중간고사 대체 과제

학과 : 컴퓨터학과

학번 : 2018320161

이름 : 송대선

제출 날짜 : 2020-05-11

1st problem

We assume that for given a diredcted graph G, each edge is labeled by an element of some closed semiring and there is special vertices ‘u’ and ‘v’.

The labels come from S where (S, +, •, 0, 1).

This is the algorithm that compute c(u,v) for all possible pair of vertices u and v that u, v ∈ V[G]

```
COMPUTE-PATH(G, u, v)
c(u, v)=0 //this “0” is an element of some closed semiring
do if u = v
  c(u, v) ← 1 //this “1” is an element of some closed semiring
for each vertex k ∈ V[G] - {u}
  do color[k] ← WHITE
      path[k] ← 0 //this “0” is an element of some closed semiring
color[u] ← GRAY
Q ← ∅
ENQUEUE(Q, u)
while Q ≠ ∅
do k ← DEQUEUE(Q)
   for each a ∈ Adj[k]
       do if color[a] = WHITE
             then color[a] ← GRAY
             if k = u
                then path[a] ← Label[(k, a)]
             else
                then if path[a] = 0 //this “0” is an element of some closed semiring
                        then path[a] ← •(path[k], Label[(k, a)])
                     else
                        then path[a] ← +(path[a], •(path[k], Label[(k, a)]))
            if a ≠ v
              then ENQUEUE(Q, a)
   color[k] ← BLACK
if c(u, v) = 1
  then c(u, v) ← +(c(u, v), path[v])
else
  then c(u, v) ← path[v]
RERUTN c(u, v)
```

We can compute **c(u, v) by using** this algorithm

2nd problem

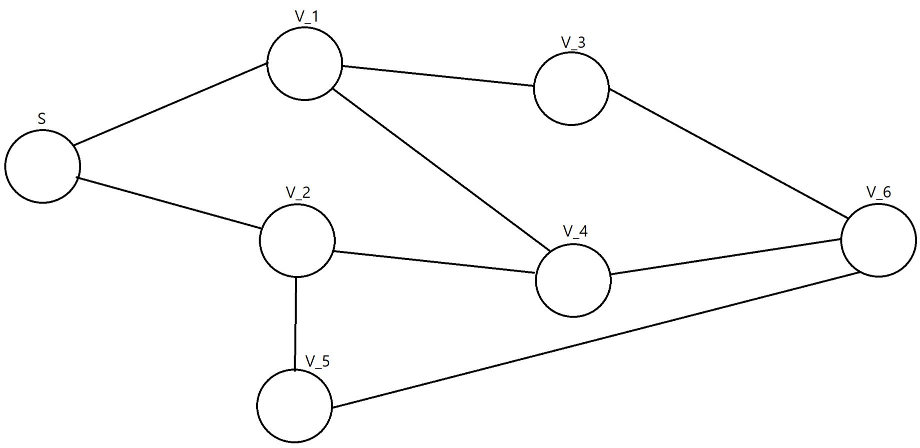
```
COMPUTE-2-COLORING(G, s)
for each vertex u ∈ V[G]
  do color[u] ← WHITE
color[s] ← RED
Q ← ∅
```

```
5 ENQUEUE(Q, u)
6 while Q ≠ ∅
7 do u ← DEQUEUE(Q)
8    for each v ∈ Adj[u]
9       do if color[v] = WHITE
10               then if color[u] = RED
11                       then color[v] ← BLUE
12                    else
13                       then color[v] ← RED
14                    ENQUEUE(Q, v)
15         else if color[v] = color[u]
16                    then RETURN NO
17 RETURN YES
```

The algorithm COMPUTE-2-COLORING is made by applying BFS.

We can solve 2-coloring problem by using this algorithm.

