1. In a connected undirected graph, the distance between two vertices is the number of edges in the shortest path between them. Suppose we denote by $P$ the following property: there exists a vertex that is a neighbour of all other vertices. Consider the following statements:

(i) If $P$ is false, then there is a pair of vertices such that the distance between them is at least 4.

(ii) If $P$ is true, then the distance between any pair of vertices is at most 2.

What can you say about these statements?

(a) Only (i) is true. 
(b) Only (ii) is true. 
(c) Both (i) and (ii) are true. 
(d) Neither (i) nor (ii) is true.

Answer:

(b) Only (ii) is true.

If $P$ is true, there is a “central” vertex that is a neighbor of all others. We can start from any vertex and reach any other by going via the center. So (ii) is true. There are graphs where $P$ is false but the distance is at most 3 between any pair of vertices, so (i) is false.

2. The symbol $|$ reads as “divides”, and $\nmid$ as “does not divide”. For instance, $2 \mid 6$ and $2 \nmid 5$ are both true. Consider the following statements:

(i) There exists a positive integer $a$ such that ($2 \mid (a^3 - 1)$) and ($2 \mid a$).

(ii) There exists a positive integer $b$ such that $6 \nmid (b^3 - b)$.

What can you say about these statements?

(a) Only (i) is true. 
(b) Only (ii) is true. 
(c) Both (i) and (ii) are true. 
(d) Neither (i) nor (ii) is true.
(d) Neither (i) nor (ii) is true.

(i) is false. Note that \((a^3 - 1) = (a - 1)(a^2 + a + 1)\), and hence \(2 \mid a\) implies that both \(a - 1\) and \(a^2 + a + 1\) are odd, and hence so is their product. Thus \(2 \nmid (a^3 - 1)\).

(ii) is false too. Note that \((b^3 - b) = (b - 1)b(b + 1)\), and at least one of these factors is even and one is divisible by 3. Hence \(6 \mid (b^3 - b)\).

3. For a regular expression \(e\), let \(L(e)\) be the language generated by \(e\). If \(e\) is an expression that has no Kleene star \(*\) occurring in it, which of the following is true about \(e\) in general?

(a) \(L(e)\) is empty.
(b) \(L(e)\) is finite.
(c) Complement of \(L(e)\) is empty.
(d) Both \(L(e)\) and its complement are infinite.

Answer:

(b) \(L(e)\) is finite.

This is proved by a simple induction on the structure of the regular expression, using the fact that \(L(a)\) is finite for each letter \(a\), and that unions and concatenations of finite languages are also finite.

4. Consider a weighted undirected graph \(G\) with positive edge weights. Let \((u, v)\) be an edge in the graph. It is known that the shortest path from a vertex \(s\) to \(u\) has weight 53 and the shortest path from \(s\) to \(v\) has weight 65. Which of the statements is always true?

(a) Weight of \((u, v)\) \(\leq\) 12.
(b) Weight of \((u, v)\) = 12.
(c) Weight of \((u, v)\) \(\geq\) 12.
(d) Nothing can be said about the weight of \((u, v)\).

Answer:

(c) Weight of \((u, v)\) \(\geq\) 12.

If the weight of \((u, v)\) is strictly less than 12, then there is a path from \(s\) to \(v\) of weight at most \(53 + 11 = 64\) (which goes from \(s\) to \(u\) and then takes the edge \((u, v)\)). This contradicts the fact that the shortest path from \(s\) to \(v\) has weight 65. Thus the weight of \((u, v)\) is at least 12.
5. A dodecahedron is a regular solid with 12 faces, each face being a regular pentagon. How many edges are there? And how many vertices?

(a) 60 edges and 20 vertices.  (b) 30 edges and 20 vertices.
(c) 60 edges and 50 vertices.  (d) 30 edges and 50 vertices.

Answer:

(b) 30 edges and 20 vertices.

Each face has five edges, and there are 12 faces. But each edge is shared between two faces. Thus the number of edges is \( (12 \times 5)/2 = 30 \). The number of vertices is given by Euler’s formula to be \(|E| - |F| + 2 = 30 - 12 + 2 = 20\). \(\Box\)

6. In the code fragment to the right, `start` and `end` are integer values and `prime(x)` is a function that returns `true` if `x` is a prime number and `false` otherwise.

At the end of the loop:

(a) \( k = i-j \).
(b) \( k = j-i \).
(c) \( k = -j-i \).
(d) Depends on `start` and `end`.

Answer:

(c) \( k = -j-i \).

