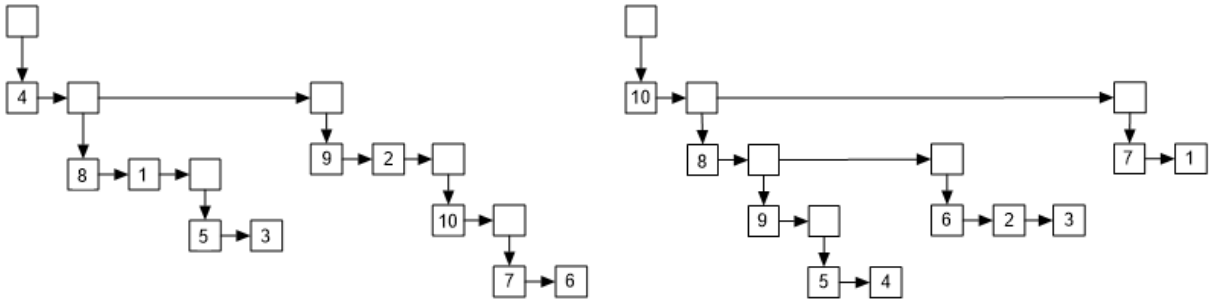
Zadanie polega na zaimplementowaniu algorytmu, który wczytuje zbiór *n* ukorzenionych drzew w formacie Newick, a następnie dla każdej z $\frac{n \cdot (n-1)}{2}$ pary drzew obliczy minimalną liczbę liści, których usunięcie z obydwu drzew spowoduje, że będą one izomorficzne. Problem znany jest w informatyce jako *Maximum agreement subtree problem*. Oto przykładowe dwa drzewa, każde z nich zawiera 10 identycznych liści oznaczonych etykietami od 1 do 10:

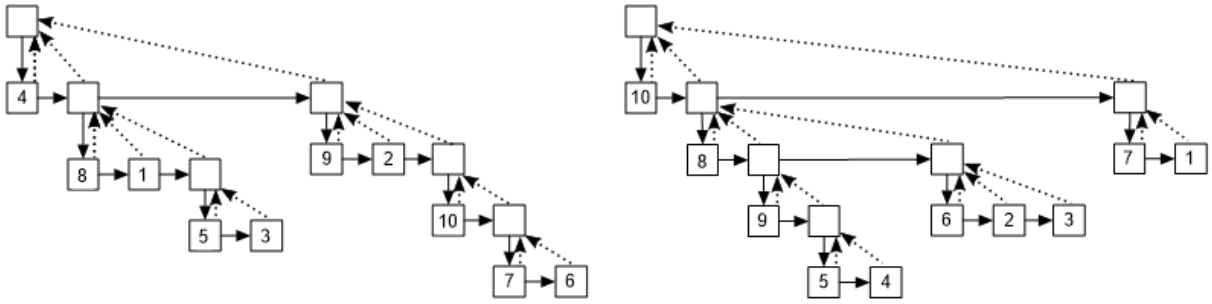
```
(4,(8,1,(5,3)),(9,2,(10,(7,6))));
```

```
(10,(8,(9,(5,4)),(6,2,3)),(7,1));
```

Podane drzewa należy zapisać w pamięci komputera w postaci drzew ogólnych, wykorzystujących jeden wskaźnik na syna, jeden na brata i jeden na rodzica. Na rysunku poniżej widać taką reprezentację ogólną obydwu drzew.



Wskaźniki (strzałki) w dół to relacja rodzic-syn a w prawo brat-brat. Wskaźników skierowanych do góry (relacja syn-rodzic) nie narysowano na pierwszym rysunku ze względu na przejrzystość. Widać je na drugim rysunku. Należy pamiętać, że każdy węzeł (poza korzeniem) taki wskaźnik posiada.

