KOSMOS

Była późna godzina piątkowego wieczoru, gdy jasnowidz January i archeolog Archibald wyszli z budynku międzygalaktycznej superkorporacji Kontener 13. Byli bardzo zmęczeni po wielogodzinnych negocjacjach, ale też zadowoleni z ich rezultatów. Podpisali kontrakt na transport poufnych danych zebranych w starożytnych ruinach w różnych zakątkach galaktyki. Jednak ich kontrahenci nie wiedzieli, że Archibald odkrył sposób na rozszyfrowanie supertajnych kodów SSH przy pomocy danych z ruin. Za rozszyfrowane kody Naukowa Akademia Szyfrowania jest gotowa zapłacić grube miliony. January natomiast jest w stanie przewidzieć, z których ruin pochodzą dane potrzebne do rozszyfrowania danego kodu. Niepokoiło ich jednak to, że z nieznanych im przyczyn negocjacje z innymi superkorporacjami takimi jak Kontener 2 czy Kontener 7 nie doszły do skutku. Jako że owe superkorporacje niespodziewanie ogłosiły bankrucwto, czego, z niewiadomych przyczyn, nie potrafił przewidzieć nawet sam January. Ale tymi i innymi problemami zajmą się dopiero w poniedziałek...

Ogólny opis rozgrywki

Gracze rywalizują ze sobą na wspólnej planszy. Plansza jest dyskretna i nieograniczona, reprezentuje kosmos.

W grze występują trzy rodzaje obiektów, które mają pozycje – planety, statki dowodzenia oraz satelity. Obiekty reprezentowane są przez punkty na płaszczyźnie, o współrzędnych całkowitoliczbowych. Planety nie ruszają się w trakcie rundy.

Każdy z graczy steruje pewną liczbą satelit i statkiem dowodzenia, na początku posiada tylko statek dowodzenia. Celem gracza jest zdobycie możliwie największej ilości punktów. Gracz otrzymuje punkty oraz paktolary (w takich samych ilościach) za dostarczanie pakietów (tajnych danych) z planet do swojego statku dowodzenia i wykonywanie zadań, czyli dostarczenie określonego zbioru pakietów do statku dowodzenia. Nagroda za dostarczenie danego pakietu i zrealizowanie zadania jest przyznawana tylko raz. Jeśli któraś z drużyn wykona wszystkie zadania lub dostarczy wszystkie pakiety to gra się kończy.

Dokładny opis rozgrywki

Rozgrywki dzielą się na tury, jedna tura trwa około $\frac{1}{FPS}$ sekundy (parametr FPS zależy od serwera). W trakcie tury gracze wykonują polecenie. Po wykonaniu poleceń, zmiany są widoczne w następnej turze. Polecenia typu ruch, transmitowanie, odbieranie można wykonać nie więcej niż 1 na turę.

Ruch

Wszystkim obiektom gracza można podać komendę ruchu, która przyjmuje wektor **pos**, pozycję, jako parametr. Jeśli odległość euklidesowa między aktualną pozycją, a **pos** jest nie większa od prędkości obiektu to w następnej turze pozycja obiektu zostanie zmieniona na **pos**.

Transmitowanie

Satelity mogą transmitować pakiety, które posiadają w swojej pamięci. W następnej turze po wysłaniu pakietu daną transmisję można odebrać. Transmisje może odebrać ktokolwiek, nawet inny gracz, jeśli spełnia wymagania zawarte w sekcji Odbieranie.

Odbieranie

Jeśli w poprzedniej turze została nadana transmisja **T** to w tej turze dany satelita **A** będzie ją mógł odebrać o ile **SINR_A** będzie większy niż czułość odbiornika. Przesył zawsze będzie udany, jeśli dwie twoje satelity będą na tym samym polu. Ponadto, zawsze można odebrać pakiet z planety jeżeli satelita jest (od niej) w odległości mniejszej niż promień tej planety.

