CHENNAI MATHEMATICAL INSTITUTE

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

Entrance Examination, 18 May 2017

(With solutions)

This question paper has 5 printed sides. Part A has 10 questions of 3 marks each. 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. The regular expression $(a^* + b)^*$ is equivalent to which of the following regular expressions:

   (a) a^*b^* (b) $(a^*b + b)^*$ (c) $(a + b^*)^*$ (d) $(a^*b)^*$

   Answer: (c) $(a + b^*)^*$

   $(a^* + b)^*$ allows any string over $\{a, b\}$ and is the same as $(a + b)^*$. Option (a) only allows strings of the form a^ib^j. Options (b) and (d) do not allow strings of the form a^i with no b's. ⊣

2. An FM radio channel has a repository of 10 songs. Each day, the channel plays 3 distinct songs that are chosen randomly from the repository.

   Mary decides to tune in to the radio channel on the weekend after her exams. What is the probability that no song gets repeated during these 2 days?

   (a) $\binom{10}{3}^2 \cdot \binom{10}{6}^{-1}$

   (b) $\binom{10}{6} \cdot \binom{10}{3}^{-2}$

   (c) $\binom{10}{3} \cdot \binom{7}{3} \cdot \binom{10}{3}^{-2}$

   (d) $\binom{10}{3} \cdot \binom{7}{3} \cdot \binom{10}{6}^{-1}$

   Answer: (c) $\binom{10}{3} \cdot \binom{7}{3} \cdot \binom{10}{3}^{-2}$

   $\binom{10}{3}$ is the number of ways of choosing 3 songs on day 1. $\binom{7}{3}$ is the number of ways of choosing 3 different songs on day 2, so $\binom{10}{3}\binom{7}{3}$ is the number of combinations that meet Mary's requirement. $\binom{10}{3}\binom{10}{3}$ is the total number of ways of choosing 3 songs on each of the two days, without any constraints. The ratio is the given probability. ⊣

3. Four siblings go shopping with their father. If Abhay gets shoes, then Asha does not get a necklace. If Arun gets a T-shirt, then Aditi gets bangles. If Abhay does not get shoes or Aditi gets bangles, the mother will be happy. Which of the following is true?

   (a) If the mother is happy, then Aditi got bangles.

(b) If Aditi got bangles, then Abhay got shoes.

(c) If the mother is not happy, then Asha did not get a necklace and Arun did not get a T-shirt.

(d) None of the above.

Answer: (c) If the mother is not happy, then Asha did not get a necklace and Arun did not get a T-shirt.

Let p denote that Asha get a necklace, q denote that Abhay gets shoes, r denote that Arun gets a T-shirt, s denote that Aditi gets bangles and w denote that mother is happy. The question is equivalent to the formula $\phi : q \to \neg p \wedge r \to s \wedge (\neg q \vee s) \to w$. A formula ψ is a valid implication iff every truth assignment to p, q, r, s, w that makes ϕ true also makes ψ true. Option (a) is the formula $\psi_a : w \to s$, option (b) is the formula $\psi_b : s \to q$ and option (c) is the formula $\psi_c : \neg w \to (\neg p \wedge \neg r)$. Neither ψ_a nor ψ_b are valid implications but ψ_c is.

$\dashv$

4. City authorities are concerned about traffic accidents on major roads. They would like to have ambulances stationed at road intersections to quickly reach the scene of any accident along these roads. To minimize response time, ambulances are to be located at intersections with traffic lights so that any segment of road can be reached by at least one ambulance that does not have to pass through a traffic light to reach the scene of the accident. If we model the road network as a graph, where intersections with traffic lights are vertices and edges represent road segments between traffic lights, the graph theoretic question to be answered is:

   (a) Find a spanning tree with minimum number of edges.

   (b) Find a spanning tree with minimum cost.

   (c) Find a minimal colouring.

   (d) Find a minimum size vertex cover.

   Answer: (d) Find a minimum size vertex cover.

   Each ambulance "covers" the adjacent roads, and all roads are covered in this way. $\dashv$

5. Let G be an arbitrary graph on n vertices with $4n - 16$ edges. Consider the following statements:

   I There is a vertex of degree smaller than 8 in G.

   II There is a vertex such that there are less than 16 vertices at distance exactly 2 from it.

   Which of the following is true:

   (a) I only
   (b) II only
   (c) Both I and II
   (d) Neither I nor II

Answer: (a) I only

By the Pigeonhole Principle, there is a vertex of degree < 8.

