Wednesday 29 November 2017

Searching Algorithms

Problem Statement

For each problem , 8-puzzle, shortest-distance, there will be entities called nodes, these nodes has to provide:

- state: It has to give its current state.
- parent: linked to the parent node.
- action: the actions that leads them to their successors
- path-cost:

But nodes are for each problem completely different data structures, for example in 8-puzzle problem, node state specifies the location of each of the eight tiles in one of the nine squares, also include the position of the blank , but in the case of shortest-distance problem just a string defines the state (the name of the location). It seems that this class node or nodesearch might be structured with the notion of class templates

```
template <class Type> class nodesearch{

nodesearch(const Type & t):state(t), next(0){}

Type state;

nodesearch* next;

};
```

For the data structure of the problem , there are several things to be defined

- The initial state which is of the same type of `state`.
- The goal state which is of the same type of `state`.
- A mapping between each `state` and the actions at that `state`.
- How the actions change the `state`.
- Path cost

These three points defined above are for each case different, for instance for the 8-puzzle the mapping is only the rule that the blank can only move up or right if it is in the left corner below and so on (Operators:blank moves left, right, up or down.). For the case of the shortest-distance we need to define a mapping :

```
std::map<std::string, std::vector<string>>
```