Wzory:

$$\textbf{siła\_transmisji}_i = \frac{\textbf{siła\_nadawania}_i}{(\textbf{dystans\_od\_źródła}+\textbf{stała}_a)^2}$$

($\textbf{stała}_a$ jest w odpowiedzi na polecenie GET_CONSTANTS)

$$\textbf{SINR}_i = \frac{\textbf{siła\_transmisji}_i}{\sum \textbf{siła\_transmisji}_j}$$

(bez transmisji wysłanej przez nasłuchującego satelitę w poprzedniej turze)

Dostarczanie

Jeśli satelita znajduje się w odległości **długości kabla** (stała podana w odpowiedzi na polecenie GET_CONSTANTS) lub jeżeli znajduje się w odległości od planety, na której jest dany pakiet, mniejszej niż jej promień, to może dostarczyć pakiet do statku dowodzenia. Takiego przesyłu informacji nie da się podsłuchać ani zakłócić.

Ulepszanie

Satelity można ulepszać. Do ulepszenia są 4 moduły:

- silnik – prędkość satelity
- pamięć – ile pakietów może przechowywać satelita
- nadajnik – jaka jest maksymalna moc nadawanego sygnału
- odbiornik – jaka jest czułość odbiornika

Opis komend

Przy parametrach przyjmowanych/zwracanych przez komendy podane są ich typy:

- (int) – liczby całkowite
- (real) – liczby zmiennoprzecinkowe
- (string) – ciąg znaków

WAIT

Natychmiastowo zwraca OK (jak każda inna komenda), po czym zwraca dodatkowe OK na początku nowej tury (tj. tuż po przetworzeniu poprzedniej tury). Przykład:

```
> WAIT
< OK
(po pewnym czasie...)
< OK
```

TURNS_LEFT

Zwraca kolejno:

- liczbę tur, które pozostały do zakończenia rozgrywki (int)
- ilość tur do końca czasu odnowienia skanu (int)
- 1 jeżeli jest to pierwsza tura nowej rozgrywki, lub 0 w przeciwnym przypadku (int)

Przykład:

```
> TURNS_LEFT
< OK
< 4534 5 0
```

GET_CONSTANTS

Podaje parametry gry. Są to kolejno:

- cena jednego satelity (int)
- prędkość statku dowodzenia (int)
- długość kabla statku dowodzenia (int)
- liczba tur potrzebna na odnowienie skanera (int)
- stała do równania na siłę transmisji (int)

Przykład:
```
> GET_CONSTANTS
< OK
< 900 75 300 10 123
```

GET_PLANETS

W pierwszej linii zwraca liczbę planet w galaktyce, w kolejnych **n** liniach zwraca 4 liczby opisujące planetę w kolejności:

- id planety (int)
- współrzędna X (int)
- współrzędna Y (int)
- promień planety (z jakiej odległości można pobierać pakiety) (int)

Przykład:
```
> GET_PLANETS
< OK
< 3
< 0 2 3 4
< 1 -10 510 14
< 2 13 -1 12
```

BUY_SATELLITE

Kupuje satelitę o podstawowych parametrach, który pojawia się w tym samym miejscu co statek dowodzenia gracza. Przykład:
```
> BUY_SATELLITE
< OK
```

GET_MOTHERSHIP

Zwraca współrzędne X (int) i Y (int) statku dowodzenia gracza. Przykład:
```
> GET_MOTHERSHIP
< OK
< 10 -10
```

GET_SATELLITES

W pierwszej linii zwraca liczbę satelit gracza, w kolejnych **n** liniach zwraca 11 + **k** liczb opisujących satelity w kolejności:

- id satelity (int)
- współrzędna X (int)
- współrzędna Y (int)
- prędkość (int)
- **k** rozmiar pamięci (int)
- maksymalna siła sygnału (int)
- minimalny SINR transmisji by móc odebrać (real)
- poziom ulepszenia silnika (int)
- poziom ulepszenia pamięci (int)
- poziom ulepszenia nadajnika (int)
- poziom ulepszenia odbiornika (int)
- zwraca **k** liczb – id pakietów w pamięci satelity (-1 oznacza, że dane miejsce w pamięci jest puste) (int)

