CHENNAI MATHEMATICAL INSTITUTE

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

Entrance Examination, 24 May 2025

Instructions

- This question paper consists of two parts. There are 10 Part A questions worth 3 marks each, and 7 Part B questions worth 10 marks each. The total marks are 100.
- Part A consists of multiple choice questions. There may be multiple correct choices. You have to select all the correct options and no incorrect option to get full marks. **There is no partial credit, and no negative marking.**
- For questions in Part A, you have to provide the answers on the computer. You only have to choose the appropriate answer(s) from the choices provided. For example, if the answer to a question is options (a) and (c), choose only (a) and (c) to score full marks for that question.
- For questions in Part B, you have to write your answer with a short explanation in the space provided for the question.
- In all questions related to graphs, unless otherwise specified, we use the word "graph" to mean a finite undirected graph with no self-loops, and at most one edge between any pair of vertices.

Part A

1. Consider the automaton in Figure 1, where the start state is q_0, and the final states (indicated by double circles) are q_0, q_3 and q_5. Which of the following strings are accepted by it?

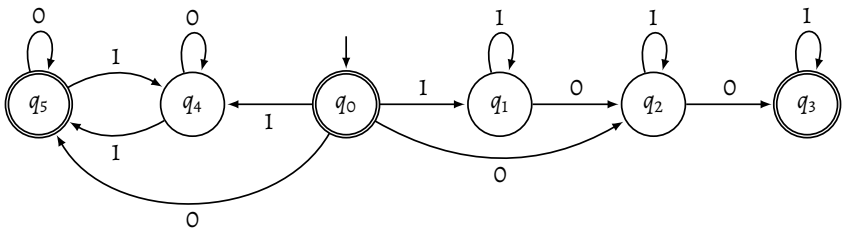


Figure 1: Automaton for Question A.1.

(a) 00111 (b) 00110110 (c) 011010 (d) ε

Answer: (a), (b) and (d).

The automaton above accepts all strings that consist of an even number of 1s, or exactly two 0s. Hence (a), (b) and (d) are accepted, while (c) is not. ⊣

2. Which of the following languages over $\{a, b\}$ can be accepted using a DFA with a *single* accepting state?

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

Answer: (b) and (d).

(a) cannot be accepted by an automaton with a single final state. For if that were the case, then a and b would end up in the same final state (call it q_f). Let $q_f \xrightarrow{a} q$. If $q_f = q$, then both aa and ba are both accepted. If $q_f \neq q$, then neither aa nor ba is accepted. But then the language accepted is not $a^* + b^*$.

If (c) were accepted by an automaton with a single final state, then ab and ba go to same final state. Similar to the reasoning in the above paragraph, consider the strings aba and baa. ⊣

3. Aruna always eats paranthas or curd rice at lunch if pizza is not available. She eats curd rice only if pickles are available. Yesterday, Aruna's mother had fresh pickles ready at lunch time, and also banned pizza delivery at home. What can you say about Aruna's lunch yesterday?

   (a) Aruna ate curd rice.

   (b) Aruna did not eat curd rice.

   (c) Aruna ate paranthas.

   (d) Aruna ate both curd rice and paranthas.

   Answer: (a), (b), (c) and (d).

   Suppose the following letters stand for the accompanying statements.

   - P: Aruna eats paranthas.
   - C: Aruna eats curd rice.
   - Z: Pizza is available.
   - K: Pickles are available.

   Then Aruna's approach to life is represented by the formulas $\neg Z \rightarrow (P \vee C)$ and $C \rightarrow K$. Her mother's actions led to the formulas K and $\neg Z$ being true yesterday. It can be seen that $\neg P \wedge C$, $P \wedge \neg C$ and $P \wedge C$ are all consistent with the four formulas above. Thus all four choices are possible. ⊣

4. Consider a biased coin which has probability p of turning up heads (and $1 - p$ of turning up tails), and consider an experiment where we toss the coin repeatedly. What is the *expected number of tosses before seeing a head*? (Recall that this is given by the formula $\sum_{k=1}^{\infty} kp_k$, where p_k is the probability that the coin turns up heads for the first time on the k-th toss.)

