Modifications (03/25)

- The program should use lower than 4GB in total, including your program code on the memory. For your information: when I tested with `default.py`, the memory usage after initial loading is 22MB.

  프로그램은 (여러분의 코드를 포함) 최대 4GB까지 메모리를 사용할 수 있습니다. 참고로, `default.py` 로 테스트했을 때, 상태 초기화 후 메모리 사용량은 22MB였습니다.

- The position of hexes and harbors will be initialized as a beginner's board, not a random board.

  육각형 판과 항구의 배치는 랜덤 배치가 아닌 초심자용 배치를 따릅니다.

- The game environment ends when your score reaches 4 points. But, the victory points (2 points) given to the longest route owner is ignored.

  여러분의 점수가 4점이 되는 순간, 게임은 끝이 납니다. 단, 최장교역로 소유자에게 지급되는 2점 승점은 무시됩니다.

CAU56753 Challenge I: Route Search

Problem definition / 문제정의

Welcome to the first programming challenge!

첫번째 프로그래밍 도전과제에 오신 여러분, 환영합니다!

In this challenge, you have to make a simple search algorithm for finding the "**LONGEST**" trading route in the given situation.

이 도전과제에서, 여러분은 주어진 상황에서 만들 수 있는 최장교역로를 찾는 알고리즘을 설계해야 합니다.

Here's PEAS description. According to the description, the system will automatically run your code and give you the evaluation result.

아래에 PEAS 상세정보가 주어져 있습니다. 평가 시스템은 PEAS 정보에 따라 여러분의 코드를 실행하고 평가하여, 평가 결과를 제공할 것입니다.

(Are you curious about how to run the evaluation? Then, go to RUN section!) (어떻게 평가를 실행하는지 궁금하다면, RUN 부분으로 넘어가세요!)

PEAS description

Performance measure (수행지표)

Presented in the order of importance in evaluation.

평가시 중요한 순서 순으로 나열됨

1. The number of errored trials (smaller is better)

   오류가 난 코드 실행의 수 (적을 수록 좋음)

2. The maximum memory usage (smaller is better; rounded down in MB)

   최대 메모리 사용량 (작을 수록 좋음; MB 단위로 버림)

3. The length of the trading route at the end (larger is better)

   게임 끝에 찾은 교역로의 길이 (길 수록 좋음)

4. The length of your solution path on the state space (smaller is better)

   상태공간 속 해답 경로의 길이 (짧을 수록 좋음)

Note: Evaluation program will automatically build a table that sort your codes' evaluation result in the order of performance measure. Also, the algorithm should finish its search procedure within 50 minutes.

참고: 평가 프로그램이 여러분 코드의 평가 결과를 수행지표 순서대로 정렬하여 표 형태로 표시해줄 예정입니다. 그리고, 알고리즘의 탐색 절차는 50분 이내에 끝나야 합니다.

Note: (Added on 03/25) The program should use lower than 4GB in total, including your program code on the memory. For your information: when I tested with `default.py`, the memory usage after initial loading is 22MB.

참고: (03/25 추가) 프로그램은 (여러분의 코드를 포함) 최대 4GB까지 메모리를 사용할 수 있습니다. 참고로, `default.py` 로 테스트했을 때, 상태 초기화 후 메모리 사용량은 22MB였습니다.

Environment (환경)

You're a player in the Settlers of Catan world. You already have two village and two path blocks on the board. Now, suppose these things for simplicity.

(Added on 03/25) The position of hexes and harbors will be initialized as a beginner's board, not a random board.

여러분은 카탄 게임의 플레이어입니다. 이미 여러분은 두개의 마을과 2개의 도로 블록을 판 위에 가지고 있습니다. 이제, 문제를 단순하게 하기 위해서 다음을 가정해봅시다.

(03/25 추가) 육각형 판과 항구의 배치는 랜덤 배치가 아닌 초심자용 배치를 따릅니다.

1. Other players will do nothing in the game until the game ends.

   다른 플레이어는 게임이 끝날 때까지 아무 행동도 하지 않습니다.

2. There's no dice here. You'll always get `1+N` resource card for each resource, for each turn. Here, `N` is the number of your cities in the board.

   여기는 주사위가 없습니다. 여러분은 매 턴마다 각각 `1+N` 개의 카드를 각 리소스마다 받게 됩니다. 여기서 `N` 은 여러분이 가진 도시의 수입니다.

3. When the dices give you 7, then nothing happens; That is, there's no thief in your game.

   주사위 눈의 합이 7일 때에는, 아무 일도 일어나지 않습니다. 즉, 이 판에는 도둑이 없습니다.

4. The only one, you, can build roads, villeages, or cities to achieve the longest trading route.

   이 판에서 여러분만 도로, 마을 또는 도시를 지어 최장교역로를 달성할 수 있습니다.

5. The number of usable blocks are restricted: 3 village, 3 cities, and 10 path blocks.

   사용가능한 블록의 개수가 제한됩니다: 3개의 마을, 3개의 도시, 10개의 도로 블록.