The value of \( i + j + k \) is 0 initially, and also at the end of each iteration of the loop. Thus \( k = -j - i \) at the end of the loop. \(\Box\)

7. Varsha lives alone and dislikes cooking, so she goes out for dinner every evening. She has two favourite restaurants, Dosa Paradise and Kababs Unlimited, to which she travels by local train. The train to Dosa Paradise runs every 10 minutes, at 0, 10, 20, 30, 40 and 50 minutes past the hour. The train to Kababs Unlimited runs every 20 minutes, at 8, 28 and 48 minutes past the hour. She reaches the station at a random time between 7:15 pm and 8:15 pm and chooses between the two restaurants based on the next available train. What is the probability that she ends up eating in Kababs Unlimited?

(a) \( \frac{1}{5} \)  (b) \( \frac{1}{3} \)  (c) \( \frac{2}{5} \)  (d) \( \frac{1}{2} \)
8. An advertisement for a tennis magazine states, “If I’m not playing tennis, I’m watching tennis. And if I’m not watching tennis, I’m reading about tennis.” We can assume that the speaker can do at most one of these activities at a time. What is the speaker doing?

(a) Playing tennis.  
(b) Watching tennis.  
(c) Reading about tennis.  
(d) None of the above.

Answer:

(b) Watching tennis.

If he is not watching tennis, then he is reading about tennis. Which means he is not playing tennis, which implies that he is watching tennis. Thus, the assumption that he is not watching tennis leads to a contradiction. Therefore he is watching tennis.

9. ScamTel has won a state government contract to connect 17 cities by high-speed fibre optic links. Each link will connect a pair of cities so that the entire network is connected—there is a path from each city to every other city. The contract requires the network to remain connected if any single link fails. What is the minimum number of links that ScamTel needs to set up?

(a) 34  
(b) 32  
(c) 17  
(d) 16

Answer:

(c) 17

If we connect the cities in a loop with 17 links, then even if any single link fails, we can connect the cities by traversing the loop the other way. If we only had 16 links connecting 16 cities, that would constitute a tree which would have only a single path between any pair of cities (and thus get disconnected by a single link failure).

10. Which of the following relationships holds in general between the scope of a variable and the lifetime of a variable (in a language like C or Java)?

(a) The scope of a variable is contained in the lifetime of the variable.  
(b) The scope of a variable is the same as the lifetime of the variable.  
(c) The lifetime of a variable is disjoint from the scope of the variable.  
(d) None of the above.
Part B

1. A group of war prisoners are trying to escape from a prison. They have thoroughly planned the escape from the prison itself, and after that they hope to find shelter in a nearby village. However, the village (marked as B, see picture below) and the prison (marked as A) are separated by a canyon which is also guarded by soldiers (marked as S). These soldiers sit in their pickets and rarely walk; the range of view of each soldier is limited to exactly 100 meters. Thus, depending on the locations of soldiers, it may be possible to pass the canyon safely, keeping the distance to the closest soldier strictly larger than 100 meters at any moment. The situation is depicted in the following picture, where the circles around S indicate range of view.

Provide an algorithm to determine if the prisoners can pass the canyon unnoticed, given the width and the length of the canyon and the coordinates of every soldier in the canyon, and assuming that soldiers do not change their locations. (Hint: Model this as a graph, with soldiers represented by the vertices.)

Answer:

Consider a graph $G = (V \cup \{N, S\}, E)$ where $V = \{v_s \mid s \text{ is a soldier}\}$. $N$ and $S$ are two special vertices representing the north and south boundary of the canyon, respectively. For two soldiers $s$ and $s'$, $(v_s, v_{s'}) \in E$ iff $s$ and $s'$ are at most 200 meters apart. For a soldier $s$, $(v_s, N) \in E$ iff $s$ is at most 100 meters from the north boundary, and similarly $(v_s, S) \in E$ iff $s$ is at most 100 meters from the south boundary. It is clear that the prisoners cannot pass the canyon unnoticed iff there is a path between $N$ and $S$. Thus it suffices to test reachability of $S$ from $N$ using any standard algorithm (for instance, one based on breadth-first search).

2. A simple path (respectively cycle) in a graph is a path (respectively cycle) in which no edge or vertex is repeated. The length of such a path (respectively cycle) is the number of edges in the path (respectively cycle).

Let $G$ be an undirected graph with minimum degree $k \geq 2$.

(a) Show that $G$ contains a simple path of length at least $k$.
(b) Show that $G$ contains a simple cycle of length at least $k + 1$. 
Answer:
Build a path in the graph as follows:

Pick an arbitrary vertex \( v_1 \) in the first step. At each subsequent step, let \( v_t \) be the vertex picked last. If all neighbours of \( v_t \) have already been picked, stop. Otherwise \( v_{t+1} \) is chosen to be an arbitrary neighbour of \( v_t \) hitherto unpicked.

