A nyakkendők irányítanak (incursion)

Most, hogy felvettél egy asszisztenst a roboteladásokból származó nyereségből, készen állsz arra, hogy a széfért menj, amely a CEOI érmeket tartalmazza!

A széf az egyik egyetemi épületben található, amely N szobából áll, ezeket $N - 1$ ajtó köti össze. Minden szoba bármelyik másikból elérhető és minden szobának legfeljebb 3 ajtaja van. Neked is és az asszisztensednek is van egy alaprajza az épületről - de sajnos két különböző kiadásban: bár mindkét terv ugyanazt az alaprajzot mutatja, *a szobák és az ajtók számozása eltérhet*.

A második versenynapon a bizottság a versenyzőktől jövő kérésekkel lesz elfoglalva. Ez a tökéletes idő, hogy megszerezd a széfet az érmekkel. Először az asszisztensed kutatja át az épületet. Mihelyt megtalálta a széfet tartalmazó szobát, odavezet téged*. Mivel a verseny helyszínére nem viheted be a telefonodat, az asszisztensed titkos jeleket hagy a szobákban. Ehhez felhasználja a tavalyi BOI-ból megmaradt, gyakorlatilag végtelen mennyiségű nyakkendőt. Mivel ezek a nyakkendők *megkülönböztethetetlenek* egymástól, így az asszisztensed által egy adott teremben hátrahagyott nyakkendők száma adhat csak információt. Mivel túl sok nyakkendő egy helyen gyanúsnak tűnhet, az *egyes szobákban hagyott nyakkendők maximális számának a lehető legkisebbnek kell lennie* (lásd az értékelésről szóló részt).

Kicsit vársz, majd kimész mosdóba, továbbosonsz az épületben, és a hátrahagyott nyakkendők segítségével megtalálod a szobát, ahol a széf van. Vigyázz, a széf el van rejtve a szobában, így a szobába lépéssel nem fogod megtalálni, helyette az ott levő nyakkendőkre kell hagyatkoznod. Mivel a távolléted nem tűnhet fel, gyorsan meg kell találnod a széfet: *legfeljebb $d + 30$ ajtón szabad áthaladnod, ahol d az ajtók száma a kiindulási helyedtől a széfet tartalmazó szobáig vezető közvetlen útvonalon.* Ha egy ajtón többször is áthaladsz, minden egyes áthaladás számít.

Írj programot, amely

- megmondja az asszisztensednek, hogy mennyi nyakkendőt hagyjon hátra az egyes szobákban, *és*
- később elvezet a széfet tartalmazó szobába.

Kommunikáció

Ez egy interaktív feladat, amelyben a programod minden egyes tesztesetnél *kétszer* fut le. A következő két függvényt kell megvalósítanod:

- **vector**⟨**int**⟩ *mark*(**vector**⟨**pair**⟨**int**, **int**⟩⟩ F, **int** *safe*) ahol
  - F az ajtók megadásával írja le az alaprajzot: F tartalmaz $N - 1$ egész (u, v) számpárt, ahol $1 \le u, v \le N$ és $u \ne v$, ami azt jelenti, hogy az u és v szobák között van ajtó;
  - *safe* a széfet tartalmazó szoba.

  A függvényednek egy T vektort kell visszaadnia, amely N egész számból áll, $T[0], \ldots, T[N-1]$, $0 \le T[i] \le 10^9$: $T[i]$ az asszisztensed által a $i + 1$. szobában hagyott nyakkendők száma. *A $T[i]$ számoknak a lehető legkisebbeknek kell lenniük.*
- **void** *locate*(**vector**⟨**pair**⟨**int**, **int**⟩⟩ F, **int** *curr*, **int** t) ahol F ugyanazt az alaprajzot írja le (de esetleg más szobaszámozással és más ajtósorrenddel!), *curr* az a szoba, amelyben éppen tartózkodsz, és t az ott talált nyakkendők száma.

* Végtére is, Te magad akarod kinyitni a széfet - nem lenne vicces, ha valaki más nyitná ki, és az érmet rögtön a verseny után adná át neked!

Ezen a függvényen belül meghívhatod az értékelő **int** *visit*(**int** *v*) függvényét, hogy az aktuális szobából a *v* szobába lépj (a Te alaprajzod szobaszámozása szerint, $1 \le v \le N$); ez a függvény visszaadja az asszisztensed által a *v* szobában hagyott nyakkendők számát. A *v* szobát és az aktuális szobát ajtónak kell összekötnie, és ezt a függvényt tesztesetenként csak korlátozott számban hívhatod meg (lásd fent).

Amikor a *locate* függvényed visszatér, a széfet tartalmazó szobában kell lenned.

Minden egyes tesztesetnél az első futtatásban a *mark* függvényt hívjuk meg, míg a második futtatásban a *locate* függvényt.

Ha a *visit* függvényt túl gyakran, érvénytelen paraméterekkel vagy a program első futtatása során hívod meg, akkor a beadás azonnal megszakad, és a megfelelő tesztesetben a **Not correct** értékelést kapod. Nem szabad a szabványos kimenetre írnod vagy a szabványos bemenetről olvasnod; ellenkező esetben a **Security violation!** értékelést kaphatod. A szabványos hibastreambe azonban szabadon írhatsz (*stderr*).

A `incursion.h` fájlt be kell építened (include) a forráskódodba. A program helyi teszteléséhez a `sample_grader.cpp` állományt használhatod, amely a CMS-ben a feladathoz tartozó mellékletek közt található. A mintatesztelő használatának leírását lásd alább, és a `sample_grader.cpp`-ben találsz segítséget arra vonatkozóan is, hogyan futtasd a programoddal. Figyelj arra, hogy az egyszerűség kedvéért ez a mintaértékelő nem futtatja le kétszer a programodat, hanem a *mark* és a *locate* függvényt (pontosan egyszer mindegyiket) egyetlen futtatás részeként hívja meg. A melléklet tartalmaz egy minta megvalósítást is, `incursion_sample.cpp` néven.