As a counterexample to II, take, for instance a cycle on $n-k$ vertices $u_1, \ldots, u_{n-k}$ (k to be fixed). Take a vertex v_0 and connect it to all the $n-k$ cycle vertices to obtain a wheel with $n-k+1$ vertices. Now take $k-1$ vertices $v_1, \ldots, v_{k-1}$. For each i in $[1, \ldots, k-1]$ connect v_i to $k+3$ equally spaced vertices C_{v_i} on the cycle so that C_{v_i} is connected to $u_i, u_{i+\lfloor((n-k)/(k+3)\rfloor}, \ldots$.

Thus there are $(n-k)+1+(k-1) = n$ vertices and $(n-k)+(n-k)+(k-1)(k+3) = 2n-3+k^2 = 4n-16$ for $k = \sqrt{(2n-13)}$

Each u_i is at distance exactly 2 from each of its non-neighbours on the cycle (via v_0) for $(n-k-3) = \Omega(n)$ vertices at distance 2. Each v_j is at distance exactly 2 from at least all the other v_k vertices (plus at least $2ku_i$ vertices if $j \neq 0$) for at least $k-1 = \Omega(sqrt(n))$ vertices at distance 2. ⊣

6. What does the following function compute in terms of n and d, for integer values of d? Note that the operation `/` denotes floating point division, even if the arguments are both integers.

```
function foo(n,d){
  if (d == 0){
     return 1;
  }else{
    if (d < 0){
      return foo(n,d+1)/n;
    }else{
      return n*foo(n,d-1);
}}}
```

(a) $\log_d n$ if $d < 0$, n^d if $d > 0$.

(b) n^d for all values of d.

(c) $n \times d$ if $d > 0$, $n \div d$ if $d < 0$.

(d) $n \times d$ for all values of d.

Answer: (b) n^d for all values of d.

For $d > 0$, this computes n^d by repeated multiplication. For d negative, the answer is $\frac{1}{n^d}$ by repeated division. ⊣

7. Consider the following functions `f()` and `g()`.

```
f(){                    g(){
  w = 5;                   z = w+1;
  w = 2*z + 2;             z = 3*z - w;
}                          print(z);
                         }
```

We start with `w` and `z` set to `0` and execute `f()` and `g()` in parallel—that is, at each step we either execute one statement from `f()` or one statement from `g()`. Which of the following is *not* a possible value printed by `g()`?

(a) `-2` (b) `-1` (c) `2` (d) `4`

Answer: (c) `2` ⊣

The possible values are : {-1,-2,3,4,7,13}, corresponding to the interleavings below.

```
w = 5; w = 2*z + 2; z = w+1; z = 3*z - w:  --- z = 7
w = 5; z = w+1; w = 2*z + 2; z = 3*z - w;  --- z = 4
w = 5; z = w+1; z = 3*z - w;               --- z = 13
z = w+1; z = 3*z - w;                      --- z = 3
z = w+1; w = 5; z = 3*z - w;               --- z = -2
z = w+1; w = 5; w = 2*z + 2; z = 3*z - w;  --- z = -1
```

8. A *stable sort* preserves the order of values that are equal with respect to the comparison function. We have a list of three dimensional points

$$[(7,1,8),(3,5,7),(6,1,4),(6,5,9),(0,2,5),(9,0,9)].$$

We sort these in ascending order by the second coordinate. Which of the folllowing corresponds to a stable sort of this input?

(a) $[(9,0,9),(7,1,8),(6,1,4),(0,2,5),(6,5,9),(3,5,7)]$

(b) $[(0,2,5),(3,5,7),(6,1,4),(6,5,9),(7,1,8),(9,0,9)]$

(c) $[(9,0,9),(7,1,8),(6,1,4),(0,2,5),(3,5,7),(6,5,9)]$

(d) $[(9,0,9),(6,1,4),(7,1,8),(0,2,5),(3,5,7),(6,5,9)]$

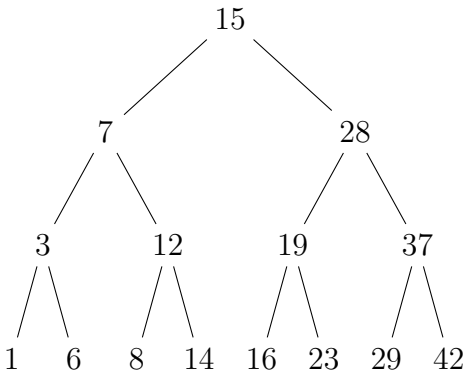
Answer: (c) $[(9,0,9),(7,1,8),(6,1,4),(0,2,5),(3,5,7),(6,5,9)]$

In a stable sort, the original order of the pairs (7,1,8),(6,1,4) and (3,5,7),(6,5,7) with equal second coordinates must be preserved in the sorted output.