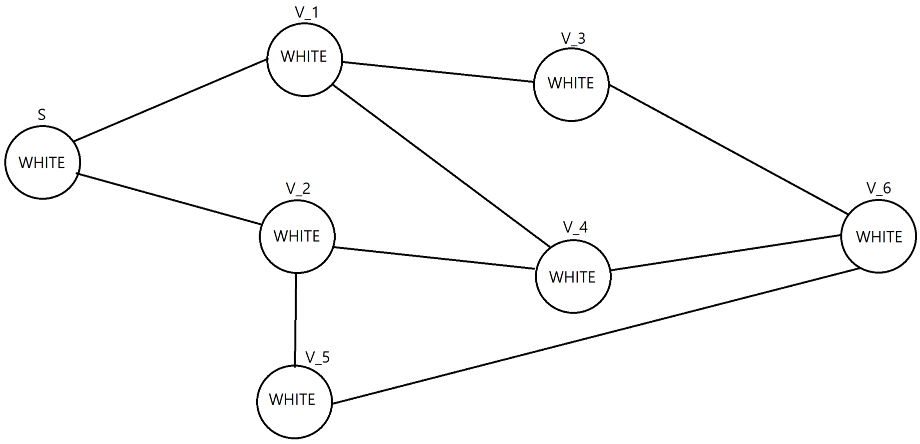
I`m going to explain how this algorithm is done with example.



The picture above is a undirected graph G that is $|V| \geq 2$, $|E| \geq 1$

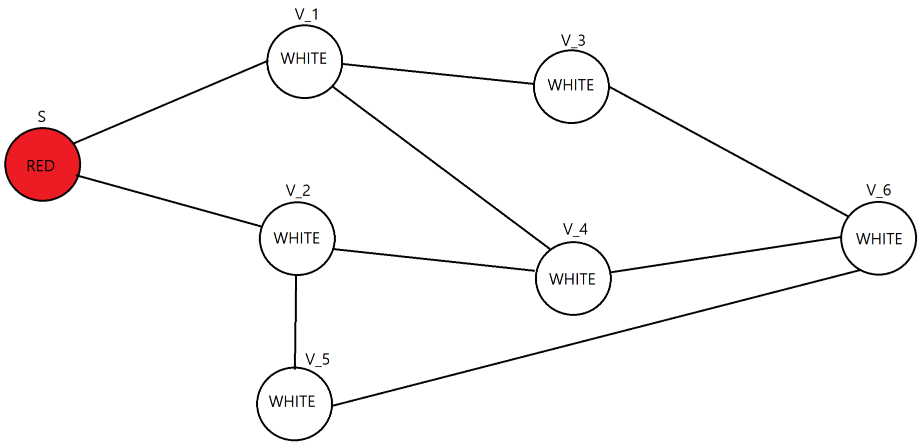
After running line 1-2, all vertices of G(S, v_1, v_2, v_3, v_4, v_5, v_6) are colored by WHITE

Graph G is as follows.



After running line 3-5, vertex S is colored by RED, QUEUE Q is initialized, and the vertex S is enqueued. So Q=(S)

Graph G is as follows.



We are in the while loop at 6-16 because Q is not empty.

Dequeue S of Q.

Adjacent vertices of S are v_1, v_2

v_1 is colored with WHITE and S is colored with RED

→ Color v_1 with BLUE

→ Enqueue v_1

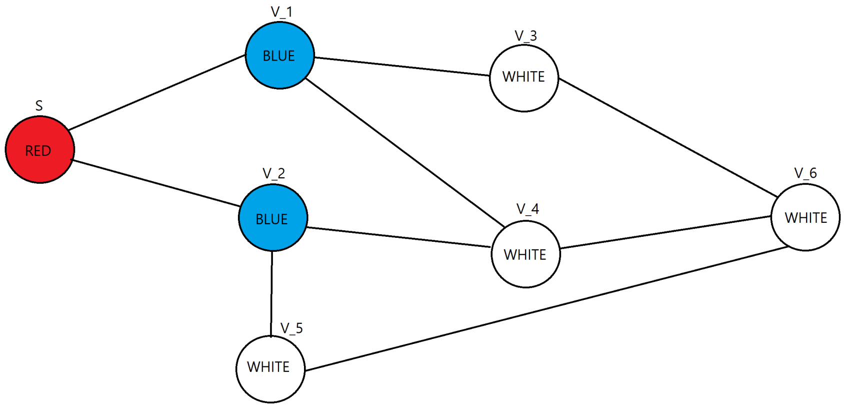
v_2 is colored with WHITE and S is colored with RED

→ Color v_2 with BLUE

→ Enqueue v_2

Q is (v_1, v_2)

Graph G is as follows.



