CHENNAI MATHEMATICAL INSTITUTE

M.Sc. / Ph.D. Programme in Computer Science

Entrance Examination, 2022

This question paper has 5 printed sides. Part A has 10 questions of 3 marks each. Each question in Part A has four choices, of which exactly one is correct. Part B has 7 questions of 10 marks each. The total marks are 100. Answers to Part A must be filled in the answer sheet provided.

Part A

1. If Vinay finishes his homework and the school closes early, then he can play in the park or eat an ice cream. He will end up at the dispensary with tummy ache if he eats ice cream and plays in the park. Which of the following can be correctly inferred?

   (a) If he doesn't end up in the dispensary with tummy ache, then he did not finish his homework or the school closed late.

   (b) If he doesn't end up in the dispensary with tummy ache, he didn't eat ice cream and he didn't play in the park.

   (c) Both (a) and (b)

   (d) None of the above.

   Answer: (d) None of the above

   (a) is not possible, because Vinay can finish his homework, the school can close early, and he plays in the park *without eating icecream*. In this case he will not end up in the dispensary, but neither of the conclusions in (a) can be inferred.

   (b) is not possible, again because Vinay can play in the park without eating icecream, or eat icecream without playing in the park. In both cases, he does not end up in the dispensary. So one cannot infer the conjunction of the two conclusions in (b).

   Hence the correct answer is (d). ⊣

2. There are n members of Chennai Mathematical Institute. Most of them are very studious, and like to own lots of books. Now the following facts have been learnt.

   - No two members own exactly the same number of books.
   - Each member owns strictly less than n books.
   - No member has exactly 200 books.

   Given the above information, which of the following is not a possible value of n?

   (a) 100

   (b) 199

   (c) 200

   (d) 201

Answer: (d) 201

Number the CMI members as $1, 2, \ldots, n$, and let b_i be the number of books owned by i. It is given that $0 \leq b_i < n$ for each $i \leq n$, so the set of b_i's is a subset of $\{0, \ldots, n-1\}$. It is also given that $b_i \neq b_j$ for distinct indices i and j, so the set of b_i's has to be of size n. From this it follows that the set is exactly $\{0, \ldots, n-1\}$. If $n = 201$, then $n - 1 = 200$, and that contradicts the fact that no one has 200 books. Thus (d) is not possible. One can verify that the other three choices are possible. ⊣

3. Which of the following assertions about regular languages is *incorrect*?

   (a) Every subset of a regular language is regular.

   (b) For every regular language L, there is a subset of L that is regular.

   (c) For every language L, there is a superset of L that is regular.

   (d) The complement of every regular language is regular.

   Answer: (a) Every subset of a regular language is regular.

   To see that (a) is incorrect, observe that $\{a, b\}^*$ is regular, while the subset $\{a^n b^n \mid n \geq 0\}$ is not regular.

   Option (b) is correct since $\emptyset$, which is regular, is a subset of every language. (c) is also correct, since Σ^* is regular, and is a superset of any regular language L over the alphabet Σ. Complements of regular languages are also regular, as one can build a DFA for any language and swap the final and non-final states to get an automaton for the complement. ⊣

4. Consider the following languages over the alphabet $\{a, b, c, d\}$

   - $L_1 = \{a^n b^n c^m d^m \mid n, m \geq 0\}$
   - $L_2 = \{a^n b^m c^n d^m \mid n, m \geq 0\}$
   - $L_3 = \{a^n b^m c^m d^n \mid n, m \geq 0\}$

   Which of these languages is/are context-free?

   (a) None of them.

   (b) Only L_1 and L_2.

   (c) Only L_1 and L_3.

   (d) All of them.

   Answer: (c) Only L_1 and L_3 are context-free.

   L_1 is generated by the grammar $S \to TU; \quad T \to aTb \mid \varepsilon; \quad U \to cUd \mid \varepsilon$.

   L_3 is generated by the grammar $S \to aSd \mid T; \quad T \to bTc \mid \varepsilon$.

   One can apply the pumping lemma for CFLs to show that L_2 is not context-free. ⊣

5. As part of a class activity, students in a class of 50 were asked to keep track of the total number of hours that they spent looking at the screen of some digital device on a specific day. It was found that the average screentime for the class was 4 hours. What is the maximum possible number of students with at least 16 hours of screentime?