⊣

9.

Suppose we constructed the binary search tree shown at the right by starting with an empty tree and inserting one element at a time from an input sequence, without any rotations or other manipulations. Which of the following assertions about the order of elements in the input sequence *cannot* be true?



(a) 8 came after 3 and 19 came after 29.

(b) 7 came before 8 and 23 came after 37.

(c) 1 came after 12 and 29 came before 42.

(d) 3 came before 14 and 16 came before 28.

Answer: (d) 3 came before 14 and 16 came before 28.

28 is an ancestor of 16, so 16 must have come after 28. In the tree only ancestor-descendant relationships matter in determining the order in which elements arrive. An ancestor must always come before any of its descendants. Incomparable elements could come in any order. ⊣

10. We have constructed a polynomial time reduction from problem A to problem B. Which of the following is a valid inference?

(a) If the best algorithm for B takes exponential time, there is no polynomial time algorithm for A

(b) If the best algorithm for A takes exponential time, there is no polynomial time algorithm for B.

(c) If we have a polynomial time algorithm for A, we must also have a polynomial time algorithm for B.

(d) If we don't know whether there is a polynomial time algorithm for B, there cannot be a polynomial time algorithm for A.

Answer: (b) If the best algorithm for A takes exponential time, there is no polynomial time algorithm for B.

If we have a polynomial time solution for B, the reduction gives us a polynomial time solution for A.

⊣

Part B

1. Let $\Sigma = \{a, b, c\}$. Let L_{even} be the set of all even length strings in Σ^*.

   (a) Construct a deterministic finite state automaton for L_{even}.

   (b) We consider an operation $\texttt{Erase}_{ab}$ that takes as input a string $w \in \Sigma^*$ and erases all occurrences of the pattern ab from w. Formally, it can be defined as follows:

   $$\texttt{Erase}_{ab}(w) := \begin{cases} w & \text{if } w \text{ does not contain the pattern } ab \\ \texttt{Erase}_{ab}(w_1)\ \texttt{Erase}_{ab}(w_2) & \text{if } w = w_1\ ab\ w_2 \text{ for some } w_1, w_2 \in \Sigma^* \end{cases}$$

   For instance, $\texttt{Erase}_{ab}(cacb) = cacb$, $\texttt{Erase}_{ab}(cabcbab) = ccb$ and $\texttt{Erase}_{ab}(ab) = \epsilon$.

   For a language L, we define $\texttt{Erase}_{ab}(L)$ to be the set of strings obtained by applying the $\texttt{Erase}_{ab}$ operation to each string in L:

   $$\texttt{Erase}_{ab}(L) := \{\texttt{Erase}_{ab}(w) \mid w \in L\}$$

   Show that $\texttt{Erase}_{ab}(L_{even})$ is a regular language.

   Answer:

   (a) L_{even} can be recognized by an automaton with two states $\{q_0, q_1\}$, where q_0 is both an initial and final state. On input letters a and b, the automaton switches from q_0 and q_1 and vice versa. An odd length input will take the automaton to q_1 and an even length input will take the automaton to q_0.

   (b) $\texttt{Erase}_{ab}(L_{even})$ is the set of all even length strings which do not contain ab.
   It is easy to construct a nondeterministic automaton with three states $\{q_0, q_1, q_2\}$ for the language L_{ab} consisting of all strings containing ab. Here, q_0 is the initial state and q_2 is the final state. There is a self loop on $\{a, b\}$ at both q_0 and q_2 and there are transitions $q_0 \xrightarrow{a} q_1$ and $q_1 \xrightarrow{b} q_2$.
   Since L_{ab} is regular, so is its complement $L_{\overline{ab}}$, the language of all strings without ab.
   $\texttt{Erase}_{ab}(L_{even})$ is the intersection of $L_{\overline{ab}}$ with L_{even}.

   ⊣

2. There are a number of tourist spots in a city and a company GoMad runs shuttle services between them. Each shuttle plies between a designated origin and destination, and has a name. Due to lack of coordination, the same name may be allotted to multiple routes.

