CHENNAI MATHEMATICAL INSTITUTE

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

Entrance Examination, 15 May 2018

This question paper has 4 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. Which of the words below matches the regular expression $a(a+b)^*b + b(a+b)^*a$?

   (a) aba (b) bab (c) $abba$ (d) $aabb$

   Answer: (d)

   Word should start with an a and end with b, or start with b and end with a. ⊣

2. Akash, Bharani, Chetan and Deepa are invited to a party. If Bharani and Chetan attend, then Deepa will attend too. If Bharani does not attend, then Akash will not attend. If Deepa does not attend, which of the following is true?

   (a) Chetan does not attend
   (b) Akash does not attend
   (c) either (a) or (b)
   (d) none of the above

   Answer: (c)

   If Deepa does not attend, then one of Chetan and Bharani is absent. The first case is (a). The second case is that Bharani does not attend – but it is given that Akash will not attend if Bharani does not attend. So in the second case, (b) holds. So either (a) holds or (b) holds, and hence the correct answer is (c). ⊣

3. In a running race, Geetha finishes ahead of Shalini and Vani finishes after Aparna. Divya finishes ahead of Aparna. Which of the following is a minimal set of additional information that can determine the winner?

   (a) Geetha finishes ahead of Divya and Vani finishes ahead of Shalini.

   (b) Aparna finishes ahead of Shalini.

   (c) Divya finishes ahead of Geetha.

   (d) None of the above.

Answer: (c)

Given: Geetha > Shalini and Divya > Aparna > Vani. The minimal extra information needed to decide the winner is a comparison between Geetha and Divya. (a) is not correct since it gives additional comparison between Vani and Shalini. (b) is not correct since it does not decide the winner. (c) is the correct option, since it decides Divya to be the winner and does not provide any extra information. (d) is not a correct option because option (c) is correct.

⊣

4. Let $G = (V, E)$ be an undirected simple graph, and s be a designated vertex in G. For each $v \in V$, let $d(v)$ be the length of a shortest path between s and v. For an edge (u, v) in G, what *can not* be the value of $d(u) - d(v)$?

   (a) 2 (b) -1 (c) 0 (d) 1

   Answer: (a)

   Note that $d(u) \leqslant d(v) + 1$ ⊣

5. How many paths are there in the plane from $(0, 0)$ to $(m, n) \in \mathbb{N} \times \mathbb{N}$, if the possible steps from (i, j) are either $(i + 1, j)$ or $(i, j + 1)$?

   (a) $\binom{2m}{n}$ (b) $\binom{m}{n}$ (c) $\binom{m+n}{n}$ (d) m^n

   Answer: (c)

   From a vertex (i, j) with either $i > m$ or $j > n$, it is not possible to reach (m, n). Therefore one cannot overshoot m on the x-axis, and n on the y-axis. The total number of steps required to reach (m, n) is $m + n$ (m steps on the x-axis and n on the y-axis). Among the $m + n$ steps, the m x-axis steps and n y-axis steps can be distributed in any manner. This is obtained by choosing n steps for the y-axis (the rest would go to the x-axis).

   ⊣

6. You are given two coins A and B that look identical. The probability that coin A turns up heads is $\frac{1}{4}$, while the probability that coin B turns up heads is $\frac{3}{4}$. You choose one of the coins at random and toss it twice. If both the outcomes are heads, what is the probability that you chose coin B?

   (a) $\frac{1}{16}$ (b) $\frac{1}{2}$ (c) $\frac{9}{16}$ (d) $\frac{9}{10}$

   Answer: (d)

   $\Pr[\text{2 heads} \mid B \text{ is chosen}] = \frac{9}{16}$ and $\Pr[\text{2 heads} \mid A \text{ is chosen}] = \frac{1}{16}$. Hence

   $$\Pr[B \text{ is chosen} \mid \text{2 heads}] = 0.9.$$

   ⊣

7. Let C_n be the number of strings w consisting of n X's and n Y's such that no initial segment of w has more Y's than X's. Now consider the following problem. A person stands on the edge of a swimming pool holding a bag of n red and n blue balls. He draws a ball out one at a time and discards it. If he draws a blue ball, he takes one step back, if he draws a red ball, he moves one step forward. What is the probability that the person remains dry?

