| INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR | |
|---|---|
| | Stamp / Signature of the Invigilator |

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **EXAMINATION ( End Semester )** | | | | | | | | | **SEMESTER ( Autumn 2024-2025 )** | | | | |
| **Roll Number** | | | | | | | | | | **Section** | | **Name** | |
| **Subject Number** | C | S | 6 | 0 | 0 | 0 | 5 | **Subject Name** | *FOUNDATIONS OF COMPUTING SCIENCE* | | | | |
| **Department / Center of the Student** | | | | | | | | **Additional sheets** | | | | | |

## Important Instructions and Guidelines for Students

1. You must occupy your seat as per the Examination Schedule/Sitting Plan.
2. Do not keep mobile phones or any similar electronic gadgets with you even in the switched off mode.
3. Loose papers, class notes, books or any such materials must not be in your possession, even if they are irrelevant to the subject you are taking examination.
4. Data book, codes, graph papers, relevant standard tables/charts or any other materials are allowed only when instructed by the paper-setter.
5. Use of instrument box, pencil box and non-programmable calculator is allowed during the examination. However, exchange of these items or any other papers (including question papers) is not permitted.
6. Write on both sides of the answer script and do not tear off any page. **Use last page(s) of the answer script for rough work.** Report to the invigilator if the answer script has torn or distorted page(s).
7. It is your responsibility to ensure that you have signed the Attendance Sheet. Keep your Admit Card/Identity Card on the desk for checking by the invigilator.
8. You may leave the examination hall for wash room or for drinking water for a very short period. Record your absence from the Examination Hall in the register provided. Smoking and the consumption of any kind of beverages are strictly prohibited inside the Examination Hall.
9. Do not leave the Examination Hall without submitting your answer script to the invigilator. **In any case, you are not allowed to take away the answer script with you.** After the completion of the examination, do not leave the seat until the invigilators collect all the answer scripts.
10. During the examination, either inside or outside the Examination Hall, gathering information from any kind of sources or exchanging information with others or any such attempt will be treated as **'unfair means'**. Do not adopt unfair means and do not indulge in unseemly behavior.

***Violation of any of the above instructions may lead to severe punishment.***

**Signature of the Student**

*To be filled in by the examiner*

| Question Number | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | Total |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Marks Obtained | | | | | | | | | | | |

| Marks obtained (in words) | Signature of the Examiner | Signature of the Scrutineer |
|---|---|---|
| | | |

**Indian Institute of Technology Kharagpur**
**Department of Computer Science and Engineering**

| | | |
|---|---|---|
| **Foundation of Computing Science (CS60005)** | | **Autumn Semester 2024-2025** |
| 20-November-2024 | **End-Semester Examination** | Maximum Marks: 100 |

**Instructions:**

- Write your answers in the question paper itself. Be brief and precise. Answer *all* questions.
- Write the answers only in the respective spaces provided. The last two blank pages may be used for rough work or leftover answers.
- In case you may need more space/pages, please ask for additional sheets in the exam hall and attach the same with this booklet while submitting.
- If you use any theorem / result / formula covered in the class, just mention it, do not elaborate. (unless the same thing has been explicitly asked to derive / prove in the question)
- Write all the proofs in mathematically / logically precise language. Unclear and/or dubious statements would be severely penalized.

— Question Paper Starts from Next Page —

**Q1.** Let $\mathbb{C}$ denote the set of complex numbers and $\mathbb{Z}[i]$ the subset $\{a+ib \mid a,b \in \mathbb{Z}\}$ of $\mathbb{C}$. Elements of $\mathbb{Z}[i]$ are called *Gaussian integers*. For $z = x+iy \in \mathbb{C}$, we denote by $|z|$ the magnitude of $z$ and by $\arg z$ the argument of $z$. Thus, $z = \sqrt{x^2+y^2}$ and $\arg z = \tan^{-1}\left(\frac{y}{x}\right)$. We take $\arg z$ in the interval $[0, 2\pi)$.