Korlátok és értékelés

Mindig teljesül $2 \le N \le 45\,000$.

Részfeladat 1 (30 pont). Nincs olyan szoba, aminek 3 ajtaja lenne.

Részfeladat 2 (30 pont). Pontosan egyetlen szobában van 2 ajtó.

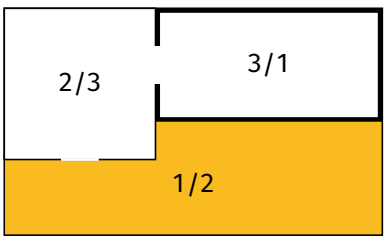
Részfeladat 3 (40 pont). Nincs további megkötés.

Részpontok. Minden részfeladatban, a tényleges pontszámod az asszisztensed által az egyes szobákban hátrahagyott nyakkendők maximális számától függ: $T_{\max}$ (azaz a *mark* függvény visszatérési értékében szereplő legnagyobb $T[i]$ számtól) az alábbi táblázat szerint:

$T_{\max}$	$[0, 1]$	2	$[3, 10^9]$
érték	100%	40%	30%

Minta interakció

Nézzük a tesztesetet az $N = 3$ értékkel és a következő alaprajzzal:



Az egyes szobákban az első szám az asszisztens számozására, a második pedig a Te számozásodra utal. A széf az alsó nagy szobában található (sárga színnel árnyékolva), míg a te kiindulási helyed a jobb felső szobában van (félkövér körvonallal rajzolva).

A programod első futtatásakor az értékelő a *mark* függvényedet *mark*({{1, 2}, {2, 3}}, 1) paraméterekkel hívja meg. Tegyük fel, hogy ez {2, 2, 0}-t ad vissza, azaz az asszisztensed két nyakkendőt hagy hátra az 1 és a 2 szobában, de a 3 szobában nincs nyakkendő.

A programod második futtatásakor az értékelő meghívja a *locate* függvényedet *locate*({{1, 3}, {2, 3}}, 1, 0) paraméterekkel. Ezután a programod és az értékelő közötti interakció a következőképpen néz ki:

A Te programod	Visszaadott érték	Magyarázat
visit(3)		A Te számozásod szerint a 3 szobát látogattad meg
	2	Ez a szoba a 2 az asszisztensed számozásában, amelyben 2 nyakkendőt hagyott
visit(1)		Újra az 1, kiindulási szobádba mész ...
	0	... az asszisztensed 3 szobájába jutsz, ahol nincs nyakkendő
visit(3)		Újra a 3 szobádba mész...
	2	... ahol még mindig 2 nyakkendőt találsz
visit(2)		A 2 szobádba mész...
	2	... Az asszisztensed 1 szobája, két nyakkendővel
return		Biztos vagy benne, hogy a 2 szoba tartalmazza a széfet
		mivel ez az asszisztensed számozásában az 1 szobának felel meg, a programod helyes.

Itt négy ajtón mentél át, míg a kiindulási helytől a széfet tartalmazó szobáig vezető közvetlen útvonalon az ajtók d száma kettő.

A fenti interakciót a `incursion_sample.cpp` a mintateszten futtatva adta.

Értékelő

A mintaértékelő először a standard bemeneten az N és a *safe* egész számokat várja. Ezután az F alaprajzot $N - 1$ darab egész (u, v) számpár ($1 \le u, v \le N$ és $(u \ne v)$) sorozataként várja.

Ezután meghívja a *mark*(F, *safe*) parancsot, és a T visszatérési értéket kiírja a standard kimenetre. Ezután a *curr* kiindulási helyedet várja, és meghívja a *locate*(F, *curr*, T[*curr* – 1]) függvényt; *megjegyzés: a mintaértékelő nem változtatja meg a szobák és ajtók számozását.* Ezután a szabványos kimenetre írja a programod által a *visit* függvény minden hívásának kimenetét.

Befejezéskor a következő üzenetek egyikét írja a standard kimenetre:

Invalid input. Az értékelő a standard bemeneten keresztül nem a fenti formátumban kapta az adatokat.

Invalid call to visit. A *visit* függvényt a *mark* függvényből vagy helytelen paraméterekkel hívtad.

Invalid return value of mark. A *mark* függvény nem az N hosszúságú T vektorral tért vissza vagy a $T[i]$ értékek valamelyike negatív vagy nagyobb mint 10^9.

Correct: at most T_{max} tie(s) per room. A *locate* függvény a *safe* szobából tért vissza, és a T vektorban a legnagyobb érték T_{max}.

Not correct: current position is *curr*. A *locate* függvény a *curr* ≠ *safe* szobából tért vissza.

Ezzel szemben a ténylegesen, a CMS-ben használt értékelő csak **Not correct** (a fenti hibák bármelyikére), a **Security violation!**, a **Partially correct** vagy a **Correct** értékelést adja. Az értékelő *adaptív*, azaz a futása függhet a programod viselkedésétől (az aktuális és a korábbi hívásoktól is). Mind a mintaértékelő, mind a programod elbírálására használt értékelő automatikusan megszakítja a programodat, ha a fenti hibák valamelyike előfordul.

Határok

Idő: 0,5 s
Memória: 512 MiB

Minden tesztesetnél az idő- és memóriakorlátok a program két futtatására *külön-külön* érvényesülnek.