   (a) $1/p^2$ (b) $1/(1-p)^2$ (c) $1/p$ (d) $1/(1-p)$

   Answer: (c).

   If the coin ends up heads for the first time on the k-th toss, it means that there are $k - 1$ tails followed by a head. Thus $p_k = (1-p)^{k-1}p$. Now letting $S = \sum_{k=1}^{\infty} kp_k$, we have that $S = p\sum_{k=1}^{\infty}(1-p)^{k-1}$. The summation is standard, and evaluates to $1/p^2$. So $S = 1/p$. ⊣

5. In a finite directed graph, every vertex has exactly three incoming edges. Which of the following statements is guaranteed to be true?

   (a) Some vertex has at least three edges leaving it.

   (b) Exactly three edges leave every vertex.

   (c) Some vertex has exactly three edges leaving it.

   (d) None of the above is true.

   Answer: (a).

   Suppose the number of vertices is n. Since every vertex has exactly three incoming edges, the number of edges is $3n$. If each vertex had at most two outgoing vertices, then the number of edges would have to be $\leqslant 2n$. Thus at least one vertex has three edges leaving it.

   (b) and (c) are not always true. Consider the graph $G = (\{a, b, c, d\}, E)$ with

   $$E = \{(a,a),(a,b),(a,c),(a,d),(b,a),(b,b),(b,c),(b,d),(c,a),(c,d),(d,b),(d,c)\}.$$

   All vertices have exactly three incoming edges, but no vertex has exactly three outgoing edges. ⊣

6. Let X be a set of size n. What is the size of the set $\{(A, B) \mid A, B \subseteq X, A \cap B = \emptyset\}$?

   (a) $3^n - 2^n$ (b) 3^n (c) 4^n (d) 2^{n+1}

   Answer: (b).

   Let $\mathscr{P} = \{(A, B) \mid A, B \subseteq X \text{ and } A \cap B = \emptyset\}$. For each $(A, B) \in \mathscr{P}$, we can associate a function $f_{AB} : X \to \{0, 1, 2\}$ defined as follows:

   $$f_{AB}(x) = \begin{cases} 0 & \text{if } x \notin A \cup B \\ 1 & \text{if } x \in A \setminus B \\ 2 & \text{if } x \in B \setminus A \end{cases}$$

   Conversely, given any function $f : X \to \{0, 1, 2\}$, we see that $(A_f, B_f) \in \mathscr{P}$, where $A_f = \{x \in X \mid f(x) = 1\}$ and $B_f = \{x \in X \mid f(x) = 2\}$.

   Thus the set $\mathscr{P}$ is in one-to-one correspondence with the set of all functions from X to $\{0, 1, 2\}$. Therefore its size is 3^n. ⊣

7. Rohit Sharma is facing Shardul Thakur's bowling in the IPL. 40% of Shardul's deliveries to Rohit are good length balls, 40% are short pitched, and 20% are overpitched. Rohit hits a boundary 40% of the time off good length balls, and 80% of the time off short pitched and overpitched balls. Given that Rohit has hit a boundary off Shardul's last ball, what is the probability that it was short-pitched?

   (a) 1/4 (b) 1/3 (c) 1/5 (d) 1/2

Answer: (d).

If Shardul bowls 100 balls to Rohit, 40 are length balls, 40 are short-pitched and 20 are over-pitched. Rohit hits 16 boundaries off the length balls, 32 boundaries off the short-pitched balls, and 16 boundaries off the overpitched balls. Thus he hits 64 boundaries off the 100 balls, and 32 of these are off short-pitched balls. So Pr(short-pitched | boundary) = 32/64 = 1/2. ⊣

The next two questions pertain to the following code which takes a non-negative integer as input.

```
function FOO(n)
    if (n = 0) then
        return 0
    else if (n = 1) then
        return 1
    else if (n = 2) then
        return 3
    else
        return n + FOO(n − 1) + FOO(n − 2)
    end if
end function
```

8. What is the value returned by FOO(5)?

(a) 10 (b) 14 (c) 26 (d) 35

Answer: (c).

