CSC373 Winter 2015 Problem Set # 5

Name: Weidong An
Student Number: 1000385095
UTOR email: weidong.an@mail.utoronto.ca
February 10, 2015

(a) **Algorithm**

```
FIND-MIN-CUT(N, s, t)
1   call Edmonds-Karp algorithm on (N, s, t)
2   S = {s}
3   T = N.V − S
4   S-queue = empty FIFO queue
5   S-queue.ENQUEUE(s)
6   while S-queue is not empty # This while loop is to make sure that every vertices in S is checked.
7       u = S-queue.DEQUEUE()
8       for each (u, v) ∈ N.E # Check all the out-edges of u
9           if v ∈ T and f(u, v) < c(u, v)
10              S-queue.ENQUEUE(v)
11              S = S ∪ {v}
12              T = T − {v}
13      for each (v, u) ∈ N.E # Check all the in-edges of u
14          if v ∈ T and f(v, u) > 0
15              S-queue.ENQUEUE(v)
16              S = S ∪ {v}
17              T = T − {v}
18  return (S, T)
```

Worst Case Running Time

Let $n = |V|$ and $m = |E|$.
Edmonds-Karp algorithm runs in $O(nm^2)$.
In the worst case, all the vertices except t are enqueued to S-*queue*. When checking each vertices, the algorithm check all the edges in the worst case. Therefore the algorithm except line 1 runs in $O(nm)$.
Totally, the algorithm runs in $O(nm^2)$ in the worst case.

(b) Instead of starting with $S = \{s\}$ and $T = N.V - S$, the following algorithm starts with $T = \{t\}$ and $S = N.V - T$. Then for all (u, v) where $u \in S$ and $v \in T$, if $f(u, v) < c(u, v)$, move u from S to T. For all (u, v) where $u \in T$ and $v \in S$, if $f(u, v) > 0$, move u from S to T. The pseudocode is as follows:

```
FIND-MIN-CUT-WITH-SMALL-T(N, s, t)
call Edmonds-Karp algorithm on (N, s, t)
T = {t}
S = N.E − T
T-queue = empty FIFO queue
T-queue.ENQUEUE(t)
while T-queue is not empty # This while loop is to make sure that every vertices in T is checked.
    u = T-queue.DEQUEUE()
    for each (v, u) ∈ N.E # Check all the in-edges of u
        if v ∈ S and f(u, v) < c(u, v)
            T-queue.ENQUEUE(v)
            T = T ∪ {v}
            S = S − {v}
    for each (u, v) ∈ N.E # Check all the out-edges of u
        if v ∈ S and f(v, u) > 0
            T-queue.ENQUEUE(v)
            T = T ∪ {v}
            S = S − {v}
return (S, T)
```

Note that this algorithm also satisfies:
- For all (u, v) where $u \in S$ and $v \in T$, $f(u, v) = c(u, v)$.
- For all (u, v) where $u \in T$ and $v \in S$, $f(u, v) = 0$.

So the algorithm returns a minimum cut.
For the algorithm in part(a), a node is added to S only when necessary (i.e otherwise the algorithm is not valid). This way yields minimum $|S|$. The same idea applies to the algorithm in part(b). A node is added to T only if it is necessary, so $|T|$ is minimum.