Nathan Bickel
CSCE 350: Section H01
Professor: Dr. Narayanan
February 24, 2023

CSCE 350 Homework 3

Problem 1 The determinant of an n-by-n matrix

$$A = \begin{bmatrix} a_{11} & \dots & a_{1n} \\ \vdots & \vdots & \vdots \\ a_{n1} & \dots & a_{nn} \end{bmatrix}$$

denoted $\det(A)$, can be defined as a_{11} for $n = 1$ and for $n > 1$, by the recursive formula

$$\det(A) = \sum_{j=1}^{n} s_j a_1 \det(A_j),$$

where $s_j = \begin{cases} 1 & j \text{ is odd} \\ -1 & j \text{ is even} \end{cases}$, a_{1j} is the element in row 1 and column j, and A_j is the $n-1$-by-$n-1$ matrix obtained from a matrix A by deleting its row 1 and column j.

(a) Set up a recursive relation for the number of multiplications made by the algorithm implementing this recursive definition. You should give the initial conditions. (Use the definition's formula to get the recurrence relation for the number of multiplications made by the algorithm.)

(b) Without solving the recurrence, what can you say about the solution's order of growth as compared to $n!$? (Investigate the right-hand side of the recurrence relation. Compare the recurrence relation of this problem with the recurrence relation for computing $n!$. Computing the first few values of $M(n)$ may be helpful, too.

Solution.

(a) We can define a recurrence relation $x(n)$ as the number of multiplications made by the recursive algorithm. Let $n \in \mathbb{N}$. The sum goes from 1 to n, and multiplies s_j by a_{1j} by $\det(A_j)$. We have that $\det(A_j)$ takes $x(n-1)$ multiplications, and multiplying by s_j and then by a_{1j} takes 2 more. Since this happens n times, we have $x(n) = n(x(n-1)+2)$. Since $\det(A)$ requires no multiplications when $A \in \text{Mat}_{1\times 1}$, we have $x(1) = 0$.

(b) From (a), we have $x(n) = n(x(n-1)+2) = nx(n-1) + 2n$. The recurrence relation for $n!$ is $x'(n) = nx'(n-1)$, where $x'(1) = 1$. Thus, the homogenous solution for both recurrence relations will be in $\Theta(n!)$, while the particular solution for $x(n)$ will be a polynomial. Then, for

$$\lim_{n\to\infty} \frac{x(n)}{x'(n)},$$

we can use L'Hôpital's rule to determine that the particular solution of $x(n)$ will become insignificant, and we will eventually be left with $\frac{n!}{n!} = 1$. So $x(n) \in \Theta(x'(n)) = \Theta(n!)$.

Problem 2 Sort the list $17, 7, 22, 33, 19, 2, 85$ in an ascending order by Insertion Sort. You should show the process of sorting step by step – You must show the intermediate results after each insertion. How many comparisons are used for this problem?

Solution.

In each step, a vertical bar separates the sorted part of the array from the remaining elements; the element being inserted is in bold. We will also indicate the number of comparisons: each element we insert is compared to each element less than it, and then when we find an element larger we shift that element and the rest of the sorted part over. So we need to compare the inserted element to every number smaller and it and the first one larger.

$$\begin{array}{rl} & \mid\ \mathbf{17}\ 7\ 22\ 33\ 19\ 2\ 85 \\ \longrightarrow & 17\ \mid\ \mathbf{7}\ 22\ 33\ 19\ 2\ 85 \quad \text{(0 comparisons)} \\ \longrightarrow & 7\ 17\ \mid\ \mathbf{22}\ 33\ 19\ 2\ 85 \quad \text{(1 comparison)} \\ \longrightarrow & 7\ 17\ 22\ \mid\ \mathbf{33}\ 19\ 2\ 85 \quad \text{(2 comparisons)} \\ \longrightarrow & 7\ 17\ 22\ 33\ \mid\ \mathbf{19}\ 2\ 85 \quad \text{(3 comparisons)} \\ \longrightarrow & 7\ 17\ 19\ 22\ 33\ \mid\ \mathbf{2}\ 85 \quad \text{(3 comparisons)} \\ \longrightarrow & 2\ 7\ 17\ 19\ 22\ 33\ \mid\ \mathbf{85} \quad \text{(1 comparison)} \\ \longrightarrow & 2\ 7\ 17\ 19\ 22\ 33\ 85\ \mid \quad \text{(6 comparisons)} \end{array}$$

So in total, we make $0 + 1 + 2 + 3 + 3 + 1 + 6 = 16$ comparisons.

