CPTS 515 Takehome Midterm

Yang Zhang 11529139

(1) This problem basically has two cases

① G has loop(s)

② G is acyclic graph

To find and testify if G is acyclic graph using following algorithm:

```
find_loop(v):
    put v → parent_list
    for each neighbor n0 of v:
        n0.pre = v
        if there is a neighbor n1 of n0 in parent_list:
            found loop, record the loop by tracing the pre
            from n0 → n1
        find_loop(n0)
    clear parent_list
```

case 1 G is acyclic graph

If G is acyclic, a greedy approach may apply here.

The basic idea is from Dijkstra's algorithm,

The cost function now is dynamically changing based on current counts of red, green, yellow

For example (C → Count, EC → EdgeCost)

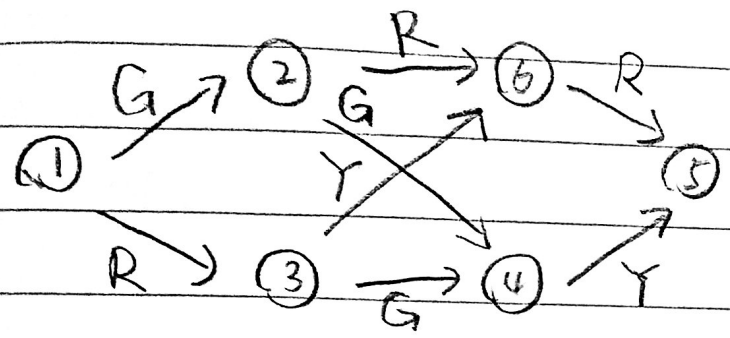
if $C_{red} = C_{green} = C_{yellow}$, $EC_{red} = EC_{green} = EC_{yellow} = 1$

if $C_{red} > C_{green} > C_{yellow}$ $EC_{red} = 3$, $EC_{green} = 2$, $EC_{yellow} = 1$

if $C_{red} > C_{green} = C_{yellow}$, $EC_{red} = 2$, $EC_{green} = EC_{yellow} = 1$

if founded path has non-equal C_{Red}, C_{yellow} and C_{green} set the cost to ∞.

Here is simple example



→ ① — ② RGY 010 cost 1 visit_list = {①}

③ 100 1

→ ② — ⑥ RGY 110 cost 2 visit_list {①, ②}

④ 200 3

among ③ ④ ⑥

choose ③

→ ③ — ⑥ RGY 101 cost 2 visit_list {①, ②, ③}

④ 110 2

among ④, ⑥

choose ⑥ visit_list {①, ②, ③, ⑥}

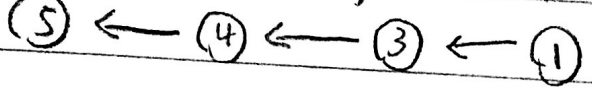
→ ⑥ — ⑤ RGY 201 cost 4

among ④ ⑥

choose ④

RGY cost
1 1 1 3

path found, backtracking with pre:

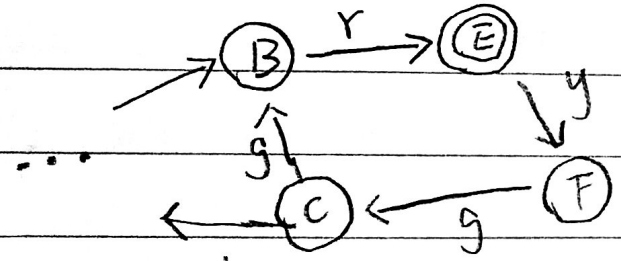


case 2 G has loop(s)

① use algorithm find_loop to get all the loops from G

② For each loop caculate the edge counter formula of each entrance, exit vertices pair

- for example:



For the above loop, B is the only entrance vertex C, E are the exit points (E is the end node; so E can be identified as an exit)

For pair (B, E):

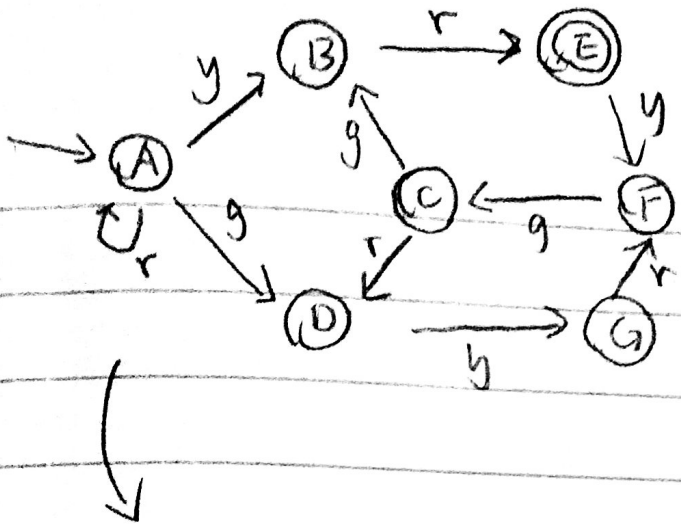
edge counter = $r + 2n \cdot g + n \cdot y + n \cdot r$

