CMPSC 465 Spring 2022 | Data Structures & Algorithms Chunhao Wang and Mingfu Shao | Solution 04

**0. (0 pts.) Acknowledgements.** The assignment will receive a 0 if this question is not answered.

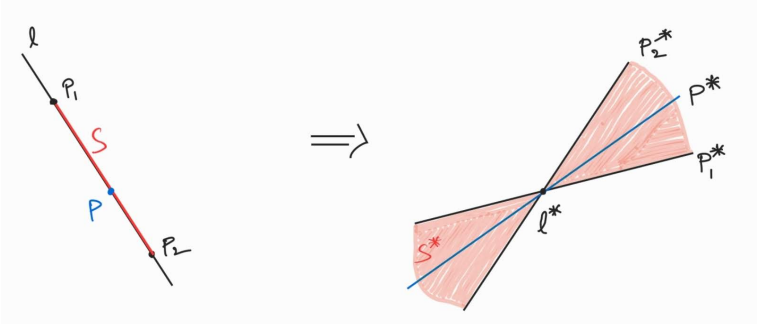
1. If you worked in a group, list the members of the group. Otherwise, write "I did not work in a group."
2. If you received significant ideas about your solutions from anyone not in your group, list their names here. Otherwise, write "I did not consult anyone except my group members".
3. List any resources besides the course material that you consulted in order to solve the material. If you did not consult anything, write "I did not consult any non-class materials."

**1. (15 pts.)** In class, we learned the following property about duality: point $p$ is on line $l$ if and only if point $l^*$ is on line $p^*$.

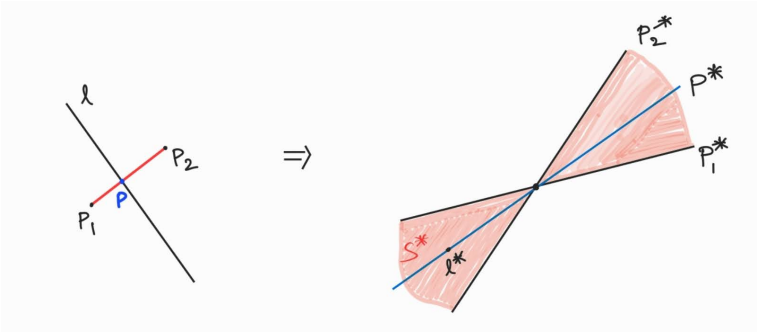
1. Using the property above, prove the following properties:
   (a) If $n$ points $p_1, ..., p_n$ are on a common line $l$, then $p_1^*, p_2^*, ..., p_n^*$ intersect at a common point $l^*$.
   (b) If $n$ lines $l_1, ..., l_n$ intersect at a common point $p$, then $l_1^*, l_2^*, ..., l_n^*$ are on a common line $p^*$.
2. If we have a line segment $s$ connecting two points $p_1$ and $p_2$, describe a region $s^*$ corresponding to the dual of $s$ in terms of $p_1^*$ and $p_2^*$. (No rigorous proof is needed.)
3. If a line $l$ intersects the line segment $s$, prove that $l^*$ is in the region $s^*$.

**Solution:**

1. (a) For each $p_i$, since $p_i$ is on the line $l$, $l^*$ must be on the line $p_i^*$ in the dual plane. This must hold for every $p_i$, so $l^*$ must be on the lines $p_1^*, p_2^*, ...$ ,$p_n^*$. Therefore, $p_1^*, p_2^*, ..., p_n^*$ intersect at the common point $l^*$.
   (b) For each $l_i$, since $p$ is on the line $l_i$, $l_i^*$ must be on the line $p^*$ in the dual plane. This must hold for every $l_i$, so $l_1^*, l_2^*, ..., l_n^*$ must be on the line $p^*$. Therefore, $l_1^*, l_2^*, ..., l_n^*$ are on the common line $p^*$.
2. The line segment $s$ coincides with a line connecting two points $p_1$ and $p_2$. Let us say this line $l$. Obviously, $p_1^*$ and $p_2^*$ intersects at the point $l^*$ in the dual plain. To figure out the region $s^*$, we investigate where the points on the line segment $s$ will be transformed to the dual plane. For every point $p$ on $s$, $p$ is also on the line $l$. Thus, $p^*$ passes through the point $l^*$. Also, $p$ is in between $p_1$ and $p_2$, so the slope of $p^*$ is in between the ones of $p_1^*$ and $p_2^*$. It means that $p^*$ should be placed in the region between $p_1^*$ and $p_2^*$. This must hold for every $p$ on $s$, hence, $s$ is transformed to the area between $p_1^*$ and $p_2^*$. Without loss of generality, assume $p_1$ is located at the left to the $p_2$, that is, the $x$ coordinate of $p_1$ is less than the one of $p_2$. Then, $s^*$ is the region sweeping from $p_1^*$ to $p_2^*$ in counter-clockwise order.



