

# Słonie

W Bajtockim Zoo ma się za chwilę odbyć parada, w której uczestniczyć będą wszystkie znajdujące się w nim słonie. Pracownicy zoo zachęcili już zwierzęta do ustawienia się w jednym rzędzie, gdyż zgodnie z zarządzeniem dyrektora zoo taka powinna być początkowa figura parady.

Niestety, na miejsce przybył sam dyrektor i zupełnie nie spodobała mu się wybrana przez pracowników kolejność słoni. Dyrektor zaproponował kolejność, w której według niego zwierzęta będą się najlepiej prezentować, i wydał pracownikom polecenie poprzesztawiania słoni zgodnie z tą propozycją.

Aby uniknąć nadmiernego chaosu tuż przed paradą, pracownicy postanowili przestawiać słonie, zamieniając miejscami kolejno pewne pary słoni. Przestawiane zwierzęta nie muszą sąsiadować ze sobą w rzędzie. Wysiłek potrzebny do naklonienia słonia do ruszenia się z miejsca jest wprost proporcjonalny do jego masy, a zatem wysiłek związany z zamianą miejscami dwóch słoni o masach  $m_1$  oraz  $m_2$  można oszacować przez  $m_1 + m_2$ . Jakim minimalnym wysiłkiem pracownicy mogą poprzesztawiać słonie tak, aby usatysfakcjonować dyrektora?

Napisz program, który:

- wczyta ze standardowego wejścia masy wszystkich słoni z zoo oraz aktualną i docelową kolejność słoni w rzędzie,
- wyznaczy taki sposób poprzesztawiania słoni, który prowadzi od kolejności początkowej do docelowej i minimalizuje sumę wysiłków związanych ze wszystkimi wykonanymi zamianami pozycji słoni,
- wypisze sumę wartości tych wysiłków na standardowe wyjście.

## Wejście

Pierwszy wiersz wejścia zawiera jedną liczbę całkowitą  $n$  ( $2 \leq n \leq 1\,000\,000$ ), oznaczającą liczbę słoni w Bajtockim Zoo. Dla uproszczenia zakładamy, że słonie są ponumerowane od 1 do  $n$ . Drugi wiersz zawiera  $n$  liczb całkowitych  $m_i$  ( $100 \leq m_i \leq 6\,500$  dla  $1 \leq i \leq n$ ), pooddzielanych pojedynczymi odstępami i oznaczających masy poszczególnych słoni (wyrażone w kilogramach).

Trzeci wiersz wejścia zawiera  $n$  różnych liczb całkowitych  $a_i$  ( $1 \leq a_i \leq n$ ), pooddzielanych pojedynczymi odstępami i oznaczających numery kolejnych słoni w aktualnym ustawieniu. Czwarty wiersz zawiera  $n$  różnych liczb całkowitych  $b_i$  ( $1 \leq b_i \leq n$ ), pooddzielanych pojedynczymi odstępami i oznaczających numery kolejnych słoni w ustawieniu proponowanym przez dyrektora zoo. Możesz założyć, że ustawienia reprezentowane przez ciągi  $(a_i)$  oraz  $(b_i)$  są różne.

## Wyjście

Pierwszy i jedyny wiersz wyjścia powinien zawierać jedną liczbę całkowitą, oznaczającą minimalny łączny wysiłek związany z poprzestawianiem słoni, w wyniku którego z ustawienia reprezentowanego przez  $(a_i)$  uzyskuje się ustawienie  $(b_i)$ .

## Przykład

Dla danych wejściowych:

6

2400 2000 1200 2400 1600 4000

1 4 5 3 6 2

5 3 2 4 6 1

poprawnym wynikiem jest:

11200

Jeden z najlepszych sposobów poprzestawiania słoni uzyskujemy, zamieniając miejscami kolejno pary słoni:

- 2 i 5 — wysiłek związany z zamianą to  $2\,000 + 1\,600 = 3\,600$ , a uzyskane ustawienie to 1 4 2 3 6 5,
- 3 i 4 — wysiłek to  $1\,200 + 2\,400 = 3\,600$ , a uzyskane ustawienie to 1 3 2 4 6 5,
- 1 i 5 — wysiłek to  $2\,400 + 1\,600 = 4\,000$ , a uzyskane ustawienie to 5 3 2 4 6 1, czyli ustawienie docelowe.