Przykład:
```
> GET_SATELLITES
< OK < 2
< 1 0 0 100 3 600 0.3 0 0 1 0 -1 3 5
< 5 10 -10 140 5 1000 0.2 2 2 5 3 10 -1 -1 13 7
```

MOVE_SATELLITE

Przesuwa satelitę na daną pozycję, o ile to możliwe. W komendzie należy podać:

- id satelity (int)
- współrzędna X nowej pozycji (int)
- współrzędna Y nowej pozycji (int)

```
> MOVE_SATELLITE 1 1000 712
< OK
```

MOVE_MOTHERSHIP

Przesuwa statek dowodzenia na daną pozycję, o ile to możliwe. W komendzie należy podać:

- współrzędna X nowej pozycji
- współrzędna Y nowej pozycji

Przykład:
```
> MOVE_MOTHERSHIP 1000 712
< OK
```

TRANSMIT_PACKET

Wysyła pakiet z satelity w ether. W komendzie należy podać:

- id satelity (int)
- numer indeksu pamięci satelity, pod którym zapisany jest pakiet (int)
- siła sygnału nadawania (int)

Przykład:
```
> TRANSMIT_PACKET 1 3 460
< OK
```

RECEIVE_PACKET

Wybrany satelita odbiera pakiet wysłany w poprzedniej turze, o ile to możliwe. W komendzie należy podać:

- id satelity (int)
- numer pakietu do odebrania (int)
- indeks pamięci, w którym pakiet zostanie zapisany, indeksowany od 0 (int)

Przykład:
```
> RECEIVE_PACKET 1 10 2
< OK
```

DELIVER_PACKET

Statek dowodzenia odbiera pakiet wysłany w poprzedniej turze, o ile w zasięgu jego kabla znajduje się satelita, który ma w pamięci dany pakiet. Gracz dostaje za niego punkty i paktolary (taką samą ilość). W komendzie należy podać:

- numer pakietu do dostarczenia (int)

Przykład:
```
> DELIVER_PACKET 10
< OK
```

GET_TRANSMISSIONS

Zwraca pakiety jakie może odebrać satelita i innych satelit (nie z planet). Jako parametr przyjmuje id satelity (int), dla której polecenie ma zwrócić dostępne transmisje. W pierwszej linii **n** znajduje się jedna liczba oznaczająca liczbę transmisji, które można odebrać, w kolejnych **n** liniach znajduje się opis ww. transmisji.

- id pakietu (int)
- id satelity źródłowego (int)
- id właściciela satelity źródłowego (int)
- współrzędna X źródła sygnału
- współrzędna Y źródła sygnału
- siła odbieranego sygnału (real)
- SINR dla danej transmisji (real)

Przykład:
```
> GET_TRANSMISSIONS
< OK
< 3
< 1 2 3 343 34 40.05 0.45
< 4 5 1 -300 24 26.7 0.3
< 10 10 10 200 -213 13.35 0.15
```

GET_TASKS

Zwraca dostępne zadania. W pierwszej linii **n** znajduje się jedna liczba oznaczająca liczbę zadań, w kolejnych **3n** liniach znajdują się 3 linijkowe opisy ww. transmiji.
1 linia opisu

- liczba **k** pakietów w zadaniu (int)
- punkty/paktolary za pakiet (int)
- punkty/paktolary za wykonanie zadania (int)

2 linia opisu - **k** par, składających się z id pakietów (int) w zadaniu i id planety (int), z której można odebrać dany pakiet
3 linia opisu - **k** liczb ze zbioru {0, 1}, gdzie 0/1 oznacza, że dany pakiet nie/- został dostarczony
Przykład:
```
> GET_TASKS
< OK
< 2
< 3 7 20
< 3 2 4 2 6 1
< 0 0 0
< 5 1 10
< 10 7 11 7 12 8 13 8 14 8
< 0 1 0 1 0
```