Reference–I used this video about insertion sort to understand the concept.

Problem 3 There are several alternative ways to define a distance between two points $p_1(x_1, y_1)$ and $p_2(x_2, y_2)$ in the Cartesian plane. In particular, the Manhattan distance is defined as

$$d_M(p_1, p_2) = |x_1 - x_2| + |y_1 - y_2|.$$

(a) Prove that d_M satisfies the following axioms, which every distance function must satisfy:

- $d_M(p_1, p_2) \geq 0$ for any two points p_1 and p_2, and $d_M(p_1, p_2) = 0$ if and only if $p_1 = p_2$.
- $d_M(p_1, p_2) = d_M(p_2, p_1)$
- $d_M(p_1, p_2) \leq d_M(p_1, p_3) + d_M(p_3, p_2)$ for any p_1,p_2, and p_3.

(b) Sketch all the points in the Cartesian plane whose Manhattan distance to the origin $(0, 0)$ is equal to 1. Do the same for the Euclidean distance.
(c) True or false: A solution to the closest-pair problem does not depend on which of the two metrics—d_E (Euclidean) or d_M (Manhattan)—is used?

Solution.

(a) We will prove that for all $p_1 = (x_1, y_1), p_2 = (x_2, y_2), p_3 = (x_3, y_3) \in \mathbb{R}^2$:

1. $d_M(p_1, p_2) \geq 0$
2. $d_M(p_1, p_2) = 0 \iff p_1 = p_2$
3. $d_M(p_1, p_2) = d_M(p_2, p_1)$

4. $d_M(p_1, p_2) \leq d_M(p_1, p_3) + d_M(p_3, p_2)$.

Proof of (1): We have $d_M(p_1, p_2) = |x_1 - x_2| + |y_1 - y_2|$. Since $|x_1 - x_2|$ and $|y_1 - y_2|$ are both non-negative, their sum is also non-negative. So $d_M(p_1, p_2) \geq 0$.

Proof of (2): First, assume $p_1 = p_2$. Then, $x_1 = x_2$ and $y_1 = y_2$, implying $x_1 - x_2 = 0$ and $y_1 - y_2 = 0$. Thus, we have $d_M(p_1, p_2) = |x_1 - x_2| + |y_1 - y_2| = |0| + |0| = 0$. So $d_M(p_1, p_2) = 0 \Longleftarrow p_1 = p_2$. Next, assume $d_M(p_1, p_2) = 0$, which implies that $|x_1 - x_2| + |y_1 - y_2| = 0$. From this, we have $|x_1 - x_2| = |y_1 - y_2| = 0$: to see this, suppose not. Then $|x_1 - x_2| = a$ and $|y_1 - y_2| = -a$ for some $a \in \mathbb{R} - \{0\}$, but then we have that an absolute value equals a negative number, a contradiction. So we have $|x_1 - x_2| = |y_1 - y_2| = 0$, so $x_1 = x_2$ and $y_1 = y_2$, and thus $p_1 = p_2$. So we have proved both directions, and therefore $d_M(p_1, p_2) = 0 \Longleftrightarrow p_1 = p_2$.

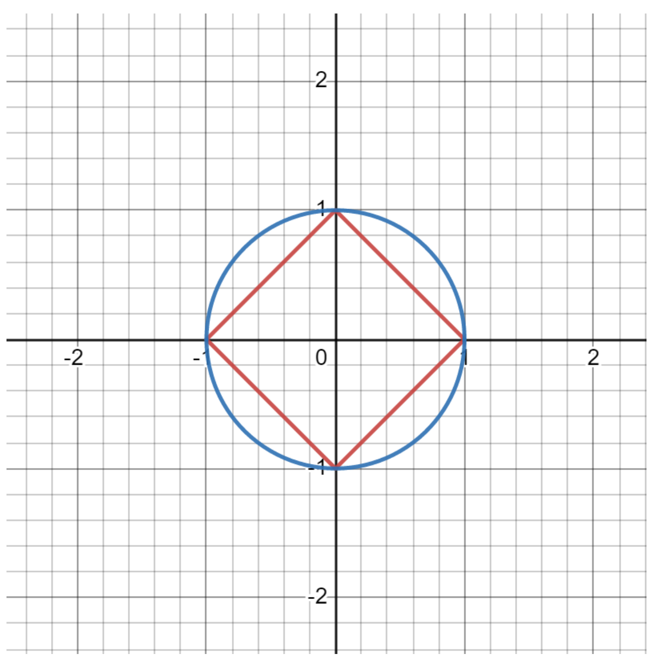
Proof of (3): We have $d_M(p_1, p_2) = |x_1 - x_2| + |y_1 - y_2| = |x_2 - x_1| + |y_2 - y_1| = d_M(p_2, p_1)$ because of the absolute values.