   (a) 11
   (b) 12
   (c) 13
   (d) 14

   Answer: (b) 12

   The total screentime (for all 50 students) is 200. 13 students spending 16 hours amounts to 208, which is more than the total. 12 students spending 16 hours amounts to 192, which is within the total time. Thus the maximum number of students with at least 16 hours of screentime is 12. ⊣

6. The Telvio mobile service provider allows each customer to choose a part of their 10-digit mobile number when getting a new connection. The first two digits of the number are fixed by the company based on the customer's region. The customer can choose the *last* four digits as they wish. The company chooses each of the remaining four digits uniformly at random, and without replacement, from the list $\{0, 1, 2, \ldots, 9\}$. Note that this means that the digits in positions $3, 4, 5$ and 6 in a Telvio number are all different.

   What is the probability that in the mobile number assigned to a new customer by Telvio, the digits in positions $3, 4, 5$ and 6 appear in increasing order when read from left to right?

   (a) $\frac{1}{4}$
   (b) $\frac{1}{16}$
   (c) $\frac{1}{24}$
   (d) $\frac{1}{32}$

   Answer: (c) $\frac{1}{24}$

   For each choice of four distinct digits, there is exactly one way to write them in increasing order. The number of possible ways of ordering four distinct digits is 24. Thus the answer is $\frac{1}{24}$. ⊣

7. Consider a random graph G on n vertices where for each pair of vertices u, v, there is an edge (u, v) with probability $p \in [0, 1]$. What is the expected number of cycles of length 3 in this graph?

   (a) $\binom{n}{3} \cdot p^3$
   (b) $\frac{p^3}{\binom{n}{3}}$
   (c) $n^3 p^3$
   (d) None of the above

Answer: (a) $\binom{n}{3} \cdot p^3$

There are $\binom{n}{3}$ unordered triples of vertices. The probability that each such vertex set forms a triangle is p^3. Thus the expected number of triangles is $\binom{n}{3} \cdot p^3$. ⊣

The next two questions refer to the following two functions.

```
int f(int m) {
    int a, b, c, d;
    a = 0; b = 0;
    c = 0; d = 1;
    while (a < m) {
        a = a + 1;
        b = b + c;
        int temp = d;
        d = c;
        c = temp;
    }
    return b;
}
```

```
int g(int m) {
    int a = 1;
    int i = 0;
    while (i < m) {
        i = i+1;
        a = 2*a;
    }
    return a;
}
```

8. What is the result of `f(100)`?

   (a) 100

   (b) 5050

   (c) 50

   (d) 1

   Answer: (c) 50

   The value of the pair (c, d) flips between $(0, 1)$ and $(1, 0)$ in each iteration of the `while` loop in `f`, starting with $(0, 1)$. In each iteration, we add 1 to a, and c to b. Thus the sequence of values taken by (a, b) at the end of each iteration is:

   $$(1, 0), (2, 1), (3, 1), (4, 2), ..., (98, 49), (99, 49), (100, 50).$$

   Hence the value of b when the loop exits is 50. In general, $f(n) = \lfloor \frac{n}{2} \rfloor$. ⊣

9. If `g(f(n)) = 32`, which of the following is a possible value of `n`?

   (a) 8

   (b) 11

   (c) 5

   (d) 64

Answer: (b) 11

At the start of each iteration of the loop in g, we maintain the invariant that $a = 2^i$. If $a = 32$ when the loop exits, it means that $i = 5$. Since $5 = f(n)$, it has to be the case that $n = 10$ or $n = 11$. The only choice that fits is (b). ⊣

10. What can you conclude from the following statements about problems A and B?

    (I) There is a polynomial-time algorithm to solve A.

    (II) There is an exponential-time algorithm to solve B.

    (III) B can be reduced to A in polynomial-time.

    (a) Not all of them can be simultaneously true.

    (b) There is a polynomial-time algorithm for B.

    (c) A cannot be reduced to B in polynomial-time.

    (d) There is no exponential-time algorithm for A.

    Answer: (b) There is a polynomial-time algorithm for B.

    If B can be reduced to A and A has a PTIME algorithm, clearly B also has a PTIME algorithm, so (b) is true.

    We cannot assert (a), since B can have both an EXPTIME algorithm and a PTIME algorithm. Since B has a PTIME algorithm, one can reduce A to B in PTIME, so we cannot assert (c). We cannot assert (d), since there can be many inefficient algorithms for A, apart from the efficient PTIME algorithm. ⊣

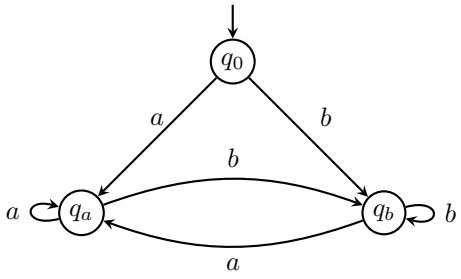
Part B

1. A *Muller automaton* is defined as a tuple $M = (Q, I, \Sigma, \rightarrow, T)$ where:

   - Q is a finite set of *states*;
   - $I \subseteq Q$ is the set of *initial states*;
   - Σ is the finite *alphabet*;
   - $\rightarrow \subseteq Q \times \Sigma \times Q$ is the *transition relation*; and
   - $T \subseteq 2^Q$ is the *accept table*.