Define a relation $\rho$ on $\mathbb{C}$ as follows. Take $z_1, z_2 \in \mathbb{C}$. We say that $z_1 \; \rho \; z_2$ if and only if

either (i) $|z_1| < |z_2|$,

or (ii) $|z_1| = |z_2|$ and $\arg z_1 \leq \arg z_2$.

Moreover, define another relation $\sigma$ on $\mathbb{C}$ as $z_1 \; \sigma \; z_2$ if and only if $|z_1| = |z_2|$. Answer the following.

**(a)** Prove that $\rho$ is a partial order on $\mathbb{C}$. **(5)**

**Solution:**

Let $z, z_1, z_2, z_3 \in \mathbb{C}$.

**Reflexive:** We have $|z| = |z|$ and $\arg z \leq \arg z$, that is, $z \; \rho \; z$.

**Antisymmetric:** Suppose $z_1 \; \rho \; z_2$ and $z_2 \; \rho \; z_1$. If $|z_1| < |z_2|$, we cannot have $z_2 \; \rho \; z_1$. Analogously, if $|z_2| < |z_1|$, we cannot have $z_1 \; \rho \; z_2$. Therefore, $|z_1| = |z_2|$. In that case, $\arg z_1 \leq \arg z_2$ and $\arg z_2 \leq \arg z_1$, that is, $\arg z_1 = \arg z_2$. It follows that $z_1 = z_2$.

**Transitive:** Let $z_1 \; \rho \; z_2$ and $z_2 \; \rho \; z_3$. This means $|z_1| \leq |z_2| \leq |z_3|$. If $|z_1| < |z_2|$ or $|z_2| < |z_3|$, then $|z_1| < |z_3|$, that is, $z_1 \; \rho \; z_3$. If $|z_1| = |z_2| = |z_3|$, we have $\arg z_1 \leq \arg z_2 \leq \arg z_3$, that is, again $z_1 \; \rho \; z_3$.

**(b)** Prove that $\sigma$ is an equivalence relation on $\mathbb{C}$. **(3)**

**Solution:**

Let $z, z_1, z_2, z_3 \in \mathbb{C}$.

**Reflexive:** Since $|z| = |z|$, we have $z \, \rho \, z$.

**Symmetric:** $z_1 \, \sigma \, z_2$ implies $|z_1| = |z_2|$, that is, $|z_2| = |z_1|$, that is, $z_2 \, \sigma \, z_1$.

**Transitive:** $z_1 \, \sigma \, z_2$ and $z_2 \, \sigma \, z_3$ imply $|z_1| = |z_2| = |z_3|$, that is, $z_1 \, \sigma \, z_3$.

**(c)** What are the equivalence classes of $\sigma$? (Provide a geometric description.) **(3)**

**Solution:**

Let $z = x + iy \in \mathbb{C}$ with $r = \sqrt{x^2 + y^2}$. Then $[z]_\sigma$ consists precisely of all complex numbers whose absolute values equal $r$, that is, $[z]_\sigma$ is the circle of radius $r$ centered at the origin.

**Q2.** Consider the context-free grammar $G$ over $\{a,b\}$, with start symbol $S$, non-terminals $\{S,A,B\}$, and the following productions.

$$S \to aaB \mid Abb, \qquad A \to a \mid aA, \qquad B \to b \mid bB.$$

**(a)** What is the language, $\mathscr{L}(G)$, generated by $G$? **(2)**

**Solution:**

$$\mathscr{L}(G) = \{a^2b^n \mid n \geq 1\} \bigcup \{a^nb^2 \mid n \geq 1\}$$

**(b)** Prove that this context-free grammar $G$ is *ambiguous*. **(3)**

**Solution:**

The string *aabb* has two distinct leftmost derivations:

$$S \Rightarrow aaB \Rightarrow aabB \Rightarrow aabb,$$
$$S \Rightarrow Abb \Rightarrow aAbb \Rightarrow aabb.$$

**(c)** Design an *unambiguous* context-free grammar for $\mathscr{L}(G)$. **(3)**

**Solution:**

In order to disambiguate this grammar, we separate out some small examples.

