Algorithm Assignment 8

10185101210 陈俊潼

25-1

1. We can do this like like a matrix multiplication. For each vertex pair (u, v) and the transitive closure matrix T, we traverse the whole graph as follows (The newly added edge is `(a, b)`):

```
for i in all vertex:
    for j in all vertex:
        if delta(i, a) != INF and delta(b, j) != INF:
            T(i, j) = 1
```

Since it will traverse every vertex twice, the complexity of this algorithm is $O(V^2)$

2. An example is a graph with two strongly connected components, each has $V/2$ vertices. The transitive closure before add a new edge will contains two complete directed graphs.
   After adding one new edge between the two strongly connected components, every vertex in the graph will be reachable in the transitive closure. The value in the adjacent matrix to be updated will be $(V/2)^2$ = $V^2/4$. No matter what algorithm is used, the updating process needs to traverse the vertices and update the value in T one by one. So the total time needed will be $\Omega(|V|^2)$.
3. From question 1 we know that the cost of inserting an edge and update the corresponding transitive closure is $O(V^2)$, so for any n sequence (n < V), running this algorithm once, the total complexity is $O(nV^2)$ where $n \leq V$, the total complexity will be $O(V^3)$.

26.1-4

From the definition of a flow, we know that a flow f iff is a flow when:

$$\sum_{v\in V} f(v,u) = \sum_{v\in V} f(u,v)$$
$$|f| = \sum_{vtV} f(s,v) - \sum_{v\in V} f(v,s)$$

So for $\alpha f_1 + (1-\alpha) f_2$, we have:

$$
\begin{aligned}
&\because f_1(u,v) < c(u,v), f_2(u,v) < c(u,v) \\
&\therefore \alpha f_1(u,v) + (1-\alpha) f_2(u,v) \leq \alpha c(u,v) + (1-\alpha) c(u,v) = c(u,v) \text{ (Satisfies the capacity limit)} \\
&\because \sum_{v \in V} f_1(v,u) = \sum_{v \in V} f_1(u,v), \\
&\sum_{v \in V} f_2(v,u) = \sum_{v \in V} f_2(u,v) \\
&\therefore \sum_{v \in V} \alpha f_1(v,u) + (1-\alpha) f_2(v,u) = \alpha \sum_{v \in V} f_1(v,u) + (1-\alpha) \sum_{v \in V} f_2(v,u) \\
&= \alpha \sum_{v \in V} f_1(u,v) + (1-\alpha) \sum_{v \in V} f_2(u,v) \\
&= \sum_{v \in V} \alpha f_1(u,v) + (1-\alpha) f_2(u,v) \\
&\therefore \alpha f_1(u,v) + (1-\alpha) f_2(u,v) \text{ is a flow.}
\end{aligned}
$$

26.2-4

A minimum cut is $\{s, v_1, v_2\}$ and $\{v_3, v_4, t\}$.

Of the augmenting paths appearing in the example, the edge (v_2, v_3) in picture (c) canceled the flow.

34.2-2

Since G is a bipartite graph with odd number of vertices, suppose the graph is partitioned into two vertex set S_1 and S_2 where there exists an edge connecting any vertex in S_1 and S_2.

We now suppose there exists a path $v_0 \to v_1 \to v_2 \to \cdots \to v_0$ be a Hamiltonian cycle in G. Since G is a bipartite graph, if v_0 is in set S_1, then v_1 is is set S_2. v_2 is in set S_3, etc. Since n is odd, then v_n must be in the same set as v_0, and there doesn't exist an edge from v_n to v_0 as there were in the same partition. This leads a contradiction, so there doesn't exist a Hamiltonian cycle in a bipartite graph G with odd number of vertices.

34.2-6

Given a certificate (a sequence of vertices) of the language HAM-PATH = {<G, u, v>: there is a hamiltonian path from u to v in graph G}. To verify it is a solution, we only need to check if the sequence contains every vertices in G and for every consequent pair `(u, v)` there's a path connecting them. Let n be the number of vertices in the graph, the complexity of verifying this problem is $O(n^2)$, which is a polynomial time. So this language belongs to NP.

34.5-7

First we prove this problem is NP. Since given a certificate (a sequence of vertices), we can verify if it is the longest simple-path of the graph in polynomial time, this problem is NP.

Then we can reduce the longest-simple-cycle problem to a Hamiltonian Cycle problem. Convert the longest simple-path problem into “Is there a simple path of length at least n-1, where n is the number of vertices in G?”, this is equivalent to find a Hamilton path that traverses every node at least once in a graph. Since Hamiltonian Cycle problem is NP-hard, this problem is also NP-hard.

Since it’s NP and NP-hard, the longest-simple-cycle problem is NP-complete.