3. Since the line $l$ intersects the line segment $s$, there should be a point $p$ on $l$ and $s$. We observe that $p^*$ is in the region $s^*$ from above. Also, we know that $l^*$ is on the line $p^*$, which means $l^*$ is in the region $s^*$.




**2. (15 pts.)** We are given a graph $G = (V, E)$; $G$ could be a directed graph or undirected graph. Let $M$ be the adjacency matrix of $G$. Let $n$ be the number of vertices so that the matrix $M$ is $n \times n$ matrix. For any matrix $A$, let us denote the element of $i$-th row and $j$-th column of the matrix $A$ by $A[i, j]$.

1. Consider the square of the adjacency matrix $M$. For all $i$ and $j$, show that $M^2[i, j]$ is the number of different paths of length 2 from the $i$-th vertex to the $j$-th vertex. It should be explained or proved as clearly as possible.
   **Solution:** For any $i$ and $j$, let us find the number of different paths of length 2 from $i$ to $j$. Assume we go through $k$ during our trip. Then, there should be edges between $i$ and $k$ and between $k$ and $j$. In other words, $M[i, k] \cdot M[k, j] = 1$. This path $i \to k \to j$ is one of the paths from $i$ and $j$. We need to consider all $k$ for counting the total number of paths, so it should be

   $$\sum_{k=1}^{n} M[i, k] \cdot M[k, j]$$

   It turns out that it is equal to $M^2[i, j]$.
2. For any positive integer $k$, show that $M^k[i, j]$ is the number of different paths of length $k$ from the $i$-th vertex to the $j$-th vertex. You may use induction on $k$ to prove it.
   **Solution:** We will use induction on $k$.
   (a) (Base case) $k = 1$
   It is clear that $M^k[i, j] = M[i, j]$ has the number of different paths from $i$ to $j$.
   (b) (Inductive step)
   Assume the result is true for $k = l$. It means that $M^l[i, j]$ is the number of different paths of length $l$ from $i$ to $j$. Note that $M^{l+1}[i, j] = \sum_{s=1}^{n} M^l[i, s] \cdot M[s, j]$ from the matrix multiplication $M^{l+1} = M^l \cdot M$. Every path of length $l + 1$ from $i$ to $j$ consists of paths from $i$ to some vertex $s$ of length $l$ and an edge $s$ to $j$. By the induction hypothesis, $M^l[i, s]$ has the number of different paths of length $l$ from $i$ to $s$. Thus, $M^l[i, s] \cdot M[s, j]$ has the number of different paths of length $l + 1$ that is from $i$ to $j$ and located at $s$ at the $l$-th step. Since $s$ is arbitrary, we need to add up over $s$ so $\sum_{s=1}^{n} M^l[i, s] \cdot M[s, j]$ is the total number of different paths of length $l + 1$ from $i$ to $j$.
   Hence, from $(a)$ and $(b)$, we conclude that $M^k[i, j]$ is the number of different paths of length $k$ from $i$ to $j$ by induction principle.
3. Assume that we are given a positive integer $k$. Design an algorithm to find the number of different paths of length $k$ from the $i$-th vertex to $j$-th vertex for all pairs of $(i, j)$. The time complexity of your algorithm should be $O(n^3 \log k)$. You can get partial credits if you design an algorithm of $O(n^3 k)$.

