Hogyan kerüld el a kizárást 75 egyszerű lépésben (avoid)

Ott állsz a nyitott széf előtt, kezedben egy éremmel. Ám diadalod kétségbeesésbe fordul, ahogy körülnézel: a szobában talált nyakkendőn lévő üzenetből kiderül, hogy az asszisztensed leleplezte a tervedet a Tudományos Bizottság előtt. Most a bizottság két elnöke az épületben rejtőzködik, hogy megakadályozza a szökésedet...

Szerencsére maradt *R* porszívórobotod a versenyzőtársaiddal folytatott kereskedelemből. Ezekkel a robotokkal akarod megtalálni a két elnököt, hogy elkerülhesd őket a menekülésed során. Minden robotot utasíthatsz arra, hogy több 1 000 pozíciót is felderítsen, ahol az elnökök lehetnek. Sajnos ezeknek a robotoknak a szoftvere elég egyszerű.* Minden robot csak azt tudja felismerni, hogy *legalább egy elnök van-e az általa felderített pozíciókban* vagy sem.

A helyzetet tovább rontja, hogy minden robotnak egy teljes órára van szüksége ahhoz, hogy felderítse a pozícióit, mielőtt visszajön az eredményével hozzád. Ez lemeríti a robotok akkumulátorát, így *minden robotot csak egyszer küldhetsz ki*.

Mivel nem akarsz elkésni az esti tevékenységekről, *legfeljebb H óra elteltével* szeretnéd tudni az elnökök pozícióit. Előfordulhat, hogy egyszerre több robotot kell kiküldened anélkül, hogy megvárnád az előző robotok visszatérését. Feltételezhetjük, hogy a két elnök mindig ugyanazon a helyen marad.†

Írj programot, amely megtervezi ezt a felderítő feladatot és meghatározza, hogy hol van a két elnök.

Interakció

Ez egy interaktív feladat. A következő függvényeket kell megvalósítanod: **pair**⟨**int**, **int**⟩ *scout*(**int** *R*, **int** *H*) ahol *R* és *H* a fent leírtaknak megfelelő. Minden egyes tesztesetnél ez a függvény pontosan egyszer hívódik meg és két egész számpárt kell visszaadnia $1 \le a, b \le 1\,000$ ($a = b$ megengedett), a két elnök pozícióját. A *scout*-n belül a következő, az értékelő által biztosított függvényeket használhatod:

- **void** *send*(**vector**⟨**int**⟩ *P*) egy robotot küld a $P[0], \ldots, P[k-1]$ pozíciók felderítésére (ahol *k* a *P* tömb hossza). A $P[i]$ pozícióknak páronként különböző egész számoknak kell lenniük 1 és 1 000 között. Ezt a függvényt tesztesetenként legfeljebb *R* alkalommal hívhatjuk meg.
- **vector**⟨**int**⟩ *wait*() egy órát vár. Ez a függvény egy olyan tömböt ad vissza, amely pontosan egy bejegyzést tartalmaz minden egyes robotra, amelyet egy órával ezelőtt küldtünk ki (a *send* hívásával az előző *wait* hívás után vagy a program kezdete után). Az i. indexű bejegyzés 1, ha az $(i+1)$. robot legalább egy elnököt észlelt a felderített pozícióiban, és `0` egyébként. Ezt a függvényt tesztesetenként legfeljebb *H* alkalommal hívhatjuk meg.

Ha a visszatérési értékek bármelyike nem felel meg a fenti feltételeknek, a program azonnal megszakad, és az adott tesztesetben **Not correct** értéket kap. Nem írhatsz a standard kimenetre és nem olvashatsz a standard bemenetről; ellenkező esetben a **Security violation!** értékelést kaphatod. Azonban a szabványos hibafolyamba (error stream) szabadon írhatsz (*stderr*).

Az `avoid.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. A melléklet tartalmaz egy minta megvalósítást is, `avoid_sample.cpp` néven.

* Hát mégiscsak eladtad a többi versenyzőnek az összes menőt!
† A késő esti feladat előkészületei miatt egyszerűen túl kimerültek ahhoz, hogy megmozduljanak.