From the code we get: FOO(0) = 0, FOO(1) = 1, FOO(2) = 3. Using these we get: FOO(3) = 3 + 3 + 1 = 7, FOO(4) = 4 + 7 + 3 = 14, and FOO(5) = 5 + 14 + 7 = 26. ⊣

9. Which of the following best describes the running time of FOO(m)?

(a) Linear in m.

(b) Quadratic in m.

(c) Cubic in m.

(d) Exponential in m.

Answer: (d).

Recall that the Fibonacci numbers are defined as: $F_0 = 0$, $F_1 = 1$ and $F_n = F_{n-1} + F_{n-2}$ for $n \geqslant 2$. Now let G_n be the number of recursive calls (including the top-level call) made to FOO(2) in the computation of FOO(n). Then we see that $G_0 = G_1 = 0 = F_0, G_2 = 1 = F_1$, and for $n \geqslant 3, G_n = G_{n-1} + G_{n-2} = F_{n-2} + F_{n-3} = F_{n-1}$. Thus the running time of FOO(m) is at least the value of F_{m-1}, which is exponential in m. ⊣

10. Consider the following two procedures `proc1` and `proc2`, which run in parallel after initialising `x` to `0`. Running in parallel means that between any two lines of code in one process, any number of lines of the other process may run. Note that `x` is a shared variable which both processes can read/modify, while `temp1` and `temp2` are local variables that only the corresponding processes can access.

```
int x = 0;

proc1 {                          proc2 {
    int temp1 = x;                   int temp2 = x;
    x = temp1 + 1;                   x = temp2 + 1;
    temp1 = x;                   }
    x = temp1 + 1;
}
```

Which of the following values are possible for `x` after both processes have come to a halt?

(a) 0 (b) 1 (c) 2 (d) 3

Answer: (b), (c) and (d).

Clearly the final value cannot be 0 since at least one of the processes will add at least 1 to `x`.

Use L.1, L.2, L.3, L.4 to number the statements of `proc1`, and R.1, R.2 to number the statements of `proc2`. Here are the execution traces that achieve the final values specified by (b), (c) and (d).

(b) R.1, L.1, L.2, L.3, L.4, R.2.

(c) L.1, R.1, R.2, L.2, L.3, L.4.

(d) L.1, L.2, L.3, L.4, R.1, R.2.

⊣

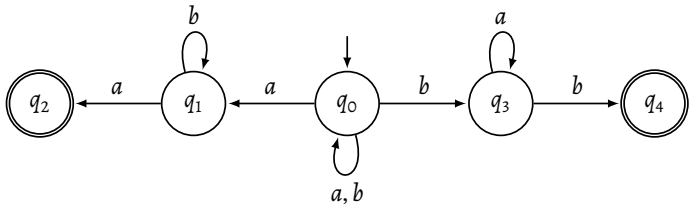
Part B

1. Consider the language

$$L = \{x \in \{a, b\}^* \mid x \text{ ends with a palindrome of length at least 2}\}.$$

Is L regular or not? Justify your answer by either constructing a finite automaton, or by giving a proof for the non-regularity of L.

Answer: L is regular. Let $\Sigma = \{a, b\}$. Any nonempty string $x \in \Sigma^*$ either ends with an a or ends with a b. In the former case, it is in L iff it is of the form $yab^n a$ for some $y \in \Sigma^*$ and $n \geqslant 0$, and in the latter case, it is in L iff it is of the form $yba^n b$ for some $y \in \Sigma^*$ and $n \geqslant 0$. Presented below is a nondeterministic automaton whose initial state is q_0 and accepting states are q_2 and q_4. It stays in the initial state till the point where it guesses that it has read the prefix y, and then guesses if it is in the first or the second case discussed above, according to which it moves left or right, respectively.



