

# CHENNAI MATHEMATICAL INSTITUTE

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

Entrance Examination, 2021

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 the milkman doesn't deliver milk or the geyser doesn't work, then Akash will be late for school and lunch will be cooked late. Suppose lunch was actually cooked on time. Which of the following is definitely true?
- (a) Akash was late for school                      (b) Akash reached school in time  
(c) Geyser worked                                      (d) Milkman did not deliver milk

**Answer:** (c) Geyser worked

If either the milkman doesn't deliver milk or the geyser doesn't work, then (in particular) lunch will be cooked late. But lunch was actually cooked on time. So it follows that the milkman delivered on time and the geyser actually worked. So option (c) is definitely true, and option (d) is definitely false. Akash can either be a dutiful student and reach school on time, or be lazy and not go to school on time. So neither (a) nor (b) can be asserted for certain. -|

2. Let  $L$  be the language over  $\{a, b\}$  that contains the same number of occurrences of  $a$  and  $b$ . Which of the following languages is regular?
- (a)  $L \cap a^*b^*$                                       (b)  $(L \cap a^*b^*) \cup a^*b^*$   
(c)  $L \cup a^*b^*$                                       (d)  $(L \cap a^*b^*) \cup b^*a^*$

**Answer:** (b)  $(L \cap a^*b^*) \cup a^*b^*$

- (a)  $L \cap a^*b^* = \{a^n b^n \mid n \geq 0\}$  is not a regular language.  
(b)  $(L \cap a^*b^*) \cup a^*b^* = \{a^m b^n \mid m, n \geq 0\}$  is a regular language.  
(c) If  $L_3 = L \cup a^*b^*$  were regular, then  $L_3 \cap b^*a^*$  would also be regular. But  $L_3 \cap b^*a^*$  is  $\{b^n a^n \mid n \geq 0\}$ , which is not regular.  
(d) If  $L_4 = (L \cap a^*b^*) \cup b^*a^*$  were regular, then  $L_4 \cap a^*b^*$  would also be regular. But  $L_4 \cap a^*b^*$  is  $\{a^n b^n \mid n \geq 0\}$ , which is not regular. -|

3. Which of the following regular expressions represents binary strings that are multiples of 3? Note that we consider the leftmost bit to be the most significant.
- (a)  $((11)0^*)^*$                                       (b)  $(10^*)^*$   
(c)  $1(01^*)^*$                                       (d)  $(11 + 101^*01 + 0)^*$

**Answer:** (d)  $(11 + 101^*01 + 0)^*$

Let  $L(r)$  denote the language represented by the regular expression  $r$ .

- (a) It can be checked that  $L(((11)0^*)^*)$  does not contain the string 1001, which represents 9.
- (b)  $L((10^*)^*)$  contains 1, which represents a non-multiple of 3.
- (c)  $L(1(01^*)^*)$  contains 101 (representing 5), as also 1 (representing 1).
- (d) This is the only remaining option. One can prove it by first constructing a DFA for binary strings that are multiples of 3, and converting it to a regular expression.

–

4. Consider the following statements about finite simple graphs  $G$ :

- (i) If each vertex of a graph  $G$  has degree at least 2 then  $G$  contains a cycle as a subgraph.
- (ii) If the number of edges of a graph  $G$  is *at least as large* as the number of its vertices, then  $G$  contains a cycle as a subgraph.

Which of the above two statements holds for all graphs?

- (a) (i) only
- (b) (ii) only
- (c) both (i) and (ii)
- (d) neither of them

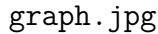
**Answer:**

(c) both (i) and (ii)

- (i) Let  $P = \langle v_1, v_2, \dots, v_t \rangle$  be a *maximal* path in  $G$ . Since  $v_t$  has degree at least 2 in  $G$ , there is an edge  $\{v_t, x\}$  in  $G$  which is different from the edge incident on  $v_t$  in the path  $P$ . Since  $P$  is a maximal path the vertex  $x$  must belong to  $P$ . So  $x = v_i$  for some  $1 \leq i < (t - 1)$ . The path  $\langle v_i, v_{(i+1)}, \dots, v_t \rangle$  together with the edge  $\{v_t, v_i\}$  forms a cycle in  $G$ .
- (ii) Suppose not, and let  $H$  be a graph with the *smallest* number of vertices which contradicts the statement. That is,  $H$  has at least as many edges as vertices, *and* does not contain a cycle. From the solution to part (a) above we get that there must be a vertex  $v$  in  $H$  whose degree is *at most* one. Deleting  $v$  from  $H$  results in a graph  $H'$  with (i) one fewer vertex, and (ii) at most one fewer edge, so  $H'$  also satisfies the premise of the statement. But then  $H'$  does not have any cycle, because  $H$  did not have any by assumption. Thus  $H'$  is a graph with *fewer* vertices than  $H$  for which the statement holds, which contradicts our assumption about  $H$ . Hence proved.