Korlátok és értékelés

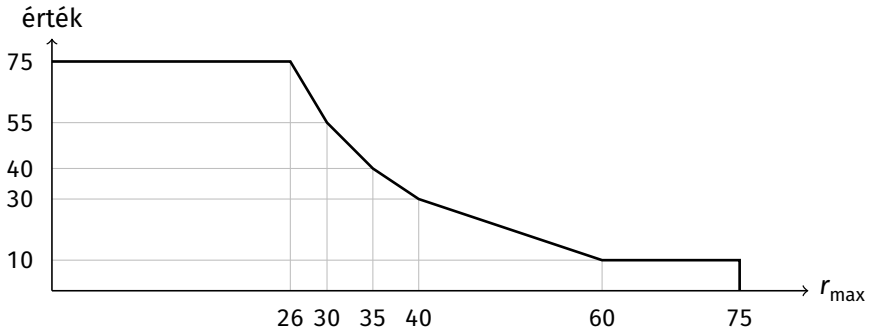
Részfeladat 1 (10 pont). $R = 10$, $H = 1$ és minden elnök ugyanabban a pozícióban van.

Részfeladat 2 (5 pont). $R = H = 20$

Részfeladat 3 (10 pont). $R = 30$, $H = 2$

Részfeladat 4 (75 pont). $R = 75$, $H = 1$

Részpontozás. A 4-es részfeladatban a tényleges pontszámod az adott részfeladat összes tesztesetében kiküldött robotok maximális $r_{\max}$ számától függ, a következő, intervallumonként lineáris függvény szerint:



A teljes pontszám eléréséhez az utolsó részfeladatban tesztesetenként nem szabad 26-nál többször meghívni a *send* függvényt. Az összes, egyéni pontszámot felsoroló táblázatot is megtalálhatod a feladathoz csatolt mellékletek közt, `score_table.txt` néven.

Minta interakció

Tekintsünk egy $R = 75$ és $H = 20$ értékű tesztesetet, ahol az elnökök a 13-as és a 37-es pozíciókban helyezkednek el. Először az értékelő meghívja a *scout* függvényedet, *scout*(75, 20) alakban. Ezután a programod és az értékelő közötti interakció a következőképpen nézhet ki:

A Te programod	Visszaadott érték	Magyarázat
send({42, 13, 37})	—	küld egy robotot a 13-as, 37-es és 42-es pozícióra
send({47, 11})	—	küld egy robotot a 11-es és 47-es pozícióra
wait()	{1, 0}	vár egy órát a robotok visszatérésére; csak az első robot észlelt egy elnököt
send({42})	—	elküld egy robotot a 42-es pozícióra
wait()	{0}	vár egy órát; nincs elnök a 42-es pozíción
return {13, 37}	—	meg vagy győződve, hogy az elnök a 13-as és a 37-es pozíciókban helyezkednek el
		a megoldás helyes és elfogadott

A {37, 13} pár visszaadása is elfogadható. Megjegyezzük, hogy a fenti lekérdezések természetesen nem elegendőek az elnökök pozícióinak biztos meghatározásához: Például az, hogy mindketten a 37-es pozícióban vannak, vagy az egyik elnök a 13-as pozícióban van, míg a másik a 100-as pozícióban, szintén összhangban lenne a *wait* összes válaszával, így az osztályozó ezt a megoldást el is utasíthatta volna.

A fenti interakciót az `avoid_sample.cpp` reprodukálja az elérhető tesztesetben.

Értékelő

A minta értékelő először a standard bemeneten az R és H egész számokat és az elnökök a és b pozícióit várja ($1 \le a, b \le 1\,000$). Ezután az értékelő meghívja a *scout*(R, H) függvényt és a standard kimenetre írja a programod által meghívott összes függvény kimenetét. A program befejezésekor a következő üzenetek egyikét írja a standard kimenetre:

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

Invalid send. A *send* függvényt helytelen paraméterekkel hívtad meg.

Out of robots. A *send* függvényt több mint R alkalommal hívtad meg.

Out of time. A *wait* függvényt több mint H alkalommal hívtad meg.

Wrong answer. A *scout* által visszaadott pozíciók nem egyeznek meg az elnökök pozícióival.

Correct: *r* robot(s) used, *h* hour(s) passed. A *scout* által visszaadott pozíciók az elnökök pozíciói, r hívás volt a *send* függvénnyel, és h hívás volt a *wait* függvénnyel.

Ezzel szemben a ténylegesen, a CMS-ben használt értékelő csak **Not correct** (a fenti hibák bármelyikére), a **Security violation!**, vagy a **Correct: *r* robot(s) used, *h* hour(s) passed** értékelést adja. Az értékelő *adaptív*, azaz az elnökök pozíciói függhetnek 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: 0,25 s
Memória: 512 MiB