**1**

Give an algorithm for MCP of running time $O(m \log \log m)$.

*Solution.*

**The pseudocode of the algorithm is in the next page.**

The correctness of the algorithm comes from below. First, we initialize our $c^*$ candidates to be all capacities of the edges in $G$. Then, in each iteration, we divide it into many divisions of approximately the same size in order, and test which division $c^*$ is in. Finally, there will be only one candidate left, and the only candidate is $c^*$.

Now we'll analyze the running time. Note that using median-of-medians algorithm to find the median of a set $S$ has running time $O(|S|)$. In "Loop 2" of function MCP, we find the median of $S_1, \ldots, S_k$ in $2^d$ iterations. Note that there is an invariant equation $\sum_{i=1}^{k} |S_k| = |C|$. So finding the median requires $O(|C| \cdot 2^d)$ in total. The GetDivision function may need to add all the edges into the graph in the worst case. Testing the connectivity will use $O(m)$ time in total if we record a set of reachable vertices as decribed in Exercise 1. So the running time of GetDivision is $O(m)$.

We now prove that $|C| \cdot 2^d = O(m)$ by induction. Initially, $|C| = m$ and $d = 0$. It obviously holds. Suppose $|C_{old}| \cdot 2^{d_{old}} = O(m)$. We have $|C_{new}| = O(|C_{old}| \cdot (\frac{1}{2})^{2^{d_{old}}})$, $d_{new} = d_{old} + 2^{d_{old}}$. So $|C_{new}| \cdot 2^{d_{new}} = O(m)$.

Now we know that a single iteration of "Loop 1" takes time $O(m)$. Note that $2^d \leqslant m$. Since $d$ increases exponentially, "Loop 1" will iterate $O(\log^* m)$ times. Finally, we get the running time is $O(m \log^* m)$. Obviously this algorithm has a better running time than $O(m \log \log m), O(m \log \log \log m), O(m \log \log \log \log m)$ when $m$ is sufficiently large. □

**2**

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

*Solution.* See Problem 2. □

**Algorithm 1** An algorithm for MCP problem.

**function** MCP($V, E = \{e_1, e_2, \ldots, e_m\}, c$)
  $C \leftarrow \{e_1, \ldots, e_m\}$
  $oldEdges \leftarrow \emptyset$
  $d \leftarrow 0$
  **while** there are multiple capacities of the edges in $C$ **do** ▷ Loop 1
    $divisions \leftarrow [C]$ ▷ Let *divisions* be a list only containing the set $C$.
    **for** $i = 1$ to $2^d$ **do** ▷ Loop 2
      Let $S_1, S_2, \ldots, S_k$ denote the edge sets in *divisions* in order, i.e., representing *divisions* as $[S_1, \ldots, S_k]$.
      Use median-of-medians algorithm $k$ times to find the median capacity of the capacities of the edges in each set(if there are two medians, choose the larger one). Denote them as $m_1, m_2, \ldots, m_k$ respectively.
      $newDivisions \leftarrow [\,]$ ▷ Let *newDivisions* be an empty list.
      **for** $j = 1$ to $k$ **do** ▷ Loop 3
        Append $\{s \in S_j : c(s) \geqslant m_j\}$ to *newDivisions*.
        Append $\{s \in S_j : c(s) < m_j\}$ to *newDivisions*.
      **end for**
      $divisions \leftarrow newDivisions$
    **end for**
    $C, oldEdges \leftarrow$ GetDivision(*divisions*, *oldEdges*)
    $d \leftarrow 2^d + d$
  **end while**
  **return** the capacity of the edges in $C$
**end function**

**function** GetDivision(*divisions*, *oldEdges*) ▷ Using the algorithm of exercise 1.
  $G \leftarrow (V, oldEdges)$
  **for** *div* in *divisions* in the order of the list **do**
    Add all edges in *div* to $G$.
    **if** $t$ is now reachable from $s$ **then**
      **return** *div*, *oldEdges*
    **else**
      $oldEdges \leftarrow oldEdges \cup div$
    **end if**
  **end for**
**end function**