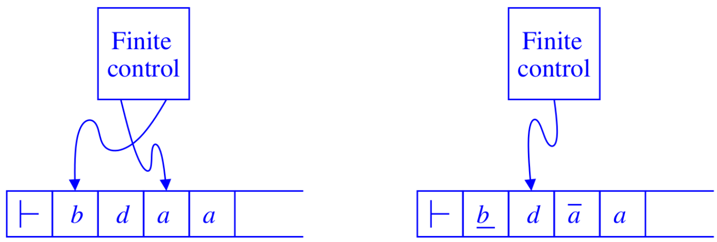
$$\begin{aligned} S &\to aab \mid abb \mid aabb \mid aaAbb \mid aaBbb \\ A &\to a \mid aA \\ B &\to b \mid bB \end{aligned}$$

**Q3.** Let $M$ be a Turing machine with one semi-infinite tape and <u>two</u> read/write heads. Each transition of $M$ is determined by the current state $p$ of the finite control, and the two symbols $a$ and $b$ scanned by the two heads. A transition of $M$ is of the form $\delta(p,a,b) = (q,c,d,D_1,D_2)$ implying that the finite control goes to state $q$, the symbol $a$ at the cell pointed by the first head is replaced by $c$, and the symbol $b$ at the cell pointed by the second head is replaced by $d$. If both the heads point to the same tape cell ($a = b$ in this case), then the symbol at this cell is replaced by $c$ (not by $d$ unless $c = d$). Finally, the first head moves by one cell in direction $D_1$ (left or right), and the second head moves by one cell in direction $D_2$ (left or right).

Argue that this two-head Turing machine $M$ can be simulated by a standard Turing machine $N$ with one semi-infinite tape and with only <u>one</u> read/write head. **(6)**

**Solution:**

Let $\Gamma$ be the tape alphabet of $M$. For each $a \in \Gamma$, we introduce three new symbols $\overline{a}$, $\underline{a}$ and $\overline{\underline{a}}$. The tape alphabet of $N$ consists of $\Gamma$ and the three new symbols introduced for each $a \in \Gamma$. The symbol $\overline{a}$ in a cell indicates that the first head of $M$ points to this cell which contains the symbol $a$, $\underline{a}$ indicates that the second head of $M$ points to this cell, whereas $\overline{\underline{a}}$ indicates that both the heads point to this cell. In order to simulate a single move of $M$, $N$ first locates the two markers $^-$ and $_-$, and remembers the corresponding symbols $a,b \in \Gamma$ in its finite control. $N$ now consults the transition function of $M$, replaces $a$, $b$ by appropriate symbols $c$, $d$, and moves the markers $^-$ and $_-$ as dictated by $\delta(p,a,b)$, where the state $p$ of $N$ is remembered in the finite control of $M$.

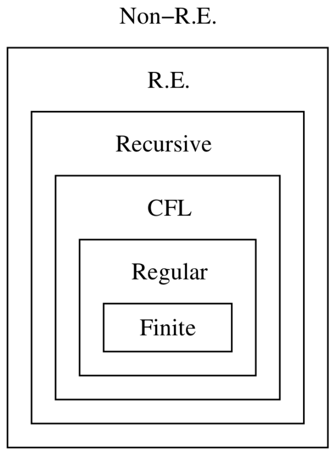



(a) A two–head machine (b) Simulation by a one–head machine

**Q4.** Consider the hierarchy of languages shown in the adjacent figure (right). Place each of the following languages, $L_1, L_2, L_3, L_4, L_5$ (given in Parts (a)-(e), respectively) at the proper place in the hierarchy, that is, state and deduce whether the language $L_i$ is:

- Finite
- Regular and Infinite
- Context-Free but Not Regular
- Recursive but Not Context-Free
- R.E. (recursively enumerable) but Not Recursive
- Non-R.E.

Provide clear and proper justifications/deductions for your judgments. No credits will be given for incorrect/incomplete/missing explanations, even though the truth value is correctly assigned.