   To make matters worse, another company GoCrazy introduces its shuttle services using the same set of shuttle names. A GoMad shuttle and a GoCrazy shuttle with the same name may start at different origins and/or end at different destinations.

   A pass from a company allows unlimited travel in all the company's shuttles. For each company, we have a list that specifies all routes allotted to each shuttle name.

   Design an algorithm to find out if there is a source s, a target t, and a sequence of shuttle names σ such that, irrespective of whether you are carrying a GoMad pass or a GoCrazy pass, you can start at s and arrive at t using the sequence σ.

Answer: Create a graph where the set of vertices are pairs (Spot 1, Spot 2) of tourist spots. There is an edge labeled "Name 1" from (Spot 1, Spot 2) to (Spot 3, Spot 4) iff there is a GoMad shuttle "Name 1" from Spot 1 to Spot 3 and a GoCrazy shuttle "Name 1" from Spot 2 to Spot 4. The answer to the question is yes iff there is a directed path from a vertex (Spot 1, Spot 1) to a vertex (Spot 2, Spot 2). ⊣

3. Let $\Sigma = \{a, b\}$. Given words $u, v \in \Sigma^*$, we say that v extends u if v is of the form xuy for some $x, y \in \Sigma^*$. Given a fixed word u, we are interested in identifying whether a finite state automaton accepts some word that extends u.

   Describe an algorithm that takes as input a finite state automaton (DFA or NFA) $\mathcal{A}$ over $\Sigma = \{a, b\}$ and a word $u \in \Sigma^*$ and reports *"Yes"* if some word in the language of $\mathcal{A}$ extends u and *"No"* if no word in the language of $\mathcal{A}$ extends u.

   Answer: Let R be the set of states that can be reached by a path from the initial state in $\mathcal{A}$ and let S be the set of states from which there is a path to one of the final states in $\mathcal{A}$.

   For each pair of states $(r, s) \in R \times S$, $\mathcal{A}_{r,s}$ is obtained from $\mathcal{A}$ by keeping the set of states and set of transitions unchanged, but making r the unique intial state and s the unique final state.

   If $u \in L(\mathcal{A}_{r,s})$, then there must be a word $xuy \in L(\mathcal{A})$, where x labels the path from the initial state to r ($r \in R$ is reachable from the initial state), u labels the path from r to s, and y labels the path from s to some final state of $\mathcal{A}$ ($s \in S$, so such a path must exist.)

   Hence, output "Yes" if $u \in L(\mathcal{A}_{r,s})$ for some $(r, s) \in R \times S$, and "No" otherwise. ⊣

4. In a party there are $2n$ participants, where n is a positive integer. Some participants shake hands with other participants. It is known that there are no three participants who have shaken hands with each other. Prove that the total number of handshakes is not more than n^2.

   Answer: Model this as a graph problem, we are looking for a graph without a triangle. We show that the maximum number of edges that a triangle-free graph on $2n$ vertices can have is n^2.

   The proof by induction on n.

   The base cases $n = 1, 2$ are easy to see by inspection.

   For the inductive step, let $n \geq 3$. If the graph G has no edges then there is nothing to prove. Otherwise, take an edge $\{x, y\}$, and consider the graph G' on $2(n-1)$ vertices obtained by deleting both the vertices $\{x, y\}$ from G.

   By the inductive hypothesis, G' has at most $(n-1)^2$ edges. Since graph G is triangle-free and $\{x, y\}$ is an edge in G, there is no vertex v in G' such that both $\{v, x\}$ and $\{v, y\}$ are edges in G. So the number of edges with one end-point in the set $\{x, y\}$ and the other in the set of vertices of G' is at most the number of vertices in G', namely $2(n-1)$.

   Now the set of edges of G consists of (i) all the edges in G', plus (ii) the one edge $\{x, y\}$, plus (iii) the at most $2(n-1)$ edges with exactly one end-point in $\{x, y\}$. So the number of edges of G is at most $(n-1)^2 + 1 + 2(n-1) \leq n^2$. ⊣

5. An undirected graph is *connected* if, for any two vertices $\{u, v\}$ of the graph, there is a path in the graph starting at u and ending at v. A *tree* is a connected, undirected graph that contains no cycle.

   (a) A *leaf* in a tree is a vertex that has degree 1. Prove that if G is a tree with at least two vertices then G contains at least two leaves.

