
| | |
|---|---|
| 과목명 | 인공지능 |
| 담당교수 | 황두성 교수님 |
| 학과 | 소프트웨어학과 |
| 학번 | 32153180 |
| 이름 | 이상민 |
| 제출일자 | 2019.10.21 |

# Problem 1

Constraint satisfaction problems(CSPs) are defined with a set of variables or objects whose state must satisfy a number of constraints. CSPs represent the entities in a problem as a homogeneous collection of finite constraints over variables, which is solved by constraint satisfaction algorithms.

*(a) Give two or three realistic examples around your environment which implement CSPs.*

① Sudoku game : Fill the empty squares with numbers ranging from 1 to 9 in such a way that no row, column has a number repeating itself. The remaining squares that are to be filled are known as variables and the range of numbers that can fill them is known as a domain. Variables take on values from the domain and these are known as constraints.

② Crossword puzzle : Variables are words that have to be filled in and domains are English words of correct length. Words have the same letters at cells where they intersect and these are constraints.

*(b) Formulate the above mentioned applications in terms of goals(or objective function) and constraints.*

① Sudoku game : Fill the empty squares with values where value$\in$(1, 10] and no row, column or a block has a number repeating itself.

② Crossword puzzle : Fill the empty squares with alphabets where alphabet$\in$ (a, z) and words have the same letters when they intersect.

## Problem 2

Graph coloring problems and applications were taught in the previous lecture. As we discussed, graph coloring is widely used. But, there is no efficient algorithm available for coloring a graph with minimum number of colors as the problem is a know NP-complete problem. There are approximate algorithms to solve the graph coloring problem.

*(a) Google NP-complete problems and describe it on your word.*

문제가 주어졌을 때, 이 문제가 현실적으로 풀 수 있는가를 정의할 때 쓰이는 개념이다.

P : Polynomial. P군에 속하는 문제들은 다항 시간에 해결할 수 있는 문제들을 의미한다.

NP : Non-Deterministic Polynomial. NP군에 속하는 문제들은 다항 시간에 확인 가능한 문제들을 의미한다.

즉, 문제의 답과 그에 대한 근거가 주어졌을 때, 그것이 옳은 근거임을 다항 시간에 확인 가능하다는 뜻이다.

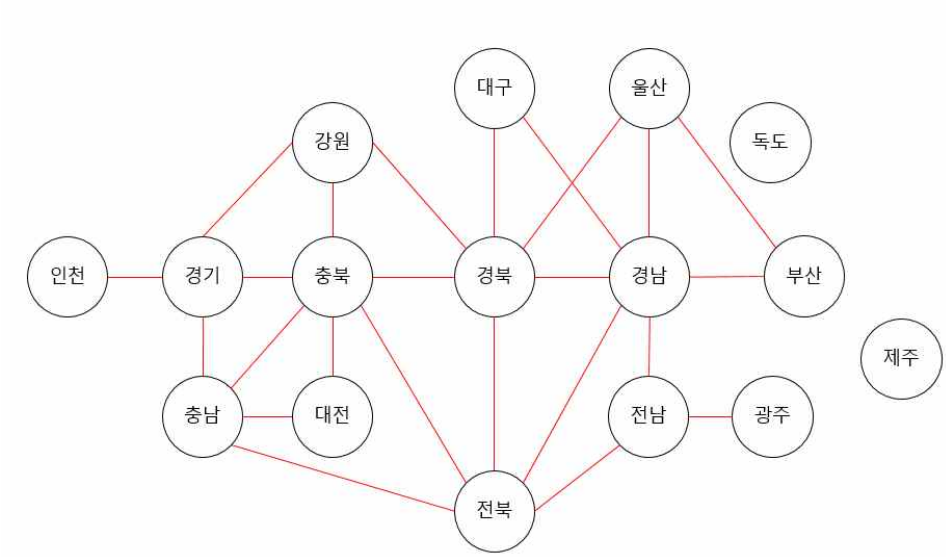
P는 NP에 포함될 것이라고 강력하게 추정되고 있을 뿐이다. 그렇지만 NP에 P가 아닌 문제도 포함되는지는 아직 아무도 증명하지 못했다.

NP-Hard : 어떤 하나의 NP 문제가 주어진 문제 A에 대하여 다항 시간 변환이 가능하면 A는 NP-Hard이다.

NP-Complete : NP-Hard보다 조금 더 좁은 개념으로 다음 두 조건을 만족한다.