We are in the while loop at 6-16 because Q is not empty.

Dequeue v_1 of Q.

Adjacent vertices of S are S, v_3, v_4

S is not colored with WHITE and colors of S and v_1 are not same

v_3 is colored with WHITE and v_1 is colored with BLUE

→ Color v_3 with RED

→ Enqueue v_3

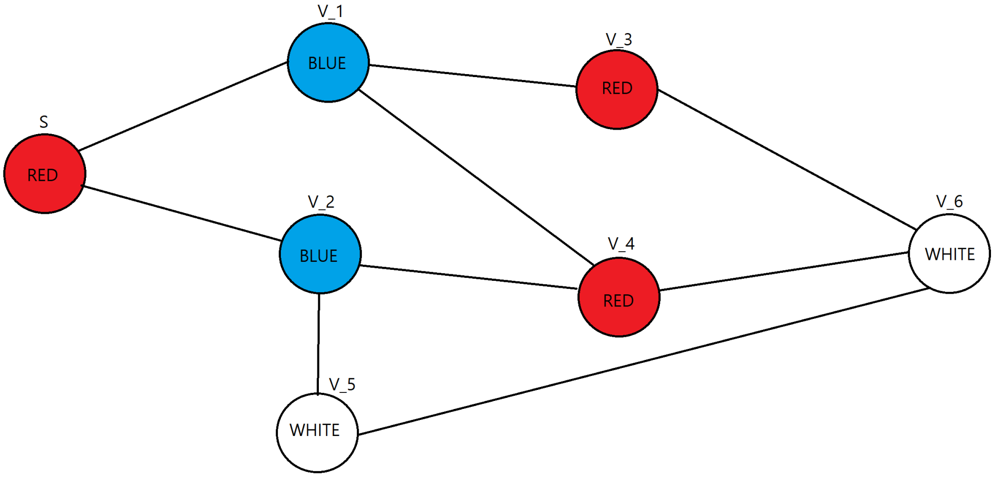
v_4 is colored with WHITE and v_1 is colored with BLUE

→ Color v_4 with RED

→ Enqueue v_4

Q is (v_2, v_3, v_4)

Graph G is as follows.



We are in the while loop at 6-16 because Q is not empty.

Dequeue v_2 of Q.

Adjacent vertices of S are S, v_4, v_5

S is not colored with WHITE and colors of S and v_2 are not same

v_4 is not colored with WHITE and colors of v_4 and v_2 are not same

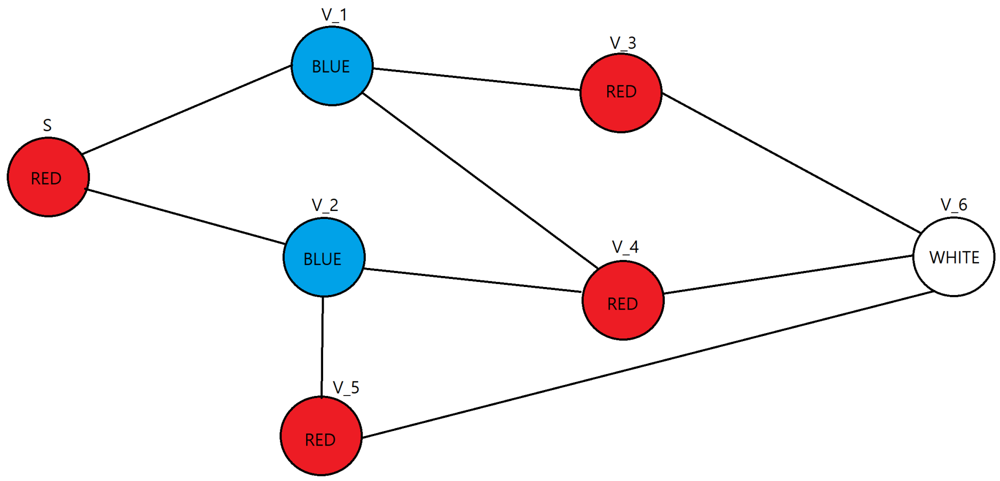
v_5 is colored with WHITE and v_2 is colored with BLUE

