236501 - Introduction to Artificial Intelligence [w2019] / hw1 / wet checking and common mistakes

Many of you did not follow the submission instructions although it was explained explicitly in the assignment instructions and in the FAQ section. We decided to find the code directory for each one of you and fix your submission. Submissions which did not follow the instructions were reducted with 5 points, according to the hw instructions.

We executed 23 tests.
Each test was executed in a dedicated process with an execution timeout limitation. Each such process has been assigned with a dedicated CPU physical core. Your submission was given 3 attempts to run each test. The timeout limitation (per test) was significantly greater than the execution time of our implementation.

The following mistakes were very common:

1. Incorrect usage of `np.random.choise(...)` in the greedy stochastic implementation.

2. Many of you forgot to insert into `close` the chosen next node to expand in greedy stochastic method `_extract_next_search_node_to_expand()`. Notice that we explicitly stated in the FAQ that this algorithm uses the `close` data-structure.

3. Calling the heuristic function on a `node.state` instead of using fields of `node` which already store this value [in the greedy stochastic implementation]. It takes time to re-calculate the heuristic function all over again.

Here are some less common mistakes (but prone to trouble) that we encountered:

1. Python assignment for mutable types does not copy the object, but only duplicates the pointer to the same object. This issue has been explained in the python notebook of tutorial 1.

2. In python, when you import a file, the whole code in this file is executed, unless this code is wrapped with the condition `if __name__ == '__main__'`. Some of you imported the `experiments/temperature.py` file from the greedy stochastic solver implementation. When we tried running the tests, it opened the plot and blocked your code execution until it reached our timeout limitation.

3. Python is sensitive to whitespace type. So if your code file uses both `spaces` indentations and also `tabs` indentations, the interpreter might fail parsing the script.