   A run of M on a word $x = a_1 \dots a_n$ is a sequence of the form $\rho = q_0 a_1 q_1 \cdots q_{n-1} a_n q_n$ where $q_0 \in I$ and $q_{i-1} \xrightarrow{a_i} q_i$ for each $i \leq n$. For a run ρ as above, we define $vs(\rho) = \{q_0, \dots, q_n\}$, the set of *visited states* along the run ρ. We say that M accepts a word x if there is a run ρ of M on x such that $vs(\rho) \in T$. The language accepted by M, denoted $L(M)$, is the set $\{x \in \Sigma^* \mid x \text{ is accepted by } M\}$.

   Consider the Muller automaton whose states and transitions are depicted below. The initial state is $\{q_0\}$.



   (a) Consider the run $\rho_1 = q_0 a q_a a q_a b q_b$ on the word aab. What is $vs(\rho_1)$? Now, consider the run $\rho_2 = q_0 b q_b b q_b$ on the word bb. What is $vs(\rho_2)$?

   (b) What is the language accepted by the above automaton when the accept table is $\{\{q_0, q_a, q_b\}\}$?

   (c) What should the accept table be in order to accept a^*?

   Answer:

   (a) $vs(\rho_1) = \{q_0, q_a, q_b\}$ and $vs(\rho_2) = \{q_0, q_b\}$.

   (b) We see that the automaton goes to state q_a exactly when it reads the letter a and goes to state q_b exactly when it reads b. The language is thus the set of all words which have at least one occurrence of a and at least one occurrence of b.

   (c) By the logic above, the accept table should be $\{\{q_0\}, \{q_0, q_a\}\}$.

   ⊣

2. Consider the language L over the alphabet $\{a, b\}$ given below.

   $$L = \{w \mid w \text{ has equal number of } a\text{'s and } b\text{'s, and there are no adjacent } a\text{'s.}\}$$

   For instance, the words $abba, abab$ are in the language but not bab and $baab$.

(a) Prove that L does not contain any word that starts and ends with a b.

(b) Give a context-free grammar for L.

Answer:

(a) Suppose, for the sake of contradiction, that L contains a word bub. Since bub has equal number of a's and b's, u contains two more a's than b's. Let u have $n+2$ a's and n b's, for some natural number n. There are $n+1$ gaps betweens the a's. Mapping n b's to these gaps will leave at least one gap vacant, and thus there are two adjacent a's in u. This contradicts the assumption that $bub \in L$.

(b)

$$\begin{aligned} S &\to S_{ab} \mid S_{ba} \mid S_{ab}S_{ba} \mid \varepsilon \\ S_{ab} &\to aS_{ba}b \mid S_{ab}S_{ab} \mid ab \\ S_{ba} &\to bS_{ab}a \mid S_{ba}S_{ba} \mid ba \end{aligned}$$

The intuition is that S_{ab} generates words from L which start with an a and ends with a b. Similarly for S_{ba}. We ensure that concatenation of the form $S_{ba}S_{ab}$ is not be allowed, to avoid consecutive occurrences of a.

⊣

3. We say that an integer a is co-prime to another integer b if $\gcd(a, b) = 1$. For any integer n, $\varphi(n)$ is the number of integers from 1 up to $|n|$ that are co-prime to n.

(a) Calculate $\varphi(5)$, $\varphi(10)$ and $\varphi(20)$.

(b) Show that $\varphi(p) = p - 1$ for any prime p.

(c) Prove that if a is co-prime to b then the remainder of a when divided by b is also co-prime to b.

Answer:

(a) $\varphi(5) = 4$, since $1, 2, 3, 4$ are all co-prime to 5. $\varphi(10) = 4$, since $1, 3, 7, 9$ are co-prime to 10. $\varphi(20) = 8$, since $1, 3, 7, 9, 11, 13, 17, 19$ are co-prime to 20.

(b) If p is prime, $\gcd(a, p) = 1$ for all $a < p$. Thus $\varphi(p) = p - 1$.

