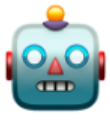

Звіт до міні-проєкту 2

Документація для бота Filler (для Notion)

Загальна інформація

Цей бот реалізує алгоритм для гри **filler**, який управляє діями гравця, обирає стратегії та розраховує оптимальні позиції для розміщення фігур. Мета бота — максимально заповнити карту своїми фігурами, дотримуючись обраної стратегії.

Функції для обробки карти та фігури

`parse_map() -> list[list[str]]`

Опис:

Читає карту з вхідних даних, що надсилаються віртуальною машиною, та перетворює її у двовимірний масив, де кожен рядок — це список символів.

Параметри:

- Немає.

Повертає:

- Двовимірний список (`list[list[str]]`), що представляє карту.

Особливості:

1. Ігнорує рядки, що не містять інформації про карту.
2. Враховує формат карти, переданий віртуальною машиною.

`parse_figure() -> list[tuple[int, int]]`

Опис:

Зчитує фігуру, яку потрібно розмістити, та формує список її активних клітин відносно верхньої лівої точки.

Параметри:

- Немає.

Повертає:

- Список координат клітин фігури відносно її верхньої лівої точки (`list[tuple[int, int]]`).

Особливості:

1. Враховує висоту фігури.
2. Ігнорує порожні клітини.

Функції для перевірки та обчислення позицій

`check_position(map: list[list[str]], figure: list[tuple[int, int]], position: tuple[int, int], player: str) -> bool`

Опис:

Перевіряє, чи можна розмістити фігуру на карті в заданій позиції згідно з правилами.

Параметри:

- `map`: двовимірний список, що представляє карту.
- `figure`: список координат фігури.
- `position`: координати верхньої лівої точки фігури.
- `player`: символ гравця (`'O'` або `'X'`).

Повертає:

- `True`, якщо фігуру можна розмістити, інакше `False`.

Особливості:

1. Гарантує перекриття рівно однієї клітини гравця.
2. Запобігає виходу за межі карти та перетину з клітинами суперника.

`get_all_positions(map: list[list[str]], figure: list[tuple[int, int]], player: str) -> list[tuple[int, int]]`

Опис:

Знаходить усі можливі позиції для розміщення фігури на карті.

Параметри:

- `map`: двовимірний список, що представляє карту.
- `figure`: список координат фігури.
- `player`: символ гравця (`'O'` або `'X'`).

Повертає:

- Список координат можливих позицій (`list[tuple[int, int]]`).

Особливості:

1. Використовує `check_position()` для перевірки кожної можливості.
2. Повертає всі валідні точки для фігури.

Функції для розрахунків

`compute_free_cells(map: list[list[str]]) -> int`

Опис:

Рахує кількість порожніх клітин на карті.

Параметри:

- `map`: двовимірний список, що представляє карту.

Повертає:

- Кількість порожніх клітин (`int`).

`compute_enemy_density(map: list[list[str]], enemy: str) -> int`

Опис:

Рахує кількість клітин, зайнятих суперником.

Параметри:

- `map`: двовимірний список, що представляє карту.

- `enemy`: символ суперника (`'O'` або `'X'`).

Повертає:

- Кількість клітин суперника (`int`).

`compute_player_density(map: list[list[str]], player: str) -> int`

Опис:

Рахує кількість клітин, зайнятих гравцем.

Параметри:

- `map`: двовимірний список, що представляє карту.
- `player`: символ гравця (`'O'` або `'X'`).

Повертає:

- Кількість клітин гравця (`int`).

`compute_distance_to_center(map: list[list[str]], player: str) -> float`

Опис:

Розраховує середню відстань клітин гравця до центру карти.

Параметри:

- `map`: двовимірний список, що представляє карту.
- `player`: символ гравця (`'O'` або `'X'`).

Повертає:

- Середня відстань до центру (`float`).

Функції для стратегій

`determine_strategy_with_thresholds(map, player, enemy) -> str`

Опис:

Визначає оптимальну стратегію для поточного стану карти.

