

## Задатак C23. MONOPOLY 2

 1 sec.  256 MB

Након што си помогао **Ирини** да направи савршену мапу за играње монопола у 2021, њој је опет потребна твоја помоћ! Кажемо да се мапа за играње монопола састоји од  $N$  поља (означена бројевима од 1 до  $N$ ) и  $M$  усмерених веза између њих. Важи  $M = N - 1$ . Усмерене везе задовољавају следеће услове - не постоји веза од поља до истог тог поља и не постоји више различитих веза између истог пара поља. Такође, везе задовољавају да је свако поље достижно од поља 1.

**Дефиниција:** Посматрајмо пермутацију бројева од 1 до  $N$ . Пермутацију зовемо *задовољавајућим* редоследом поља ако испуњава следећи услов - за сваку везу од поља  $i$  до поља  $j$ ,  $i$  се мора налазити **пре**  $j$  у пермутацији.

**Иринин** проблем је то што је она изгубила савршену мапу за играње монопола. На сву срећу, њен друг Михајло се сећа мапе али је одлучио да игра игру са њом пре него што јој каже везе. На почетку, Михајло каже **Ирини** да је *задовољавајући* редослед поља секвенца бројева  $1, 2, \dots, N$ . Након тога Михајло ће одговорити на нека **Иринина** питања како би јој помогао да открије све везе између поља. Свако питање ће бити о томе колико је *задовољавајућа* нека пермутација бројева од 1 до  $N$  (за више детаља погледати секцију "Детаљи имплементације"). Наша јунакиња је затражила твоју помоћ да смисли и имплементира стратегију користећи што мање упита.

### Задатак

Написати програм **monopoly2** који налази непознате везе са што је могуће мање постављених питања Михајлу. Он мора имати функцију `find_connections` која ће бити компајлирана на комисијском грејдеру (у улози Михајла).

### Детаљи имплементације

Твоја функција `find_connections` мора бити следећег формата:

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

Она ће бити позвана једном од стране комисијског програма са једним параметром - број поља. Функција мора вратити листу уређених парова који представљају пронађене везе између поља. Редослед уређених парова у листи није битан.

Функција за постављање питања Михајлу мора бити следећег формата:  
`std::vector <bool> check (std::vector <int> p);`

Параметар  $p$  је пермутација бројева од 1 до  $N$  за коју желите да добијете одговор. Резултат је вектор (vector) булова (bool)  $b$  дужине  $N$ , где је  $b_i = 1$  ( $0 \leq i < N$ ) ако и само ако бар једно од следећег важи:

- постоји веза од поља  $p_j$  до поља  $p_i$  где важи  $i < j$ ;
- постоји поље  $p_j$  за које је  $b_j = 1$  и можеш се кретати преко веза од поља  $p_j$  до поља  $p_i$ .

Приметити да једнозначност вредности следи из услова задатка.

Ако секвенца није валидна пермутација бројева од 1 до  $N$ , добићеш `Wrong answer` за тест пример. Сложеност функције је  $O(N)$ . Ти можеш позвати функцију

## XVI INTERNATIONAL ADVANCED TOURNAMENT IN INFORMATICS SHUMEN 2025

највише  $\frac{10^8}{N}$  пута, иначе добићеш Wrong answer.

Твој програм **monopoly2** мора имплементирати функцију `find_connections`. Он такође може садржати други код, функције, и глобалне променљиве, али он не сме садржати `main` функцију. Такође, ти не треба да читаш улаз са стандардног улаза или исписујеш у стандардни излаз. Твој програм мора укључити (`include`) хедер фајл `monopoly2.h` по инструкцији за претпроцесор:

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

### Ограничења

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

### Подзадаци

Подзадатак	Поени	Неопходни подзадаци	$N$	Остала ограничења
0	0	—	—	Мапа из текста задатка.
1	5	—	$\leq 1\,000$	$1, 2, \dots, N$ је једини задовољавајући редослед поља.
2	6	0	$\leq 6$	—
3	17	1	$\leq 1\,000$	$1, 2, \dots, N$ може се добити на следећи начин: почињемо са пољем 1, након тага додајемо у листу поља са директним везама од 1, након тога додајемо у листу поља са директним везама од друго додатог поља и тако даље.
4	13	0, 2	$\leq 300$	—
5	59	0 — 4	$\leq 1\,000$	Ако почнемо од поља 1 и пратимо везе, ми не можемо користити више од 25 веза.