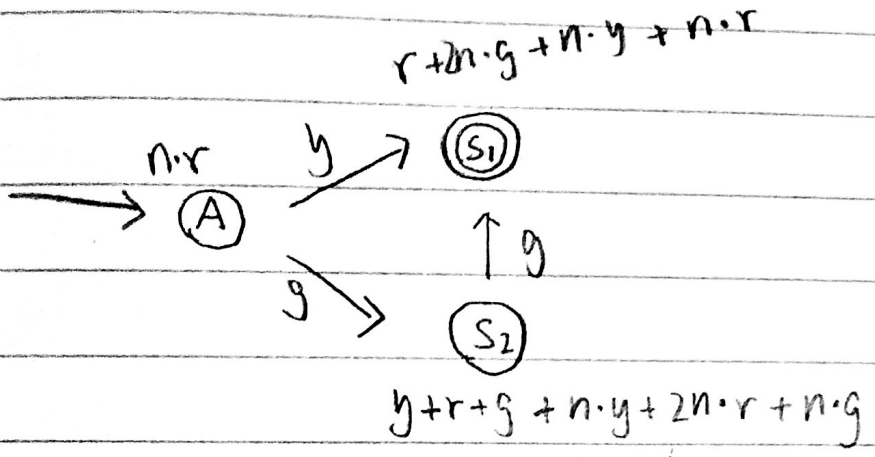
where n is the full cycle iterations

For pair (B, C):

edge counter = $r + y + g + 2n \cdot g + n \cdot y + n \cdot r$

③ redraw the graph G by subtituting loops with super node with the calculated edge counter formula (elimnates loops)





Then the problem is reduced to path from A to S_1

So that the total edge count is $n \cdot g + n \cdot r + n \cdot y$.

e.g. when $n = 1$, $edge\_count(S_1) = r + 2 \cdot g + y + r$

$= 2 \cdot r + 2 \cdot g + y$

$2 \cdot r + 2 \cdot g + 2 \cdot y - edge\_count(S_1) = y$

Therefore, the objective is to find a path from A to S_1 with $y = 1, g = 0, r = 0$. By using basic BFS, we found the path.

(2) Firstly, check if input G is acyclic graph by using find loop (from part 1). If G is acyclic return false, because there are limited paths in G.

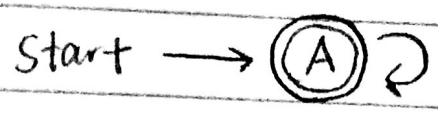
Secondly, if G has loop(s), check each loop to see

if there is a loop that has equal number of edges within it
if yes, return true, because the edge count for this loop is $n \cdot c \cdot g + n \cdot c \cdot y + n \cdot c \cdot r$ where c is a constant n is the iteration time. As long as the start vertex is also end vertex inside the loop, there are infinity number of walks that have equal number of g, y, r edges.

else (there is no such loop that has edge count = $n \cdot c \cdot g + n \cdot c \cdot y + n$
using the method from part 1 to super compose each loop int
a super node, and then check if there is a combination of
super nodes that can give out the edge count = $n \cdot c \cdot g +$
$n \cdot c \cdot y + n \cdot c \cdot r$.

2. Define: If (G_1, v_1, u_1) has more path than (G_2, v_2, u_2)
There are more loops between v_1, u_1 in G_1 than
between v_2, u_2 in G_2
Even though regardless of the number of loops in G_1, G_2
they both have ∞ walks, but the patterns are limited
The number of patterns are decided by loop count
For example:

G_1:



G_2: Start → A ⇄ B

G_1 has only one loop $\ell_0: A \to A$

so, the only possible pattern is $n \cdot \ell_0$ (n is the iteration tim

G_2 has two loops $\ell_0: A \to A$, $\ell_1: A \to B \to A$

so the possible patterns are

① $n \cdot \ell_1$

② $n \cdot \ell_1 + m \cdot \ell_0$

Therefore, from the perspective of patterns, G_2 has more walks than G_1.