ECE 606, Fall 2019, Assignment 6

Zhijie Wang, Student ID number: 20856733

zhijie.wang@uwaterloo.ca

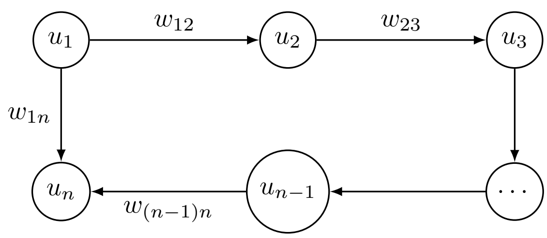
October 22, 2019

1. Let m constraints input as $c_1, c_2 \dots c_m \in C$, where each constraint is stored as $[a, b, 1]$ representing $x_a = x_b$ or $[c, d, -1]$ representing $x_c \neq x_d$.

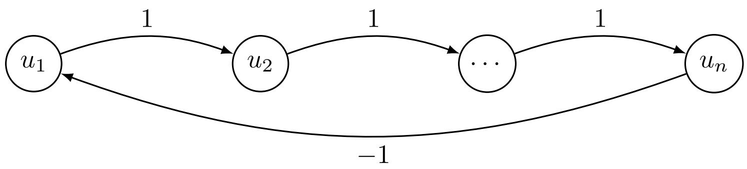
   CheckConstraints(C)

```
for each i from 1 to n do
    for each j from 1 to n do
        if i = j then
            r[i][j] ← 1
        else
            r[i][j] ← 0
for each i from 1 to m do
    a ← C[i][0], b ← C[i][1], s ← C[i][2]
    if r[a][b] = 0 then
        for each j from 1 to n do
            if r[a][j] = 1 and r[b][j] = 0 then
                r[b][j] ← s, r[j][b] ← s
            if r[b][j] = 1 and r[a][j] = 0 then
                r[a][j] ← s, r[j][a] ← s
    else if r[a][b] ≠ s then
        return False
return True
```

2. (a) $\mathcal{G} = \{G =< V, E >\}$, where each G is a directed graph that
   $V = \{u_1, u_2, \dots, u_n, n \in \mathbb{N}\}$,
   $E = \{(u_1, u_2, w_{12}), (u_2, u_3, w_{23}), \dots, (u_{n-1}, u_n, w_{(n-1)n}), (u_1, u_n, w_{1n})\}$.
   If $w_{12} > w_{1n} > 0$, $(w_{12} + w_{23} + \dots w_{(n-1)n}) < w_{1n}$, then $\mathcal{G}$ is satisfied with requirements. Firstly, there is no negative-weight cycle. Secondly if Dijkstra's algorithm starting from u_1, then the minimum weight path from u_1 to u_n will be marked as $\{u_1, u_n, w_{1n}\}$, but obviously $\{(u_1, u_2, w_{12}), (u_2, u_3, w_{23}), \dots, (u_{n-1}, u_n, w_{(n-1)n})\}$ is shorter. Therefore, Dijkstra's algorithm is not correct.



   (b) $\mathcal{G} = \{G =< V, E >\}$, where each G is a directed graph that
   $V = \{u_1, u_2, \dots, u_n, n \in \mathbb{N}\}$,
   $E = \{(u_1, u_2, 1), (u_2, u_3, 1), \dots, (u_{n-1}, u_n, 1), (u_n, u_1, -1)\}$.
   Firstly there is only one cycle in the graph which has a positive-weight. Secondly, if Dijkstra's algorithm starting from u_n, then obviously it is correct.



3. *Proof.* Here we consider about an output, $G = \{g_1, g_2, \ldots, g_k\}$ that this greedy choice result in, vs. an optimal set of meetings, $T = \{t_1, t_2, \ldots, t_m\}$. Suppose we represent the start time of a meeting as a function, $s(\cdot)$, and finish time as a function, $f(\cdot)$. We assume, without any loss of generality, that the meetings in the two sets are ordered by latest start time.

   Firstly, we need to prove that for every $i = 1, \ldots k, s(g_i) \geq s(t_i)$. By induction on i. For the base case, consider $i = 1$. The greedy choice is to pick a meeting with the latest start time. This guarantees that $s(g_1) \geq s(t_1)$. For the step, assume that the assertion is true for $i = 1, \ldots, p-1$. For $i = p$, we know that $s(g_{p-1}) \geq s(t_{p-1}) \geq f(t_p)$. Thus, the meeting t_p dose not conflict with g_{p-1}, and therefore, is available to be chosen after g_{p-1} is chosen. Thus, $s(g_p) \geq s(t_p)$ because we choose the meeting with the latest start time.

   Hence, we need to prove $k = m$. Assume otherwise, for the purpose of contraction, and that $m > k$. Then, there exists a meeting t_{k+1} in T. But by the claim above, $s(g_k) \geq s(t_k)$, thus, the meeting t_{k+1} dose not conflict with g_k, and is available to be chosen after g_k is chosen, which contradicts the claim that no more meetings are left that can be chosen after g_k is chosen.

4. a6p4.py