(a) $\frac{C_n}{2^{2n}}$ (b) $\frac{C_n}{\binom{2n}{n}}$ (c) $\frac{n \cdot C_n}{(2n)!}$ (d) $\frac{n \cdot C_n}{\binom{2n}{n}}$

Answer: (b)

The total number of ways of choosing $2n$ balls where n balls are identical of one type, and the other n balls are identical of a different type is $\frac{(2n)!}{n! \cdot n!} = \binom{2n}{n}$. For the person to remain dry, at any point of time he should have chosen more red balls than blue balls. This is given by C_n. Hence the answer is (b). ⊣

8. There are 7 switches on a switchboard, some of which are *on* and some of which are *off*. In one move, you pick any 2 switches and toggle each of them—if the switch you pick is currently off, you turn it on, if it is on, you turn it off. Your aim is to execute a sequence of moves and turn all 7 switches on. For which of the following initial configurations is this *not* possible? Each configuration lists the initial positions of the 7 switches in sequence, from switch 1 to switch 7.

(a) (off,on,off,on,off,off,on) (b) (off,on,on,on,on,on,off)
(c) (on,off,on,on,on,on,on) (d) (off,off,off,off,off,on,off)

Answer: (c)

The parity of switches in each position is unchanged after each move. If all 7 switches are on at the end, the final parity of "off" is even and of "on" is odd. So we can only achieve this if we start with a configuration where the number of "off" switches is even and the number of "on" switches is odd. The exact order does not matter since we can pick any two to toggle at each step. ⊣

9. Your college has sent a contingent to take part in a cultural festival at a neighbouring institution. Several team events are part of the programme. Each event takes place through the day with many elimination rounds. Your contingent is multi-talented and each individual has the skills to take part in a subset of the events. However, the same individual cannot be part of the team for two different events because of a possible clash in timings. Your aim is to create teams to take part in as many events as possible.

To do this, you decide to model the problem as a graph where the nodes are the events and edges represent pairs of events where the team that you plan to send shares a member. In this setting, the graph theoretic question to be answered is:

(a) Find a maximum length simple cycle

(b) Find a maximum size independent set

(c) Find a maximum matching

(d) Find a maximal connected component

Answer: (b)

⊣

10. What does the following function compute in terms of n and d, for integer values of n and d, $d > 1$? Note that `a//b` denotes the quotient (integer part) of `a` $\div$ `b`, for integers `a` and `b`. For instance `7//3` is 2.

```
function foo(n,d){
  x := 0;
  while (n >= 1) {
    x := x+1;
    n := n//d;
  }
  return(x);
}
```

(a) The number of ways of choosing d elements from a set of size n.

(b) The number of ways of rearranging d elements from a set of size n.

(c) The number of digits in the base d representation of n.

(d) The number of ways of partitioning n elements into groups of size d.

Answer: (c)

The code computes $\lfloor \log_d(n) \rfloor + 1$. ⊣

Part B

1. Consider the following non-deterministic finite automata (NFA) $\mathcal{A}_1$ and $\mathcal{A}_2$:

(a) Give an example of a word which is accepted by both $\mathcal{A}_1$ and $\mathcal{A}_2$.

(b) Give an example of a word which is accepted by $\mathcal{A}_1$, but not by $\mathcal{A}_2$.

(c) Draw the deterministic finite automaton (DFA) equivalent to $\mathcal{A}_1$.

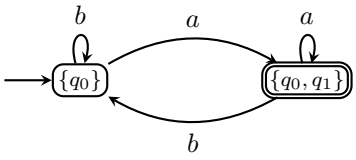

Figure 1: Question 1 (c)

Answer:

(a) ba

(b) a

(c) The equivalent DFA is given in Figure 1.

⊣

2. A student requests a recommendation letter from a professor. The professor gives three sealed envelopes. Each envelope contains either a good recommendation letter or a bad recommendation letter.

   (a) Make a list of all the possible scenarios.

   (b) Suppose now the professor tells the student that exactly one envelope contains a good recommendation letter and the other two contain bad recommendation letters. In the list of scenarios you prepared above, mark the ones that are still possible.

   (c) On envelope 1 is written the clue "This contains a bad recommendation letter". On envelope 2 is written the clue "This contains a bad recommendation letter". On envelope 3 is written the clue "Envelope 2 contains a good recommendation letter". Suppose now the professor gives the additional information that exactly one of these three clues are true and the other two are false. Can the student find out the contents of the envelopes without breaking their seals?

Answer:

(a) For $i \in \{1, 2, 3\}$, let g_i denote that letter i is good, and b_i denote that it is bad. Here are the possible scenarios:

1. b_1 b_2 b_3
2. b_1 b_2 g_3
3. b_1 g_2 b_3
4. b_1 g_2 g_3
5. g_1 b_2 b_3
6. g_1 b_2 g_3
7. g_1 g_2 b_3
8. g_1 g_2 g_3

(b) 2, 3 and 5.

