UNIVERSITY OF SASKATCHEWAN

Department of Computer Science
176 Thorvaldson Building
110 Science Place, Saskatoon, SK, S7N 5C9, Canada
Telephine: (306) 966-4886, Facimile: (306) 966-4884

CMPT 317
Winter 2023
Introduction to Artificial Intelligence

Assignment 3

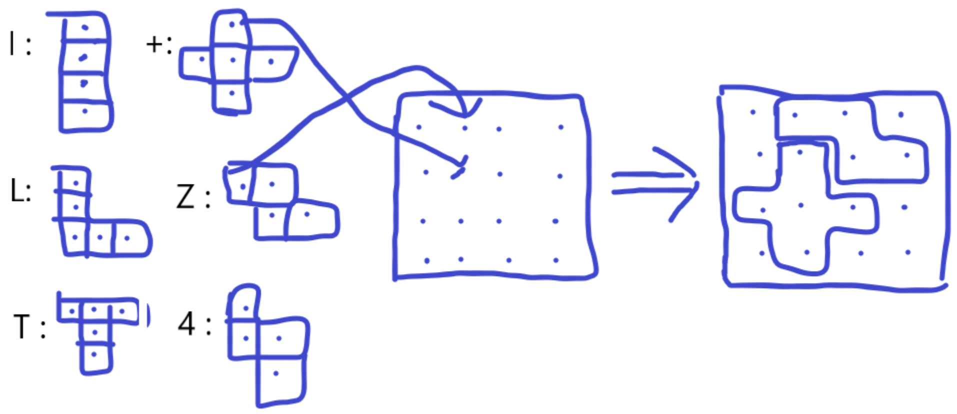
Local Search

Date Due: Thursday February 9, 5:00pm **Total Marks: 18**

General Instructions

- **This assignment is individual work.** You may discuss questions and problems with anyone, but the work you hand in for this assignment must be your own work.
- If you intend to use resources not supplied by the instructor, please request permission prior to starting. **You should not use any resource that substantially evades the learning objectives for this assignment.** You must provide an attribution for any external resource (library, API, etc) you use. If there's any doubt, ask! No harm can come from asking, even if the answer is no.
- Each question indicates what the learning objective is, what to hand in, and how you'll be evaluated.
- **Do not submit folders, or zip files, even if you think it will help.**
- Assignments must be submitted to Canvas.

The Block Tiling problem



Suppose you have an empty NxN grid, and some blocks that each have one of the 6 possible shapes shown above. For convenience, each different shape has a single character that represents it, as shown in the picture.

You are allowed to place a block anywhere on the grid, so long as it fully fits and does not cover any portion of another block. You are NOT allowed to rotate the blocks.

The goal is to cover as much of the grid as possible with the blocks. The order in which the blocks are placed does not matter, and it does not matter which blocks are used or if there are any blocks left over. Minimizing the number of empty grid spaces is all that counts.

Data Files

You will be given a series of data files representing Block Tiling problems. Each file represents a single problem. The following is an example that corresponds to the situation in the picture above.

```
4
1 +
1 L
1 |
1 Z
1 T
1 4
```

The `4` on line 1 specifies N, which is the dimension of the grid. Each subsequent line specifies the availabilty of a certain shape of block. For example, line 2 indicates that one `+`-shaped block is available, and line 3 indicates that one `L`-shaped block is available.

Question 1 (3 points):

Purpose: To understand an optimization problem

AIMA Chapter(s): None really

Part 1

By hand, without using any program, find what you think is a good solution to the Block Tilings problem represented in tiles1.txt. You can draw a picture or use ASCII text. Any method that makes it clear where you have placed each block is fine. Indicate how many grid spaces you had to leave blank in your solution.

Part 2

Do you think you've found the optimal solution to the problem? If no, what's your best guess as to how many grid spaces the optimal solution will cover? Do you think you can write a program that will find it?

What to Hand In

- A file named `a3q1.pdf` (or .txt, .rtf, .docx) containing your answers to the parts above

Be sure to include your name, NSID, student number, and course number at the top of your file

Evaluation

- 2 marks: Solution given for tiles1.txt
- 1 mark: Discussion shows reasonable effort

Question 2 (10 points):

Purpose: To practice problem specification design for local search

AIMA Chapter(s): 4.1

For this question, you will write the code to apply Local Search to the Block Tiling problem.