## Rozwiązanie

### Wstęp

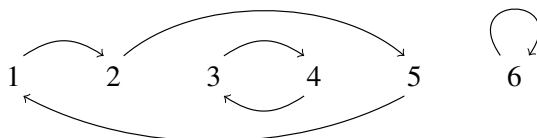
Aby łatwiej wyobrazić sobie zadanie, jakie przed pracownikami zoo postawił dyrektor, spróbujemy przedstawić je graficznie. W tym celu zdefiniujemy funkcję  $p: \{1, 2, \dots, n\} \rightarrow \{1, 2, \dots, n\}$  w następujący sposób:

$$p(b_1) = a_1, \quad p(b_2) = a_2, \quad \dots, \quad p(b_n) = a_n.$$

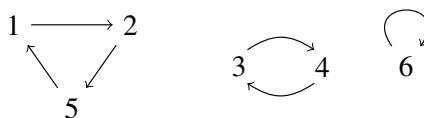
Zauważmy, że wówczas  $p(x) = y$  będzie oznaczało, że słoń o numerze  $x$  powinien znaleźć się w końcowym ustawieniu w miejscu, które jest aktualnie zajmowane przez słonia o numerze  $y$ . Wszystkie liczby  $b_i$  są różne, zatem  $p$  jest poprawnie zdefiniowaną funkcją, a ponieważ wszystkie liczby  $a_i$  są różne,  $p$  jest permutacją zbioru  $\{1, 2, \dots, n\}$ . Sytuację z zadania możemy zatem przedstawić w postaci grafu skierowanego, w którym wierzchołkami są numery  $1, 2, \dots, n$  słoni, krawędziami natomiast wartości funkcji  $p$  (patrz rys. 1).

Dalej, jak każdą permutację, funkcję  $p$  można rozłożyć na tak zwane *cykle proste*  $C_1, C_2, \dots, C_c$ . W tym celu należy wystartować z dowolnego wierzchołka grafu i podążać po krawędziach, aż dojdzie się z powrotem do tego wierzchołka (dlaczego zawsze trafia

się w końcu w wierzchołek początkowy trasy?), po czym usunąć znaleziony cykl z grafu i kontynuować proces aż do wyczerpania wszystkich wierzchołków — patrz rys. 2.

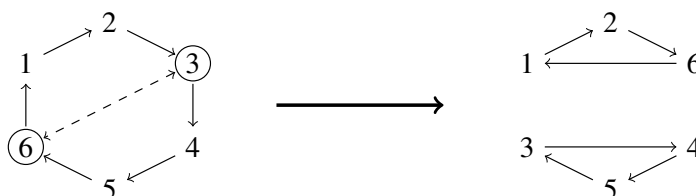


Rys. 1: Graf wyznaczony przez funkcję  $p$  dla przykładu z treści zadania. Wierzchołki grafu reprezentują numery słoni, natomiast strzałka z  $x$  do  $y$  obrazuje relację „słoń  $x$  powinien zająć miejsce słonia  $y$ ”.



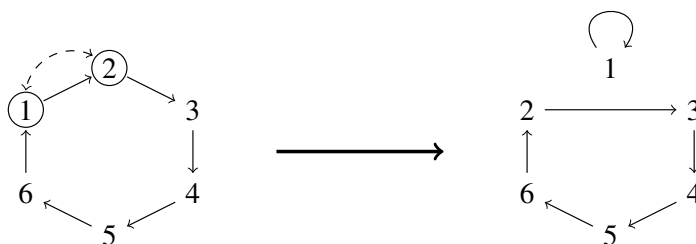
Rys. 2: Rozkład grafu z rys. 1 na cykle proste: trójelementowy, dwuelementowy i jednoelementowy.