Proof of (4): We will use the triangle inequality, which states that for all $a, b \in \mathbb{R}$, $|a + b| \leq |a| + |b|$. We have

$$\begin{aligned}
d_M(p_1, p_2) &= |x_1 - x_2| + |y_1 - y_2| && \text{(definition)} \\
&= |x_1 - x_2 + (x_3 - x_3)| + |y_1 - y_2 + (y_3 - y_3)| && (x_3 - x_3 = y_3 - y_3 = 0) \\
&= |(x_1 - x_3) + (x_3 - x_2)| + |(y_1 - y_3) + (y_3 - y_2)| && \text{(associativity)} \\
&\leq (|x_1 - x_3| + |x_3 - x_2|) + (|y_1 - y_3| + |y_3 - y_2|) && \text{(triangle inequality)} \\
&= |x_1 - x_3| + |y_1 - y_3| + |x_3 - x_2| + |y_3 - y_2| && \text{(associativity)} \\
&= d_M(p_1, p_3) + d_M(p_3, p_2). && \text{(definition)}
\end{aligned}$$

So $d_M(p_1, p_2) \leq d_M(p_1, p_3) + d_M(p_3, p_2)$. □

(b) The red points are Manhattan distance 1 from the origin, while the blue points are Euclidean distance 1:



(c) This is false. For example, consider $S = \left\{ p_1 = (0, 0), p_2 = \left(\frac{5}{2}, 0\right), p_3 = \left(-\sqrt{2}, \sqrt{2}\right) \right\}$. Then, we note:

- $d_E(p_1, p_2) = \sqrt{\left(0 - \frac{5}{2}\right)^2 + (0 - 0)^2} = 2.50$ $\qquad d_M(p_1, p_2) = |0 - \frac{5}{2}| + |0 - 0| = 2.50$
- $d_E(p_1, p_3) = \sqrt{\left(0 + \sqrt{2}\right)^2 + \left(0 - \sqrt{2}\right)^2} = 2.00$ $\qquad d_M(p_1, p_3) = |0 + \sqrt{2}| + |0 - \sqrt{2}| \approx 2.83$
- $d_E(p_2, p_3) = \sqrt{\left(\frac{5}{2} + \sqrt{2}\right)^2 + \left(0 - \sqrt{2}\right)^2} \approx 4.16$ $\qquad d_M(p_2, p_3) = |\frac{5}{2} + \sqrt{2}| + |0 - \sqrt{2}| \approx 5.33$

So if we choose to use Euclidean distance, the closest-pair solution for S is p_1 and p_3, but if we choose to use Manhattan distance, the closest-pair solution for S is p_1 and p_2. Therefore, using different metrics can sometimes yield different solutions.

Problem 4 How many comparisons (both successful and unsuccessful) will be made by the brute-force algorithm in searching for each of the following patterns in the binary text of one thousand zeros?
(a) 00001
(b) 10000
(c) 01010

Solution.

Since each bit string is of length 5, there are 995 possible locations for the string to be in the bit string of 1000 zeroes. The brute-force algorithm starts at the beginning of each possible location, and starts matching the pattern to the string until either there is a complete match or a mismatch.

(a) There will be 5 comparisons per candidate location, since the 4 zeroes will match and then the 1 will be unsuccessfully compared to the fifth zero in the candidate. So there will be $(995)(4) = 3,980$ successful comparisons, 995 unsuccessful comparisons, and $4,975$ total comparisons.