(c) Observe that $\gcd(a \bmod b, b) = \gcd(a, b)$. If $\gcd(a, b) = 1$ then $\gcd(a \bmod b, b) = 1$ as well. Thus the remainder of a when divided by b is also co-prime to b.

⊣

4. You are organizing a party involving $2n$ diplomats. Each pair of diplomats are either friends or enemies. You have managed to invite an excellent set of guests, each of whom has more friends than enemies (among the other guests). Can you now seat them at a round table so that everyone has two friends as their neighbours. (**Hint:** Model this situation as an appropriate graph so that the desired seating arrangement is a Hamiltonian path.)

Answer: Let the wizards be numbered $1, \ldots, 2n$. Form a graph $G = (V, E)$ with $V = \{1, \ldots, 2n\}$ and $E = \{(i, j) \mid i \text{ and } j \text{ are friends}\}$. Treat the edges to be undirected. Since each wizard has more friends than enemies, it means that the degree of each vertex in G is at least n. If we find a Hamiltonian cycle in this graph, we can seat wizards around the table so that each wizard has friends on both sides. By *Dirac's Theorem*, We know that a Hamiltonian cycle always exists for such graphs. A proof of this fact can be found here. ⊣

5. For any set S of natural numbers, we say that a relation $R \subseteq S \times S$ is a 2-spanner of S if it satisfies the following conditions:

   - $(i, j) \in R \Rightarrow i < j$;
   - $i < j \Rightarrow [((i, j) \in R) \text{ or } (\exists k : (i, k) \in R \wedge (k, j) \in R)]$.

   For example, $\{(1, 2), (2, 3)\}$ is a 2-spanner for $\{1, 2, 3\}$, and

   $$R_1 = \{(1, 2), (1, 4), (2, 3), (2, 4), (3, 4), (4, 5), (4, 6), (4, 7), (5, 6), (6, 7)\}$$

   is a 2-spanner for $S = \{1, \ldots, 7\}$. There are other 2-spanners for S, of course. $R_2 = \{(i, j) \mid i, j \in \{1, \ldots, 7\}, i < j\}$ is an example. But R_1 is of size 10, while R_2 is of size 21. We would like to find 2-spanners that are as small as possible.

   (a) Suppose you are given 2-spanners R_1 and R_2 for $\{1, \ldots, 7\}$ and $\{9, \ldots, 15\}$ respectively, each of size 10. Use them to construct a 2-spanner R for $\{1, \ldots, 15\}$. Try to get R of size 34.

   (b) Generalize the above construction to show that any set S of size $2^k - 1$ (for $k > 2$) has a 2-spanner of size $(k - 2)2^k + 2$.

   Answer: For $k > 0$, let $S(k) = \{1, \ldots, 2^k - 1\}$. W.l.o.g. we will construct 2-spanners for $S(k)$. That can be adapted to any S of size $2^k - 1$. Let $T(k)$ denote the size of the minimal 2-spanner for $S(k)$.

   For $S(1) = \{1\}$, the 2-spanner is $\emptyset$, of size 0.

   For $S(2) = \{1, 2, 3\}$, the 2-spanner is $\{(1, 2), (2, 3)\}$, of size 2.

   For $S(3) = \{1, \ldots, 7\}$, we have already shown a 2-spanner of size $10 = (3 - 2)2^3 + 2$.

   For $k > 3$, assume there are 2-spanners R_1 and R_2 of size $T(k - 1) = (k - 3)2^{k-1} + 2$ for $\{1, \ldots, 2^{k-1} - 1\}$ and $\{2^{k-1} + 1, \ldots, 2^k - 1\}$. Form R by adding the pairs $(i, 2^{k-1})$ for all $i < 2^{k-1}$ and the pairs $(2^{k-1}, j)$ for $j > 2^{k-1}$ to $R_1 \cup R_2$. The number of such pairs is $2(2^{k-1} - 1) = 2^k - 2$. So we have

   $$\begin{aligned} T(k) &= 2T(k - 1) + 2^k - 2 \\ &= 2[(k - 3)2^{k-1} + 2] + 2^k - 2 \\ &= (k - 3)2^k + 4 + 2^k - 2 \\ &= (k - 2)2^k + 2 \end{aligned}$$

   ⊣

6. There is a treasure hunt game in which parts of a treasure are hidden across n islands, numbered 1 to n. You start in island 1, and aim to collect all parts of the treasure, by hopping from island to island. Each island has a list of other islands you can directly go to. (Note that if you can directly go from island i to island j, it does not necessary mean that you can directly go from j to i.) You can revisit the same island multiple times during your search. Design an algorithm that takes as input the number n of islands, the n neighbourhood lists, and determines if you can succeed in collecting all parts of the treasure. The algorithm should run in time $O(n^2 \cdot 2^n)$.