(c) The professor says that exactly one of the three clues is true, and from part (b), we see that exactly one letter is good. Let clue i denote the clue on envelope i. Now it cannot be the case that both clue 1 and clue 2 are false, since then both envelope 1 and envelope 2 would contain good letters, contradicting part (b). Therefore one of clue 1 and clue 2 is true, and clue 3 is definitely false. Since clue 2 and clue 3 assert opposite facts, this means that clue 2 is true (and hence clue 1 is false). Thus we have that both g_1 and b_2 hold. Since exactly one letter is good, it follows that b_3 holds. Hence the exact scenario is $g_1\ b_2\ b_3$.

⊣

3. Let G be a simple graph on n vertices.

   (a) Prove that if G has more than $\binom{n-1}{2}$ edges then G is connected.

   (b) For every $n > 2$, find a graph G_n which has exactly n vertices and $\binom{n-1}{2}$ edges, and is not connected.

   Answer:

   (a) We show this by an induction on n. When $n = 1$, there is a single vertex in the graph, and hence the statement is vacuously true.

   Suppose the statement is true for all graphs with $\leqslant k$ vertices. Consider a graph with $k+1$ vertices which has more than $\binom{k}{2}$ edges. There can be no isolated vertex in such a graph, since the maximum number of edges possible in a graph of $k+1$ vertices that contains an isolated vertex is $\binom{k}{2}$. Furthermore, we can assume without loss of generality that there is a vertex of degree at most $k-1$, else the graph is complete and hence connected. Pick an arbitrary (non-isolated) vertex of degree at most $k-1$ and delete all edges incident on it. The remaining graph (on k vertices) has more than $\binom{k}{2} - (k-1) = \binom{k-1}{2}$ edges, and hence is connected by the inductive hypothesis. This implies that the graph on $k+1$ vertices is also connected.

   (b) For the second part, consider graphs consisting of an $n-1$-clique and an isolated vertex.

   ⊣

4. You are given a sorted array of n elements which has been *circularly shifted*. For example, $\{35, 42, 5, 12, 23, 26\}$ is a sorted array that has been circularly shifted by 2 positions.

   Give an $\mathcal{O}(\log n)$ time algorithm to find the largest element in a circularly shifted array. (The number of positions through which it has been shifted is unknown to you.)

   Answer: Let n be the number of elements in the array. If the array is not shifted at all, then the first element is smaller than the last element. Say the array is shifted by m positions, for $m > 0$. If $m < \frac{n}{2}$, then the first element would be larger than the middle element (and the largest element lies in the first half). If $m >= \frac{n}{2}$, then the

last element would be smaller than the middle element (and the largest element lies in the second half). This suggests a recursive algorithm which reduces the search space by half at each call, and hence gives a running time of $\mathcal{O}(\log n)$.

The following code describes this algorithm: `a/b` gives the integer part of dividing `a` by `b`. The required solution is obtained by calling `f(T, 0, n-1)`, where `n` is the length of `T`.

```
function f(T, first, last)
    if (T[first] <= T[last])
        return T[last]
    else {
        mid := first + (last - first)/2
        if (T[mid] > T[mid+1])
            return T[mid]
        else if (T[first] > T[mid])
            return f(T, first, mid-1)
        else if (T[mid] >= T[last])
            return f(T, mid+1, last)
    }
```

⊣

5. Let $G = (V, E)$ be an undirected graph and $V = \{1, 2, \ldots, n\}$. The input graph is given to you by a 0-1 matrix A of size $n \times n$ as follows. For any $1 \leq i, j \leq n$, the entry $A[i, j] = 1$ if and only if (i, j) is an edge in G.

   A connected component in G is a subgraph in which any two vertices are connected to each other by paths. Give a simple algorithm to find the number of connected components in G. Analyze the time taken by your procedure.

   Answer: We assume that there is a separate boolean array of size n that tells which vertex is marked. Initially all vertices are unmarked. Start at vertex 1 and perform a BFS (breadth-first search). The set of vertices marked during this procedure constitute the connected component that 1 belongs to. The time taken is $\mathcal{O}(n \times$ size of this connected component$)$. Now perform a linear search for the first unmarked vertex, and start a BFS with that as starting point. Once the BFS terminates, we search for the next unmarked vertex. We repeat this procedure till all vertices are marked. The number of times we start a BFS with a new vertex in the above procedure is the number of connected components.

   Each BFS takes time $\mathcal{O}(n \times$ size of that connected component$)$. Since the connected components partition V, and since we never run a BFS on each connected component more than once, the overall time taken by all the breadth-first searches is $\mathcal{O}(n^2)$. Since we search for a new unmarked vertex (to start a new BFS) at most nn times, and since each search takes at most n time, the overall time taken by the above procedure is still $\mathcal{O}(n^2)$. ⊣

6. You are playing an old-style video game in which you have to shoot down alien spaceships as they fly across the screen from left to right. Each spaceship flies across the

screen at a specified height. You have an antiaircraft gun set to shoot down all spaceships at a certain height. Spaceships fly one at a time, so if your gun is set to fire at the correct height, it will shoot down the spaceship currently flying across the screen.

