

دستور کار بازی

Javascript Client

AI Cup

2

0

2

3

کوش مصنوعی را متفاوت تجربه کنید!

اجرای کلاینت

- ابتدا از نصب nodejs روی سیستم خود مطمین شوید . برای دانلود به سایت رسمی به آدرس <https://nodejs.org/en/> مراجعه کنید .
- همیز پروژه کلاینت را در ترمینال باز کرده و با استفاده از دستور `npm install` تماشی `dependency` های پروژه را نصب کرده
- در انتهای دستور `npm start` کلاینت را اجرا کنید

توضیح بازی

- استراتژی خود را باید در دو تابع `initializer` و `turn` که در فایل `main.js` قرار دارد بنویسید.
- تابع `initialize` در بازی ۳۵ بار صدای زده میشود و هر بار که صدای زده میشود شما باید یک نیرو در یکی از خانه ها قرار دهید
- تابع `turn` حداقل ۲۰ بار برای شما صدای زده میشود و شما در آن باید ۴ استیت نوبت را جلو ببرید و استراتژی خود را پیاده سازی کنید. در ورودی این تابع یک آبجکت از کلاس `Game` به شما داده میشود که در آن توابع مورد نیاز برای پیاده سازی استراتژی قرار دارد.

• **توجه:**

- آبجکت game که به عنوان پارامتر به توابع initializer و turn داده میشود دارای متدهایی است که همه به صورت await بوده و برای استفاده از آنها باید از کلمه کلیدی async استفاده کرد. همچنین همه متد ها در صورت بروز خطا و ناموفق بودن عملیات، اکسپشن throw میکنند به همین دلیل برای هندل کردن خطا های احتمالی و نیز جلوگیری از کرش کردن برنامه هنگام اجرا میباشد تمامی متدها را در بلاک try catch صدا بزنید. نمونه کد زیر مثالی برای نحوه درست صداردن متدهای آبجکت game میباشد :

```
let adj_nodes;
try{
adj_nodes = await game.get_adj();
} catch(err){
console.log("an error happen during getting adj
nodes : ", err);
adj_nodes = null; }
```

- برای آشنایی بیشتر با نحوه استفاده از متدها میتوانید از کد نمونه ای که در فایل main.js قرار دارد استفاده کنید و اطلاعات بیشتری کسب کنید.

توابع کلاس Game

get_owners()

- این تابع یک آبجکت برمی‌گرداند که کلید های آن، شماره هر سیاره است و هر کلید مقداری برابر با شماره صاحب آن سیاره دارد . اگر سیاره‌ای صاحب نداشت مقدار آن برابر ۰ است.
- نمونه خروجی:**

```
{
  "0": 0,
  "1": 2,
  "2": -1,
  "3": 1,
  { 2 :"4"
```

get_number_of_troops_to_put()

- این تابع یک آبجکت برمی‌گرداند که کلید آن برابر با number_of_troops است و مقدار آن برابر با تعداد سربازی است که آن بازیکن در اختیار دارد ولی هنوز از آن استفاده نکرده است.
- نمونه خروجی:**

```
{
  number_of_troops: 12
}
```

get_state()

- این تابع یک آبجکت برمی‌گرداند که کلید آن برابر با state است و مقدار آن مشخص می‌کند که بازیکن چه عملیاتی می‌تواند انجام دهد.
- اگر مقدار آن برابر ۱ باشد یعنی بازیکن توانایی این را دارد که سرباز های خود را در یک سیاره مستقر کند. (استقرار سرباز)
- اگر مقدار آن برابر با ۲ باشد یعنی می‌تواند به سیاره‌ای حمله کند.
- اگر مقدار آن برابر ۳ باشد می‌تواند نیرو های خود را به سیاره ای دیگر منتقل کند. (جابجایی نیرو)
- اگر برابر با ۴ باشد یعنی در مرحله تبدیل نیرو های دفاعی هستید

{

state : 2

}

• نمونه خروجی:

get_turn_number()

- این تابع یک آبجکت برمی‌گرداند که کلید آن برابر با turn_number است و نشان می‌دهد بازی در نوبت چندم است

•

- **نمونه خروجی:**

```
{
    turn_number : 1
}
```

get_adj()

- این تابع یک آبجکت برمی‌گرداند که کلیدهای آن برابر با شماره های سیاره ها است و هر کلید مقداری برابر با یک آرایه دارد که این آرایه شامل شماره تمام همسایه های این سیاره است.

