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 $O(n + m)$.

Solution Because the edges has been sorted by their cost, so we can consider add edges to the edge set E' from largest to smallest. We need to maintain a vertex set S represent the reachable vertex from s in (V, E') The S will change if and only if the new edge (u, v) join E' and $u \in S, v \notin S$ So we can simply BFS from v, and add vertex along the way to S.

In all BFS, we will just access each vertex v at most the in-degree of v times, This is because if we access v from u through a certain edge e, we

Algorithm 1 Find the MCP with sorted edges list $E = \{e_1, e_2, \cdots, e_m\}$

```
1: S ← {s}
2: E' ← ∅
3: for e_i = (u, v) ∈ E do
4:    E' ← E' ∪ {e_i}
5:    if u ∈ S and v ∉ S then
6:       BFS from v in (V/S, {(u, v)|(u, v) ∈ E', u, v ∉ E'}), add all passing
         vertex to S, , add all passing edge to E
7:       if t ∈ S then
8:          return c(e_i)
9:       end if
10:   end if
11: end for
```

must have $u \notin S$ before BFS. Obviously, u will not be added in S twice. And we also have $e \notin E'$,so we will just access each edge only once because of the same reason, so the complexity is $O(n + m)$

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.

Algorithm 2 A O(mlog $m \log m$) algorithm for MCP Problem

Require: $G = (V, E)$: the graph; $m = |E|$: the number of edge; $e \in E$: an edge of the graph; $c_e \in \mathbb{N}$: the capacity of edge e; s: the source point; t: the target point;

Ensure: The answer of MCP Problem

1: Initialize IterCount $\leftarrow$ 0,E' $\leftarrow$ E,L $\leftarrow min_{e\in E} c_e$,U $= max_{e\in E} c_e$
2: **while** $IterCount < \log s(m)$ **do**
3: Determin the median M of $\{c_e : e \in E', c_e \leq U\}$
4: $T := \{e \in E' : c_e \leq M\}$,$F := \{e \in E' : c_e > M\}$
5: **if** (V, F) is s-t-connected **then**
6: $E' \leftarrow F, L \leftarrow M$
7: **else**
8: $U \leftarrow M$
9: **end if**
10: IterCount $\leftarrow$ IterCount + 1
11: **end while**
12: Number the t edges in $\{e \in E' : c_e \leq U\}$ according to increasing weights:$e_1, \ldots, e_t$ (We need to use something like quick sort)
13: Solve the instance by the method in Question 1 with the following ordering:

$$order(e) = \begin{cases} 1 & if\ c_e \leq L \\ i+1 & if\ e \in E', e = e_i \\ m & if\ c_e > U \end{cases}$$

Solution

Apparently, we can solve the MCP problem with the algorithm used in Exercise 1 if we've got the edges sorted. But by using the algorithm above,

we've estimated an upper bound of MCP problem. So that we don't have to distinguish the edges which have larger capacity than the upper bound. What we finally need to do is it to determine the bottleneck value by sorting edges in E' and run the algorithm used in Exercise 1.

Complexity Analysis We use BFS algorithm to check whether the graph is connected with complexity $O(m)$. In this part, the total complexity is $O(m \log s(m))$.

The other part is sorting edges in E', which is the set containing bottleneck edge. The complexity is $O(\frac{m}{s(m)} \log \frac{m}{s(m)})$

Taking $s(m)$ as $\log(m)$, we can see that the complexity of the algorithm is $O(m \log \log(m))$

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?

Solution In iteration i, we denote |E'| as m_i, line 3 runs in $O(m_i p_i)$ time, and $m_i \leq \frac{m}{\Pi_{j=1}^{i} p_j}$.And line 22 runs in $O(m)$ time, so the total running time becomes $O(\sum_{i=1}^{k} m_{i-1} p_i + km + m_i log m_i)$

Let $p_1 = p_2 = 2, p_i = p_{i-1}^2$, then $p_i = 2^{2^{i-1}}$ and $m_i * p_i = m$. Let $k = logloglogm + 1$,then $m_k log m_k \leq \frac{m}{logm} * logm = m$.

In order to run MCP faster, we can reduce the running time of line 3. Line 3 can be running in time $O(m_i log p_i)$, so we can let k = logloglog...logm with arbitary number of log. Then we can make MCP running in time of $O(mlogloglo...logm)$.

Algorithm 3 MCP

Require: *input*: $V, E, c, s, t, k, p_1, p_2, \cdots, p_k$

$E' \leftarrow E$
for i from 1 to $\lfloor k \rfloor$ **do**
  find $\lceil \frac{1}{p_i} m \rceil, \cdots, \lceil \frac{p_i - 1}{p_i} m \rceil$-th largest member in E' as $c_1, c_2, c_3, \cdots, c_{p_i - 1}$
  for $e \in E'$ **do**
    if $c(e) < c_1$ **then**
      add e to E_1
    else if $c(e) == c_j$ and $1 \leq j \leq p_i - 1$ **then**
      add e to E_{2j}
    else if $c_j < c(e) < c_{j+1}$ and $1 \leq j < p_i - 1$ **then**
      add e to E_{2j+1}
    else
      add e to $E_{2p_i - 1}$
    end if
    if $c(e) < min\{c(e) | e \in E'\}$ **then**
      $k_e = 1$
    else if $e \in E_i$ **then**
      $k_e = i + 1$
    else
      $k_e = 2p_i + 1$
    end if
  end for
  use exercise 1 to solve MCP with k(order) and let $q = k_{MC} - 1$
  if q is even **then**
    $c_{q/2}$ is maximum capacity, exit
  else
    $E' \leftarrow E_q$
  end if
end for
sort elements in E' as $e_1, e_2, \cdots, e_l$
if $c(e) < min\{c(e) | e \in E'\}$ **then**
  $k_e = 1$
else if $e = e_i \in E'$ **then**
  $k_e = i + 1$
else if $c(e) > max\{c(e) | e \in E'\}$ **then**
  $k_e = l + 1$
end if
use exercise 1 to solve MCP with k(order)