–

5. One day, Dumbledore assigns Harry Potter the task of obtaining the Philosopher's Stone that lies in an inner chamber surrounded by many rooms. To guide him along, he is given the Marauder's Map which contains the following graph.



Harry wishes to color every wall with colors such that walls that meet at a corner get different colors. Harry also wishes to color every room so that adjacent rooms get different colors. What is the minimum number of colors to accomplish this?

- (a) 4 colors for walls and 4 colors for the rooms
- (b) 3 colors for walls and 4 colors for the rooms
- (c) 2 colors for walls and 3 colors for the rooms
- (d) 2 colors for walls and 5 colors for the rooms

**Answer:**

(b) 3 colors for walls and 4 colors for the rooms

Consider the undirected graph  $G$  where every vertex corresponds to a wall and there is an edge between two vertices whenever two walls are incident on (or adjacent to) each other. Now, coloring for walls is the same as coloring the vertices of  $G$  such that no two adjacent vertices get the same color. This can be done with 3 colors and the graph  $G$  is often called *the line graph* (of the image given above).

Consider the undirected graph  $G'$  where every region (polygon in the above image) is a vertex and there is an edge between two vertices whenever the regions share a boundary. Now, coloring the rooms is the same as coloring the vertices of  $G'$  such that no two adjacent vertices get the same color. This can be done using 4 colors. Interestingly, note that you can draw  $G'$  on paper (or a plane) such that no two edges cross each other. Such a graph is said to be a *planar graph*. The question is a rephrasing of the *Planar 4-Color Theorem*.

→

6. In the chamber containing the Philosopher's stone, Harry sees a deck of 5 cards, each with a distinct number from 1 to 5. Harry removes two cards from the deck, one at a time. What is the probability that the two cards selected are such that the first card's number is exactly one more than the number on the second card?

- (a)  $1/5$
- (b)  $4/25$
- (c)  $1/4$
- (d)  $2/5$

**Answer:**

(a)  $1/5$

Letting  $(i, j)$  be the event that card  $i$  was drawn first and then card  $j$ , the sample space is  $\{(i, j) \mid i, j \in \{1, \dots, 5\}, i \neq j\}$ . Its size is 20. Of these, the set of desired outcomes is  $\{(5, 4), (4, 3), (3, 2), (2, 1)\}$ , and its size is 4. So the answer is  $1/5$ . →

7. You are given a sequence of letters  $x_1x_2 \dots x_n$  where each  $x_i \in \{a, b, c, d, e, f, g, h\}$ . You can form a new sequence from the given sequence as follows. Start with  $x_1$ . Place  $x_{m+1}$  either to the left or to the right of the sequence already built from  $x_1x_2 \dots x_m$ .

For instance, from  $dcbeb$  you can form  $ecdbb$  through the steps  $d \rightarrow cd \rightarrow cdb \rightarrow ecdb \rightarrow ecdbb$  and  $bbcde$  through the steps  $d \rightarrow cd \rightarrow bcd \rightarrow bcde \rightarrow bbcde$ .

What is the largest sequence in lexicographic (dictionary) order that you can form from the input sequence  $becgdfg$ ?

- (a)  $ggebcbdf$                       (b)  $ggfedcb$                       (c)  $ggebcbdf$                       (d)  $ggfebcd$

**Answer:**

(a)  $ggebcbdf$

We can generate  $ggebcbdf$  through the sequence  $b, eb, ebc, gebc, gebcd, gebcbdf, ggebcbdf$ . We cannot generate  $ggfedcb$ , since after the first  $g, d$  and  $f$  have to come to its right in the same order. We cannot generate  $ggebcbdf$ , because it is not possible to generate  $ecb$  from  $\{b, e, c\}$ . We cannot generate  $ggfebcd$ , since after the first  $g, d$  and  $f$  have to come to its right in the same order. ←

8. There is a basket full of apples, oranges, mangoes, pears and pomegranates. You want to pick three fruits from the basket. Multiple pieces of the same fruit can be picked. However, if you pick mangoes, you cannot pick apples.