⊣

2. (a) Construct a finite automaton for the language L consisting of all binary strings with equal number of occurrences of 01 and 10.

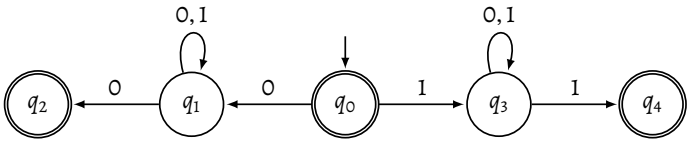
   (b) Consider the language $L = \{1^x0^y1^z \mid x > 0,\ y \geqslant 0,\ z > 0\}$ over the alphabet $\Sigma = \{0, 1\}$. Construct a 3-state NFA for L and prove that it is correct.

Answer:

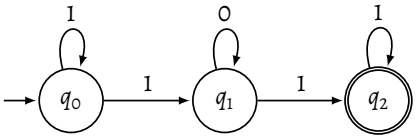
(a) Any string x over $\{0, 1\}$ falls in one of the following categories:

- $x = \varepsilon$ or $x = 0^n$ or $x = 1^n$ for some $n > 0$. Then x has zero occurrences of 01 and 10, so $x \in L$.
- $x = 0^{n_1}1^{n_2} \ldots 0^{n_{2k-1}}1^{n_{2k}}$ for some $k > 0$, with $n_i > 0$ whenever $0 \leqslant i \leqslant 2k$. Then x has k occurrences of 01 and $k - 1$ occurrences of 10, so $x \notin L$.
- $x = 1^{n_1}0^{n_2} \ldots 1^{n_{2k-1}}0^{n_{2k}}$ for some $k > 0$, with $n_i > 0$ whenever $0 \leqslant i \leqslant 2k$. Then x has $k - 1$ occurrences of 01 and k occurrences of 10, so $x \notin L$.
- $x = 0^{n_1}1^{n_2} \ldots 0^{n_{2k-1}}$ or $x = 1^{n_1}0^{n_2} \ldots 1^{n_{2k-1}}$ for some $k > 0$, with $n_i > 0$ whenever $0 \leqslant i \leqslant 2k - 1$. Then x has $k - 1$ occurrences of 01 and $k - 1$ occurrences of 10, so $x \in L$.

Thus, $x \in L$ iff $x = \varepsilon$ or if x begins and ends with the same letter. Presented below is a nondeterministic automaton for L.



(b) Presented below is a nondeterministic automaton whose initial state is q_0 and accepting state is q_2.



Letting δ be the nondeterministic transition function, we prove by induction on length of strings the following facts in order.

(a) $\delta^*(q_0, u) = \{q_0, q_1\}$ iff $u = 1^x$ for some $x > 0$.

(b) $\delta^*(q_1, v) = \{q_1\}$ iff $v = 0^y$ for some $y \geqslant 0$.

(c) $\delta^*(q_1, w) = \{q_2\}$ iff $w = 0^y1^z$ for some $y \geqslant 0$ and $z > 0$.

Combining (a) and (c) above, along with the fact that $\delta(q_0, 0) = \varnothing$, it follows that $\delta^*(q_0, s) = \{q_0, q_2\}$ iff $s = 1^x0^y1^z$ for some $x, z > 0$ and $y = 0$, while $\delta^*(q_0, s) = \{q_2\}$ iff $s = 1^x0^y1^z$ for some $x, y, z > 0$. In any case, $q_2 \in \delta^*(q_0, s)$ iff $s = 1^x0^y1^z$, for some $x, z > 0$ and $y \geqslant 0$.

⊣

3. (a) Let X, Y be finite sets. Show that a function $f : X \to Y$ is a bijection if and only if $f(X \setminus A) = Y \setminus f(A)$ for every subset A of X.

   (b) Let S be a nonempty set and P the set of all subsets of S. Let $f : P \to P$ be a function satisfying the following property: if $X \subseteq Y$, then $f(X) \subseteq f(Y)$. Show that there exists some subset T of S such that $f(T) = T$.