→ Color v_5 with RED

→ Enqueue v_5

Q is (v_3, v_4, v_5)

Graph G is as follows.



We are in the while loop at 6-16 because Q is not empty.

Dequeue v_3 of Q.

Adjacent vertices of S are v_1, v_6

v_1 is not colored with WHITE and colors of v_1 and v_3 are not same

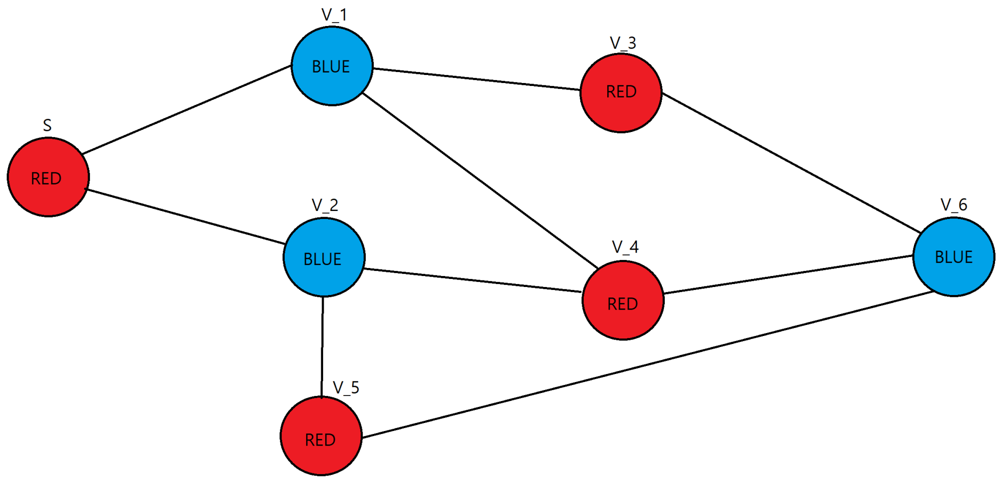
v_6 is colored with WHITE and v_3 is colored with RED

→ Color v_6 with BLUE

→ Enqueue v_6

Q is (v_4, v_5, v_6)

Graph G is as follows.



We are in the while loop at 6-16 because Q is not empty.

Dequeue v_4 of Q.

Adjacent vertices of S are v_1, v_2, v_6

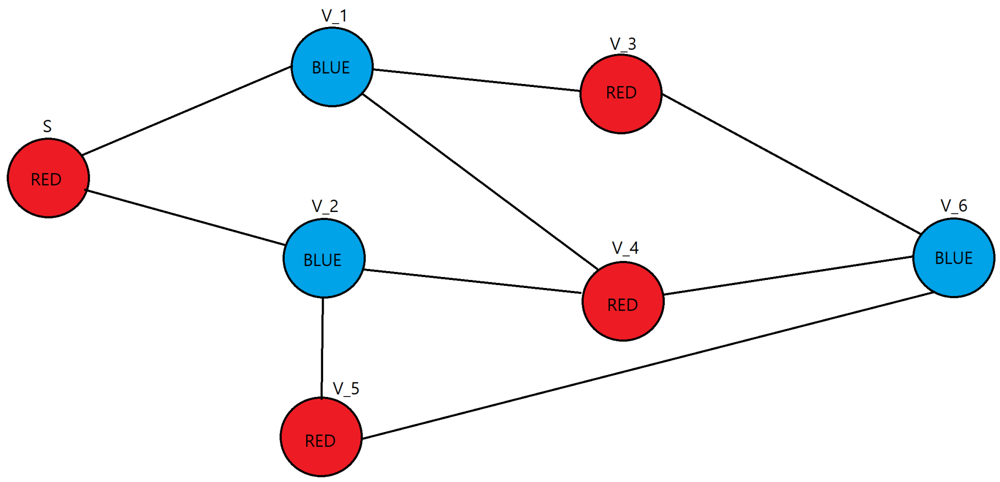
v_1 is not colored with WHITE and colors of v_1 and v_4 are not same