- **نمونه خروجی:**

```
{
    "1": [2, 3, 4],
    "2": [1, 3],
    "3": [1, 2, 4],
    "4": [1, 3, 5],
    "5": [4]
}
```

get_player_id()

- این تابع یک آبجکت برمی‌گرداند که کلید آن player_id است و مقدار آن برابر با شماره بازیکن شما است.

```
{ player_id: 2}
```

- **نمونه خروجی:**

get_strategic_nodes()

- این تابع یک آبجکت برمی‌گرداند که شامل ۲ تا کلید است :
- کلید `strategic_nodes`: مقدار آن آرایه شش سیاره‌ای است که به عنوان سیاره استراتژیک در بازی معرفی شده‌اند.
- کلید `score`: مقدار آن آرایه‌ای از امتیاز نظیر هر کدام از سیاره‌های استراتژیک است.
- **نمونه خروجی:**

```
{  
    strategic_nodes : [1, 5, 10, 20, 7, 9],  
    scores: [1, 2, 8, 4, 5, 3]  
}
```

get_number_of_troops()

- این تابع یک آبجکت برمی‌گرداند که کلیدهای آن شماره سیاره‌ها است و مقدار هر کلید برابر است با تعداد نیروهایی که در آن سیاره وجود دارد.

```
{  
    "0": 4,  
    "1": 0,  
    "2": 5,  
    "3": 11,  
    "4": 20  
}
```

get_reachable(node_id)

- این تابع به عنوان ورودی شماره یک سیاره می‌گیرد و به عنوان خروجی یک آبجکت برگشته که گردداند که کلید آن reachable است و مقدار آن برابر با آرایه ای از سیاره‌هایی است که بازیکن می‌تواند از سیاره ای که انتخاب کرد، به آن‌ها نیرو منتقل کند.

- **نمونه خروجی:**

```
{  
    "reachable": [1, 2, 3, 4]  
}
```

get_number_of_fort_troops()

- این تابع یک آبجکت برمی‌گرداند که کلیدهای آن شماره‌های سیاره‌ها است و مقدار هر کدام از آن برابر با تعداد نیروهای دفاعی موجود در آن سیاره است.

- **نمونه خروجی:**

```
{  
    "0": 4,  
    "1": 0,  
    "2": 1  
}
```

put_one_troop(node_id)

- این تابع به عنوان ورودی شماره سیاره‌ای را می‌گیرد که بازیکن در فاز شروع بازی قصد دارد درون آن یک نیرو قرار بدهد و در صورت موفقیت آمیز بودن، به عنوان خروجی یک آبجکت می‌دهد که کلید آن برابر با message است و مقدار آن برابر با troop added successfully است.

- در صورتی که این کار امکان پذیر نباشد برنامه برای شما یک اکسپشن throw می‌کند که آبجکتی است با کلید error و مقدار آن دلیل عدم موفقیت را نشان میدهد

- نمونه خروجی موفق:

```
{  
    message :"troop added successfully"  
}
```

- نمونه خروجی ناموفق:

```
{  
    error :"You can not put more than one troop in a turn"  
}
```

put_troop(node_id , num)

- این تابع برای گذاشتن نیرو در استیت اول هر نوبت استفاده میشود و با استفاده از آن مشخص میکنید که در کدام خانه چه تعداد نیرو قرار بگیرد
- این تابع را تا زمانی که در استیت اول نوبت باشید می توانید صدابزنید و نیروهایی که درون نقشه نگذاشته اید را درون نقشه بگذارید
- این تابع ۲ ورودی میگیرد:

- شماره سیاره‌ای که قصد دارد درون آن نیرو قرار دهد: node_id

- تعداد سربازی که قصد دارد روی آن سیاره قرار دهد: num

• نمونه خروجی موفق:

```
{
    message :"troop added successfully"
}
```

• نمونه خروجی ناموفق:

```
{ error :"This node is already owned by another player" }
```

attack(attacking_id, target_id, fraction, move_fraction)

- این تابع برای حمله به دیگر سیاره‌ها پس از مرحله قرار گیری نیروها استفاده می‌شود و ۴ تا ورودی دریافت می‌کند:

- شماره سیاره‌ای که قصد داریم حمله از آن سیاره آغاز شود: β attacking_id

- شماره سیاره‌ای که قصد داریم به آن حمله کنیم: β target

- حمله تا زمانی که نسبت نیروهای هم‌اجم به نیروهای مدافعان

- حداقل چه مقداری است، ادامه داشته باشد: β fraction

- در صورت موفقیت آمیز بودن حمله، چه کسری از نیروهای باقیمانده از حمله به سیاره تصاحب شده انتقال یابند: β

`move_fraction`

• خروجی:

- در صورت موفقیت آمیز بودن یک آبجکت به عنوان خروجی می‌دهد که کلید آن برابر با `message` است و مقدار آن برابر با `attack is successful` است.