Параметри:

- `map`: двовимірний список, що представляє карту.
- `player`: символ гравця (`'O'` або `'X'`).
- `enemy`: символ суперника (`'O'` або `'X'`).

Повертає:

- Назву стратегії (`"expansion"`, `"blocking"`, `"centralization"`).

`expansion_score(map, figure, position) -> float`

Опис:

Розраховує оцінку для стратегії розширення.

`blocking_score(map, figure, position) -> float`

Опис:

Розраховує оцінку для стратегії блокування.

`centralization_score(map, position) -> float`

Опис:

Розраховує оцінку для стратегії централізації.

Основний цикл

`main()`

Опис:

Основний цикл гри. Приймає дані від віртуальної машини, обирає стратегію, знаходить оптимальну позицію для фігури та надсилає координати назад.

Особливості:

1. Читає карту та фігуру на кожному кроці.
2. Визначає стратегію.
3. Обчислює оптимальну позицію.
4. Завершує гру, якщо більше немає доступних ходів.

Ідея та підхід до реалізації

Основна концепція

Ми створили алгоритм, який ділить завдання на два основних етапи:

1. **Визначення стратегії** — обирається оптимальна стратегія залежно від стану карти, положення фігур гравця та суперника, а також кількості порожніх клітин.
2. **Пошук найкращої позиції** — для заданої стратегії обчислюється позиція, яка максимізує значення обраної стратегії.

Підхід до визначення стратегії

Ми використали функції аналізу стану карти, які оцінюють:

- Щільність клітин гравця та суперника.
- Відстань до центру карти.
- Кількість доступних порожніх клітин.

Ключова ідея:

Кожна стратегія залежить від певного набору метрик:

- **Розширення** — максимізує кількість порожніх клітин навколо фігури.
- **Блокування** — мінімізує можливості суперника шляхом обмеження його рухів.
- **Централізація** — рух до центру карти для створення переваги в контролі над картою.

Пошук найкращої позиції

Для кожної можливої позиції ми обчислюємо оцінку залежно від обраної стратегії. Усі позиції перевіряються через функцію `check_position`, яка гарантує дотримання правил гри. Потім проводиться перебір позицій, щоб знайти ту, яка дає максимальний результат.

Функції для обробки карти та фігур

`parse_map()`

- Зчитує карту з вхідних даних.
- Перетворює її у двовимірний список символів.

`parse_figure()`

- Зчитує активні клітини фігури.
- Формує список координат відносно верхньої лівої точки.

Функції для визначення стратегій

`determine_strategy_with_thresholds(map, player, enemy)`

- Визначає, яка стратегія є оптимальною для поточного стану гри.
- Використовує функції аналізу карти, щоб оцінити такі параметри:
  - Кількість порожніх клітин.
  - Щільність клітин суперника.
  - Відстань до центру карти.

Функції для оцінки стратегій

Стратегія "Розширення"

- Використовує функцію `expansion_score`, яка обчислює оцінку позиції на основі:
  - Щільності порожніх клітин навколо фігури.
  - Відстані до країв карти.
  - Площі, яку можна зайняти.

Стратегія "Блокування"

- Використовує функцію `blocking_score`, яка обчислює:
  - Щільність клітин суперника навколо позиції.
  - Відстань до найближчих фігур суперника.

Стратегія "Централізація"

- Використовує функцію `centralization_score`, яка враховує:
  - Відстань до центру карти.
  - Вільну площу поблизу центру.

Головна функція

`main()`

- Основний цикл гри:
  1. Зчитує карту та фігуру.
  2. Обирає стратегію через `determine_strategy_with_thresholds`.
  3. Обчислює найкращу позицію через `find_best_position`.
  4. Надсилає координати у віртуальну машину.

Переваги підходу

1. **Адаптивність:** алгоритм аналізує стан гри та динамічно обирає стратегію.
2. **Модульність:** кожна стратегія реалізована окремими функціями, що спрощує тестування та модифікацію.
3. **Оптимізація:** враховуються як поточні, так і майбутні можливості для гравця.