Zastanówmy się teraz, jak na tle opisanego rozkładu na cykle proste wygląda operacja zamiany miejscami słoni o numerach  $e$  oraz  $f$ . Koszt takiej zamiany to  $m_e + m_f$ . Jeżeli słonie  $e$  oraz  $f$  znajdują się w tym samym cyklu, to następuje wówczas podział tego cyklu na dwa rozłączne, z których jeden zawiera jednego z tych słoni, a drugi drugiego — patrz rys. 3.



Rys. 3: Zamiana miejscami trzeciego i szóstego słonia w cyklu prowadzi do powstania dwóch cykli trójelementowych.

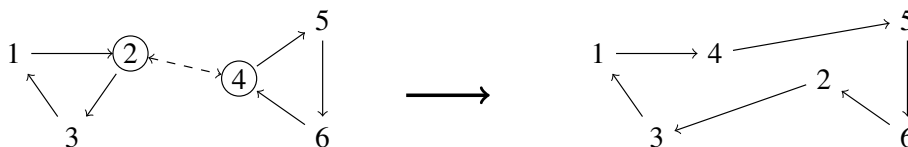
W szczególności, w wyniku zamiany miejscami dwóch słoni, które sąsiadują na cyklu, jeden z powstałych cykli jest jednoelementowy, co oznacza dokładnie tyle, że po tej zamianie jeden ze słoni znajduje się na swojej pozycji docelowej (rys. 4).



Rys. 4: Zamiana miejscami pierwszego i drugiego słonia w cyklu powoduje, że pierwszy słoń trafia na właściwą pozycję.



Z kolei jeżeli słonie  $e$  oraz  $f$  należą do różnych cykli, to zamiana ich miejscami powoduje sklejenie tych cykli w jeden (rys. 5).



Rys. 5: Zamiana miejscami słoni należących do różnych cykli.

## Rozwiązanie wzorcowe<sup>1</sup>

Przyjmijmy następujące oznaczenia:

- $|C|$  — długość cyklu  $C$ , czyli liczba wierzchołków grafu w nim zawartych
- $\text{suma}(C)$  — suma mas słoni należących do cyklu  $C$
- $\text{min}(C)$  — masa najlżejszego słonia na cyklu  $C$
- $\text{min}$  — masa najlżejszego słonia w ogóle.

Naszym celem jest sprowadzenie układu cykli reprezentowanego przez permutację  $p$  do takiego, który będzie złożony wyłącznie z cykli jednoelementowych. W rozwiązaniu wzorcowym każdy cykl  $C$  rozpatrujemy osobno, a wspomniane przekształcenie realizujemy, stosując do  $C$  jedną z następujących metod przestawiania słoni.

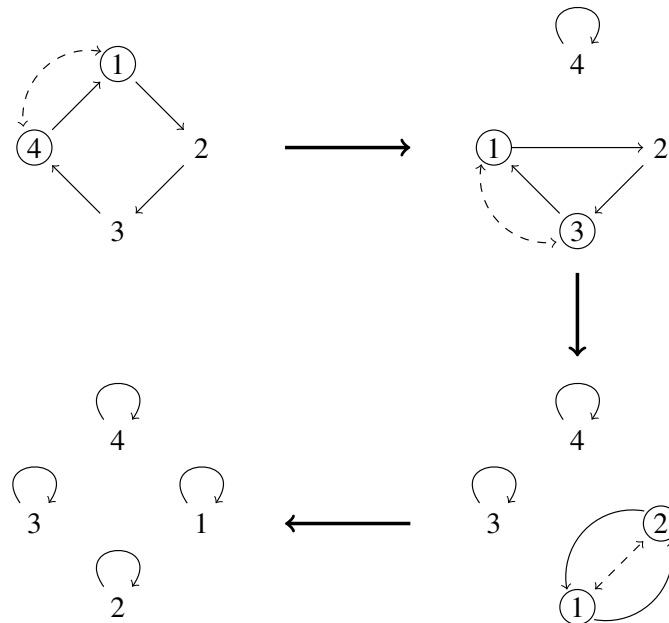
**Metoda 1.** Porządkujemy cały cykl pojedynczymi zamianami sąsiednich słoni (jak na rys. 4). Za każdym razem zamieniamy najlżejszego słonia z całego cyklu ( $\text{min}(C)$ ) z jego poprzednikiem na cyklu, w wyniku czego poprzednik ten trafia na właściwe miejsce (patrz rys. 6). W ten sposób najlżejszego słonia przemieszczamy łącznie  $|C| - 1$  razy, natomiast każdego z pozostałych słoni z cyklu — dokładnie raz. Łączny koszt porządkowania cyklu tą metodą wynosi zatem:

$$\text{metoda}_1(C) = \text{suma}(C) + (|C| - 2) \cdot \text{min}(C). \quad (1)$$

**Metoda 2.** Tym razem postępujemy bardzo podobnie, jednakże do obsłużenia całego cyklu wykorzystujemy globalnie najlżejszego słonia ( $\text{min}$ ). W tym celu zamieniamy go z najlżejszym słoniem cyklu ( $\text{min}(C)$ ), a następnie za jego pomocą przestawiamy kolejno wszystkie słonie z cyklu  $C$  (poza  $\text{min}(C)$ ) jak poprzednio, po czym z powrotem zamieniamy  $\text{min}$  z  $\text{min}(C)$ . Na ten ciąg zamian można też spojrzeć jak na sklejenie  $C$  z cyklem zawierającym  $\text{min}$ , po którym następuje ustawienie wszystkich słoni z  $C$  na właściwych miejscach za pomocą pojedynczych zamian ze słoniem  $\text{min}$ . W ten sposób najlżejszy słoń w całym zoo zostaje przemieszczony  $|C| + 1$  razy, najlżejszy w cyklu — 2 razy, a każdy z pozostałych słoni tego cyklu — dokładnie raz. Łączny koszt wszystkich przestawień to wówczas:

$$\text{metoda}_2(C) = \text{suma}(C) + \text{min}(C) + (|C| + 1) \cdot \text{min}. \quad (2)$$

<sup>1</sup> Duża część opisu rozwiązania wzorcowego została zaczerpnięta z pracy [37].



Rys. 6: Porządkowanie czteroelementowego cyklu za pomocą pierwszej metody. Zakładamy, że najlżejszy słon ma numer 1.

Okazuje się, że w całym rozwiązaniu wystarczy wziąć pod uwagę jedynie dwie opisane metody i dla każdego cyklu w rozkładzie do uporządkowania użyć korzystniejszej z nich<sup>2</sup>. Ostatecznie otrzymujemy następujący minimalny koszt przestawiania słoni:

$$\sum_{i=1}^c \min(\text{metoda}_1(C_i), \text{metoda}_2(C_i)). \quad (3)$$

## Implementacja

Jako podsumowanie słownego opisu rozwiązania wzorcowego umieszczamy poniższy jego pseudokod. Łatwo sprawdzić, że cały algorytm ma złożoność czasową  $O(n)$ . Konkretnie implementacje tego algorytmu można znaleźć w plikach `slo.cpp`, `slo1.java` oraz `slo2.pas`.

```

1: { Konstrukcja permutacji  $p$ . }
2: for  $i := 1$  to  $n$  do  $p[b_i] := a_i$ ;
3:
4: { Rozkład  $p$  na cykle proste. }
5:  $odw : \text{array}[1..n] := (\text{false}, \text{false}, \dots, \text{false})$ ;
6:  $c := 0$ ;
7: for  $i := 1$  to  $n$  do
8:   if not  $odw[i]$  then begin
9:      $c := c + 1$ ;  $x := i$ ;