   Answer: Let $I = \{1, \ldots, n\}$. Consider a graph G where the nodes are pairs (S, i), where $S \subseteq \{1, \ldots, n\}$, and there is an edge from (S, i) to (S', j) if $S' = S \cup \{j\}$ and j is a neighbour of i. The question amounts to asking if there is some j such that (I, j) is reachable from $(\{1\}, 1)$ in G, and can be solved using BFS in time $O(|V| + |E|)$. The number of vertices in the graph is $n.2^n$ and the number of edges is $|E|.2^n$ (if j is a neighbour of i, we have an edge from (S, i) to $(S \cup \{j\}, j)$ for each $S \subseteq \{1, \ldots, n\}$). Since $|E| \leq n^2$, a naive algorithm for reachability runs in time $O(n^2 \cdot 2^n)$.

   An alternate solution, with better complexity is as follows. Consider a graph G' with vertices $\{1, \ldots, n\}$ and edges given by the neighbourhood lists. This is the graph mapping the islands and the paths between them. Decompose G' into SCCs in $O(n^2)$ time. These SCCs will be connected as a DAG H' (each SCC will be a vertex in this DAG). There is a way to collect all parts of the treasure iff there is a path in H' that visits all the vertices. In other words, there is a solution iff the longest path in the DAG H' equals the number of nodes in H'. Longest paths in DAGs can be found in linear time. Hence the overall complexity comes to $O(n^2)$. $\dashv$

7. Consider the following inventory problem. You are running a company that sells lorries. Predictions tell you the quantity of sales to expect over the next n months. Let d_i denote the number of sales expected in month i. We assume that sales happen on the first of the month, and that lorries not sold are stored till the start of the next month. You can store at most C lorries, and it costs R to store each lorry for a month. You receive lorries from the manufacturer in shipments, each of which has a transportation fee of F (regardless of the number of lorries ordered). You start out with no lorries. Your aim is to place orders (say l_i is the number of lorries ordered in month i) so as to satisfy the following constraints:

   - For each i, the number of lorries on hand at the start of month i (l_i+ whatever is stored from the previous month) is enough to meet the demand d_i.
   - For each i, the number of lorries left over after meeting the demand d_i should not exceed the storage capacity C.

   The aim is to determine the orders $(l_1, \ldots, l_n)$ that will minimize the overall transportation fee and the overall storage cost.

   For example, if $n = 4$, and the demands for each month is given by $10, 11, 8, 12$, and if $F = 50$, while $R = 2$ and $C = 10$, then here are a few possible scenarios.

Month	1	2	3	4	Total
d_i	10	11	8	12	
l_i	10	11	8	12	
Cost	50	50	50	50	200
l_i	20	11	0	10	
Cost	$50 + 10 \times 2$	$50 + 10 \times 2$	2×2	50	194
l_i	10	19	0	12	
Cost	50	$50 + 8 \times 2$	0	50	166

Let $c_i(S)$ denote the minimal cost of transportation and storage to meet demands from month i till month n, given that we have S lorries left over at the start of month i, while satisfying the storage requirements.

(a) Write an expression for $c_n(S)$.

(b) Express $c_i(S)$ in terms of $c_{i+1}(S')$ for appropriate values of S'.

(c) Convert the above equations into a dynamic programming algorithm that computes $c_1(0)$. Your algorithm must run in time polynomial in n and C.

Answer: Let $c_i(S)$ be the optimal cost of ordering lorries for the rest of the semester starting from month i, given a stock of S of lorries left over. We need to finally compute $c_1(0)$.

The recurrence for $c_i(S)$ can be derived as follows. You can order anywhere from $j = d_i - S$ to $j = d_i + C - S$ pieces this month. If we order j pieces, the cost incurred is F (the delivery fee) + $R(j + S - d_i) + c_{i+1}(j + S - d_i)$. So we take minimum over all possible values of j. The base case is $c_n(S) = F$ (there is no storage till next month, so only the delivery fee is incurred) if $S \geq d_n$, and $c_n(S) = 0$ otherwise.

$$c_i(S) = \begin{cases} F + \min_{j=d_i}^{d_i+C-S} \left(R(j + S - d_i) + c_{i+1}(j + S - d_i)\right) & \text{if } i < n \\ F & \text{if } i = n \text{ and } S \geq d_n \\ 0 & \text{otherwise} \end{cases}$$

The $c_i(S)$ values can be systematically computed by filling in a suitable array that stores the values of the recursive calls. ⊣