In how many ways can you pick three fruits satisfying this condition?

- (a) 14                                      (b) 30                                      (c) 28                                      (d) 20

**Answer:**

(b) 30

There are the 35 different ways to pick three fruits from five kinds of fruits:

- all three fruits are of the same kind (5 ways).
- two are of one kind and the third is different ( $5 \times 4 = 20$  ways).
- each fruit is of a different kind ( $\binom{5}{3} = 10$  ways).

Of these, we should avoid the following 5 choices (letting  $M, A, P, G, O$  represent mango, apple, pear, pomegranata and orange respectively):

$MMA \quad MAA \quad MPA \quad MGA \quad MOA$

Thus the final answer is 30. ←

*The next two questions refer to the following procedure.*

The procedure operates on three arrays  $A[0..99]$ ,  $B[0..99]$  and  $C[0..99]$ , which are initialized with integer values.

```

procedure mystery() {
    for (i = 0; i < 100; i++) { C[i] = A[i]; }
    p = 99;
    for (i = 0; i < 100; i++) {
        B[p] = C[0];
        p = p-1;
        for (j = 1; j < 100; j++) {
            C[j-1] = C[j];
        }
    }
}

```

9. When the procedure terminates, which of the following statements can be asserted about the array B?
- (a) All elements of B are equal to A[0]
  - (b) B contains the elements of A sorted in descending order
  - (c) All values of B are the same
  - (d) B contains the elements of A in reverse order

**Answer:**

(d) B contains the elements of A in reverse order

We copy A into C to begin with. In each iteration of the outer for loop, we place the first element of C at position p of B. Since we decrement p in each iteration, we are essentially placing the elements in positions 99, 98, ..., 0 of B, in order. At each iteration, we shift C left by one place, so what was C[i+1] in the previous iteration is C[i] in the current iteration. So essentially we are copying A[0], A[1], ..., A[99] into B[99], B[98], ..., B[0]. ←

10. When the procedure terminates, which of the following statements can be asserted about the array C?
- (a) It contains the elements of A sorted in ascending order
  - (b) It contains the elements of A sorted in descending order
  - (c) All values are equal to A[99]
  - (d) All values are equal to A[0]

**Answer:**

(c) All values are equal to A[99]

As explained above, each iteration of the outer loop shifts C to the left by one place. At the end of iteration i, C[99], which is the same as A[99], is copied to positions 99-i, 99-i+1, ..., 99 of C. Since the outer loop is run for 100 iterations, the result follows. ←

## Part B

1. Let  $A = (\sqrt{3} + \sqrt{2})^{1000}$ ,  $B = (\sqrt{3} - \sqrt{2})^{1000}$ . Prove the following statements.

- (a)  $A + B$  is an integer.
- (b) The digit immediately after the decimal point in  $B$  is 0.
- (c) The digit immediately after the decimal point in  $A$  is 9.
- (d) Both  $A, B$  are irrational.

### Answer:

- (a)  $A + B$  is an integer, because for odd  $i$ , the  $\binom{1000}{i}$  terms in  $A$  and  $B$  cancel out, and for even  $i$ , the  $\binom{1000}{i}$  terms in both  $A$  and  $B$  are of the form  $\binom{1000}{i}(\sqrt{3})^i(\sqrt{2})^{1000-i}$ , which is an integer (since both  $i$  and  $1000 - i$  are even).
- (b)  $\sqrt{3} - \sqrt{2} < 0.318$ , and  $(\sqrt{3} - \sqrt{2})^6 < 0.0011$ . So the first two digits after the decimal in  $B$  are 0.
- (c) The above fact forces the first digit after the decimal in  $A$  to be 9, since  $A + B$  is an integer.
- (d)  $A$  and  $B$  are both irrational, since we can express each as a linear combination of an integer and  $\sqrt{6}$ , and  $\sqrt{6}$  is irrational.