   (b) A *bipartite graph* is one in which the vertex set V can be partitioned into two disjoint sets V_1 and V_2 so that for every edge $\{u, v\}$, u and v lie in different partitions—that is, $u \in V_1$ and $v \in V_2$ or vice versa. Prove that if G is a tree with at least two vertices, then G is bipartite.

   Answer:

   (a) Consider a maximal path $\rho = v_0 v_1 \dots v_k$ in the tree. If v_0 is not a leaf, then it has a neighbour $v \neq v_1$, and the path ρ can be extended to a longer one $v\rho$ since there are no cycles. This contradicts the fact that ρ is maximal, thereby showing that v_0 is a leaf. A similar argument shows that v_k is also a leaf.

   (b) If we root the tree at any node, we can colour the tree level by level by alternating two colours. This 2-colouring gives us the partition of the vertex set.

   ⊣

6. We are given a sequence of pairs of integers $(a_1, b_1), (a_2, b_2), \dots (a_n, b_n)$. We would like to compute the largest k such that there is a sequence of numbers $c_{i_1} \leq c_{i_2} \leq \dots \leq c_{i_k}$ with $1 \leq i_1 < i_2 < ... < i_k \leq n$ and for each $j \leq k$, $c_{i_j} = a_{i_j}$ or $c_{i_j} = b_{i_j}$. Describe an algorithm to solve this problem and explain its complexity.

   Answer: Compute two arrays A and B with $A[i]$ and $B[i]$ giving the length of the longest such sequence in $(a_1, b_1), \dots (a_i, b_i)$ with a_i and b_i as the last elements, respectively.

   Note that $A[1] = 1$ and $B[1] = 1$. To compute $A[i]$ and $B[i]$ for $i \geq 2$, we first compute the following quantities:

$$\alpha_i^1 = \begin{cases} 0 & \text{if } a_j > a_i \text{ for all } j < i \\ \max\{\ A[j] \mid j < i \text{ and } a_j \leq a_i\} & \text{otherwise} \end{cases}$$

$$\alpha_i^2 = \begin{cases} 0 & \text{if } b_j > a_i \text{ for all } j < i \\ \max\{\ B[j] \mid j < i \text{ and } b_j \leq a_i\} & \text{otherwise} \end{cases}$$

$$\beta_i^1 = \begin{cases} 0 & \text{if } a_j > b_i \text{ for all } j < i \\ \max\{\ A[j] \mid j < i \text{ and } a_j \leq b_i\} & \text{otherwise} \end{cases}$$

$$\beta_i^2 = \begin{cases} 0 & \text{if } b_j > b_i \text{ for all } j < i \\ \max\{\ B[j] \mid j < i \text{ and } b_j \leq b_i\} & \text{otherwise} \end{cases}$$

   Then, we get $A[i] = \max(\alpha_i^1, \alpha_i^2) + 1$ and $B[i] = \max(\beta_i^1, \beta_i^2) + 1$.

   Arrays A and B can be computed using dynamic programming in time $O(n^2)$. The required answer is the maximum over all elements $A[1], \dots, A[n], B[1], \dots, B[n]$. ⊣

7. Consider the following function that takes as input a sequence A of integers with n elements, $A[1], A[2], \ldots, A[n]$ and an integer k and returns an integer value. The function `length(S)` returns the length of sequence `S`. Comments start with `//`.

```
function mystery(A, k){
  n = length(A);
  if (k > n) return A[n];

  v = A[1];
  AL = [ A[j] : 1 <= j <= n, A[j] < v ];  // AL has elements < v in A
  Av = [ A[j] : 1 <= j <= n, A[j] == v ]; // Av has elements = v in A
  AR = [ A[j] : 1 <= j <= n, A[j] > v ];  // AR has elements > v in A

  if (length(AL) >= k) return mystery(AL,k);
  if (length(AL) + length(Av) >= k) return v;
  return mystery(AR, k - (length(AL) + length(Av)));
}
```

(a) Explain what the function computes.

(b) What is the worst-case complexity of this algorithm in terms of the length of the input sequence A?

(c) Give an example of a worst-case input for this algorithm.

Answer:

(a) *mystery*(A, k) finds the k^{th} element in A in ascending order. This is a divide-and-conquer selection algorithm, similar to quicksort.

(b) The worst case complexity is $O(n^2)$ because of the fixed choice of pivot, as in quicksort.

(c) A typical worst case input would be, for instance, where A is in ascending order and $k = n$.

⊣