In case of justifying Recursive or R.E. languages, construct appropriate Turing machines and/or supply appropriate reductions (note that, halting may be a choice of Turing machines, so do not use Rice's theorem).

**(a)** $L_1 = \mathscr{L}(G)$ where grammar $G$ is defined over $\{a, b\}$, with start symbol $S$, non-terminals $\{S, T\}$, and the following productions. **(4)**

$$S \to \varepsilon \mid abTS, \qquad T \to \varepsilon \mid Tb.$$

$L_1$ is Regular and Infinite.

**Solution:**

*Proof:*

It is an easy matter to check that $\mathscr{L}(T) = b^*$ and it follows that $L_1 = \mathscr{L}(G) = \mathscr{L}(S) = (abb^*)^*$.

**(b)** The set $L_2$ of all strings $\alpha \in \{a,b,c\}^*$ containing an equal number of occurrences of $a$'s, $b$'s and $c$'s. **(6)**

$L_2$ is Recursive but Not Context-Free .

**Solution:**

*Proof:*

We have, $L_2 = \big\{\{a,b,c\}^* \mid \nu_a(\alpha) = \nu_b(\alpha) = \nu_c(\alpha)\big\}$, where $\nu_x(\alpha)$ denotes the number of occurrences of $x \in \{a,b,c\}$ in $\alpha \in \{a,b,c\}^*$.

If $L_2$ is context-free, consider the pumping lemma constant $n$ for $L_2$ (Better use the stronger version of the lemma.) and using the string $a^n b^n c^n \in L_2$ arrive at a contradiction. So $L_2$ is not context-free.

It is R.E., since we can design a TM $M_2$ to accept $L_2$ . One may go for a four-tape machine, where the numbers of occurrences of $a$, $b$ and $c$ are counted and stored in tapes 2 through 4 as the strings $0^{\nu_a(\alpha)}$, $0^{\nu_b(\alpha)}$ and $0^{\nu_c(\alpha)}$. Subsequently the lengths of these strings can be easily compared. Clearly, $M_2$ can be so constructed as to halt on every input. Thus $L_2$ is recursive as well.

**(c)** The complement of $L_2$ (from Part-(b)), that is, $L_3 = \{a,b,c\}^* \setminus L_2$. **(6)**

$L_3$ is ______Context-Free but Not Regular______ .

**Solution:**

*Proof:*

If $L_3$ is regular, $L_2 = \overline{L_3}$ will also be regular, but by Part (b), $L_2$ is not even context-free. One can write $L_3 = L_{31} \cup L_{32}$ , where $L_{31}$ (resp. $L_{32}$) consists of all strings $\alpha \in \{a,b,c\}^*$ with $\nu_a(\alpha) \neq \nu_b(\alpha)$ (resp. $\nu_b(\alpha) \neq \nu_c(\alpha)$). Since context-free languages are closed under union, it is sufficient to show that $L_{31}$ and $L_{32}$ are context-free.

To this end, we can construct a CFG $G = \big(\{a,b,c\},\ \{S,A,B,C,E\},\ S,\ R\big)$ for $L_{31}$ with the following rules:

$$
\begin{aligned}
S &\rightarrow A \mid B \\
E &\rightarrow C \mid CaEbC \mid CbEaC \mid EE \\
A &\rightarrow CaE \mid CaA \mid CbAA \\
B &\rightarrow CbE \mid CbB \mid CaBB \\
C &\rightarrow \varepsilon \mid cC
\end{aligned}
$$

An analogous CFG for $L_{32}$ can be written.

**(d)** $L_4 = \{M \mid M$ is a Turing machine that halts on exactly 2024 input strings$\}$. **(6)**

$L_4$ is Non-R.E. .

**Solution:**

*Proof:*

We propose a reduction $\overline{\mathsf{HP}} \le_m L_4$ which maps $M\#\alpha$ to $N$ such that $M$ does not halt on $\alpha$ if and only if $N$ halts on exactly 2024 input strings. The reduction algorithms uses any 2024 constant strings $\gamma_1, \gamma_2, \ldots, \gamma_{2024}$ (distinct from one another). For example, we may have $\gamma_i = 0^i$ for $i = 1, 2, \ldots, 2024$.