–

2. Imagine you are playing a computer game that consists of different types of coins. You have the power to cast two magic spells  $s_1$  and  $s_2$ . Each spell consumes some number of coins of each type and produces some number of coins of each type. Spell  $s_1$  consumes one coin of type  $t_1$  and produces two coins of type  $t_2$ , written in short as  $(-1 t_1, +2 t_2)$ . Spell  $s_2$  is  $(-1 t_2, +1 t_1)$ . You are allowed to cast the two spells any number of times, but a spell cannot be cast if there aren't enough coins present for consumption. For example,  $s_2$  cannot be cast if there are 0 coins of type  $t_2$ .

- (a) Suppose you start with  $n$  coins of type  $t_1$  and 0 coins of type  $t_2$ . If you repeatedly cast spell  $s_1$  till it can no longer be cast, what is the total number of coins (of both types) at the end?
- (b) Suppose you start with  $n$  coins of type  $t_1$  and 0 coins of type  $t_2$ . You are allowed to cast both the spells  $s_1$  and  $s_2$ . Give a sequence of spells that will lead you to  $4n$  total coins. Can you extend the sequence to produce more than  $4n$  coins?
- (c) Suppose we provide you with a third type of coin  $t_3$ , and you start with  $n$  coins of type  $t_1$  and 0 coins of types  $t_2$  and  $t_3$ . Come up with a new spell  $s_3$  satisfying the following three properties:
  - Spell  $s_3$  can add or remove at most two coins of any type.
  - Using spells  $s_1$  and  $s_3$  repeatedly in some order, you can reach  $4n$  total coins.
  - There is no sequence of spells using only  $s_1$  and  $s_3$  that can produce more than  $4n$  total coins.

**Answer:**

- (a) A configuration is given by a pair of natural numbers  $(i, j)$ , where  $i$  is the number of coins of type  $t_1$  and  $j$ , the number of coins of type  $t_2$ . Starting from  $(n, 0)$  and repeatedly casting the spell  $s_1$ , we get the following sequence of configurations:

$$(n, 0) \ (n-1, 2) \ (n-2, 4) \ \cdots \ (n-i, 2i) \ \cdots \ (2, 2(n-2)) \ (1, 2(n-1)) \ (0, 2n)$$

At this point we can no longer cast  $s_1$ , since it requires a non-zero number of coins of type  $t_1$ . So we end up with  $2n$  coins at the end.

- (b) We use  $(i, j) \xrightarrow{s} (i', j')$  to denote that applying spell  $s$  in configuration  $(i, j)$  results in configuration  $(i', j')$ . Observe that we can go from  $(i, j)$  to  $(i, j+4)$  as follows:

$$(i, j) \xrightarrow{s_1} (i-1, j+2) \xrightarrow{s_2} (i, j+1) \xrightarrow{s_1} (i-1, j+3) \xrightarrow{s_2} (i, j+2) \xrightarrow{s_1} (i-1, j+4)$$

Starting from  $(n, 0)$  and repeating the above sequence  $n$  times, we reach  $(0, 4n)$ . We can extend this sequence to produce more coins. For example, we can cast  $s_2$  repeatedly to reach  $(n, 3n)$  and repeat the above sequence  $n$  times to reach  $(0, 7n)$ , and so on. In fact, we can go on forever if allowed to use both rules.

- (c) The new spell  $s_3$  is  $(-1 t_2, +2 t_3)$ . If we start with configuration  $(i, j, k)$  and cast  $s_1$  followed by  $s_3$ , we arrive at  $(i-1, j, k+2)$ . Starting from  $(n, 0, 0)$  and repeating this  $n$  times, we get  $(0, 0, 4n)$ . If we start from  $(i, 0, k)$  and apply a sequence of spells using only  $s_1$  and  $s_3$ , the number of times  $s_3$  is cast is at most the number of times  $s_1$  is cast. (Since  $s_3$  consumes one coin of type  $t_2$  and  $s_1$  produces one coin of type  $t_2$ .) Also the number of times  $s_1$  is cast is at most  $i$ , since  $s_1$  consumes one coin of type  $t_1$  and  $s_3$  does not add any coin of type  $t_1$ . Thus, starting from  $(n, 0, 0)$ , a longest sequence consisting of  $s_1$  and  $s_3$  will have  $n$  applications of  $s_1$  and  $n$  applications of  $s_3$ . We cannot end with more than  $4n$  coins this way.

