**Assigned:** Tuesday 15 September

**Due:** Monday 21 September, 11:59 p.m.

**Objectives:**

- Understand details of **informed search** algorithm implementations
- Explore **performance** implications of uninformed versus informed search algorithms
- Learn how to develop **admissible heuristics**
- Practice relevant data-gathering for **analysis**

**Collaboration:** This laboratory will be completed in pairs assigned by the instructor.

# 1 Introduction

In the last assignment you implemented a variety of tree-search based uninformed search algorithms, measuring their efficiency and solution quality. In this assignment we'll expand our horizons (nodes? frontiers?) and use information about the problem to implement $A^*$, an informed search routine.

# 2 Code and environment

The infrastructure code and problem for this assignment are the same as the last. In addition, an optional heuristic for the hex turn problem, `(weinman-hexturn-heuristic` *`state`*`)`, is provided with the other compiled files of the prior assignment. To use it, copy the completed search directory, for example:

```
$ cp -R ~weinman/courses/CSC261/code/searchc ./astar
$ drracket astar/astar.scm
```

# 3 Lab assignment

## Problem 1 - Implementing $A^*$ search [16 points]

Document, implement, and test the $A^*$ search routine `(a-star-search start-state problem heuristic)` in the file `astar.scm`. You should use the compiled general tree search routine `search`, as documented in the previous assignment.

Your solution may be based closely on a specific search routine implementation from the previous lab assignment, so long as it is properly attributed.

## Problem 2 - Designing a heuristic [24 points]

Design the best heuristic for the hex turn problem you can think of. This should be neither the trivial always-zero heuristic nor a perfect heuristic that simply solves the problem using full breadth-first search. You should strive to come up with an admissible heuristic, even if you do not end up implementing it. If you cannot implement your supposedly admissible heuristic, you should implement a simpler one that is not admissible but may outperform the other searches for efficiency (i.e., node expansions).

### Part A

Write a clear, unambiguous description of your best heuristic in a mixure of high-level English and pseudocode, as appropriate. Your description ought to allow a skilled programmer to create and *understand* a working implementation.

### Part B

Prove, or at least explain, why your heuristic is or is not admissible.

### Part C

Implement your heuristic as *username1*`-`*username2*`-hexturn-heuristic` in a file called `heuristic.scm`.

## Problem 3 - Experimentation [12 points]

In this problem, you will do some comparative experimentation of your search routines. Note, only a table is required for this portion—no integrative analytical essay is required. However, be sure you have organizational clarity and include your name, mailbox, and a title. Remember to acknowledge any parties responsible for the search or heuristic source code used for analysis.

The IDS routine is available within the compiled code as `(iterative-deepening-search` *`start-state problem`*`)`. You may use your own $A*$ search implementation by loading it after you `(require "search.scm")` or use the compiled version (with the same parameters) therein.

Construct a table similar to the one in AIMA Figure 3.29, comparing IDS and $A^*$ with progressively larger hex turn puzzles. You should include at least six rows in your table with the problem size and solution depth for each.

You can calculate the effective branching factor `b` in Maple (interactive mathematical software available on the MathLAN via the shell command `maple`) with the command:

```
fsolve( (b^d-1)*b/(b-1)=N,b=4,0..infinity);
```

where `d` is your solution depth and `N` is the number of nodes expanded.

### Extra Credit [8 points]

Add column(s) to your table for various heuristics you have designed and tried (4 points) with a paragraph comparing and analyzing (i.e., explaining) the results (another 4 points).

## What to turn in

Your submission should include the following

- Your `astar.scm` and `heuristic.scm` files
- A short driver program (`.scm` file) that demonstrates your heuristic applied to several meaningful states and your search algorithm applied to some problems that do not take long to run.
- A single PDF containing (merged)
  - Your Scheme file(s)
  - A transcript of your driver program's output
  - Your heuristic design and admissibility statements
  - Your empirical data
  - Any optional extra credit

  **A missing PDF or files in any other format will receive a zero.**