

Task C23. MONOPOLY 2

🕒 1 sec. 📁 256 MB

După ce ai ajutat-o pe **Deni** să creeze planșa de joc perfectă pentru monopoly în 2021, ea are nevoie din nou de ajutor. Ne amintim că planșa de joc pentru monopoly este alcătuită din N spații (numerotate de la 1 la N) și de M conexiuni directe între acestea. Aici, $M = N-1$. Conexiunile direcționate respectă următoarele condiții - nu există o conexiune de la un spațiu la el însuși și nu există conexiuni diferite între aceeași pereche de spații. De asemenea, conexiunile au proprietatea că orice spațiu poate fi atins din spațiul 1.

Definiție: Fie o permutare a numerelor de la 1 la N . Spunem că această permutare este o ordonare *apropiată* a spațiilor dacă următoarea condiție este îndeplinită - pentru fiecare conexiune de la spațiul i la spațiul j , i trebuie să fie **înainte** de j în permutare.

Problema lui **Deni** este că a pierdut planșa perfectă pentru jocul de Monopoly. Din fericire, prietenul ei **Bobi** își amintește cum arată planșa, dar a decis să se distreze puțin cu ea înainte de a-i dezvălui conexiunile. Mai întâi, **Bobi** îi spune lui **Deni** că o ordonare *apropiată* a spațiilor este șirul $1, 2, \dots, N$. După aceea, **Bobi** va răspunde la câteva întrebări ale lui **Deni** pentru a o ajuta să afle care sunt conexiunile dintre spații. Fiecare întrebare va fi despre cât de *apropiată* este o anumită permutare a numerelor de la 1 la N (pentru mai multe detalii, vedeți secțiunea "**Detalii de implementare**"). Eroina noastră are nevoie de ajutorul tău pentru a găsi și implementa o strategie cu cât mai puține întrebări posibile.

Cerință

Scrieți programul **monopoly2** care determină conexiunile necunoscute utilizând cele mai puține întrebări posibile adresate lui **Bobi**. Acesta trebuie să conțină funcția `find_connections` care va fi compilată și executată împreună cu programul comisiei (pentru rolul lui **Bobi**).

Detalii de implementare

Funcția ta `find_connections` trebuie să aibă următorul format:

```
std::vector <std::pair <int, int> > find_connections (int N);
```

Funcția va fi apelată o singură dată de către programul comisiei cu un singur parametru - numărul de spații. Funcția trebuie să returneze o listă de perechi ordonate ce descriu conexiunile găsite între spații. Ordinea perechilor în listă nu contează.

Funcția pentru adresarea întrebărilor către **Bobi** are următorul format:

```
std::vector <bool> check (std::vector <int> p);
```

Parametrul p reprezintă permutarea numerelor de la 1 la N din întrebare. Rezultatul este un vector boolean b de dimensiune N , unde $b_i = 1$ ($0 \leq i < N$) dacă și numai dacă este îndeplinită una dintre următoarele condiții:

- există o conexiune de la spațiul p_j la spațiul p_i cu $i < j$;
- există un spațiu p_j pentru care $b_j = 1$ și se poate ajunge de la p_j la p_i parcurgând conexiunile.

Rețineți că neambiguitatea valorilor rezultă din proprietățile conexiunilor.

Dacă șirul nu este o permutare validă a numerelor de la 1 la N , vei primi **Wrong**

**XVI INTERNATIONAL ADVANCED TOURNAMENT IN INFORMATICS
SHUMEN 2025**

answer pentru test. Complexitatea funcției este $O(N)$. Poți apela această funcție de maximum $\frac{10^8}{N}$ ori, altfel vei primi Wrong answer.

Programul tău **monopoly2** trebuie să implementeze funcția `find_connections`. Poate conține și alt cod, funcții și variabile globale, dar nu trebuie să conțină funcția `main`. De asemenea, nu trebuie să citești din intrarea standard sau să afișezi la ieșirea standard. Programul trebuie să includă fișierul antet `monopoly2.h` prin includerea următoarei secvențe de cod:

```
#include "monopoly2.h"
```

Restricții

- ♣ $1 \leq N \leq 1\,000$;
- ♣ $M = N - 1$.