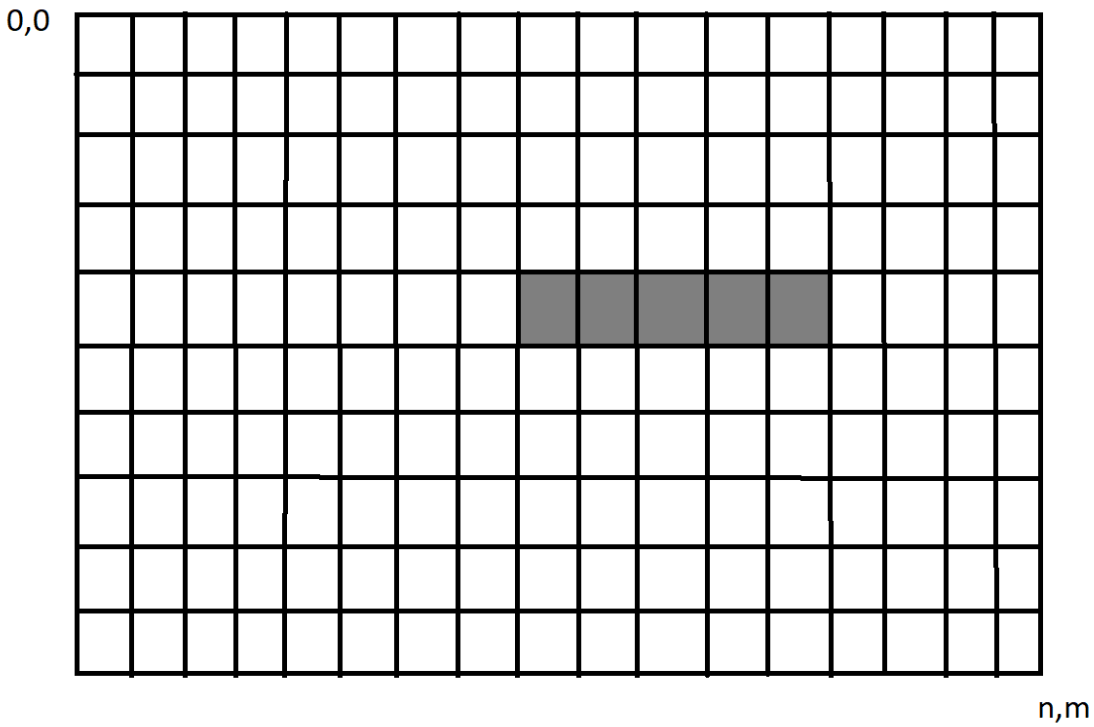
**Solution:** Once we have $M^k$ for given $k$, we can get the number of different paths of length $k$ from $i$ to $j$ for any pairs $(i, j)$. It means that we need to design an algorithm to get $M^k$. As the matrix multiplication takes $O(n^3)$, we can come up with a naive algorithm of doing $k-1$ matrix multiplications. In this case, the running time is $O(n^3 k)$. If we reduce the number of matrix multiplications, we will get a better algorithm.