Since the graph is finite, the above procedure terminates eventually. Let \( \pi = v_1 \cdots v_t \) be the path produced at the end. By construction, it is a simple path. It follows that all neighbours of \( v_t \) have already been picked and they occur among \( \{v_1, \ldots, v_{t-1}\} \). Since the minimum degree of the graph is \( k \), it follows that \( k \leq t - 1 \). Thus there are at least \( k + 1 \) vertices and hence at least \( k \) edges in \( \pi \).

Let \( i \) be the least index such that \( v_i \) is adjacent to \( v_t \). It is clear that the path from \( v_i \) to \( v_t \) is of length at least \( k \). Adding the edge \( (v_i, v_t) \) yields a simple cycle of length at least \( k + 1 \).

3. An undirected graph can be converted into a directed graph by choosing a direction for every edge. Here is an example:

\[
\begin{array}{c}
\begin{array}{c}
\text{before} \quad \text{after}
\end{array}
\end{array}
\]

Show that for every undirected graph, there is a way of choosing directions for its edges so that the resulting directed graph has no directed cycles.

Answer:
Assign a distinct natural number \( N(v) \) to each vertex \( v \) of the graph. For an edge between \( v \) and \( w \), choose the direction \( v \rightarrow w \) provided \( N(v) < N(w) \). It is easy to prove that if there is a directed path from \( v \) to \( w \) in this scheme, then \( N(v) < N(w) \). Suppose there is a directed cycle involving \( v \). This means that there is a directed path from \( v \) to itself, whence \( N(v) < N(v) \), which is a contradiction.

4. Let \( \Sigma = \{0, 1\} \). Let \( A, B \) be arbitrary subsets of \( \Sigma^* \). We define the following operations on such sets:

\[
\begin{align*}
A + B & := \{ w \in \Sigma^* \mid w \in A \text{ or } w \in B \} \\
A \cdot B & := \{ uv \in \Sigma^* \mid u \in A \text{ and } v \in B \} \\
2A & := \{ ww \in \Sigma^* \mid w \in A \}
\end{align*}
\]

Is it true that \((A + B) \cdot (A + B) = A \cdot A + B \cdot B + 2(A \cdot B)\) for all choices of \( A \) and \( B \)? If yes, give a proof. If not, provide suitable \( A \) and \( B \) for which this equation fails.
Answer:
The equation fails for $A = \{0\}$ and $B = \{1\}$. In that case $(A + B) \cdot (A + B) = \{00, 01, 10, 11\}$, whereas $A \cdot A + B \cdot B + 2(A \cdot B) = \{00, 11, 0101\}$. ⊟

5. For a string $x = a_0 a_1 \cdots a_{n-1}$ over the alphabet $\{0, 1, 2\}$, define $val(x)$ to be the value of $x$ interpreted as a ternary number, where $a_0$ is the most significant digit. More formally, $val(x)$ is given by
\[
\sum_{0 \leq i < n} 3^{n-1-i} \cdot a_i.
\]
Design a finite automaton that accepts exactly the set of strings $x \in \{0, 1, 2\}^*$ such that $val(x)$ is divisible by 4.

Answer:
Let $L = \{x \in \{0, 1, 2\}^* \mid val(x)$ is divisible by 4\}. For any string $x$ and $b \in \{0, 1, 2\}$:

(a) $x \in L$ iff $val(x)$ mod 4 = 0.
(b) if $val(x) = i$ mod 4, $val(xb) \equiv (3i + b)$ mod 4, for $b = 0, 1, 2$.

So to construct a DFA for $L$, it suffices to take 4 states $q_0, q_1, q_2, q_3$, with $q_i$ recording the fact that the value of the string read so far is $i$ mod 4. Under this interpretation, $q_0$ will be the initial and (only) final state, and the automaton will transition from state $q_i$ to $q_j$ on reading letter $b \in \{0, 1, 2\}$, where $j = (3i + b)$ mod 4.

![Diagram of DFA for strings whose value is divisible by 4.](image)

Figure 1: Automaton for strings whose value is divisible by 4.

6. An automatic spelling checker works as follows. Given a word $w$, first check if $w$ is found in the dictionary. If $w$ is not in the dictionary, compute a dictionary entry that is close to $w$. For instance if the user types `ocurrance`, the spelling checker should
suggest occurrence, which belongs to the dictionary. Similarity between words such as occurrence and occurance is quantified in terms of alignment.

An alignment between two strings $w_1$ and $w_2$ (over the alphabet \{a, b, c, \ldots, z\}) is obtained by inserting hyphens in the two strings such that the modified strings align (i.e., the modified strings are of equal length, and at each position, either both strings have the same letter or one of the strings has a hyphen).

Here are three examples of alignments. The first is between occurance and occurrence and the second and third are between ctatg and ttaagc.