```

$p(x) = y$  będzie oznaczało, że słon o numerze  $x$  powini

<sup>2</sup>Dodajmy tylko, że dla niektórych cykli zachodzi  $\min(C) = \min$ , i wówczas druga metoda traci sens, jednakże nie trzeba się tym faktem przejmować, gdyż wówczas i tak  $\text{metoda}_2(C) > \text{metoda}_1(C)$ .

```

10:   while not odw[x] do begin
11:       odw[x] := true;
12:        $C_c := C_c \cup \{x\}$ ;
13:        $x := p[x]$ ;
14:   end
15: end
16:
17: { Wyznaczenie parametrów cykli. }
18: min := ∞;
19: for i := 1 to c do begin
20:     suma(Ci) := 0; min(Ci) := ∞;
21:     forall e ∈ Ci do begin
22:         suma(Ci) := suma(Ci) + me;
23:         min(Ci) := min(min(Ci), me);
24:     end
25:     min := min(min, min(Ci));
26: end
27:
28: { Obliczenie wyniku. }
29: w := 0;
30: for i := 1 to c do begin
31:     metoda1(Ci) := suma(Ci) + (|Ci| - 2) · min(Ci);
32:     metoda2(Ci) := suma(Ci) + min(Ci) + (|Ci| + 1) · min;
33:     w := w + min(metoda1(Ci), metoda2(Ci));
34: end
35: return w;

```

## Uzasadnienie poprawności

Na sam koniec opisu rozwiązania wzorcowego pozostawiliśmy jego dowód poprawności. Jest on, niestety, trochę skomplikowany, jednakże jest to jedyna niezawodna metoda sprawdzenia tego, czy w rozwiązaniu nie zapomnieliśmy o żadnym z przypadków.

Jeżeli  $C$  jest cyklem, to przez

$$\text{koszt}(C) = \min(\text{metoda}_1(C), \text{metoda}_2(C))$$

oznaczymy koszt uporządkowania tego cyklu w rozwiązaniu wzorcowym. Niech dalej  $OPT(p)$  oznacza najmniejszy możliwy koszt poprzestawiania wszystkich słoni zgodnie z zarządzeniem dyrektora. Aby wykazać poprawność rozwiązania wzorcowego, wystarczy udowodnić, że

$$OPT(p) \geq \sum_{i=1}^c \text{koszt}(C_i). \quad (4)$$

Nierówność (4) udowodnimy przez indukcję względem liczby zamian wykonywanych w rozwiązaniu optymalnym. Przypadek zerowej liczby zamian jest trywialny. Załóżmy

więc, że (4) zachodzi dla wszystkich permutacji, dla których algorytm optymalny wykonuje  $k - 1$  zamian, i niech  $p$  będzie dowolną permutacją, do której uporządkowania algorytm ten potrzebuje  $k$  zamian. Musimy udowodnić, że (4) zachodzi także dla  $p$ .

Przyjrzyjmy się pierwszemu krokowi rozwiązania optymalnego „OPT” uruchomionego dla  $p$ . Załóżmy, że są w nim zamieniane słonie  $e$  oraz  $f$ . Permutację po ich zamianie oznaczmy przez  $p'$  — uporządkowanie jej przez algorytm optymalny wymaga  $k - 1$  zamian, więc wiadomo, że nierówność (4) zachodzi dla  $p'$ . W zależności od wzajemnego położenia  $e$  oraz  $f$  wyróżniamy teraz kilka przypadków.

**Przypadek 1:**  $e$  i  $f$  należą do tego samego cyklu w  $p$ . Dla ustalenia uwagi niech  $e, f \in C_1$ . Po zamianie słoni cykl  $C_1$  rozpada się na dwa mniejsze cykle (patrz rys. 3). Oznaczmy te cykle przez  $A$  i  $B$  oraz niech  $e \in A$  i  $f \in B$ . Wówczas rozkład  $p'$  na cykle proste wygląda tak:  $\{A, B, C_2, C_3, \dots, C_c\}$ , a stąd i z (4) dla  $p'$  mamy:

$$OPT(p) = OPT(p') + m_e + m_f \geq \text{koszt}(A) + \text{koszt}(B) + \sum_{i=2}^c \text{koszt}(C_i) + m_e + m_f.$$

