# CMPT 225 FINAL PROJECT DOCUMENT

1. **The choice of data structure, algorithm, heuristics and classes:**

For this project, I choose to approach solving the problem by using greedy-first search, which is an extension of A\* algorithm but neglecting the g-cost (how far we have moved from the start node).

I create 2 additional classes, the first class called fifteenpuzzle which read the input file and convert it into 2d-array and store the size of the board, the solution is the same as the assignment 1 solution you provide for us. The second class called Node to store that fifteenpuzzle class and to calculate the heuristic for each node (calculate the distance from that current node to the goal node-which is the board we want to find).

In the fifteenpuzzle class, I have to write 2 override function which are equals and compare in order to use it to compare and store it into Priority Queue. Also in order to keep track of the movement of the board, I have to create a member variable called “String move” to keep track of the movement of the black tile when I find the neighbors.

About calculating the heuristics, I use Manhattan distance, which calculate how far each tile from its right position and then add them up.

The algorithm I just follow mostly your pseudo code that TA gave us, but instead of setting parent and compare the f, because I already use Priority Queue so for each time I travel a new node, I just add it in both Open queue and Priority Queue, simple as that.

There are a few noticeable additional functions to support my algorithm such as generate neighbors, construct solution path, write solution to file. The generate neighbors is quite long to explain fully in detail, but in general, I just keep track of the position of the blank tile, if it can move up, move down, etc, then swap the number with that blank tile then add it up to a list called neighbors, then return that list.

The constructSolutionPath function is for back tracking then print out the movement (which tile move up, down, etc). I have to be really careful cause I have to print the movement in right order, so I add it reversely when I back-tracking it.

1. **Was one of heuristics always better?**

I don’t have time to explore other different heuristic, when I research about this project, Manhattan distance is one of the most popular and easy to approach so I just use it right away. But I believe that for larger board up to 7x7, as TA said, I have to create a heuristic that solve by column and row until there is only 3x3 sub-graph of the 2d array that haven’t solved, apply Manhattan and it could solve in a tickle of time.

1. **Describe which parts were harder?**

To be honest, the hardest part is to convert your pseudo code into actual code, after that is creating a Node class. I encounter a problem that I only travel again and again the same node and impossible to find the final solution because I didn’t override the .equals() function, a minor problem but cause me 2 days to find it!

The most easiest part definitely about reading the file, I just copy your solution of the assignment 1, add 1 more line to read the size of the board, delete unused part, then finish!

1. **Data structure that I didn’t include in the project?**

For me, the closed set at first I used HashSet, but later on I found out that if I use a hash map with the same key and value, using such function like .containsKey(node) is generally faster than .contains() of the HashSet. So that is only data structure that I used to write but now I don’t use it anymore.