v_2 is not colored with WHITE and colors of v_2 and v_4 are not same

v_6 is not colored with WHITE and colors of v_6 and v_4 are not same

Q is (v_5, v_6)

Graph G is as follows.



We are in the while loop at 6-16 because Q is not empty.

Dequeue v_5 of Q.

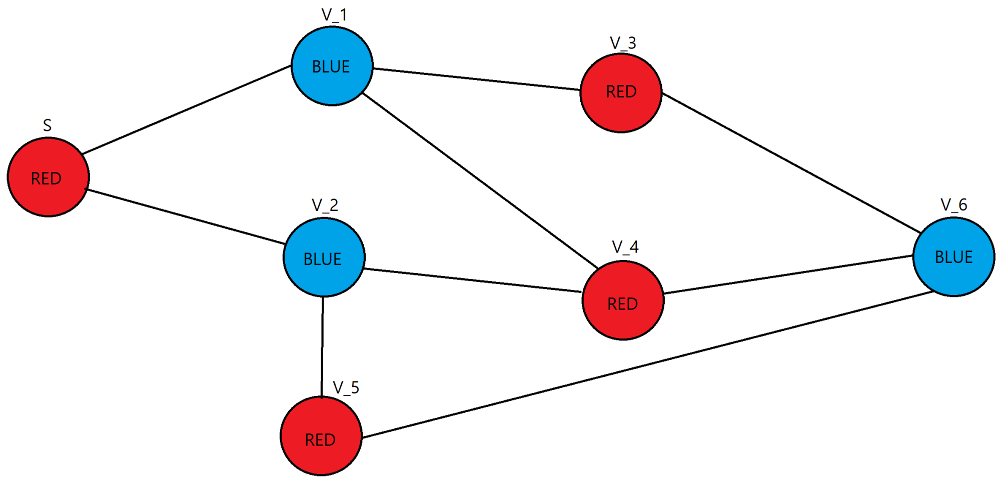
Adjacent vertices of S are v_2, v_6

v_2 is not colored with WHITE and colors of v_2 and v_4 are not same

v_6 is not colored with WHITE and colors of v_6 and v_4 are not same

Q is (v_6)

Graph G is as follows.



We are in the while loop at 6-16 because Q is not empty.

Dequeue v_6 of Q.

Adjacent vertices of S are v_3, v_4, v_5

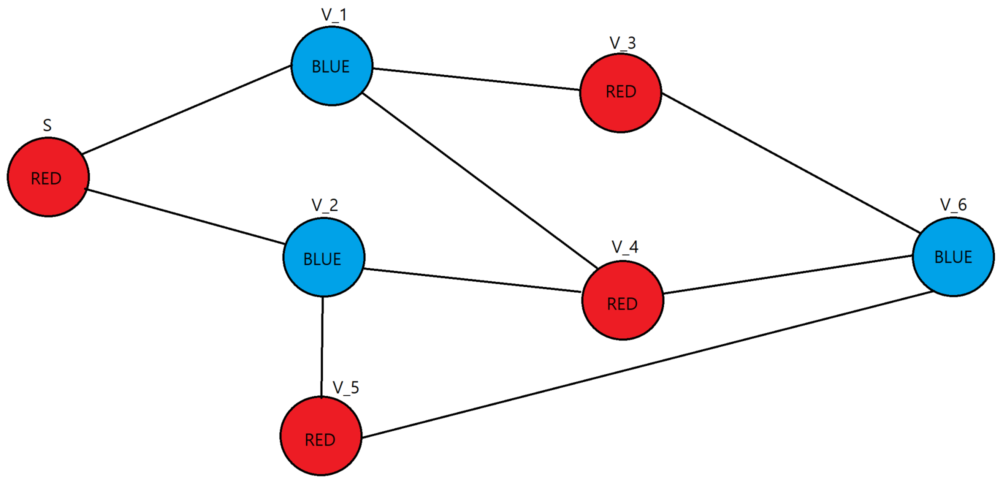
v_3 is not colored with WHITE and colors of v_3 and v_6 are not same

