Name: _____

1. (20 points) A network topology specifies how computers, printers, and other devices are connected over a network. Figure 1 below illustrates three common topologies of networks: the ring, the star, and the fully connected mesh.

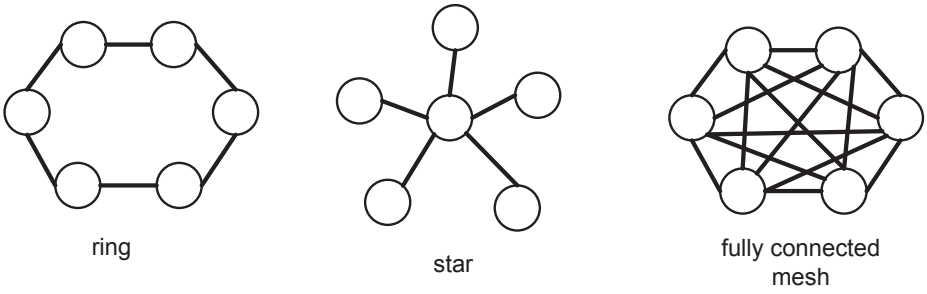

Figure 1: Topologies

You are given a boolean matrix $A[0..n-1, 0..n-1]$, where $n > 3$, which is supposed to be the adjacency matrix of a graph modeling a network with one of these topologies, if any, the matrix represents. Design a brute-force algorithm (express it in pseudocode) for this task and indicate its time efficiency class.

[Solution] Each vertex of a ring is connected to exactly two other vertices, so if we sum up the entires in a row we get 2. For a star, each of the $n-1$ vertices is connected to one other vertex and the central vertex is connected to $n-1$ other vertices. For a fully connected mesh, each vertex is connected to exactly $n-1$ other vertices. The pseudocode is given in $WhichTopology$.

Time efficiency is linear.

2. (25 points) Consider the following algorithm:

   (a) Apply *CountingSort* algorithm to sort the list $\langle 2, 5, 3, 0, 2, 3, 0, 3\rangle$. Specifically, show the states of the array C just after line no. 9 and just after line no. 12. And also show the state of the array B after each iteration of the **for** loop on line no. 13.

   [Solution]

   Initially, $A = \langle 2^a, 5, 3^a, 0^a, 2^b, 3^b, 0^b, 3^c\rangle$. The superscripts are to help decide on stability.

   Just after line no. 9: $C = \langle 2, 0, 2, 3, 0, 1\rangle$. C now contains say how many 2's are there in A at $C[2]$, etc. Just after line no. 12: $C = \langle 2, 2, 4, 7, 7, 8\rangle$. Now C contains say how many elements of A are less or equal to 2 at $C[2]$, etc.

   The states of B after iterations are as follows:

$$
\begin{aligned}
[i = 8]: &\quad \langle \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, 3^c, \emptyset\rangle \\
[i = 7]: &\quad \langle \emptyset, 0^b, \emptyset, \emptyset, \emptyset, \emptyset, 3^c, \emptyset\rangle \\
[i = 6]: &\quad \langle \emptyset, 0^b, \emptyset, \emptyset, \emptyset, 3^b, 3^c, \emptyset\rangle \\
[i = 5]: &\quad \langle \emptyset, 0^b, \emptyset, 2^b, \emptyset, 3^b, 3^c, \emptyset\rangle \\
[i = 4]: &\quad \langle 0^a, 0^b, \emptyset, 2^b, \emptyset, 3^b, 3^c, \emptyset\rangle \\
[i = 3]: &\quad \langle 0^a, 0^b, \emptyset, 2^b, 3^a, 3^b, 3^c, \emptyset\rangle \\
[i = 2]: &\quad \langle 0^a, 0^b, \emptyset, 2^b, 3^a, 3^b, 3^c, 5\rangle \\
[i = 1]: &\quad \langle 0^a, 0^b, 2^a, 2^b, 3^a, 3^b, 3^c, 5\rangle
\end{aligned}
$$

Algorithm 1 WhichTopology($A[0..n-1, 0..n-1]$)

```
1: {Determines whether the graph is a ring, a star, or a fully connected mesh}
2: {Input: Boolean adjacency matrix A[0..n − 1, 0..n − 1] of the graph}
3: {Output: 1 denoting a ring, 2 denoting a star, or 3 denoting a fully connected mesh}
4: d_0 ← 0
5: for c ← 1 to n − 1 do
6:    d_0 ← d_0 + A[0, c]
7: end for
8: if d_0 = 2 then
9:    return 1
10: end if
11: if d_0 = 1 then
12:    return 2
13: end if
14: d_1 ← 0
15: for c ← 0 to n − 1 do
16:    d_1 ← d_1 + A[1, c]
17: end for
18: if d_0 = d_1 then
19:    return 3
20: else
21:    return 2
22: end if
```

(b) Do we need to consider the worst-case, the average-case, and the best-case while analyzing the algorithm? Explain your answer. What is the time efficiency class of this algorithm?

[Solution] The algorithm never compares two elements and its operation is independent of the specific elements in A. So, we do not need to consider various cases. The time efficiency class is linear.

(c) Is it a stable sorting algorithm? Explain your answer. Give one situation where we may prefer a stable sorting algorithm over an unstable one.