Поени за подзадатак се освајају само ако сви тестови у њему и неопходним подзадацима су **успешно** прошли, и поени су једнаки најмањем резултату неког примера у њему и неопходним подзадацима, помножени са бројем поена подзадатка.

### Бодовање

Сваки тест добија резултат који је децимални број између 0 и 1 укључујући и 0 и 1. Ако тест има позитиван резултат, он се сматра **успешним** за твоје решење. Тест има позитивни резултат ако ти успешно пронађеш све везе између поља.

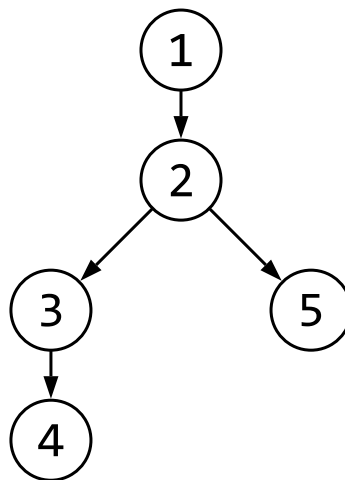
Нека је `cnt` број позива функције `check` за неки тест, онда се резултат тог теста рачуна на следећи начин:

- За све подзадатке: ако  $cnt > \frac{10^8}{N}$ , онда је резултат једнак 0.
- Подзадаци 0, 1, и 2.

- Ако је  $cnt \leq \frac{10^8}{N}$ , онда је резултат једнак 1.
- Подзадатак 3.
  - Ако је  $cnt \leq 10\,000$ , онда је резултат једнак 1.
  - Ако је  $10\,000 < cnt \leq 15\,000$ , онда је резултат једнак  $\min(\frac{1000}{cnt-9000}, 1)$ .
  - Ако је  $15\,000 < cnt \leq \frac{10^8}{N}$ , онда је резултат једнак 0.1.
- Подзадатак 4.
  - Ако је  $cnt \leq 50\,000$ , онда је резултат једнак 1.
  - Ако је  $50\,000 < cnt \leq 200\,000$ , онда је резултат једнак  $\min(\frac{25000}{cnt-25000}, 1)$ .
  - Ако је  $200\,000 < cnt \leq \frac{10^8}{N}$ , онда је резултат једнак 0.1.
- Подзадатак 5.
  - Ако је  $cnt \leq 170$ , онда је резултат једнак 1.
  - Ако је  $170 < cnt \leq 1000$ , онда је резултат једнак  $\min((\frac{170+3000}{cnt+3000})^{\frac{3000}{x}}, 1)$ .
  - Ако је  $1000 < cnt \leq 20\,000$ , онда је резултат једнак  $\min((\frac{170}{cnt})^{0.4}, 0.5)$ .
  - Ако је  $20\,000 < cnt \leq \frac{10^8}{N}$ , онда је резултат једнак 0.1.

#### Комуникација из текста задатка (Sample communication)

Нека имамо следећу илустрацију мапе са 5 поља и 4 везе:



Кораци твог програма	Кораци и одговори комисије
	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}}	

#### Локално тестирање

За локално тестирање дати су следећи фајлови: monopoly2.h, Lgrader.cpp, пример фајла monopoly2.cpp за твој програм и фајл са мапом из комуникације из текста задатка. Када су дати фајлови у истој датотеци (folderu), ти можеш

компајлирати заједно `monopoly2.cpp` и `Lgrader.cpp`. Ово ће направити програм који ће проверити тачност твоје функције.

Програм ће захтевати са стандарног улаза следећу секвенцу бројева:

- на првој линији: један позитиван број – број поља  $N$ ;
- на свакој од следећих  $N - 1$  линија два позитивна цела броја који представљају усмерену везу.

Ако ти не пратиш протокол за комуникацију, ти ћеш добити одговарајућу поруку о грешци. У супротном, ако је програм успешан, ти ћеш добити поруку "Correctly found connections."