sumę innych cykli, musimy procedurę powtórzyć na innych kawałkach grafu. Patrzymy więc na wierzchołek 2 (a potem na 3, 4, i tak dalej). Możliwa jest jedna z trzech sytuacji:

- Wierzchołek 2 był już odwiedzony (czyli `zaznaczone[2]` nie jest równe 0) – wtedy pomijamy go i przechodzimy do następnego wierzchołka;
- Wierzchołek 2 nie był odwiedzony – wtedy zaczynamy od niego całą procedurę jeszcze raz – a procedura doprowadzi nas do nowego cyklu. Wtedy sprawdzamy, czy nowy cykl nie jest lepszy niż poprzedni, i kontynuujemy od wierzchołka 3;
- Wierzchołek 2 nie był odwiedzony, ale leżał na innej odnodze któregoś z wcześniej odwiedzonych cykli. Wtedy powinniśmy przerwać procedurę, gdy tylko zorientujemy się, że w tym kawałku grafu już byliśmy.

Aby odróżnić ostatnią sytuację (dojścia do wierzchołka, który widzieliśmy w poprzedniej iteracji) od przedostatniej (nowy kawałek i nowy cykl), wpisujemy do tablicy `zaznaczone` za drugim razem nie liczbę 1, ale 2 – a ogólnie, numer wierzchołka, od którego zaczęliśmy procedurę. Jeśli więc dojdziemy do wierzchołka, który ma taką samą liczbę, jak rozpoczynający, oznacza to, że znaleźliśmy nowy cykl. Jeśli zaś do wierzchołka, który nie ma 0, ale jakąś mniejszą liczbę, oznacza to, że w tym miejscu już byliśmy wcześniej, a więc nie powinniśmy tej iteracji kontynuować.

Nie musimy pamiętać sum wszystkich znalezionych cykli, a tylko najmniejszą z nich, i poprawiać ją, kiedy pojawi się nowy, lepszy kandydat. Zauważmy, że nasz algorytm odwiedzi każdy wierzchołek co najwyżej dwa razy – raz w czasie przeglądania (pilnujemy wtedy, żeby nie pójść dwa razy tę samą ścieżką!), i ewentualnie drugi raz w czasie obliczania sumy jego cyklu. Ostateczna złożoność czasowa i pamięciowa tego algorytmu wynosi zatem $O(N)$.

spe.py

```
def main():

  # Wczytujemy dane z wejścia.
  N = int(input())
  T = [0] + [int(x) for x in input().split()]

  # Tworzymy tablicę zaznaczone, w której będziemy zapmiętywać kiedy odwiedziliśmy
  # dany poziom.
  zaznaczone = [0 for _ in range(N+1)]
  # Ustalamy wynik na odpowiednią dużą liczbę, wiemy że wynik będzie mniejszy niż N*N.
  wynik = N * N

  # Dla każdego poziomu od 1 do N.
  for i in range(1, N+1):
    # Jeżeli poziom i nie był jeszcze odwiedzony.
    if zaznaczone[i] == 0:
      aktualny = i
      # Dopóki nie trafimy do już wcześniej odwiedzonego poziomu.
      while zaznaczone[aktualny] == 0:
        # Zazancz poziom jako odwiedzony.
        zaznaczone[aktualny] = i
        # I przejdź do kolejnego.
        aktualny = T[aktualny]
      # Kiedy weszliśmy do tego samego poziomu w tym samym przejściu,
```


Olimpiada finansowana jest ze środków Ministerstwa Edukacji Narodowej w ramach zadania publicznego „Organizacja i przeprowadzenie olimpiad i turniejów w latach szkolnych 2019/2020, 2020/2021, 2021/2022".

```
      # to znaczy że jesteśmy na cyklu.
      if zaznaczone[aktualny] == i:
        # Musimy policzyć sumę indeksów na tym cyklu.
        suma_na_cyklu = 0
        # Zapamiętujemy gdzie znajduje się koniec cyklu.
        koniec_cyklu = aktualny
        # Przechodzimy ponownie po cyklu i dodajemy numery indeksów.
        suma_na_cyklu += aktualny
        aktualny = T[aktualny]
        while koniec_cyklu != aktualny:
          suma_na_cyklu += aktualny
          aktualny = T[aktualny]
        # Na końcu bierzemy wynik z wyniku i sumy na tym cyklu.
        wynik = min(suma_na_cyklu, wynik)

  print(wynik)


main()
```

spe.cpp

```
#include "bits/stdc++.h"

using namespace std;

int main() {
  // Wczytujemy dane z wejścia.
  int N;
  cin >> N;
  vector <int> T(N+1);
  for (int i=1; i<=N; i++) cin >> T[i];

  // Tworzymy tablicę zaznaczone, w której będziemy zapmiętywać kiedy odwiedziliśmy
  // dany poziom.
  vector <int> zaznaczone(N+1);
  // Ustalamy wynik na odpowiednią dużą liczbę, wiemy że wynik będzie mniejszy niż N*N.
  long long wynik = (long long)N * N;

  // Dla każdego poziomu od 1 do N.
  for (int i=1; i<=N; i++) {
    // Jeżeli poziom i nie był jeszcze odwiedzony.
    if (zaznaczone[i] == 0) {
      int aktualny = i;
      // Dopóki nie trafimy do już wcześniej odwiedzonego poziomu.
      while (zaznaczone[aktualny] == 0) {
        // Zazancz poziom jako odwiedzony.
        zaznaczone[aktualny] = i;
        // I przejdź do kolejnego.
        aktualny = T[aktualny];
      }
      // Kiedy weszliśmy do tego samego poziomu w tym samym przejściu,
      // to znaczy że jesteśmy na cyklu.
      if (zaznaczone[aktualny] == i) {
        // Musimy policzyć sumę indeksów na tym cyklu.
        long long suma_na_cyklu = 0;
        // Zapamiętujemy gdzie znajduje się koniec cyklu.
        int koniec_cyklu = aktualny;
        // Przechodzimy ponownie po cyklu i dodajemy numery indeksów.
        do {
```


Olimpiada finansowana jest ze środków Ministerstwa Edukacji Narodowej w ramach zadania publicznego „Organizacja i przeprowadzenie olimpiad i turniejów w latach szkolnych 2019/2020, 2020/2021, 2021/2022".

```
39            suma_na_cyklu += aktualny;
40            aktualny = T[aktualny];
41          } while (koniec_cyklu != aktualny);
42          // Na końcu bierzemy wynik z wyniku i sumy na tym cyklu.
43          wynik = min(suma_na_cyklu, wynik);
44        }
45      }
46    }
47    cout << wynik << "\n";
48  }
```

W języku C++ należy użyć zmiennych typu `long long` do obliczania sumy numerów wierzchołków z cyklu, gdyż ta może być nawet rzędu 10^{12}.

Olimpiada finansowana jest ze środków Ministerstwa Edukacji Narodowej w ramach zadania publicznego „Organizacja i przeprowadzenie olimpiad i turniejów w latach szkolnych 2019/2020, 2020/2021, 2021/2022".