

Tutorial on A*

Prof. Qiang Yang

TA: Derek Hao Hu

Algorithm 1 A***Input:** initial state, goal state**Output:** solution path

```
1:  $Q$  and  $visited$  are 2 empty containers
2: add initial state to both  $Q$  and  $visited$ 
3: while  $Q$  is not empty do
4:   Extract  $C$  with a minimum  $f$  value from  $Q$ 
5:   if  $C$  is a goal state then
6:     print the solution and terminates
7:   end if
8:   for every direction  $d$  the MAN can go do
9:     the MAN goes toward  $d$ , we can get a new state  $next$  if we can move.
10:    if  $next$  is not in  $visited$  then
11:      add it to  $visited$ 
12:      calculate its  $f$  value and add it to  $Q$ 
13:    end if
14:  end for
15: end while
```

The pseudocode above may confuses you in :

1. Q: What is a container?

A: A container is something that you can put a collection of data in it. The simplest container maybe an array. In our code, $visited$ is a container that holds all the nodes we have generated, so we can use it avoid regenerating.

2. Q: How to implement a container?

A: We need to see what operations we need to do with a specific container. In our code, Q is a container has INSERT and EXTRACTMIN¹. You can use an array do these 2 operations, but not efficiently! $visited$ does not need EXTRACTMIN, but needs FIND instead to be implemented efficiently. So although I called them containers, they require different implementations.

3. Q: What is the efficient implementation of Q ?

A: Heaps. Refer to your 171 textbook.

4. Q: What is the efficient implementation of $visited$?

B: Search trees or hash tables. Refer to your 171 textbook.

¹EXTRACTMIN means that you delete the node with smallest f value in a container and return it.

5. YOU DO NOT HAVE TO IMPLEMENT THEM AT ALL if you use C++ or Java. In C++, you may want to use `set`, `map` classes. In Java, you may want to use `HashSet`, `PriorityQueue` classes.

6. Q: How to represent a state?

A: This part should be done by yourself.

7. Q: How to build a solution path.

A: Remember that every node has only one parent except the initial one, which does not have a parent. It is easy to show that it is correct because we don't generate the same node twice. So for every node, add a link variable to its parent. When we reach a goal node, we simply walk back to the initial node using these parent links to find a solution path.

8. Q: How to write f function?

A: Again this is a core of the programming assignment, which I will not tell you how to do exactly. $f = c + h$, c is what we can get from $next$: $c(initialstate) = 0$, $c(next) = c(C) + 1$. The value of $h(next)$ is how many steps at least we need to reach the goal. Firstly it is always an underestimate of the real needs. Secondly, it is better to be close to $h * (next)$.