CS 217 – Algorithm Design and Analysis

Shanghai Jiaotong University, Fall 2019

Handed out on Thursday, 2019-10-10
First submission and questions due on Monday, 2019-10-14
You will receive feedback from the TA.
Final submission due on Monday, 2019-10-21

4 Bottleneck Paths

Let $G = (V, E)$ be a directed graph with an edge capacity function $c : E \to \mathbb{R}^+$. For a path $p = u_0 u_1 \dots u_t$ define its *capacity* to be

$$c(p) := \min_{1 \le i \le t} c(\{u_{i-1}, u_i\}) \ . \tag{1}$$

Maximum Capacity Path Problem (MCP). Given a directed graph $G = (V, E)$, an edge capacity function $c : E \to \mathbb{R}^+$, and two vertices $s, t \in V$, compute the path p^* maximizing $c(p)$. We denote by p^* the optimal path and by $c^* := c(p^*)$ its cost.

Exercise 1. Suppose the edges $e_1, \dots, e_m$ are sorted by their cost. Show how to solve MCP in time $O(n + m)$.

Proof. Design a algorithm following Pseudocode shows(Suppose the edges are sorted decreased):

And then we think about the correctness and complexity.

The algorithm means we can enum the answer. When we find a edge between the point visited and not visited, we can go through all the edges which's costs is higher than this edge. If now s is connected to t, means this edge is the largest edge while going through all edges higher than it from s to t. That fits the answer we want.

Now, let's think about the complexity. For every node, it may be in queue at least once. And for every edge, it may be in G' and used in bfs at least once. So the time complexity is $O(n+m)$. $\square$

Exercise 2. Give an algorithm for MCP of running time $O(m \log\log m)$. **Hint:** Using the median-of-medians algorithm, you can determine an edge e such that at most $m/2$ edges are cheaper than e and at most $m/2$ edges are more expensive than e. Can you determine, in time $O(n+m)$, whether $c^* < c(e)$, $c^* = c(e)$, or $c^* > c(e)$? Iterate to shrink the set of possible values for c^* to $m/4$, $m/8$, and so on.

Exercise 3. Give an algorithm for MCP that runs in time $O(m \log\log\log m)$? How about $O(m \log\log\log\log m)$? How far can you get?

Exercise 4. Suppose we modify the Ford-Fulkerson method so that, in every round, it finds a path of *maximum capacity*, as opposed to be shortest-$s-t$-path method employed by Edmonds-Karp. Show that this algorithm terminates after a number of rounds that is polynomial in n and m.

Algorithm 1 Solve MCP in time $\Theta(n+m)$ with all the edges sorted by their cost

```
procedure MCP($G, s, t$)
    $visited[s]$ = True
    $G'$ = NULL
    for $e \in G$ do
        if $visited[e.from]\&\&!visited[e.to]$ then
            $bfs\_graph(e.to, G')$
        else
            $G'.add\_edge(e)$
        end if
        if $visited[t]$ then
            return $e.weight$
        end if
    end for
end procedure
procedure BFS_GRAPH($s, G$)
    $visited[s]$ = True
    $queue.push(s)$
    while !queue.empty() do
        $top = queue.top()$
        $queue.pop()$
        for $e \in G[top]$ do
            if $!visited[e.to]$ then
                $visited[e.to]$ = True
                $q.push(e.to)$
            end if
        end for
    end while
end procedure
```