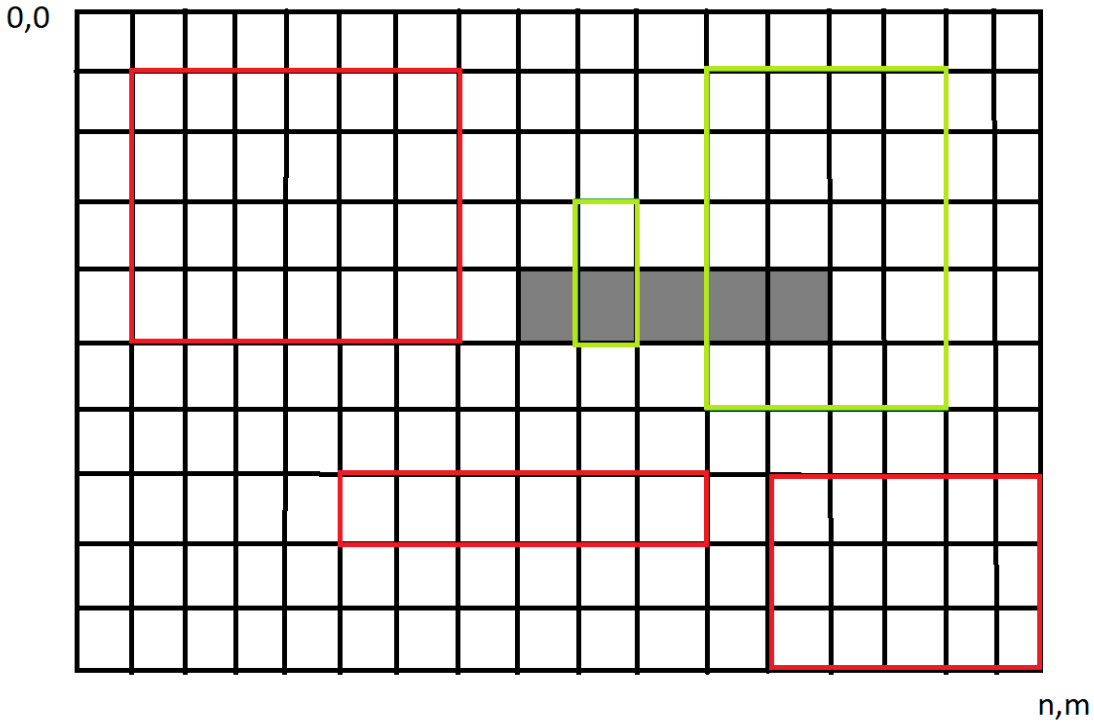
Let $l$ be the highest power of 2 less than or equal to $k$ and $p$ be the exponent of $l$. In other words, $p = \lfloor \log_2 k \rfloor$ and $l = 2^p$. To do this, let us precompute $M, M^2, M^4, M^8, ..., M^l$. Since $M^{2^i} \cdot M^{2^i} = M^{2^{i+1}}$, we can get $M^2 = M \cdot M$, $M^4 = M^2 \cdot M^2$, $M^8 = M^4 \cdot M^4$, and so on. This precomputation needs $p$ times of matrix multiplication, so it takes $O(n^3 p)$. Next we represent $k$ as a binary number. It has $p+1$ digits as $2^p \leq k < 2^{p+1}$.

First, we start with the identity matrix $I$. If the first digit is 1, multiply it by $M$. Otherwise, do nothing. After that, if the second digit is 1, multiply it by $M^2$, otherwise, do nothing. If the third digit is 1, multiply it by $M^4$. Repeat this process for $p+1$ digits. With this algorithm, we obtain $M^k$ with at most $p+1$ times of matrix multiplications. It also takes $O(n^3 p)$. Therefore, our algorithm takes $O(n^3 p) = O(n^3 \log k)$ to get $M^k$ for given $k$.

Example: Let $k = 26$. Then, $l$ is the highest power of 2 less than or equal to $k$, so it is 16, and $p = 4$. Precompute $M, M^2, M^4, M^8$ and $M^{16}$. If we represent $k$ as a binary number, it is $11010_{(2)}$. This implies that $26 = 2^4 + 2^3 + 2^1$. Therefore, $M^{26} = M^{16} \cdot M^8 \cdot M^2$.

**3. (10 pts.)** You are given a grid of size $n \times m$ and within that grid there is a horizontal rod (arbitrary width but height of 1). You aim is to locate it. The only thing you can use to find the location of the rod is make queries of the form IsPresent $(x_1, y_1, x_2, y_2)$ where $x_1 \leq x_2, y_1 \leq y_2$. This returns True if part of the rod is present in the grid from $(x_1, y_1)$ to $(x_2, y_2)$ and False if not. See figures below. Design an algorithm which uses queries $O(\log(n+m))$ times to find the location of the rod, i.e., the leftmost coordinates and rightmost coordinates. *Hint: consider using binary search in your algorithm.*






