# Artificial Intelligence Homework 1
# Search

2014 / 10 / 15

# Question 1~4 – Graph Search

- Motivation: By abstracting the problems, we can solve them using this ***problem-independent*** method.
- To abstract a problem, we define
  - the initial state
  - the goal state
  - the successors (or children) of each state
  - (optional) the cost of each action (i.e. state transition)

  of the problem.

# Question 1~4 – Graph Search

- It is recommended that you begin with a function "graphSearch" defined by yourself
- def graphSearch(problem, search):

GRAPH-SEARCH(*problem*)

```
1  initialize the frontier using the initial state of problem
2  initialize the explored set to be empty
3  repeat
4       if the frontier is empty
5            return failure
6       choose a leaf node and remove it from the frontier.
7       if the node contains a goal state
8            return the corresponding solution
9       add the node to the explored set
10      expand the chosen node
11      if not in the frontier or explored set
12           add the resulting nodes to the frontier
```

# Question 1~4 – Graph Search

- graphSearch(problem, search): (in search.py)

```
_____                                  code initializing the frontier (see util.py)
frontier.push(problem.getStartState())
explored = set()
actionList = []
transitionTable = dict()
node = problem.getStartState()
while ( True ):
  ..............................
  for leaf in leaves:
    if ..............................
      frontier.....................
      _________ record in transitionTable
_____                                  backtracing
return actionList
```

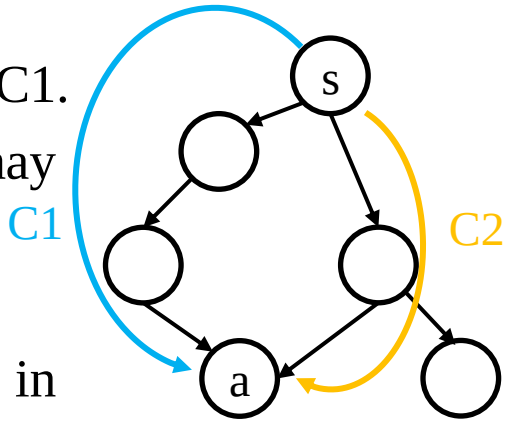
GRAPH-SEARCH(*problem*)

```
1  initialize the frontier using the initial state of problem
2  initialize the explored set to be empty
3  repeat
4      if the frontier is empty
5          return failure
6      choose a leaf node and remove it from the frontier.
7      if the node contains a goal state
8          return the corresponding solution
9      add the node to the explored set
10     expand the chosen node
11     if not in the frontier or explored set
12         add the resulting nodes to the frontier
```

Note on Transition Table:
dict() is like map in C++ STL
Key: child, Element: [parent, action]

# Question 1~4 – Graph Search

- Question 1: DFS – Stack
- Question 2: BFS – Queue
- Question 3: UCS – Priority Queue, Tricky!
  - The first time when node “a” is explored, the cost is C1.
  - Since we are considering graph search, we may encounter “a” second time, with cost C2.
  - What if C2<C1?
  - My method: define function “replace” and “has” in PriorityQueue. (If you plan to do so, please redefine the class PriorityQueue in search.py.)
  - Or you may use “lambda” keyword in Python.
  - Or any method you like.
- Question 4: A* – with heuristic




| Priority Queue | |
|---|---|
| Node | Cost |
| … | … |
| a | ~~C1~~ C2 |
| … | … |

# Question 1~4 – Graph Search

- Reminder: Please make sure your graph search program is ***problem-independent***!
  - By testing “python eightpuzzle.py”

# Question 5 – Defining Game States

- Objective: Abstract the Corner Problem
  - There are four foods at each corner at the beginning.
  - Once the Pacman wanders to a corner with food, the food will be eaten and no longer exist.
  - The goal is the Pacman eating up all foods, which declares the end of the game.