CSC 505 Bonus Homework, Spring 2020

I. Options:

You may complete an INDIVIDUAL programming assignment, **or** a collaborative creation of a podcast. The podcast team may contain up to 3 people.

The usual academic integrity restrictions apply: Do not discuss it at all or share information about the assignment with anyone outside of your team until after the assignment is turned in.

Do not forget to cite any resources you used, including our textbook. Citations can go into comments in the code.

1. Option 1: Programming assignment

In this assignment you will implement one of two programs. You may choose either one.
Deadline: *To be negotiated with the instructor.*

Option 1-A*

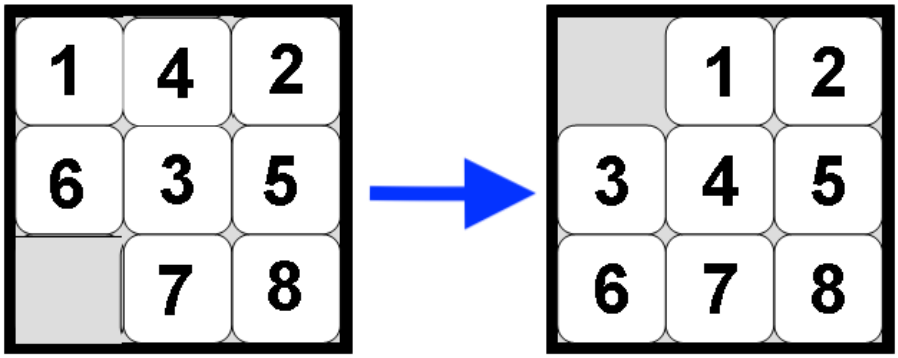
Implement A* search, and use it to solve the 8-puzzle from homework 6 with two different heuristics. Heuristic 1 is the sum of the Manhattan distances of each square from its current to its goal position. Heuristic 2 is the constant 0.

Your implementation of A* should be general. You should pass as arguments the graph, start and goal nodes, and the heuristic function.

The starting configuration will be supplied on the command line:

```
puzzle 1 4 2 6 3 5 _ 7 8
```

which corresponds to the starting state of the example below, using _ (a single underscore character) to represent the blank space.



The output should be a sequence of puzzle states. Each puzzle state should be printed using 3 rows, so that either a human or another program could understand it. There should be a blank line between puzzle states. The last state should represent the solved state shown in the diagram above.

E.g.

```
$ puzzle 1 4 2 6 3 5 _ 7 8
 1  4  2
 6  3  5
 _  7  8

...

 _  1  2
 3  4  5
 6  7  8
$
```

Option 1-Parallel

With the COVID-19 situation, we lost some time that we would have used to talk about parallel and distributed programming. If you want to write a parallel program anyway, as a means of learning by doing, here is a program to implement: a simple parallel merge sort.

Write a version of mergesort which creates 2 threads at a time (one for each half of the array) and waits until they finish before performing the merge.

It would have to be written in a language that has proper threads (which eliminates python). Maybe Java or C++ or C or Go or Rust? Feel free to suggest a different language.

Your program should accept a file name on the command line (the input file to sort) and it should write the sorted results to the standard output device. This makes it easy to redirect the output to /dev/null once it is working properly, or to a file for unit testing, or simply allow it to go to the terminal during debugging.

After the file name, the program should accept two numeric configuration parameters: min, and $threads$. The first parameter specifies the minimum array size for which 2 threads would be created. The idea is that when the arrays get small, the sorting will be done in the usual recursive way. A value of 1 would create threads for every sort phase.

The second configuration parameter is a maximum number of threads that should be created. Presumably, if we run your program with this parameter set to the number of cores on our machine, we will get the maximum performance benefit from the parallel implementation. But allowing many more threads will test hyper-threading and, beyond that, the efficiency of the thread implementation provided by the operating system.

2. Option 2: Podcast assignment

Generally, we might say there are 3 different flavors of algorithms:

- exact algorithms which compute the exact solution of a computational problem;
- heuristic algorithms that compute 'acceptable' solutions for computational problems, but for which there are no formal correctness proofs; and
- approximation algorithms which compute solutions that are provably close to the exact solution.

The goal of this assignment is to illustrate these different algorithm types by making a short podcast. Podcasting offers the opportunity to broadcast engaging audio content, which the audience can then listen to at any time. One of the great advantages of the podcast format is that it can be consumed in otherwise wasted time, or alongside a routine activity like commuting or doing the dishes.

In future versions of CSC 505, we would like to offer student-made podcasts, which were recorded by students from previous semesters. These will be short, and will supplement the lectures by focusing on key topics.

For this topic (the 3 kinds of algorithms listed above), we envision a single podcast episode, 5-7 minutes long, that will address the following:

1. What is an "exact algorithm"? What are some examples?
2. How does an "approximation algorithm" differ? When would I use an approximation algorithm? Is there a familiar algorithm that falls into this category?
3. How does a "heuristic algorithm" differ from an approximation algorithm? Why do such algorithms exist? Is there a familiar algorithm that falls into this category?

The assignment is to produce such a podcast, starting with an outline, then a written draft, and finally an audio recording (the podcast itself).

To plan your podcast, please first try to identify a scenario that is suitable to demonstrate each algorithm flavor. Try to find an example or application that the audience can easily relate to.

Your outline should consist of a definition and an example for each algorithm flavor.

Please share your outline with me so that I may offer feedback before you proceed further.

After the outline, develop a storyboard (written draft) for your podcast. In your storyboard, in addition to the text that will be spoken, address the following questions:

- What is the story's driving question?
- How will I engage my audience — and hold them?
- What will the audience remember when it's over?

Please share your storyboard with me so that I may offer feedback before you proceed further.

Finally, record your podcast!

A helpful guide to create a podcast is:
https://www.npr.org/2018/11/15/662070097/starting-your-podcast-a-guide-for-students

Deliverables:

- Outline
- Storyboard
- Podcast (5-7 minutes long)

Deadlines: *To be negotiated between the instructor and the team producing the podcast.*