1. A는 NP이다.
2. A는 NP-Hard이다.

*(b) Given the Korea administrative region map(Figure 1), convert the map into a graph.*



*(c) Find the minimum number of colors for the Korea map.*

- *Design your algorithm in pseudo code.*

```
backtrack(city_name, cur_assign) {
it (city_name == {})
return cur_assign
remove a variable X from city_name;
for (each value a in 'X's domain) {
if (X=a is consistent with cur_assign) {
add X=a to cur_assign;
res = search(city_name, cur_assign);
if res != false
return res
remove X=a from cur_assign;
}
}
return false;
}
```

*(d) Implement your algorithm to find the minimum number of colors.*

- *Submit your code and results.*
- *Discuss about how to analyze the space and time complexity of your own algorithm.*


Figure 1: South Korea administrative region map

Time complexity : $O(n \times m^n)$  n : number of vertex / m : number of color

```
from simpleai.search import CspProblem, backtrack

def constraint_func(names, values):
    return values[0] != values[1]

if __name__ == '__main__':
    names = ('인천', '경기', '강원', '경북', '울산', '충북', '전북', '대구',
             '충남', '대전', '경남', '부산', '광주', '전남', '독도', '제주도')
    colors = dict((name, list(i for i in range(1, len(names) + 1))) for name in names)

constraints = [
(('인천', '경기'), constraint_func), (('경기', '강원'), constraint_func),
(('경기', '충북'), constraint_func), (('경기', '충남'), constraint_func),
(('강원', '충북'), constraint_func), (('강원', '경북'), constraint_func),
(('경북', '충북'), constraint_func), (('경북', '전북'), constraint_func),
(('경북', '대구'), constraint_func), (('경북', '울산'), constraint_func),
(('경북', '경남'), constraint_func), (('울산', '경남'), constraint_func),
(('울산', '부산'), constraint_func), (('충북', '충남'), constraint_func),
(('충북', '대전'), constraint_func), (('충북', '전북'), constraint_func),
(('부산', '경남'), constraint_func), (('경남', '대구'), constraint_func),
(('경남', '전북'), constraint_func), (('경남', '전남'), constraint_func),
(('경남', '전남'), constraint_func), (('전남', '광주'), constraint_func),
(('전남', '전북'), constraint_func), (('전북', '충남'),constraint_func),
(('충남', '대전'), constraint_func)
]

problem = CspProblem(names, colors, constraints)

output = backtrack(problem)
print('<< Color mapping >>\n')
min_color = 0
for k, v in output.items():
    if v == 1:
        print(k, '==> red')
    elif v == 2:
        print(k, '==> green')
    elif v == 3:
        print(k, '==> blue')
```

```
    elif v == 4:
        print(k, '==> black')

    if min_color < v:
        min_color = v
print('\nNumber of Color:', min_color)
```

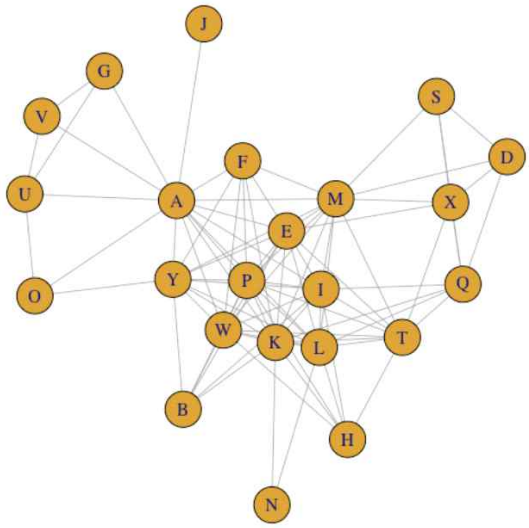



Figure 2: A undirected graph

# Problem 3

For the following graph in Figure 2, answer the questions:

*(a) Report the order of the vertices encountered on a breadth-first search starting from vertex A. Choose the vertices in alphabetical order.*