6. The game environment ends when your score reaches 4 points. (Added 03/25) But, the victory points (2 points) given to the longest route owner is ignored.

   여러분의 점수가 4점이 되는 순간, 게임은 끝이 납니다. (03/25 추가) 단, 최장교역로 소유자에게 지급되는 2점 승점은 무시됩니다.

In terms of seven characteristics, the game can be classified as:

환경을 기술하는 7개의 특징을 기준으로, 게임은 다음과 같이 분류됩니다:

- Fully observable (완전관측가능)

  You already know everything required to decide your action.

  여러분은 이미 필요한 모든 내용을 알고 있습니다.

- Single-agent (단일 에이전트)

  The other agents are actually doing nothing. So you're the only one who pursues those performance measure.

  다른 모든 에이전트는 사실 아무것도 안 합니다. 그러므로, 그 판에서 여러분만이 수행지표를 최대화하려는 유일한 에이전트입니다.

- Deterministic (결정론적)

There's no probabilistic things or unexpected chances when deciding the next state. Everything is deterministic.

다음 상태를 결정할 때 확률적으로 결정되는 것이나, 예상치 못한 변수가 없습니다. 모든 것은 결정되어 있습니다.

- Sequential actions (순차적 행동)

  You need to handle sequence of actions. But, don't panic. The solution path in a search space will give you the right sequence of actions.

  행동을 순차적으로 기술해야 합니다. 하지만 걱정하지 마세요. 탐색 공간에서 찾아내는 해답 경로가 정확한 행동의 순서를 줄 것입니다.

- Semi-dynamic performance (준역동적 지표)

  Note that performance metrics 1, 3 are static, but the metrics 2, 4 is dynamic metric, which changes its measure by how your algorithm works. So you need some special effort for achieving good performance measure on 2 and 4, when designing your algorithm.

  지표 1과 3은 정적이고, 지표 2와 4는 동적입니다. 특히 지표 2와 4는 여러분의 알고리즘 작동에 따라 변화하는 지표입니다. 그래서 알고리즘을 설계할 때, 지표 2/4의 변화에 신경써서 설계하는 노력이 필요합니다.

- Discrete action, perception, state and time (이산적인 행동, 지각, 상태 및 시간개념)

  All of your actions, perceptions, states and time will be discrete, although you can query about your current memory usage in the computing procedure.

  여러분의 모든 행동, 지각, 상태 및 시간 흐름은 모두 이산적입니다. 여러분이 계산 도중 메모리 사용량을 시스템에 물어볼 수 있다고 하더라도 말입니다.

- Known rules (규칙 알려짐)

  All rules basically follows the original Catan game, except the rules specified above.

  모든 규칙은 위에 작성된 예외를 제외하면 기본적으로 원래의 카탄 게임을 따릅니다.

Actions

You can take one of the following actions.

다음 행동 중의 하나를 할 수 있습니다.

- **PASS**: Pass the turn to the next person

  다음 사람에게 차례 넘기기

- **ROAD(e)**: Build a road at a specific edge e.

특정한 모서리 e 에 도로 짓기

Here, the list of applicable edges will be given by the board.

도로 짓기가 가능한 모서리의 목록은 board가 제공합니다.

- **VILLAGE(v)**: Build a village at a specific node v.

  특정한 꼭짓점 v 에 마을 짓기

  Here, the list of applicable nodes will be given by the board. The list may be empty if you reached the maximum number of villages, i.e., three.

  마을 짓기가 가능한 꼭짓점의 목록은 board가 제공합니다. 단, 가능한 마을의 수가 최대치인 3개에 도달한 경우, 목록은 빈 리스트로 제공될 수 있습니다.

- **UPGRADE(v)**: Upgrade a village at v to a city.

  꼭짓점 v 에 위치한 마을을 도시로 업그레이드하기.

  Here, the list of applicable nodes will be given by the board. The list may be empty if you don't have any villages left.

  도시 업그레이드가 가능한 꼭짓점의 목록은 board가 제공합니다. 단, 업그레이드가 가능한 마을이 없으면 목록은 빈 리스트로 제공될 수 있습니다.

- **TRADE(given, request)**: Getting a new resource card by giving multiple cards of the same resources, through the port(2:1 or 3:1) or bank(4:1).

  무역을 통해 동일한 자원카드 여러장을 다른 카드 한장으로 바꿀 수 있습니다. 이 때, 교역률은 항구에서는 2:1 또는 3:1이며, 은행을 통하는 경우 4:1입니다.

  Here, 2:1 or 3:1 trade is applicable only when you have a villege or city at the port which allows 2:1 or 3:1 trading of that very resources. Otherwise, you can't use that trade option.

  2:1과 3:1 무역 조건은 여러분이 2:1 또는 3:1 조건으로 해당 리소스를 무역할 수 있는 항구에 도시나 마을을 지었을 때에만 가능합니다. 그렇지 않으면 해당 무역 조건은 사용할 수 없습니다.