Answer:

(a) ($\Rightarrow$): Suppose f is a bijection and $A \subseteq X$. Since f is one-to-one, $f(a) \neq f(b)$ for any $a \in A$ and $b \notin A$. Thus $f(A) \cap f(X \setminus A) = \varnothing$ (†). Since f is onto, for any $y \in Y$, there is x such that $f(x) = y$. If $x \in A$ then $y \in f(A)$, and if $x \notin A$ then $y \in f(X \setminus A)$. Thus $f(A) \cup f(X \setminus A) = Y$ (‡). Combining (†) and (‡), we have that $f(A) = Y \setminus f(X \setminus A)$, or in other words, $f(X \setminus A) = Y \setminus f(A)$.

($\Leftarrow$): Suppose f is not a bijection, which means that it is either not one-to-one, or not onto. If f is not one-to-one, let $u, v \in X$ such that $u \neq v$ and $f(u) = f(v)$. Consider the set $A = \{u\} \subseteq X$. Now $f(u) \in f(A)$, but also $f(u) = f(v) \in f(X \setminus A)$. Thus $f(A) \neq Y \setminus f(X \setminus A)$. If, on the other hand, f is not onto, let $u' \notin f(X)$. Taking $A = X$, we have $u' \notin f(A)$ but $u' \in Y = Y \setminus \varnothing = Y \setminus f(\varnothing) = Y \setminus f(X \setminus A)$. Thus, once again, we have that $f(A) \neq Y \setminus f(X \setminus A)$.
In sum, if $f(X \setminus A) = Y \setminus f(A)$ for every subset A of X, then f is bijective.

(b) A function f is called *monotone* if it satisfies the property that $f(X) \subseteq f(Y)$ whenever $X \subseteq Y$.

- Here is an argument that works when S is finite.
  Define a sequence $S_0, S_1, \ldots$ as follows:

$$S_0 = \varnothing$$
$$S_{i+1} = f(S_i), \text{ for } i \geqslant 0.$$

  Clearly $S_0 = \varnothing \subseteq S_1$. If we assume that $S_i \subseteq S_{i+1}$ for some i, then, by monotonicity of f, $S_{i+1} = f(S_i) \subseteq f(S_{i+1}) = S_{i+2}$. Thus we see that $S_0, S_1, \ldots$ is a non-decreasing sequence of subsets of S. Let S have n elements. Then there cannot be more than $n+1$ distinct S_i-s. This means that $S_k = S_{k+1} = f(S_k)$ for some $k \geqslant 0$. We take that S_k to be our T.
- Here is a different argument which is much slicker, and which works for all sets S, whether finite or infinite.
  Consider the collection $\mathscr{C} = \{T \subseteq S \mid f(T) \subseteq T\}$. This collection is nonempty – since $f(S) \subseteq S$, we have $S \in \mathscr{C}$. Letting $T_0 = \bigcap_{T \in \mathscr{C}} T$, we have $T_0 \subseteq T$ for all $T \in \mathscr{C}$. By monotonicity of f, it follows that $f(T_0) \subseteq f(T) \subseteq T$ for each $T \in \mathscr{C}$. We therefore have that $f(T_0) \subseteq \bigcap_{T \in \mathscr{C}} T = T_0$. By monotonicity of f again, $f(f(T_0)) \subseteq f(T_0)$, and thus $f(T_0) \in \mathscr{C}$. Thus, by definition of T_0, we have $T_0 \subseteq f(T_0)$. We conclude that there is a $T \subseteq S$ such that $f(T) = T$.

⊣

4. Seventeen students take part in a tournament where each student plays against every other student exactly once. In each contest, the pair of students can choose to play one of three games – Chess, Go or Hex. Prove that there are three students that play the same game among themselves (i.e., each of the three contests involving these players is a game of Chess, or each is a game of Go, or each is a game of Hex).

   Hint: Use the pigeonhole principle.

   Answer: We say that a *solution group* is a subset of three students who all play the same game against each other, which we call their *solution game*.