GET_POINTS

Zwraca liczbe punktów (int) zdobytą przez gracza i posiadane paktolary (int). `> GET_POINTS`
```
< OK
< 666 333
```

GET_UPGRADES

Zwraca 4 linie z dostępnymi poziomami ulepszeń, ich poziomami, pierwszy poziom to poziom podstawowy i cenami. Pojedyńcza linia składa się z:

- nazwa ulepszanego parametru (string)
- liczba **k** poziomów ulepszenia (int)
- **k**+1 wartości parametru w zależności od poziomu ulepszenia (int)
- **k** cen kupienia kolejnych poziomów ulepszeń (int)

Przykład:
```
> GET_UPGRADES
< OK
< engine 3 40 50 100 200 200 400 800
< memory 3 3 4 5 6 50 100 120
< transmitter 3 80 100 200 300 150 200 250
< receiver 3 0.4 0.3 0.2 0.1 100 200 300
```

UPGRADE_SATELLITE

Ulepsza jeden z parametrów satelity. W komendzie należy podać:

- id satelity (int)
- nazwa ulepszanego parametru (string)
- liczba poziomów o jakie chcemy ulepszyć dany parametr (int)

Przykład:
```
> UPGRADE_SATELLITE 4 engine 2
< OK
```

SCAN

Uruchamia skaner wypisując pozycje wszystkich wrogich satelitów. Ma czas odnowienia, patrz GET_CONSTANTS.
Przykład:
```
> SCAN
< OK
< 3
< 12 13
< -5 -1450
< 500 -2222
```

Możliwe błędy

- 202 Not enough money to buy – nie wystarczjące środki do zakupu satelity
- 211 Wrong satellite id number – nieprawidłowy numer id satelity
- 212 Already moved – już wykonano ruch obiektem
- 213 Position out of ship's range – pozycja poza zasięgiem statku
- 231 This satellite already transmitted packet – satelita juz wysłał pakiet w tej turze
- 232 No given packet in satellite's memory – nie ma takiego pakietu w pamięci satelity
- 233 Signal strength out of range – siła sygnału poza dostępnym zakresem
- 241 Satellite already received packet – satelita juz odebrał pakiet w tej turze
- 242 Wrong satellite‘s memmory address – nieprawidłowy adres pamięci satelity
- 243 Given packet can not be received or does not exist – podana pakiet nie może byc odebrany lub nie istnieje
- 251 Cannot deliver packet with id -1 – nie mozna dostarczyć pakietu o identyfikatorze
- 252 Given packet can not to be delivered or does not exist – podany pakiet nie znajduje się na satelicie w zasięgu kabla
- 261 Engine result level higher than maximum – poziom silnika po ulepszeniu przekracza maksymalny
- 262 Not enough money for engine upgrade – nie wystarczjące środki do zakupu ulepszenia silnika
- 263 Memory result level higher than maximum – poziom pamięci po ulepszeniu przekracza maksymalny
- 264 Not enough money for memory upgrade – nie wystarczjące środki do zakupu ulepszenia pamięci
- 265 Transmitter result level higher than maximum – poziom nadajnika po ulepszeniu przekracza maksymalny

- 266 Not enough money for transmitter upgrade – nie wystarczjące środki do zakupu ulepszenia nadajnika
- 267 Receiver result level higher than maximum – poziom odbiornika po ulepszeniu przekracza maksymalny
- 268 Not enough money for receiver upgrade – nie wystarczjące środki do zakupu ulepszenia odbiornika
- 269 Wrong module name – nieprawidłowa nazwa modułu
- 275 Scanner used too soon – skaner nie jest gotowy do użycia, patrz TURNS_LEFT