v_4 is not colored with WHITE and colors of v_4 and v_6 are not same

v_5 is not colored with WHITE and colors of v_5 and v_6 are not same

Q is ()

Graph G is as follows.



We are out of while loop at 6-16 because Q is empty.

After running line 17, the algorithm gives us the anwser "YES"

3rd prblem

polynomial-time computable problems은 in practice에서 훨씬 적은 시간을 필요로 한다. polynomial-time computable problems을 다항시간내에 해결하는 모델이 있다면 그 문제를 다항시간 내에 해결하는 또 다른 모델이 있다.
polynomial-time computable problems class는 addition, multiplication, and composition에 대해 closed되어 있다. 즉, 다항식 시간 알고리즘은 여러개를 겹쳐서사용해도 다항식 시간 알고리즘이다.

Abstract problems

"problem"을 정의해보자. 문제 Q는 instance의 set I와 colution의 set S에 대한 binary relation으로 정의된다. theory of NP-completeness은 yes/no solution을 가지는 decision problems들로 범위를 제한한다. 즉, 이 경우에는 abstract decision problem을 instance를 [0, 1]의 solution으로 mapping시키는 일종의 function이라 볼 수 있다. 대부분의 problem들은 특정 값들을 minimize 또는 maximize시키는 problem이다. minimize 또는 maximize시키는 problem을 decision problem으로 바꾸는 것은 어렵지 않다.

Encodings

문제를 컴퓨터가 인식하려면 컴퓨터가 인식할 수 있는 방식으로 instance를 mapping해줘야 한다. 주로 {0, 1, 10, 11, 100,...}같은 binary strings으로 mapping한다. mapping은 단순 숫자 뿐만 아니라, 다각형, 그래프, 함수, 순서 쌍, 프로그램 등 거의 모든 것을 binary strings으로 mapping가능하다. instance set이 binary strings으로 mapping된 problem을 concrete problem이라고 부른다. instance의 길이가 n인데 알고리즘이 O(n^k)시간 내로 solution을 줄 수 있는 경우, 그 concrete problem은 polynomial-time solvable하다고 말할 수 있다. The complexity class P는 polynomial-time solvable한 concrete decision problems의 set이라고 말할 수 있다. abstract problem의 solution과 그 problem을 mapping하여 만든 concrete problem의 solution은 동일하다. 알고리즘의 실행시간은 그 문제의 encoding방식에 많이 의존된다. input으로 {0, 1}*을 받아 output으로 f(x)를 생성하는 polynomial-time algorithm A이 존재하면, function f : {0, 1}* → {0,1}*은 polynomial-time computable하다고 한다.

A formal-language framework

Σ : 일종의 alphabet이다. 유한개의 symbols의 set이다.
L : language이다. Σ을 사용하는 L은 Σ의 symbols으로 만든 strings의 set이다.
ε : empty string
Ø : empty language
Σ* : 가능한 모든 string들로 이루어진 L
Σ을 사용하는 모든 L은 Σ*에 포함된다.
Union, intersection, concatenation, closure, Kleene star와 같은 operation들이 있다.
formal-language는 다양한 problem들을 mapping하는데 사용가능하다.