Aby pokazać (4) dla  $p$ , wystarczy zatem udowodnić, że:

$$\text{koszt}(A) + \text{koszt}(B) + m_e + m_f \geq \text{koszt}(C_1). \quad (5)$$

Dalsze uzasadnienie możemy podzielić na trzy podprzypadki:

**Przypadek 1.1:**  $\text{koszt}(A) = \text{metoda}_1(A)$  i  $\text{koszt}(B) = \text{metoda}_1(B)$ . Korzystając z faktu, że zbiór słoni zawartych w cyklu  $C_1$  jest sumą zbiorów słoni odpowiadających  $A$  oraz  $B$ , oraz z tego, że każda z wartości  $\min(A)$ ,  $\min(B)$ ,  $m_e$ ,  $m_f$  jest nie mniejsza niż  $\min(C_1)$ , mamy:

$$\begin{aligned} \text{metoda}_1(A) + \text{metoda}_1(B) + m_e + m_f &= \\ &= \text{suma}(A) + (|A| - 2) \cdot \min(A) + \text{suma}(B) + (|B| - 2) \cdot \min(B) + m_e + m_f \geq \\ &\geq \text{suma}(C_1) + (|C_1| - 2) \cdot \min(C_1) = \text{metoda}_1(C_1) \geq \text{koszt}(C_1). \end{aligned}$$

Intuicyjnie, pokazaliśmy, że zamiast rozbijać cykl  $C_1$  na cykle  $A$  i  $B$  (za pomocą zamiany  $e$  z  $f$ ) i każdy z nich porządkować za pomocą pierwszej metody, można równie dobrze od razu uporządkować cały cykl  $C_1$  za pomocą tej metody.

**Przypadek 1.2:**  $\text{koszt}(A) = \text{metoda}_1(A)$  i  $\text{koszt}(B) = \text{metoda}_2(B)$  (lub odwrotnie). Podobnie jak poprzednio, w poniższej nierówności liczba składników nie zmienia się, a jedynie zastępujemy cięższe słonie przez  $\min(C_1)$  lub  $\min$ :

$$\begin{aligned} \text{metoda}_1(A) + \text{metoda}_2(B) + m_e + m_f &= \\ &= \text{suma}(A) + (|A| - 2) \cdot \min(A) + \text{suma}(B) + \min(B) + (|B| + 1) \cdot \min + m_e + m_f \geq \\ &\geq \text{suma}(C_1) + \min(C_1) + (|C_1| + 1) \cdot \min = \text{metoda}_2(C_1) \geq \text{koszt}(C_1). \end{aligned}$$

Tym razem intuicja stojąca za powyższym ciągiem przekształceń jest taka, że zamiast wprowadzać najlżejszego słonia tylko do cyklu  $B$ , można go wprowadzić od razu do całego  $C_1$ , co nie przynosi żadnej straty, a może się dodatkowo opłacić przy porządkowaniu fragmentu  $C_1$  odpowiadającego cyklowi  $A$ .

**Przypadek 1.3:**  $\text{koszt}(A) = \text{metoda}_2(A)$  i  $\text{koszt}(B) = \text{metoda}_2(B)$ . I tym razem do wyprowadzenia (5) wykorzystujemy te same spostrzeżenia. W tym przypadku strata wynikła z rozbicia cyklu  $C_1$  jest ewidentna: intuicyjnie, zamiast wprowadzać słonia  $\min$  osobno do każdego z cykli  $A, B$ , można na samym początku wprowadzić go do całego  $C_1$ :

$$\begin{aligned} \text{metoda}_2(A) + \text{metoda}_2(B) + m_e + m_f &= \\ &= \text{suma}(A) + \min(A) + (|A| + 1) \cdot \min + \text{suma}(B) + \min(B) + (|B| + 1) \cdot \min + m_e + m_f \geq \\ &\geq \text{suma}(C_1) + \min(C_1) + (|C_1| + 1) \cdot \min = \text{metoda}_2(C_1) \geq \text{koszt}(C_1). \end{aligned}$$