(1) oc-urr-ance
(2) ct-at-g-
(3) ctat---g--
occurre-nce -tta-agc ---ttaagc

A mismatch in an alignment is a position where one of modified strings has a hyphen and the other does not. There are three mismatches in the first alignment given above, five mismatches in the second, and seven mismatches in the third.

Use dynamic programming to give an efficient algorithm that takes two strings $x$ and $y$ (over the alphabet \{a, b, c, \ldots, z\}) as its input, and computes the minimum number of mismatches among all alignments of $x$ and $y$. What is the running time of your algorithm (in terms of the lengths of $x$ and $y$)?

Answer:

Let $x = a_1a_2 \cdots a_m$ and $y = b_1b_2 \cdots b_n$. For any string $w$ and any $i \in \{0, \ldots, |w|\}$, we let $w[\ldots i]$ denote the prefix of $w$ of length $i$. For $i \in \{0, \ldots, m\}$ and $j \in \{0, \ldots, n\}$, we let $\text{OPT}(i, j)$ denote the minimum number of mismatches among all alignments of $x[\ldots i]$ and $y[\ldots j]$. It is easy to see that $\text{OPT}(0, j) = j$ and $\text{OPT}(i, 0) = i$. When $i, j > 0$, an optimal alignment of $x[\ldots i]$ and $y[\ldots j]$ either aligns $x[\ldots i-1]$ with $y[\ldots j-1]$ and tries to match $a_i$ and $b_j$ (if possible), or aligns $y[\ldots j]$ with $x[\ldots i-1]$, leaving $a_i$ unmatched, or aligns $x[\ldots i]$ with $y[\ldots j-1]$, leaving $b_j$ unmatched.

This is reflected by the following recurrence:

$$\text{OPT}(i, j) = \min\{\text{OPT}(i-1, j-1) + c_{ij}, \text{OPT}(i-1, j) + 1, \text{OPT}(i, j-1) + 1\}.$$  

where $c_{ij}$ denotes the cost of aligning $a_i$ (the string containing just the one letter) with $b_j$. It is easy to see that

$$c_{ij} = \begin{cases} 0 & \text{if } a_i = b_j \\ 2 & \text{otherwise} \end{cases}$$

The following algorithm (which runs in $O(mn)$ time) computes the minimum number of mismatches among alignments of $x$ and $y$.

array A[0\ldots m, 0\ldots n]  
array c[0\ldots m, 0\ldots n]  
initialize A[i,0] = i for each i  
initialize A[0,j] = j for each j  
for $j = 1, \ldots, n$
for i = 1,...,m
    if a[i] == b[j] then c[i,j] = 0 else c[i,j] = 2
endfor
endfor
for j = 1,...,n
    for i = 1,...,m
    endfor
endfor
return A[m,n]

7. Consider the function $M$ defined as follows:

$$M(n) = \begin{cases} 
n - 10 & \text{if } n > 100 \\
M(M(n+11)) & \text{if } n \leq 100
\end{cases}$$

(a) Compute the following.
   i. $M(101)$
   ii. $M(99)$
   iii. $M(87)$

(b) Give a constant-time algorithm that computes $M(n)$ on input $n$. (A constant-time algorithm is one whose running time is independent of the input $n$.)

Answer:

(a) $M(101) = M(99) = M(87) = 91$

(b) The following is a simple algorithm for computing $M$. It just compares $n$ with 100, and either subtracts 10 from $n$ or returns a constant, all of which are constant-time operations.

    if (n> 100) return (n-10) else return 91

To prove that the algorithm is correct, we need to show that $M(n) = 91$ for $n < 101$. Notice first that $M(101) = 91$. From the following claim, it follows that $M(n) = 91$ for all $n \leq 100$

Claim: For all $k \geq 1$ and all $n$ such that $101 - 11k \leq n < 101 - 11(k - 1)$, $M(n) = 91$.

The proof of the claim is by induction on $k$.

Suppose $k = 1$ and $90 \leq n < 101$.

$$M(n) = M(M(n+11))$$
$$= M(n+11 - 10) \quad (\text{since } n + 11 \geq 101)$$
$$= M(n + 1)$$
Therefore, $M(90) = M(91) = \cdots = M(100) = M(101) = 91$.
Assume the claim holds for $k \leq m$. Consider $n$ such that $101 - 11k \leq n < 101 - 11(k - 1)$ for $k = m + 1$.

$$M(n) = M(M(n + 11)) = M(91) \quad (M(n + 11) = 91, \text{ by IH}) = 91 \quad \text{by base case}$$

Thus the claim holds for all $k$, and hence $M(n) = 91$ for all $n < 101$. 

\[\triangleright\]