A

E F G I J K L M O P U V W Y

F G I J K L M O P U V W Y B T

O P U V W Y B T H N Q D S X

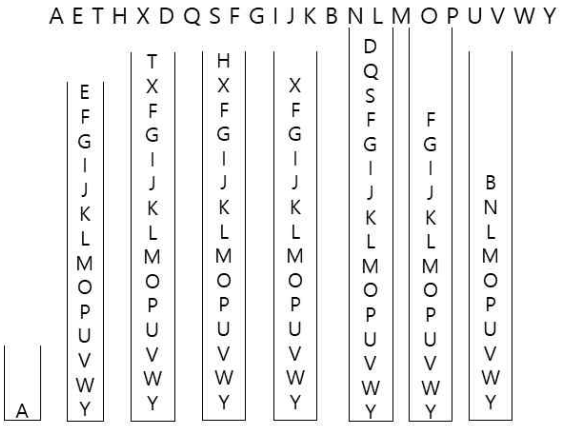
아래와 같은 순서로 탐색

A E F G I J K L M O P U V W Y B T H N Q D S X

*(b) Report the order of the vertices encountered on a depth-first search starting from vertex A. Choose the vertices in alphabetical order.*

다음과 같은 순서로 탐색

A E T H X D Q S F G I J K B N L M O P U V W Y




## Problem 4

Propose a PEAS description of the task of designing an automated taxi driver. PEAS is an abbreviation of Performance measure, Environment, Actuators, and Sensors.

| | |
|---|---|
| **Performance measure** | safe, fast, legal, comfortable trip, maximize profits |
| **Environment** | roads, other traffic, pedestrians |
| **Actuators** | steering wheel, accelerator, brake, signal, horn |
| **Sensors** | cameras, sonar, speedometer, GPS, odometer, engine sensors, keyboard |

## Problem 5

Find the solution of the graph in Figure 3. The edge weight is distance between two nodes and the weight below a node is the cost to node G.

*(a) When we start at node A and ignore the given cost values, find the breadth-first search of the graph.*


<Queue contents>

Breadth-first traversal : **A B C D E F G**

*(b) When we start at node A and ignore the given cost values, give the depth-first search of the graph.*


<Stack contents>

Depth-first traversal : **A B E F G D C**

*(c) Find the optimal solution from A to G by applying the A* algorithm.*

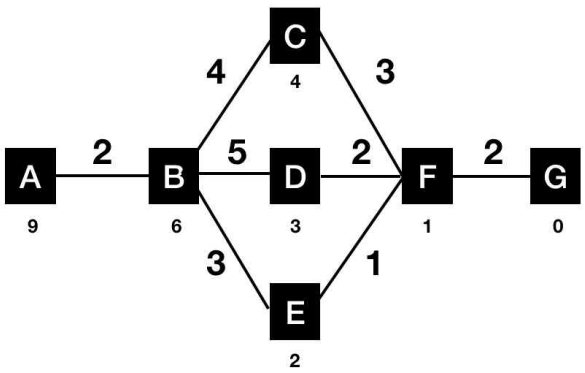



Figure 3: A graph example

## Problem 6

For any event and in a sample space, prove

$$P(x|y)P(y) = P(y|x)P(x).$$

$P(x|y)P(y) = P(y|x)P(x)$ 에서

$$P(x|y) = \frac{P(x \cap y)}{P(y)},$$

$$P(y|x) = \frac{P(y \cap x)}{P(x)} \text{ 이다.}$$

$x, y$가 같은 공간에서

$P(x \cap y) = P(y \cap x)$ 이다.

$$\therefore P(x|y)P(y) = P(x \cap y) = P(y \cap x) = P(y|x)P(x)$$

## Problem 7

Given the discrete probability distribution in Table 1, answer the questions.

*(a) Compute* $P(W)$.

| P(W) | |
|---|---|
| W | P |
| sun | 0.6 |
| rain | 0.4 |

*(b) Compute* $P(W|S = winter)$.

| P(W\|S = winter) | | |
|---|---|---|
| S | W | P |
| winter | rain | 0.625 |
| | sun | 0.375 |

*(c) Compute* $P(W|S = winter, T = cold)$.

| P(W\|S = winter, T = cold) | | | |
|---|---|---|---|
| S | T | W | P |
| winter | cold | rain | 0.67 |
| | | sun | 0.33 |

Table 1: A discrete probability distribution

| S | T | W | P |
|---|---|---|---|
| summer | hot | sun | 0.3 |
| summer | hot | rain | 0.05 |
| summer | cold | sun | 0.15 |
| summer | cold | rain | 0.1 |
| winter | hot | sun | 0.05 |
| winter | hot | rain | 0.05 |
| winter | cold | sun | 0.1 |
| winter | cold | rain | 0.2 |