[Solution] It is a stable algorithm which is obvious from the final state of B. It is stable because it starts putting the elements of A in sorted order from right to left. One situation could be, students are already sorted alphabetically, now we want to sort them by their grade but want to preserve alphabetic order for the students having the same grade.

(d) Is it an in-place sorting algorithm? Explain your answer.

[Solution] It is not in-place, because it uses extra arrays whose size depends on the input size, n.

3. (15 points) Give an example of a text of length n and a pattern of length m (> 1) that constitutes a worst-case input for the brute-force string matching algorithm (given below). Exactly how many character comparisons will be made for such input?

[Solution] Text could be "aab" whereas the pattern could be "ab". The worst case number of comparisons happens, $m(n - m + 1) = 2(3 - 2 + 1) = 4$.

4. (10 points) How can pseudorandom numbers be useful in the empirical analysis of an algorithm?

[Solution] We need to generate random inputs to perform the average-case analysis of an algorithm empirically. We can use pseudorandom numbers to generate random sample sizes and also the individual random input-elements.

Algorithm 2 CountingSort($A[1..n], k$)

```
1: {Sort the elements in A[1..n] in nondecreasing order}
2: {Input: An array A[1..n] of nonnegative integers and the value (k) of the the highest element in A[1..n]}
3: {Output: An array B[1..n] containing the elements of A[1..n] sorted in nondecreasing order}
4: for i ← 0 to k do
5:    C[i] ← 0
6: end for
7: for i ← 1 to n do
8:    C[A[i]] ← C[A[i]] + 1
9: end for
10: for i ← 1 to k do
11:    C[i] ← C[i] + C[i − 1]
12: end for
13: for i ← n downto 1 do
14:    B[C[A[i]]] ← A[i]
15:    C[A[i]] ← C[A[i]] − 1
16: end for
17: return B
```

Algorithm 3 BruteForceStringMatching(T[0..$n-1$], P[0..$m-1$])

```
1: {Implements brute-force string matching}
2: {Input: An array T[0..n−1] of n characters representing a text and an array P[0..m−1] of m characters
   representing a pattern}
3: {Output: The index of the first character in the text that starts a matching substring or −1 if the search
   is unsuccessful}
4: for i ← 0 to n − m do
5:    j ← 0
6:    while j < m and P[j] = T[i + j] do
7:       j ← j + 1
8:    end while
9:    if j = m then
10:      return i
11:   end if
12: end for
13: return −1
```

5. (10 points) Given $f(n) = n2^n$ and $g(n) = 3^n$, which one do you think is true: $f(n) \in O(g(n))$, $f(n) \in \Omega(g(n))$, or $f(n) \in \Theta(g(n))$. Prove your answer.

   [Solution]

$$\begin{aligned}
\lim_{n\to\infty} \frac{n2^n}{3^n} &= \lim_{n\to\infty} \frac{n}{\left(\frac{3}{2}\right)^n} \\
&= \lim_{n\to\infty} \frac{1}{\left(\ln \frac{3}{2}\right)\left(\frac{3}{2}\right)^n} \quad \text{[By L'Hospital's rule]} \\
&= \frac{1}{\ln \frac{3}{2}} \lim_{n\to\infty} \left(\frac{2}{3}\right)^n \\
&= 0.
\end{aligned}$$

   So, $f(n) \in O(g(n))$.

6. (20 points) Solve the following recurrence relation (assume that n is a power of 2):

$$T(n) = \begin{cases} 1 & \text{when } n = 1, \\ 2T\left(\frac{n}{2}\right) + 6n - 1 & \text{when } n \geq 2 \end{cases}$$

   Given $T(n) \in O(g(n))$, $g(n) = ?$

   [Solution] Lets assume $n = 2^m$ or $m = \lg n$.

$$\begin{aligned}
T(n) &= 2T\left(\frac{n}{2}\right) + 6n - 1 \\
&= 2\left[2T\left(\frac{n}{2^2}\right) + 6\frac{n}{2} - 1\right] + 6n - 1 \\
&= 2^2 T\left(\frac{n}{2^2}\right) + 6n - 2 + 6n - 1 \\
&= 2^2\left[2T\left(\frac{n}{2^3}\right) + 6\frac{n}{2^2} - 1\right] + 2 \cdot 6n - (1 + 2) \\
&= 2^3 T\left(\frac{n}{2^3}\right) + 3 \cdot 6n - \left(2^0 + 2^1 + 2^2\right) \\
&= 2^m T\left(\frac{n}{2^m}\right) + m \cdot 6n - \left(2^0 + 2^1 + 2^2 + \ldots + 2^{m-1}\right) \\
&= 2^m T(1) + 6mn - (2^m - 1) \\
&= 2^m + 6mn - 2^m + 1 \\
&= 6n \lg n + 1.
\end{aligned}$$

   So, $T(n) \in O(n \lg n)$.

1 Useful formulas

1. $\sum_{i=l}^{u} 1 = u - l + 1$
2. $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$
3. $\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$
4. $\sum_{i=0}^{n} a^i = \frac{a^{n+1}-1}{a-1} (a \neq 1)$
5. $\lg n = \frac{\ln n}{\ln 2}$
6. $\frac{\mathrm{d}}{\mathrm{d}n} n^m = mn^{m-1}$
7. $\frac{\mathrm{d}}{\mathrm{d}n} \ln n = \frac{1}{n}$
8. $\frac{\mathrm{d}}{\mathrm{d}n} a^n = (\ln a) a^n$