(b) There will be 1 comparison per candidate location, since the 1 at the beginning of the pattern will be unsuccessfully compared to the first zero in the candidate. So there will be 0 successful comparisons and 995 unsuccessful/total comparisons.

(c) There will be 2 comparisons per candidate location, since the first zero will match the first zero in the candidate and then the 1 will be unsuccessfully compared to the second zero in the candidate. So there will be 995 successful comparisons, 995 unsuccessful comparisons, and $1,990$ total comparisons.

Problem 5 A network topology specifies how computers, printers, and other devices are connected over a network. The figure below illustrates three common topologies of networks: the ring, the star, and the fully connected mesh. You are given a boolean matrix $A[0 \dots n-1, 0, \dots, n-1]$ where $n > 3$, which is supposed to be the adjacency of a graph modeling a network with one of these topologies. Your task is to determine which of these three topologies, if any, the matrix represents. Design a brute-force algorithm for this task and indicate its time efficiency class.

Solution.

We will assume that the adjacency matrices are for simple graphs, and thus they are symmetric with zero entries on the main diagonal. We will represent the i, j-entry of A as $A_{i,j}$.

Since each our algorithm loops through the matrix a constant number of times, the time complexity is in $\Theta(n^2)$.

```
// Input: Matrix A ∈ Mat_{n×n}
// Output: True if A is a ring and false otherwise
function DetermineRing(A)
    // Find beginning of cycle
    current ← 1
    numVerticesTraversed ← 1
    next ← −1
    end ← −1
    for j ← 0 to n − 1 do
        if A_{1,j} = true then
            if next = −1 then
                next ← j
            else if end = −1 then
                end ← j
            else
                return false // Vertex 1 has more than 2 connections
            end if
        end if
    end for
    while next ≠ end do
        for j ← 0 to n − 1 do
            if A_{next,j} = true then
                foundNext ← false
                if j ≠ current then
                    if foundNext = false then
                        next ← j // Follow the cycle to the next value
                        foundNext ← true
                        numVerticesTraversed+ = 1 // We've added another vertex to our cycle
                    else
                        return false // We found a vertex with three connections
                    end if
                end if
            end if
        end for
    end while
    // We made it back to the start vertex with no problems
    // Return if we've covered all vertices
    return numVerticesTraversed = n
end function
```

```
// Input: Matrix A ∈ Mat_{n×n}
// Output: True if A is a star and false otherwise
function DeterminеStar(A)
    // We need to find the central vertex
    center ← −1
    for j ← 1 to n − 1 do
        if A_{1,j} = true then
            // We found a connection, need to check if vertex 1 or j is the center
            if j = n − 1 or A_{1,j+1)} = false then
                center ← j // Vertex 1 isn't connected to everything, so it can't be the center
            else
                center ← 1 // Vertex 1 has more than one connection, so it must be the center
            end if
            break
        end if
    end for
    if center = −1 then
        return false // We didn't find an appropriate center
    end if
    for i ← 0 to n − 2 do
        for j ← i + 1 to n − 1 do
            if (i = center or j = center) and A_{i,j} = false then
                return false // We have found a vertex the center isn't connected to
            else if i ≠ center and j ≠ center and A_{i,j} = true then
                return false // We have a found a connection not adjacent to the center
            end if
        end for
    end for
    return true // The matrix satisfies our conditions
end function
```

```
// Input: Matrix A ∈ Mat_{n×n}
// Output: True if A is fully connected and false otherwise
function DetermineMesh(A)
    for i ← 0 to n − 2 do
        for j ← i + 1 to n − 1 do
            if A_{i,j} = false then
                return false // We've found two vertices that aren't adjacent
            end if
        end for
    end for
    return true // We've checked every pair of vertices and they're all adjacent
end function
```

Algorithm 1 TopologyDeterminer($A[0 \dots n-1, 0, \dots, n-1]$)

```
// Input: Matrix A ∈ Mat_{n×n}
// Output: "Star" if A is a star, "Ring" if A is a ring,
// "Mesh" if A is a fully connected mesh, "None" if A is none of these
function TopologyDeterminer(A)
    if DetermineRing(A) then
        return "Ring"
    else if DetermineStar(A) then
        return "Star"
    else if DetermineMesh(A) then
        return "Mesh"
    else
        return "None"
    end if
end function
```