- در صورتی که این کار امکان پذیر نباشد برنامه برای شما یک اکسپشن `throw` می‌کند که آبجکتی است با کلید `error` که مقدار آن دلیلی عدم موفقیت را توضیح میدهد

• نمونه خروجی موفق:

```
{
  message :"attack is successful"
}
```

• نمونه خروجی ناموفق:

```
{
  error :"fraction is not provided"
}
```

• شرایط حمله:

1. شماره سیاره ای که میخواهیم از آن حمله کنیم حتماً فقط باید شامل ارقام باشد (۰-۹)
2. شماره سیاره‌ای که میخواهیم از آن حمله کنیم در بین سیاره‌های بازی باید باشد
3. سیاره‌ای که میخواهید از آن حمله کنید متعلق به خودتان باید
4. سیاره‌ای که میخواهید به آن حمله کنید متعلق به شما نباشد
5. هیزان نیروهای سیاره‌ای که میخواهیم از آن حمله کنیم از ۲ کمتر نباشد
6. شماره سیاره‌ای که میخواهیم به آن حمله کنیم، حتماً باید فقط شامل ارقام باشد (۰-۹)
7. به سیاراتی که متعلق به هیچ بازیکنی نیستند نمیتوانید حمله کنید
8. مقدار `fraction` داده شده باید قابلیت تبدیل شدن به عدد اعشاری را داشته باشد
9. مقدار `move_fraction` داده شده باید قابلیت تبدیل شدن به عدد اعشاری را داشته باشد
10. باید مقداری بین ۰-۱ داشته باشد `move_fraction`.

move_troop(**source**, **destination**, **troop_count**)

- این تابع برای انتقال نیروها پس از مرحله حمله استفاده می‌شود و ۳ ورودی دریافت می‌کند:
 - نیروهای کدام سیاره باید انتقال بیابند (مبدأ): β **source**
 - نیروهای سیاره انتخاب شده به کدام سیاره باید منتقل شوند β **destination**: (مقصد)
 - چه میزان نیرو باید انتقال داده شود (تعداد نیروها): β **troop_count**
- **خروجی:**
 - در صورت موفقیت آمیز بودن یک آبجکت به عنوان خروجی می‌دهد که کلید آن برابر با **message** است و مقدار آن برابر با **troops moved successfully** است.
 - در صورتی که این کار اهکان پذیر نباشد برنامه برای شما یک اکسپشن **throw** می‌کند که آبجکتی است با کلید **error** که مقدار آن دلیلی عدم موفقیت را توضیح میدهد

• نمونه خروجی موفق :

```
{
    message :"troops moved successfully"
}
```

• نمونه خروجی ناموفق :

```
{
    error :"troop_count is not provided"
}
```

• شرایط انتقال نیرو :

- باید در فاز اصلی و استیت سوم نوبت باشیم
- باید سیاره‌ی مبدأ و مقصد متعلق به خودمان باشد
- باید مسیری بین این دو سیاره وجود داشته باشد که تمام سیارات آن متعلق به خودمان باشد

next_state()

- زمانی که در نوبت خود می‌خواهید به استیت بعدی بروید باید این تابع را صدا بزنید مثلا در استیت گذاشتن نیرو هستید و می‌خواهید به استیت حمله بروید

- **خروجی:**

- در صورت موفقیت آمیز بودن درخواست خروجی برابر با یک آبجکت است که شامل ۲ کلید است:
 - کلید game_state که مقدار آن برابر استیت جدید بازی است که یکی از سه وضعیت : قرار دادن نیرو/حمله/انتقال نیرو است.
 - کلید message که مقدار success دارد.

- **نمونه خروجی:**

```
{  
    game_state : 2,  
    message : "success"  
}
```

- توضیح مقادیر مختلف game_state:
 - 1. مرحله قرار دادن نیرو
 - 2. مرحله حمله
 - 3. مرحله انتقال نیرو
 - 4. مرحله تقویت نیرو

fort(node_id, troop_count)

- این تابع ۲ تا ورودی می‌گیرد:
- ورودی node_id برابر با شماره سیاره است.
- ورودی troop_count برابر با تعداد نیروهایی است که بازیکن می‌خواهد آن را به نیروی دفاعی تبدیل کند.
- در صورت موفقیت آمیز بودن عملیات خروجی یک آبجکت است که کلید آن برابر با success است و مقدار آن برابر با there is not enough troops in the node است.
- **نمونه خروجی:**

```
{  
    success :"the fortification ability is applied  
    successfully"  
}
```



AICUP
2023

[aicup_official](https://www.instagram.com/aicup_official/)

t.me/aicup

aicup2023.ir

هوش مصنوعی را متفاوت تجربه کنید!