# Intro to AI - Blokus memory hints

Yoni Sher

March 2020

## 1 Introduction

In the field of artificial intelligence, most of the problems we are trying to solve belong to a computation class called NP-Hard (at least). While no efficient (in the complexity theory sense) algorithm exists for these problems, we must still do our best to make the algorithms we use run in a reasonable amount of time. Search is the standard workhorse of classical artificial intelligence, so it is worth taking the time to do it right (also your exercise grade will likely depend on it).

## 2 Memory management

The first exercise in this course involves writing search algorithms - BFS and DFS - in python. Theoretically, python does memory management for you, so you don't have to worry about any of this. Practically, nothing is ever that simple.

All the information you use when running an algorithm must reside somewhere in the computer's memory. While python allocates and frees memory resources for you (such as objects, lists and other variables), doing so takes time. A typical memory allocation (such as creating a new list or object) takes several milliseconds, or about as long as several million calculation operations. This (almost) does not depend on the size of the object created. While milliseconds does not sound like very long, for a search algorithm that opens (even without expanding) hundreds of thousands of nodes, memory allocation alone will take minutes or even hours. All this memory must later be freed, which takes similar amounts of time.

### 2.1 Reducing memory usage

This exercise does not use very much memory (in modern terms), but for your exercise to run fast you must be smart about memory allocation. Here are some general practical tips:

- Avoid copying objects whenever possible. If an object is needed in several parts of your code, hold a single copy and pass it by reference.

- Avoid creating new lists when you can. This means that list comprehension is not your friend anymore, as it creates a new list by filtering and applying functions to each element of the old list. A simple loop might be faster, and should be considered before turning to advanced (high-level) functions such as map. If you are not sure, use the profiling tool (Section 2.2) to test both and see which is faster.

Specifically, this means you should consider the following:

- The only place where you absolutely must copy a board is in creating a new search state for a newly opened node.
- Think carefully what objects should be stored in the visited list, and how.
  - Checking if an element is in a set is faster than checking if it is in a list. What functions does this require? When should two search nodes/states be considered equal?
  - What do you want to store in visited list? Do you need the entire search node?

### 2.2 Checking your code's efficiency

PyCharm has a built-in profiling tool, which you can run from the button to the right of the "run" and "debug" buttons (the one with the clock). This will run your code and generate a report that tells you how much time each part of you code is taking. Some parts of your code will inevitably take time, but you might find that others are taking unexpectedly long - these are the parts you should look at and try to improve.

## 3 Expected times

If you have written your code efficiently, questions 1-9 should each take less than 30 seconds. The school solution (run on school computers) takes less than 15 seconds for each. The mini-contest (question 10) takes significantly more time. With the same heuristic used in the previous questions, the school solution takes 226 seconds to find an optimal solution. You will need a very smart and efficient heuristic to beat the 30 second cut-off for this question!

## 4 Getting Help

This explanation page includes concepts and terms introduced in courses you may not yet have taken or do not feel confident about (computation complexity and memory management). You should feel free to come to your TA for help with these topics if you feel they are preventing you from doing your best work on this exercise.