   Consider an arbitrary student A. She plays all the 16 others, so there is at least one game, say Chess, she plays against 6 others (for otherwise, she would play each game against at most 5 people, which only adds up to 15 opponents at most).

   Suppose now that these six students play only Go and Hex against each other. Pick an arbitrary member of this group, say B. Now B has played five games against members of this group, so must have played the same game, say Go, against three others (let us name them C_1, C_2, C_3). If C_1, C_2 and C_3 only play Hex against each other, then $\{C_1, C_2, C_3\}$ is a solution group with Hex as the solution game. Otherwise, two of these, say C_1 and C_2, play Go against each other. But now we have $\{B, C_1, C_2\}$ as a solution group, with Go as the solution game.

   Otherwise at least two students out of the six, say B_1 and B_2, play Chess against each other. In this case $\{A, B_1, B_2\}$ is a solution group, with Chess as the solution game. ⊣

5. Let G be a connected graph on n vertices, with no multiple edges or self-loops. Let $deg(v)$ denote the degree of vertex v in G. Let s, t be two vertices in G, let P be a shortest path from s to t in G, and let $V(P)$ be the set of vertices in the path P. Prove that

$$\sum_{v \in V(P)} \deg(v) \leqslant 3n.$$

Answer: If P has at most three vertices then the claim is easily seen to be true, since each vertex has degree at most $(n-1)$. So let us consider the case when P has four or more vertices, and let these vertices be $\{v_1, v_2, \dots, v_t\}$. Consider the partition of $V(P)$ into the three sets $X = \{v_1, v_4, v_7, \dots\}, Y = \{v_2, v_5, v_8, \dots\}, Z = \{v_3, v_6, v_9, \dots\}$. No two vertices in X can have a common neighbour in G, since this would imply a strictly shorter s-t path than P. So the neighbourhoods of the vertices in X form a partition of some subset of the vertex set of G. It follows that the sum of the degrees in G of all the vertices in X is at most n. Repeating this argument for Y and Z completes the proof. ⊣

6. We have an array A of n numbers, where n is a power of 2.

   We build a full binary tree on top of the array A. The elements of the array are leaves of the tree, numbered $1, 2, \ldots, n$, from left to right. At each internal node of the tree, we store the sum of the values of the array elements in the subtree rooted under it.

   We wish to use this data structure to support operations $psum(i), 1 \leqslant i \leqslant n$, and $add(y, i), 1 \leqslant i \leqslant n$, defined below.

$$psum(i) : \text{return the value of } \sum_{j=1}^{j=i} A[j]$$

$$add(y, i) : \text{update } A[i] \text{ to } A[i] + y$$

   (a) Draw the data structure built on top of the array $A = [-2, 1, 4, -3, 6, 7, 9, -5]$.

   (b) How will you update the data structure when you perform $add(y, i)$? Assuming arithmetic operations are free, what is the time complexity of this update, as a function of n?

   (c) Give an algorithm to implement $psum(i), 1 \leqslant i \leqslant n - 1$ using this data structure? Assuming arithmetic operations take unit time, what is the time complexity of $psum(i)$? What is the complexity of $psum(n)$?

Answer:

(a) The data structure is given in Figure 2.

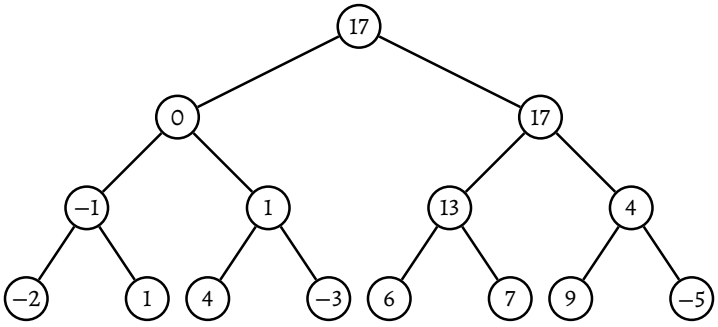


Figure 2: Data structure for the array $A = [-2, 1, 4, -3, 6, 7, 9, -5]$.

