**Full Name: Nguyễn Thế Huy**

**Class: K57CA**

**Student Code: 12020174**

**Assignment Report – Artificial Intelligence**

1. **How to detect and avoid repeated state in A\* implementation ?**

In my solution, I use set (a kind of data structure is supported in C++) to check duplicate

Sets of C++ are containers that store unique elements following a specific order and implemented in binary search tree.

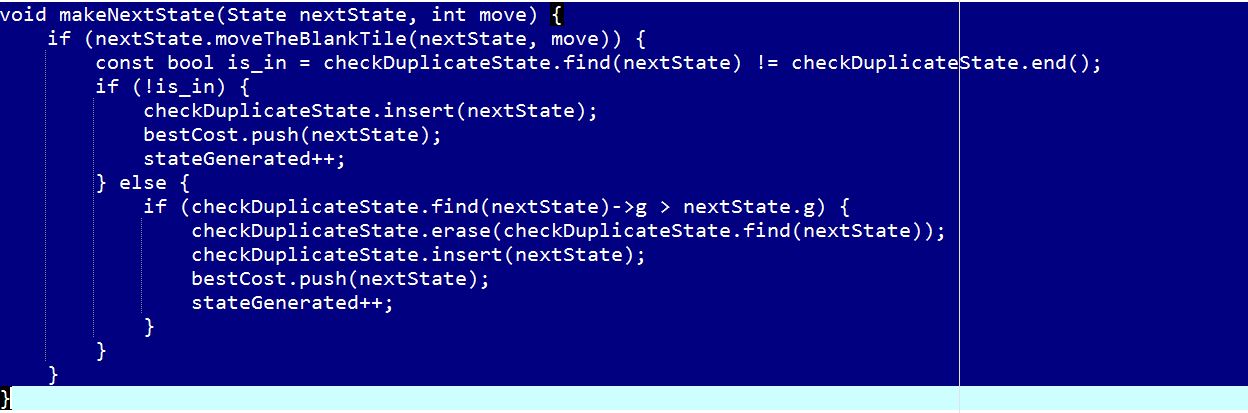
Here is my code to check if a state existed in set:



Explain:

We check whether a state is duplicated by its value in board. In a set of C++, it provides find() method to find a specific element. If found, find() return pointer to element is found. If no, find() return pointer to the last element of set. So, we can check: if find(state) is not equal to last element (provided by end() method), this state existed in our set. If no duplicate, we can insert it to the set. But if we have the same state (means that the same board), we have to check g (cost from initial state to current state). If new state has g better than current state in set, we delete old state and insert new state

Full code:



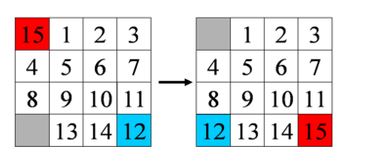
Because of implementing in binary search tree, set is fast to find and check duplicated element with time complexity average: O(log n)

1. **My heuristic is by flying distance:**

**The formula:**



We consider this example:



The flying distance in this example 

In a right triangle, the side opposite the [right angle](http://en.wikipedia.org/wiki/Right_angle) is equal to the sum of the squares of the other two sides (Pythagorean theorem) and less than sum of them.

Flying distance represents the length of the hypotenuse and Manhantan distance is  the sum of the right triangle's other two sides. So: flying distance always less than Manhantan distance. On the other hand, manhantan distance is admissible. So we have proved that, flying distance always admissible

Flying distance is worse than Manhantan distance, but better than missed place