**Przypadek 2:**  $e$  i  $f$  należą do różnych cykli. Dla ustalenia uwagi niech tym razem  $e \in C_1, f \in C_2$ . Po zamianie słoni cykle  $C_1$  i  $C_2$  łączą się w jeden cykl  $A$  (patrz rys. 5), stąd rozkład  $p'$  na cykle to:  $\{A, C_3, C_4, \dots, C_c\}$ . Stąd i z (4) dla  $p'$  otrzymujemy:

$$\text{OPT}(p) = \text{OPT}(p') + m_e + m_f \geq \text{koszt}(A) + \sum_{i=3}^c \text{koszt}(C_i) + m_e + m_f.$$

Aby pokazać (4) dla  $p$ , wystarczy udowodnić, że:

$$\text{koszt}(A) + m_e + m_f \geq \text{koszt}(C_1) + \text{koszt}(C_2). \quad (6)$$

Bez straty ogólności założmy, że  $\min(C_1) \leq \min(C_2)$ , czyli  $\min(A) = \min(C_1)$ . Tym razem mamy dwa podprzypadki:

**Przypadek 2.1:**  $\text{koszt}(A) = \text{metoda}_1(A)$ . Korzystając z tego, że cykl  $A$  jest sumą (pod względem zbioru zawartych w nim słoni) cykli  $C_1$  oraz  $C_2$ , a także z założenia  $\min(A) = \min(C_1)$  oraz nierówności:  $m_f \geq \min(C_2)$  i  $m_e, \min(A) \geq \min$ , otrzymujemy następujący ciąg przekształceń:

$$\begin{aligned} \text{metoda}_1(A) + m_e + m_f &= \\ &= \text{suma}(A) + (|A| - 2) \cdot \min(A) + m_e + m_f \geq \\ &\geq \text{suma}(C_1) + (|C_1| - 2) \cdot \min(C_1) + \text{suma}(C_2) + \min(C_2) + (|C_2| + 1) \cdot \min = \\ &= \text{metoda}_1(C_1) + \text{metoda}_2(C_2) \geq \text{koszt}(C_1) + \text{koszt}(C_2). \end{aligned}$$

Intuicja tym razem jest taka, że zamiast łączyć cykle  $C_1$  i  $C_2$  i porządkować otrzymany cykl  $A$  metodą 1, można sam cykl  $C_1$  uporządkować tą metodą (tu nic nie tracimy, gdyż  $\min(A) = \min(C_1)$ ), natomiast cykl  $C_2$  połączyć nie z  $C_1$ , ale z cyklem zawierającym najlżejszego słonia  $\min$ , co odpowiada zastosowaniu do  $C_2$  drugiej metody porządkowania.

**Przypadek 2.2:**  $\text{koszt}(A) = \text{metoda}_2(A)$ . Podobnie jak poprzednio, na mocy warunków  $\min(A) = \min(C_1)$ ,  $m_f \geq \min(C_2)$  oraz  $m_e \geq \min$ , mamy:

$$\begin{aligned} \text{metoda}_2(A) + m_e + m_f &= \\ &= \text{suma}(A) + \min(A) + (|A| + 1) \cdot \min + m_e + m_f \geq \\ &\geq \text{suma}(C_1) + \min(C_1) + (|C_1| + 1) \cdot \min + \text{suma}(C_2) + \min(C_2) + (|C_2| + 1) \cdot \min = \\ &= \text{metoda}_2(C_1) + \text{metoda}_2(C_2) \geq \text{koszt}(C_1) + \text{koszt}(C_2). \end{aligned}$$



Intuicyjnie, nie opłaca się ponosić kosztu połączenia cykli  $C_1$  i  $C_2$ , żeby potem uporządkować wynikowy cykl  $A$  za pomocą najbliższego słonia  $min$ , gdyż na pewno nie gorszym rozwiązaniem jest wprowadzenie słonia  $min$  do każdego z cykli  $C_1$ ,  $C_2$  z osobna. Innymi słowy,  $min + min(A) + m_e + m_f \geq 2 \cdot min + min(C_1) + min(C_2)$ .