(b) We assume that each node of the tree (including each leaf) has three pointer fields: *par*, *lchild* and *rchild*, that point respectively to the parent node, left child and right child. We also assume that there is a special *nil* node, and that the *par* field of the root, and the *lchild* and *rchild* fields of each leaf, point to *nil*. Each node also has a field *val*, which stores the value at that node. Further we assume that A itself is now an array of leaf nodes.

The following pseudocode implements $add(y, i)$.

procedure $add(y, i)$
    $v \leftarrow A[i]$;
    while $(v \neq nil)$ **do**
        $v.val \leftarrow v.val + y$;
        $v \leftarrow v.par$
    end while
end procedure

Since the procedure touches each node on the path from leaf i to root, and since the length of each such path in a full binary tree with n nodes is $O(\log n)$, the time taken for the procedure is $O(\log n)$.

(c) We assume the same fields as in the previous part of the question. We also assume that there is a variable *root*, that points to the root of the tree. The following pseudocode implements $psum'(r, s, i)$, which computes the sum of the leftmost i values of a tree with root r, and with s leaves.

function $psum'(r, s, i)$
    if $(i = s)$ **then**
        return $r.val$
    else if $(i \leqslant s/2)$ **then**
        return $psum'(r.lchild, s/2, i)$
    else
        return $psum'(r.lchild, s/2, s/2) + psum'(r.rchild, s/2, i - s/2)$
    end if
end function

function $psum(i)$
    return $psum'(root, n, i)$
end function

Note that we adjust the third argument, i, when we make recursive calls to $psum'$. One can prove that when r is a leaf, then the call would be of the form $psum'(r, 1, 1)$. The time complexity of $psum'(r, s, i)$ in general is $O(s)$, as can be proved by induction on s. So the time complexity of $psum(i)$ is $O(n)$ in general. But for the special case of $psum(n) = psum'(root, n, n)$, the time complexity is $O(1)$.

⊣

7. The *snakes and ladders* game is played on a board with 100 squares, numbered 1 to 100. There are some ladders and some snakes. Each ladder stands on some square and leads to a higher square. Each snake has its head on some square and tail on a lower square. It is possible that there are squares which neither contain the head of a snake, or the foot of a ladder. There are no squares which contain both the head of a snake and the foot of a ladder.

   The game starts by placing a token on square 1, and the aim of the game is to make the token reach square 100. We do this over a number of rounds. At each round, we choose a number between 1 and 10 and advance the token by so many squares. If the token lands on the head of a snake, it automa:wtically goes to the tail. If it lands at the foot of a ladder, it automatically goes to its top.

   (a) Is it possible to have boards where one can never reach square 100?

   (b) Model the snakes and ladders board as a graph.

   (c) Devise an algorithm to find the smallest number of rounds required to go from square 1 to square 100.

   Answer:

   (a) It is possible to have such a board. If all squares from 90 to 99 have the head of a snake on them, and 100 does not have the top of a ladder, then there is no way to reach square 100. Since we start at 1 and since the maximum number we can choose is 10, and since 100 cannot be reached by a ladder, we cannot reach 100 without reaching a square from 90 to 99. But each such square contains the head of a snake, so one would fall below.

   (b) We can model a board as a directed graph. Each square as a vertex, and each possible transition from square x to square y in one move (either by choosing a number, or by following a snake or a ladder) is an edge between x and y. Further, we attach a weight to each edge, 0 is it is by following a ladder or a snake, and n if $y = x + n$. We also retain the edge with least weight, in case there are multiple edges between two vertices. This models the fact that if a square x is at the bottom of a ladder or at the head of a snake, the player does not have option to choose a number n and go to $x + n$. They must necessarily follow the snake or ladder.

   (c) We can apply Dijkstra's algorithm to find the path with the least total weight from 1 to 100, if any path exists at all.

   ⊣