Note that you will get resources automatically on each dice roll, without claming your resources. Also, you cannot buy a development card, because it is not relevant to the current problem's goal.

참고로, 모든 자원 카드는 자원을 달라고 요청할 필요 없이 주사위를 굴릴 때마다 여러분에게 지급됩니다. 또한, 발전카드는 이 문제와 관련이 없으므로, 발전카드는 살 수 없습니다.

Sensors

You can perceive the game state as follows:

- The board (게임 판)

    - All the place of hexes(resources), villages, roads and ports

      모든 육각형(자원), 마을, 도로, 항구의 위치

    - You can ask the board to the list of applicable actions for:

      가능한 행동에 대해서 게임판 객체에 물어볼 수 있습니다:

- Your resource cards (여러분의 자원카드 목록)

Structure of evaluation system

평가 시스템의 구조

The evaluation code has the following structure.

평가용 코드는 다음과 같은 구조를 가지고 있습니다.

```
/                   ... The root of this project
/README.md          ... This README file
/evaluate.py        ... The entrance file to run the evaluation code
/board.py           ... The file that specifies programming interface with the board
/actions.py         ... The file that specifies actions to be called
/util.py            ... The file that contains several utilities for board and action definiti
/agents             ... Directory that contains multiple agents to be tested.
/agents/__init__.py ... Helper code for loading agents to be evaluated
/agents/load.py     ... Helper code for loading agents to be evaluated
/agents/default.py  ... A default DFS agent
/agents/_skeleton.py... A skeleton code for your agent. (You should change the name of file to
```

All the codes have documentation that specifies what's happening on that code (only in English).

모든 코드는 어떤 동작을 하는 코드인지에 대한 설명이 달려있습니다 (단, 영어로만).

To deeply understand the `board.py` and `actions.py`, you may need some knowlege about `pyCatan2` library.

`board.py` 와 `actions.py` 를 깊게 이해하고 싶다면, `pyCatan2` library 라이브러리에 대한 지식이 필요할 수 있습니다.

What should I submit?

You should submit an agent python file, which has a similar structure to `/agents/default.py`. That file should contain a class name `Agent` and that `Agent` class should have a method named `search_for_longest_route(board)`. Please use `/agents/_skeleton.py` as a skeleton code for your submission.

`/agents/default.py` 와 비슷하게 생긴 에이전트 코드를 담은 파이썬 파일을 제출해야 합니다. 해당 코드는 `Agent` 라는 클래스가 있어야 하고, `Agent` 클래스는 `search_for_longest_route(board)` 메서드를 가지고 있어야 합니다. 편의를 위해서 `/agents/_skeleton.py` 를 골격 코드로 사용하여 제출하세요.

Also, you cannot use the followings to reduce your search time:

그리고 시간을 줄이기 위해서 다음에 나열하는 것을 사용하는 행위는 제한됩니다.

- multithreading / 멀티스레딩
- multiprocessing / 멀티프로세싱
- using other libraries other than basic python libraries. / 기본 파이썬 라이브러리 이외에 다른 라이브러리를 사용하는 행위

The TA will check whether you use those things or not. If so, then your evaluation result will be marked as zero.

조교가 여러분이 해당 사항을 사용하였는지 아닌지 확인하게 됩니다. 만약 그렇다면, 해당 평가 점수는 0점으로 처리됩니다.

RUN

실행

To run the evaluation code, do the following:

1. (Only at the first run) Install the required libraries, by run the following code on your terminal or powershell, etc:

   (최초 실행인 경우만) 다음 코드를 터미널이나 파워쉘 등에서 실행하여, 필요한 라이브러리를 설치하세요.

   ```
   pip install -r requirements.txt
   ```

2. Place your code under `/agents` directory.

   여러분의 코드를 `/agents` 디렉터리 밑에 위치시키세요.

3. Execute the evaluation code, by run the following code on a terminal/powershell:

   다음 코드를 실행하여 평가 코드를 실행하세요.

   ```
   python evaluate.py
   ```

   If you want to print out all computational procedure, then put `--debug` at the end of python call, as follows:

만약, 모든 계산 과정을 출력해서 보고 싶다면, `--debug` 을 파이썬 호출 부분 뒤에 붙여주세요.

```
python evaluate.py --debug
```

4. See what's happening.

   어떤 일이 일어나는지를 관찰하세요.

Note: All the codes are tested both on (1) Windows 11 (23H2) with Python 3.9.13 and (2) Ubuntu 22.04 with Python 3.10. Sorry for Mac users, because you may have some unexpected errors.

모든 코드는 윈도우 11 (23H2)와 파이썬 3.9.13 환경과, 우분투 22.04와 파이썬 3.10 환경에서 테스트되었습니다. 예측불가능한 오류가 발생할 수도 있어, 미리 맥 사용자에게 미안하다는 말을 전합니다.