You can set the initial height at which the gun fires. As the game progresses, you can reset the height, but only to a lower value. You are given in advance the height at which each spaceship flies. There are N spaceships numbered $1, 2, \ldots, N$ in the order in which they fly across the screen. For $1 \leq i \leq N$, $h[i]$ denotes the height at which spaceship i flies.

(a) Let $V[i]$ denote the maximum number of spaceships from $i, i+1, \ldots, N$ that you can shoot down with a single gun. Write a recurrence for $V[i]$ and describe a strategy to compute $V[i]$ using dynamic programming. What is the space and time complexity of your solution?

(b) Describe an algorithm to compute the minimum number of guns required to shoot down all the spaceships. Each gun can be initialized separately to a firing height and each gun can be separately reset to a lower value.

Answer:

(a) If we view $h[1], \ldots, h[N]$ as a list of numbers, we are essentially asked to find the length of the *longest descending subsequence*. The recurrence is as follows:

$$V[N] = 1$$
$$V[i] = 1 + \max\{V[j] \mid j > i \text{ and } h[i] \geqslant h[j]\}$$

The dynamic programming algorithm maintains an array V of size N, to be filled in from $V[N]$ to $V[1]$. Computing the value of $V[i]$ takes $\mathcal{O}(N)$ time, since it involves potentially examining all entries from $V[i+1]$ to $V[N]$. Thus the time taken overall is $\mathcal{O}(N^2)$. The space required is $\mathcal{O}(N)$.

(b) The above algorithm can be modified to produce not just the length of the longest descending subsequence, but the subsequence itself. Once we do this, we delete these entries from the h list and find the longest descending subsequence now. We repeat this process till the original h list has been decomposed into disjoint descending subsequences. Since each of these subsequences can be handled by one gun, the minimum number of guns required to shoot down all the spaceships is the number of disjoint subsequences computed above.

⊣

7. A First In First Out queue is a data structure supporting the operations *Enque*, *Deque*, *Print*. *Enque*(x) adds the item x to the tail of the queue. *Deque* removes the element at the head of the queue and returns its value. *Print* prints the head of the queue.

(a) You are given a queue containing 5 elements. Using a single additional temporary variable X, write down a sequence of statements each being one of *Enque*, *Deque*, *Print* so that the output of the program results in the 5 elements of the queue being printed in reverse order.

(b) If the queue had n elements to begin with, how many statements would you need to print the queue in reverse order?

Answer:

(a) Assume the queue contains 1,2,3,4,5 in order, with 1 at the head. We display below the sequence of instructions and the state of the queue whenever it changes.

Initial queue	12345
$X = Deque; Enque(X);$	23451
$X = Deque; Enque(X);$	34512
$X = Deque; Enque(X);$	45123
$X = Deque; Enque(X);$	51234
$Print;$	prints 5
$X = Deque; Enque(X);$	12345
$X = Deque; Enque(X);$	23451
$X = Deque; Enque(X);$	34512
$X = Deque; Enque(X);$	45123
$Print;$	prints 4
$X = Deque; Enque(X);$	51234
$X = Deque; Enque(X);$	12345
$X = Deque; Enque(X);$	23451
$X = Deque; Enque(X);$	34512
$Print;$	prints 3
$X = Deque; Enque(X);$	45123
$X = Deque; Enque(X);$	51234
$X = Deque; Enque(X);$	12345
$X = Deque; Enque(X);$	23451
$Print;$	prints 2
$X = Deque; Enque(X);$	34521
$X = Deque; Enque(X);$	45123
$X = Deque; Enque(X);$	51234
$X = Deque; Enque(X);$	12345
$Print;$	prints 1

As we can see the number of statements in the above program is 45.

(b) Each time we do an $X = Deque$ followed by an $Enque(X)$, we move the element at the head to the tail. Repeating these two statements $n-1$ times moves the last element to the head, while preserving the order among the rest of the list. Issuing a *Print* instruction now prints the current head of the list (which was originally the last element).

Thus we need $2(n-1)+1$ instructions to move the last element to the head and print it. At this point, the last element is what was originally the last but one element. If we now repeat the above block of $2(n-1)+1$ instructions, we print the second last element of the original queue, while bringing the last two elements of the original queue to the head.

Extending this logic, we see that if repeat the block of code n times, we end up printing the queue in reverse. The number of statements required is $n(2n-1)$.

$\dashv$