–

3. A *cut edge* of a connected graph  $G$  is any edge  $e = \{u, v\}$  of  $G$  such that the graph  $G - e$  obtained by deleting edge  $e$  (and *not* deleting  $u$  or  $v$ ) is disconnected. In fact,  $G - e$  will have exactly two connected components. Let  $G$  be an arbitrary finite simple connected graph in which every vertex has an **odd** degree, and let  $e$  be an arbitrary cut edge of  $G$ . Let  $H_1, H_2$  be the two connected components of  $G - e$ .

Prove or disprove the following statement: Each of  $H_1, H_2$  must necessarily have an **odd number** of vertices.

To disprove this statement it suffices to exhibit a graph which contradicts the statement. A proof must take the form of an argument.

**Answer:**

The sum of the degrees of any graph is twice the number of its edges. So the number of vertices of odd degree in any graph is even. It follows that  $G$  has an **even number** of vertices. So the counts of vertices of  $H_1, H_2$  are either both odd, or both even. Each of  $H_1, H_2$  has exactly one vertex of even degree: this is the erstwhile end-point of the edge  $e$  which ends up in that graph. If the vertex count of  $H_1$  is even, then this means that  $H_1$  has an odd number of vertices of odd degree, which is impossible. Similarly for  $H_2$ . So each of  $H_1, H_2$  has an odd number of vertices. –

4. For a language  $L$  over an alphabet  $\Sigma$ , define

$$\text{SW}(L) := \{ y \in \Sigma^* \mid \exists x \in \Sigma^* \text{ s.t. } xyx \in L \}$$

Prove that if  $L$  is regular,  $\text{SW}(L)$  is also regular.

**Answer:**

Suppose  $L$  is regular, and let  $A$  be an automaton recognizing  $L$ . Nondeterministically pick a state  $q$ . Run  $A$  on a guessed string  $x$  from the start state  $s$  as well as from  $q$ . At any point if the simulation from  $q$  reaches a final state and the first component is at some state  $p$  you may nondeterministically move to a phase which checks if the input word  $y$  can take you from  $p$  to  $q$ .

Formally, if  $A$  is given by  $(Q, s, \rightarrow, F)$ , with  $t \notin Q$ , we define  $B$  (an NFA with  $\varepsilon$ -transitions) recognizing  $\text{SW}(L)$  to be  $B = (Q', s', \Rightarrow, F')$  where:

- $Q' = \{(p, q, r) \mid p \in S \cup \{t\}, q, r \in S\}$
- $s' = \{(t, q, q) \mid q \in S\}$
- $(p, q, r) \xRightarrow{a} (p', q', r')$  iff one of the following conditions holds:
  - $p = p' = t, q = q'$  and  $r \xrightarrow{a} r'$  (this is the phase that runs on  $y$ );
  - $p = t, p' = q, q' = s, r' = r$  and  $a = \varepsilon$  (this is the phase change);
  - $p \neq t, p' = p, a = \varepsilon$  and for some  $b \in \Sigma, q \xrightarrow{b} q'$  and  $r \xrightarrow{b} r'$  (this is the phase that checks that there are appropriate runs on a guessed  $x$ ).
- $F' = \{(q, q, f) \mid q \in S, f \in F\}$ .

Any accepting run of  $B$  on  $y$  is of the following form:

$$(t, p, p) \xRightarrow{y} (t, p, q) \xRightarrow{\varepsilon} (p, s, q) \xRightarrow{*} (p, p, f).$$

Furthermore, the run before the phase shift is witnessed by an  $A$ -run on  $y$  (from  $p$  to  $q$ ), and the run after the phase shift is witnessed by two  $A$ -runs on the same string (say  $x$ ), one from  $s$  to  $p$ , and another from  $q$  to  $f$ . Thus there is an accepting  $B$ -run on  $y$  iff there is an accepting  $A$ -run on  $xyx$ , for some  $x$ .  $\dashv$

5. Given a list  $A$  of  $N$  elements and a number  $K$ , your task is to output the  $N - K + 1$  values listing the minimum among  $A[i] \cdots A[i + K - 1]$  for every  $1 \leq i \leq N - K + 1$ . Provide an algorithm for this task, that runs in time at most  $\mathcal{O}(N \log K)$ .