## Inne rozwiązania

Wśród potencjalnych rozwiązań błędnych można wyróżnić przede wszystkim takie, które przy porządkowaniu cykli zapominają o jednej z metod: drugiej (plik `slob1.cpp`, 20% punktów) lub pierwszej (plik `slob2.cpp`, 10% punktów). Przypomnijmy, że takich błędów można uniknąć, jeżeli przeprowadzi się dowód poprawności rozwiązania lub chociażby sprawdzi poprawność swojego rozwiązania na większej grupie losowych testów, porównując jego wyniki z jakimkolwiek rozwiązaniem na pewno poprawnym, chociażby wykładniczym. Innym błędem było wykonywanie wszystkich obliczeń na liczbach całkowitych 32-bitowych — takie rozwiązanie, wskutek błędu przepełnienia typu, zdobywało 60% punktów za to zadanie (implementacja w pliku `slob3.cpp`).

Wśród rozwiązań wolniejszych można wymienić rozwiązanie kwadratowe względem  $n$  (plik `slos1.cpp`), będące nieefektywną implementacją rozwiązania wzorcowego i zdobywające 40% punktów, oraz zaimplementowane w pliku `slos2.cpp` i uzyskujące 10% punktów rozwiązanie siłowe, rozważające wszystkie możliwości zamian słoni, poczynając od najtańszych.

## Testy

Zadanie było sprawdzane na 10 zestawach danych testowych. Wszystkie testy za wyjątkiem tych z grupy b to testy w jakimś sensie losowe. Większość testów zawiera losową permutację  $p$  słoni. Ponieważ zupełnie losowa permutacja zawiera statystycznie stosunkowo mało cykli, to w testach 4 i 10a wygenerowano permutacje o dużych liczbach cykli. Poza tym specjalną postać mają testy 9a oraz te z grupy b — patrz opisy poniżej.

W następującym zestawieniu testów  $n$  oznacza liczbę słoni, natomiast pozostałe parametry charakteryzują własności permutacji  $p$ :  $c_1$  to liczba cykli jednoelementowych (czyli takich, które nie wymagają żadnych zamian),  $m_1$  to liczba cykli, których optymalne uporządkowanie otrzymuje się za pomocą metody 1, natomiast  $m_2$  to liczba cykli, które należy porządkować metodą 2.

Nazwa	n	$c_1$	$m_1$	$m_2$	Opis
<i>slo1.in</i>	10	1	1	1	test losowy
<i>slo2.in</i>	100	2	4	1	test losowy
<i>slo3.in</i>	1 000	2	8	0	test losowy
<i>slo4.in</i>	10 000	24	55	24	test losowy o zwiększonej liczbie cykli
<i>slo5.in</i>	100 000	2	8	1	test losowy

80 *Słonie*

Nazwa	n	c <sub>1</sub>	m <sub>1</sub>	m <sub>2</sub>	Opis
<i>slo6.in</i>	920 000	1	9	7	test losowy
<i>slo7.in</i>	960 000	2	6	11	test losowy
<i>slo8a.in</i>	980 000	0	6	8	test losowy
<i>slo8b.in</i>	980 000	979 998	1	0	potrzeba tylko jednej zamiany
<i>slo9a.in</i>	1 000 000	904 788	44 788	424	90% słoni na swoim miejscu
<i>slo9b.in</i>	1 000 000	0	500 000	0	wszystkie cykle dwuelementowe
<i>slo10a.in</i>	1 000 000	307	330	1212	test losowy o zwiększonej liczbie cykli
<i>slo10b.in</i>	1 000 000	0	1	0	jeden długi cykl