SOME GRAPH EXPLORATION

1. Basics

Let $G = (V, E)$ be a graph with $V = \{1, \dots, n\}$. Define $\mathbb{1}$ to be the $n \times 1$ matrix (i.e., column vector) with 1 in every slot. Let A be the adjacency matrix of G.

Proposition 1. *$D_{i,i}$ is the number of neighbors of i.*

Proof. □

Definition 1. $L \coloneqq D - A$

Proposition 2. $L_{i,j} = \begin{cases} D_{i,i} & i = j \\ -1 & (i,j) \in E \\ 0 & \textit{else} \end{cases}$

Proof. □

2. Search

I have at least two questions:

(1) How does the fact that nodes have 2, 3, or 4 neighbors manifest in the adjacency matrix?
(2) Can we exploit this structure to parallelize exploration?

2.1. Structure of the graph.

The first goal here is to write down the adjacency matrix for the 100 or so nodes surrounding the 3×3 solved state. This is to start addressing question (1) above. See `explore_puzzle_space.py` for progress.

We'll be considering, e.g., a 3×3 puzzle configuration as a permutation of the set

$$\{1, 2, 3, 4, 5, 6, 7, 8, 0\}.$$

There are some particularities about where 0 goes in the null/solved permutation, but that shouldn't be too important now. Permutations will be denoted by σ.

Let $\sigma_0, \dots, \sigma_{99}$ be the configurations of the 100 nodes surrounding the solved state. Let $\lambda(\sigma)$ denote the Lehmer encoding of the permutation σ. So $\lambda(\sigma) \in \{0, \dots, 99\}$. Let $i_0, \dots, i_{99}$ be such that $\lambda(\sigma_{i_0}) < \cdots < \lambda(\sigma_{i_{99}})$, and define f by $f(\lambda(\sigma_{i_k})) \coloneqq k$.

Now, whenever a new node n is discovered from parent p, keep track of number $\lambda(n.\sigma)$ as well as the pair

$$(\lambda(n.\sigma), \lambda(p.\sigma)).$$

The size of the set $\{n.\sigma\}$ will tell us how big to make the adjacency matrix, and the pairs $(\lambda(n.\sigma), \lambda(p.\sigma))$ will tell us which entries are nonzero.