Subtasks

| Subtask | Punctaj | Subtask-uri necesare | N | Alte restricții |
|---------|---------|----------------------|---------------|---|
| 0 | 0 | — | — | Planșa din exemplu. |
| 1 | 5 | — | $\leq 1\,000$ | $1, 2, \dots, N$ este singura ordonare adecvată a spațiilor. |
| 2 | 6 | 0 | ≤ 6 | — |
| 3 | 17 | 1 | $\leq 1\,000$ | $1, 2, \dots, N$ se poate obține și astfel: începând cu spațiul 1, apoi listând spațiile cu conexiuni directe din 1, apoi listând spațiile cu conexiuni directe din al doilea spațiu adăugat, și așa mai departe. |
| 4 | 13 | 0, 2 | ≤ 300 | — |
| 5 | 59 | 0 – 4 | $\leq 1\,000$ | Dacă începem de la spațiul 1 și urmăm conexiunile, nu putem folosi mai mult de 25 de conexiuni. |

Punctele pentru un subtask se acordă doar dacă toate testele pentru acesta și pentru subtask-urile necesare sunt trecute cu **succes**, iar punctajul este egal cu scorul minim obținut la testele din acesta și din subtask-urile necesare, înmulțit cu punctele subtask-ului.

Punctare

Fiecare test primește un scor care este un număr fracționar între 0 și 1 inclusiv. Dacă un test are un scor pozitiv, acesta este considerat **succes** pentru soluția ta. Un test are scor pozitiv dacă găsești cu succes conexiunile dintre spații.

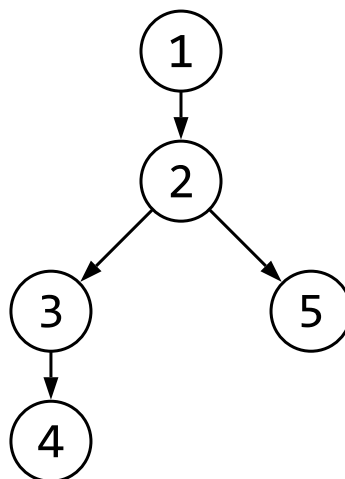
Dacă *cnt* este numărul de apeluri ale funcției `check` pentru un anumit test, atunci scorul testului se calculează în următorul mod:

- Pentru toate subtask-urile: dacă $cnt > \frac{10^8}{N}$, atunci scorul este 0.

- Subtask-urile 0, 1 și 2:
 - Dacă $cnt \leq \frac{10^8}{N}$, atunci scorul este 1.
- Subtask-ul 3:
 - Dacă $cnt \leq 10\,000$, atunci scorul este 1.
 - Dacă $10\,000 < cnt \leq 15\,000$, atunci scorul este $\min(\frac{1000}{cnt-9000}, 1)$.
 - Dacă $15\,000 < cnt \leq \frac{10^8}{N}$, atunci scorul este 0.1.
- Subtask-ul 4:
 - Dacă $cnt \leq 50\,000$, atunci scorul este 1.
 - Dacă $50\,000 < cnt \leq 200\,000$, atunci scorul este $\min(\frac{25000}{cnt-25000}, 1)$.
 - Dacă $200\,000 < cnt \leq \frac{10^8}{N}$, atunci scorul este 0.1.
- Subtask-ul 5:
 - Dacă $cnt \leq 170$, atunci scorul este 1.
 - Dacă $170 < cnt \leq 1000$, atunci scorul este $\min((\frac{170+3000}{cnt+3000})^{\frac{3000}{x}}, 1)$.
 - Dacă $1000 < cnt \leq 20\,000$, atunci scorul este $\min((\frac{170}{cnt})^{0.4}, 0.5)$.
 - Dacă $20\,000 < cnt \leq \frac{10^8}{N}$, atunci scorul este 0.1.

Exemplu de comunicare

Să considerăm următoarea ilustrare a unei hărți cu 5 spații și 4 conexiuni:



| Programul tău | Răspunsul juriului |
|---|------------------------|
| | find_connections(5) |
| check({2, 3, 4, 5, 1}) | return {1, 1, 1, 1, 0} |
| check({1, 5, 2, 3, 4}) | return {0, 1, 0, 0, 0} |
| check({1, 4, 3, 2, 5}) | return {0, 1, 1, 0, 0} |
| return {{1, 2}, {2, 3}, {3, 4}, {2, 5}} | |

Testare Locală

Pentru testare locală sunt furnizate următoarele fișiere: `monopoly2.h`, `Lgrader.cpp`, fișierul exemplu `monopoly2.cpp` pentru programul tău și fișierul cu planșa din exemplul de comunicare. Când aceste fișiere sunt în același director, poți compila împreună programul tău `monopoly2.cpp` și `Lgrader.cpp`. Aceasta va crea un program care verifică corectitudinea funcției tale.

Programul va cere de la intrarea standard următoarea secvență de numere:

- Pe prima linie: un număr întreg pozitiv - numărul de spații N ;
- Pe fiecare dintre următoarele $N - 1$ linii: două numere întregi pozitive care descriu conexiunile direcționate.

Dacă nu respectați protocolul de comunicare, veți primi un mesaj de eroare corespunzător. În caz contrar, dacă programul reușește, veți primi mesajul "Correctly found connections."