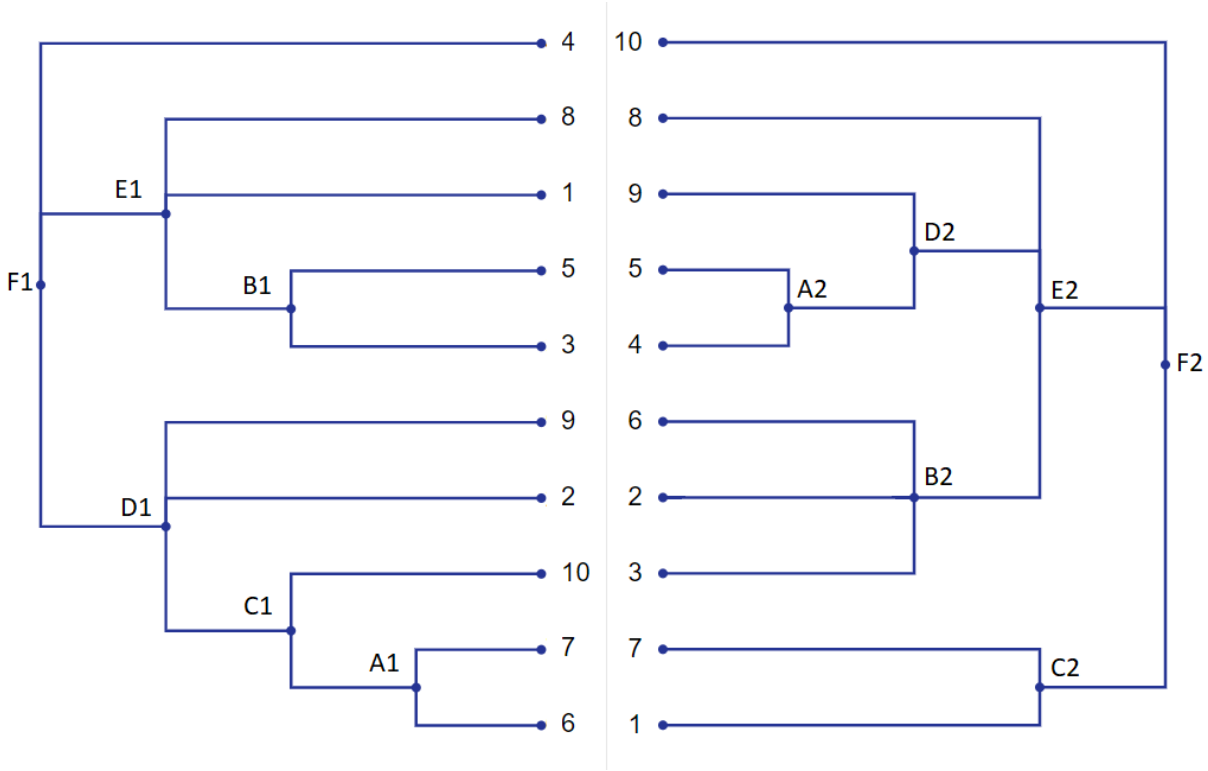


Rozwiązanie brute force (50%)

Można wygenerować wszystkie $n!$ możliwych podzbiorów liści. Dla każdego podzbioru należy utworzyć poddrzewa indukowane przez ten zbiór z obydwu drzew wejściowych. Jeśli poddrzewa są izomorficzne, to poddrzewo jest zgodne (agreement subtree). Następnie ze wszystkich znalezionych zgodnych poddrzew należy wyszukać największe. Liczba liści tego drzewa, l, jest rozwiązaniem. Jako wynik działania programu trzeba jednak podać minimalny zbiór liści, których usunięcie pozwala otrzymać to poprawne rozwiązanie, zatem jako wynik wysłana powinna zostać wartość $k - l$, gdzie k oznacza liczbę wszystkich liści pojedynczego drzewa. Takie podejście wymaga implementacji eleganckiego sposobu iteracji po wszystkich podzbiorach liści oraz porównywania indukowanych poddrzew. Jest ono również mocno nieefektywne obliczeniowo.

Rozwiązanie O(n^2) (100%)

Dla drzew widocznych powyżej należy nadać identyfikatory wierzchołkom wewnętrznym (nie muszą to być kombinacje litera-liczba, tak jak w przykładzie, mogą to być po prostu kolejne liczby większe od tych przypisanych liściom). Pamiętać trzeba, że drzewa te nadal reprezentowane są za pomocą notacji drzew ogólnych – na rysunku poniżej zastosowano "zwykłe drzewa" jedynie w celu poprawienia czytelności.



Następnie tworzona jest tablica, w której każdy wymiar zawiera reprezentację wszystkich wierzchołków jednego drzewa. Taką tablicę wypełnia się najpierw wstawiając jedynki dla wszystkich liści o zgodnych etykietach. Jeśli w wierzchołku wewnętrznym jednego drzewa znajduje się liść z odpowiadającą mu etykietą z pierwszego drzewa, to wpisywana jest 1. Najważniejsza część algorytmu zaczyna się podczas porównywania wierzchołków wewnętrznych jednego drzewa z wierzchołkami wewnętrznymi drugiego.