**Answer:**

Let  $M[i][j]$  denote the minimum among  $A[i] \cdots A[i + 2^j - 1]$ , for  $j \leq K$  and  $1 \leq i \leq N - 2^j + 1$ . Our aim is to compute  $M[i][K]$  for  $1 \leq i \leq N - K + 1$ . Observe that  $M[i][j + 1] = \min(M[i][j], M[i + 2^j][j])$ . This suggests the following dynamic programming algorithm.

```

for (i = 1; i <= N; i++) {
    M[i][0] = A[i];
}
s = 1; l = log(K);

```



```

for (j = 1; j <= 1; j++) {
    for (i = 1; i <= N-2*s+1; i++) {
        M[i][j] = min(M[i][j-1], M[i+s][j-1]);
    }
    s = 2*s;
}
output M[1][log(K)], M[2][log(K)], ..., M[N-K+1][log(K)];

```

The inner loop runs in time  $\mathcal{O}(N)$ , and there are  $\log K$  iterations of the outer loop. So we have the desired running time.  $\dashv$

6. Let  $A = A[1]A[2] \cdots A[n]$  be an array of  $n$  numbers where  $n = 2^k - 1$  for some  $k > 0$ . Consider a binary tree  $T$  built from the array in the following manner. The root of  $T$  is the middle element of  $A$ . Let  $A_L$  and  $A_R$  denote the left and right subarrays resulting from the removal of the middle element from  $A$ . The left and right subtrees of the root node are binary trees defined similarly on  $A_L$  and  $A_R$ , respectively.

Each element  $A[i]$  of the array is a node in this tree  $T$ . Let  $S_i$  denote the indices occurring in the unique path from the root node to the element  $A[i]$ . We say that index  $i$  is *good* if for every  $j_1, j_2$  in  $S_i$  with  $j_1 < j_2$ , we have  $A[j_1] \leq A[j_2]$ .

Prove that the subarray  $A[i_1]A[i_2] \cdots A[i_m]$  consisting only of good indices is sorted.

**Answer:**

Consider any two good indices  $i$  and  $j$ , with  $i < j$ . Let  $A[k]$  be the lowest common ancestor of  $A[i]$  and  $A[j]$  in the tree  $T$  (it is possible that  $k \in \{i, j\}$ ). Clearly  $i \leq k \leq j$ . Now  $i, k \in S_i$  and  $j, k \in S_j$ , and both  $i$  and  $j$  are good. Therefore  $A[i] \leq A[k]$  and  $A[k] \leq A[j]$ , and hence  $A[i] \leq A[j]$ . The desired result follows.  $\dashv$

7. Consider the functions `foo` and `bar` described in pseudocode below, where `//` denotes quotient (integer division) and `%` denotes remainder.

```

int foo(int n) {
    int i = 1;
    while (bar(i) < n) {
        i = 2*i;
    }
    return(i);
}

int bar(int n) {
    if (n == 0) {
        return(1);
    }
    int x = bar(n // 2);
    if (n % 2 == 0) {
        return(x*x);
    } else {
        return(2*x*x);
    }
}

```

```
    }  
}
```

- (a) What function does `bar(n)` compute? Justify your answer.
- (b) If `foo(n)` computes `x`, how do `n` and `x` relate to each other. Justify your answer.
- (c) How many recursive calls to `bar` are made in a computation of `foo(100)`?

**Answer:**

- (a) `bar(n)` computes  $2^n$ . This can be shown easily by induction. The base case, when  $n = 0$ , returns 1. Otherwise, if  $n = 2m$ , we compute  $2^m \cdot 2^m = (2^m)^2 = 2^{2m} = 2^n$ , and if  $n = 2m + 1$ , we compute  $2 \cdot 2^m \cdot 2^m = 2(2^m)^2 = 2^{2m+1} = 2^n$ .
- (b) In `foo(n)`, we are repeatedly computing  $2^1, 2^2, 2^4, 2^8, \dots$  till we reach  $x$  such that  $2^x \geq n$ . So the `x` computed by `foo(n)` is the smallest number  $m$  such that  $m$  is a power of 2 and  $2^m \geq n$ .
- (c) In `foo(100)`, there are recursive calls to `bar(1)`, `bar(2)`, `bar(4)` and `bar(8)`. We stop with this since  $2^8 \geq 100$ . Now, `bar(2i)` has  $i + 1$  further calls to `bar`. So overall we have a total of  $2 + 3 + 4 + 5 = 14$  calls to `bar`.

–