**Solution:**

1. We try to find at which row the rod is located. When making queries, we fix $x_1$ as 0 and $x_2$ as $n$. Then, we find the upper coordinate and the lower coordinate of the rod by varying $y_1$ and $y_2$ in the same manner as binary search. This step makes $O(\log m)$ queries.
2. We fix $y_1$ and $y_2$ in any queries later since we found the upper and lower coordinates of the rod. Then, we make queries to find the left coordinate of the rod by varying $x_1$ and $x_2$ in a binary search fashion as well. More specifically, we set $x_1$ as 0 and $x_2$ as $n$ in the beginning, along with $prev\_x_2$ being $n$. Then, we start continuously making the query IsPresent $(x_1, y_1, x_2, y_2)$. If the current query returns True, we update $prev\_x_2$ to be $x_2$, $x_2$ to be $x_1 + \lfloor \frac{x_2 - x_1}{2} \rfloor$, and make another query. On the other hand, if the current query returns False, we update $x_1$ to be $x_2$, $x_2$ to be $x_1 + \lfloor \frac{prev\_x_2 - x_1}{2} \rfloor$, and make another query. When the query returns True and $x_2 - x_1 \leq 1$, we find the leftmost block in the rod, and the leftmost coordinate is $x_1$. This step makes $O(\log n)$ queries.
3. We find the rightmost coordinate of the rod in a very similar way as step 2, though we have $prev\_x_1$ and set it to 0 in the beginning. Now, if the current query returns True, we update $prev\_x_1$ to be $x_1$, $x_1$ to be $x_1 + \lfloor \frac{x_2 - x_1}{2} \rfloor$, and make another query. If the current query returns False, we update $x_2$ to be $x_1$, $x_1$ to be $prev\_x_1 + \lfloor \frac{x_1 - prev\_x_1}{2} \rfloor$, and make another query. In the end where the query returns True and $x_2 - x_1 \leq 1$, we find the rightmost block instead, and the rightmost coordinate is $x_2$. Same as step 2, this step makes $O(\log n)$ queries.

Overall, this algorithm makes $O(\log m) + O(\log n) + O(\log n) = O(\log m + \log n) = O(\log mn)$ queries.
Lemma: $O(\log mn) = O(\log(m + n))$
Proof:

- $m + n \leq mn + mn = 2mn$ ($m$ and $n$ can't be positive floats that smaller than 1, based on the semantic of the problem). So, $\log(m + n) = O(\log mn)$.

- $mn \leq (m+n)^2 = m^2 + 2mn + n^2$. So, $\log mn = O(\log(m+n))$.
- Consequently, $\log mn = \Theta(\log(m+n))$ and $O(\log mn) = O(\log(m+n))$.

From the lemma, our algorithm indeed makes $O(\log mn) = O(\log(m+n))$ queries.

# Rubrics:

**Problem 1, 15 pts**

1. (a) 3 points : proved/explained 1.a correctly.
   (b) 3 points : proved/explained 1.b correctly.
2. 5 points : proved/ explained region correctly.
3. 3 points : partially correct explanation of region.
4. 4 points : proved that $l^*$ is in the region $s^*$
5. 2 points : partially correct explanation of $l^*$ is in the region $s^*$.

**Problem 2, 15 pts**

1. (a) 3 points : Provided a correct explanation or proof.
2. (a) 6 points : Provided a correct proof.
   (b) 3 points : Provided an incomplete proof or explanation
3. (a) 6 points : Provided an algorithm in $O(n^3 \log k)$
   (b) 4 points : Provided an algorithm in $O(n^{\log_2 7}k)$ ($n^{\log_2 7}$ may come from the optimized matrix multiplication in the textbook)
   (c) 3 points : Provided an algorithm in $O(n^3 k)$
4. 1.5 point : I don't know how to answer this question.

**Problem 3, 10pts**

- 3 points: Described a correct way to find at which row the rod is i.e. the uppermost and lowermost coordinates.
- 4 points: Described a correct way to find the leftmost and rightmost coordinates.
- 3 points: Provided a reasonable running time analysis for the algorithm, and it runs in $O(\log(m+n))$ or $O(\log(mn))$.
- 1 point: I don't know how to answer this question.