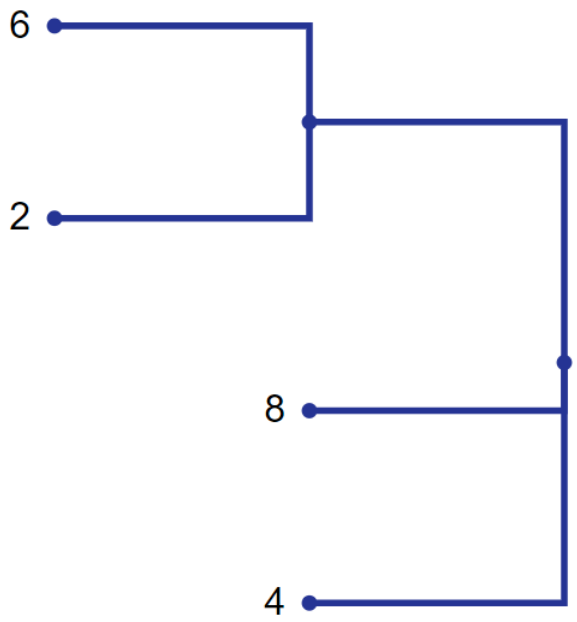
Należy to robić w odpowiedniej kolejności, tzw. odwrotnym przeszukiwaniem wszerz. Dla każdej pary należy rozpatrzyć 3 przypadki i wybrać ten, który daje największy wynik (największą liczbę zgodnych liści). Porównując dwa wierzchołki wewnętrzne – *AX* oraz *BX*:

1. należy sprawdzić, czy jeden z synów węzła *AX* można utożsamić z węzłem *BX*,
2. należy sprawdzić, czy jeden z synów węzła *BX* można utożsamić z węzłem *AX*,
3. należy powiązać synów węzła *AX* z synami węzła *BX* (niekoniecznie wszystkimi), tak aby suma tych powiązań była jak największa.

Ostatni podpunkt można zaimplementować algorytmem brute force, przeglądając pełną przestrzeń możliwych przypisań wierzchołków i wybierając najlepszy wynik. Kończąc porównanie dwóch korzeni uzyskuje się końcowy wynik, czyli liczbę liści maksymalnego zgodnego poddrzewa *l*. Należy pamiętać, że jako wynik wysłana powinna zostać wartość *k* - *l*.

	10	8	9	5	4	6	2	3	7	1	F	2	E	2	D	2	A	2	B	2	C	2
4	0	0	0	0	1	0	0	0	0	0	1	1	1	1	0	0						
8	0	1	0	0	0	0	0	0	0	0	1	1	0	0	0	0						
1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1					
5	0	0	0	1	0	0	0	0	0	0	1	1	1	1	0	0						
3	0	0	0	0	0	0	0	1	0	0	1	1	0	0	1	0						
9	0	0	1	0	0	0	0	0	0	0	1	1	1	0	0	0						
2	0	0	0	0	0	0	1	0	0	0	1	1	0	0	1	0						
10	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0						
7	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	1						
6	0	0	0	0	0	1	0	0	0	0	1	1	0	0	1	0						
F1	1	1	1	1	1	1	1	1	1	1	4	4	2	2	2	2						
E1	0	1	0	1	0	0	0	1	0	1	3	2	1	1	1	1						
B1	0	0	0	1	0	0	0	1	0	0	2	2	1	1	1	0						
D1	1	0	1	0	0	1	1	0	1	0	2	2	1	0	2	1						
C1	1	0	0	0	0	1	0	0	1	0	2	1	0	0	1	1						
A1	0	0	0	0	0	1	0	0	1	0	2	1	0	0	1	1						

W tabeli powyżej widoczna jest wypełniona tablica po przebiegu algorytmu. Jak widać, już zmapowanie wierzchołka *F1* (korzenia pierwszego drzewa) z wierzchołkiem *E2* dało wynik optymalny $l = 4$. W kolejnym kroku, podczas mapowania wierzchołków wewnętrznych *F1* i *F2* (korzeni obu drzew) wierzchołek *E2* został wybrany jako najlepszy wybór z 3 opisanych wcześniej warunków. Wynikiem końcowym jest zatem 6 (10 - 4). Jest to minimalna liczba liści wymaganych do usunięcia w każdym z porównywanych drzew, aby były one izomorficzne. W tym przypadku tymi liśćmi są wierzchołki oznaczone etykietami 1, 3, 5, 7, 9, 10. Po ich usunięciu otrzymane zostanie poniższe zgodne poddrzewo:



W programie nie wolno korzystać z bibliotek STL, <algorithm> ani podobnych. Dopuszczalne jest użycie bibliotek <iostream>, <cstdio> oraz <string>.

Wejście

Wartość *n* oznaczająca wielkość zbioru ukorzenionych drzew.

n ukorzenionych drzew w formacie Newick.

Wyjście

$\frac{n \cdot (n-1)}{2}$ porównań drzew wejściowych (każde z każdym). Kolejność porównywania: 1-2, 1-3, ..., 1-*n*, 2-3, 2-4, ..., 2-*n*, ..., (*n*-1)-*n*.

Przykład

Wejście

4

(1,5,(10,6,3),(2,(8,7)),(9,4));

((7,(3,(4,9,(1,2)))),8,(5,(10,6)));

(7,((6,(9,5),(8,3)),(1,(2,10,4))));

(7,(8,3,4,6,1),(9,5,10),2);

Wyjście

5

7

6

6

6

6