The following basic Local Search strategies have been provided for you:

- Random Guessing: Randomly choose a new state from all possible states.
- Random Search: Randomly move to a neighbouring state.
- Hill Climbing: Move to the best neighbouring state.
- Random Restart Hill Climbing: Repeat Hill Climbing, with random initial positions.

These functions are complete and have been tested, and you should be confident that they work. To apply these methods, you will implement a State class and a Problem class, as per the following interface.

Problem Class

A Problem class has been provided for you in which most of the work relating to the problem specification is already done. You will only need to implement the following methods:

- `random_state(self)`: returns a completely random state
- `neighbors(self, state)`: returns a list of all neighbors of the given state

You should not change any of the other provided methods, but you may add any number of additional helper methods that you wish.

Note that defining the **neighborhood** of a state (i.e. the states that are considered adjacent to a given state) is up to you! There is no single correct answer. This is a design problem, and like all design problems, there are multiple reasonable designs and a great many bad ones. Along with your code, you need to submit a short written paragraph that describes your design.

State Class

In the State class, you will need to implement the following method:

- `get_score(self)`: Returns the **fitness score** for this state

Deciding on how to calculate the fitness score for this problem will likely be quite easy. Recall that this is the score that we are trying to minimize. Include a description of your fitness score in your written document.

You should not change any of the other methods in State, but you will likely need to read them in detail to understand how a State is represented so that you can write all of the code you need for this problem.

What to Hand In

- A file named `blockTiling.py` containing your implementation of the State and Problem classes.
- A written document named `a3q2.txt` (or .rtf, .pdf, .docx) that describes your definition of `adjacency` for the Block Tiling problem, and the definition of your objective function.

Be sure to include your name, NSID, student number, and course number at the top of all documents.

Evaluation

- 4 marks: Your `random_state()` method is capable of generating ANY possible state, with a reasonable attempt at uniform probability
- 5 marks: Your neighborhood specification is reasonable, well-described, and correctly implemented
- 1 marks: Your fitness function is well-described and correctly implemented

Question 3 (5 points):

Purpose: To explore the performance of Local Search algorithms

AIMA Chapter(s): 4.1

For this question, you will run the provided Local Search algorithms, using the State and Problem classes that you created for the previous question, on the provided Block Tiling data files.

A solver script is provided to you, which you can use to run the tests and produce your data. **For each problem file**, you should submit a neat and readable table of results in the following format:

	Step Limit					
	200 steps		1000 steps		5000 steps	
Strategy	Score	Time	Score	Time	Score	Time
Random Guessing						
Random Search						
Hill-climbing						
Hill-climbing, Long Restarts						
Hill-climbiing, Frequent Restarts						

The **Score** column is the fitness of the best solution found by each stategy, as defined by your objective function. The **Time** column is the run time in seconds, which you can round to 2 decimal places.

Note that there are two versions of Random-Restart Hill-climbing. They each use the given number of total steps, but differ in how long they wait before trying a random restart.

1. Long Restarts: 100 steps before restarting
2. Frequent Restarts: 20 steps before restarting

In addition, for each problem file, include the SINGLE best state found by any of the runs (it is fine to just copy/paste the text output from the solver script).

Finally, write a short paragraph where you discuss the following questions:

- Which algoritihm found the best solution? What does this tell you about your choice of neighborhood for the Block Tilings problem?
- For the biggest problem (tiles3.txt), how well do you think the Search Tree methods from Chapter 3 would have performed? Why?

What to Hand In

- Your document that includes your data tables and your concluding paragraph as described above. You can use a plain text file, or you can submit a MSWord document, or PDF. Name your document `a3q3` with an appropriate extension (e.g., txt, pdf, docx, etc).

Be sure to include your name, NSID, student number, and course number at the top of each document or file.

Evaluation

- 2 marks: Results are plausible and well-formatted for all data files.
- 3 mark: The conclusion shows insight into the behaviour of local search and its advantages/disadvantages relative to Search Tree approaches