$N$, upon input $\beta$, does the following:

(1) Check whether $\beta = \gamma_i$ for some $i = 1, 2, \ldots, 2024$. If so, halt (after accepting or rejecting).

(2) Simulate $M$ on $\alpha$.

(3) If the simulation of Step (2) halts, halt (after accepting or rejecting).

It follows that if $M$ halts on $\alpha$, then $N$ halts on every input $\beta$. On the other hand, if $M$ does not halt on $\alpha$, then $N$ halts only on the 2024 input strings $\gamma_1, \gamma_2, \ldots, \gamma_{2024}$.

**(e)** $L_5 = \{M \mid M$ is a Turing machine that halts on at least 2024 input strings$\}$. **(6)**

$L_5$ is ______ R.E. but Not Recursive ______ .

**Solution:**

*Proof:*

We can design a Turing machine $K$ that simulates $M$ on all possible input strings on a time-sharing basis. If any 2024 of the simulations halt, $K$ accepts and halts. If 2024 strings are never found, the parallel simulation of $K$ never stops. Thus, $L_5$ is recursively enumerable. (Alternatively, $K$ can be a non-deterministic Turing machine which simulates $M$ on 2024 distinct choices of inputs. The simulations may proceed in parallel or one after another.)

In order to prove that $L_5$ is not recursive, we make a reduction HP $\leq_m L_5$ that maps $M\#\alpha$ to $N$ such that $M$ halts on $\alpha$ if and only if $N$ halts on at least 2024 input strings.

$N$, upon input $\beta$, does the following:

(1) Simulate $M$ on $\alpha$.

(2) If the simulation of Step (1) halts, halt (after accepting and rejecting).

It follows that if $M$ halts on $\alpha$, then $N$ halts on all input strings (in particular, on at least 2024 input strings). On the other hand, if $M$ does not halt on $\alpha$, then $N$ does not halt on any input string $\beta$.

**Q5.** *Prove* or *disprove* the following statements (Parts (a)–(e)). Give clear justifications. No credits will be given for incorrect/incomplete/missing explanations, even though the truth value is correctly assigned.

**(a)** Prove/Disprove: Let $C$ be an NP-complete problem. If $C \in$ co-NP, then NP = co-NP. **(6)**

**Solution:**

This statement is true.

Let $X$ be a problem in NP, that is, $X \in$ NP. As $C$ is NP-complete, $X \leq_P C$. Now, as per problem statement $C \in$ co-NP. So, we have $X \in$ co-NP. This implies, NP $\subseteq$ co-NP.

Let $Y$ be a problem in co-NP, that is, $Y \in$ co-NP. So, we have $\overline{Y} \in$ NP. Now, as $C$ is NP-complete, $\overline{Y} \leq_P C$. Again, as $C \in$ co-NP, $\overline{Y} \in$ co-NP. So, $Y \in$ NP. This implies, co-NP $\subseteq$ NP.

Both these together *proves* the given assertion.

**(b)** Prove/Disprove: An $O(n^k)$ reduction algorithm from $A$ to $B$ followed by a deterministic $O(n^k)$ algorithm for $B$ yields an $O(n^k)$ deterministic algorithm for $A$.
(Here $n$ is the input size, and $k$ is a positive integer constant $> 1$.) **(4)**

**Solution:**

This statement is false.

Let $\alpha$ be an input of size $n$ for $A$. Call the reduction map $f$, that is, $f(\alpha)$ is an input for $B$. Since $f$ is computable in time $O(n^k)$, the string $f(\alpha)$ can be of length as big as $O(n^k)$. Subsequent application of the algorithm for $B$ then runs in $O((n^k)^k)$, that is, $O(n^{k^2})$, time. For $k > 1$ we have $k^2 > k$.

**(c)** Prove/Disprove: The class P is closed under Kleene star, that is, if $L \in \mathrm{P}$, then $L^* \in \mathrm{P}$. **(8)**

**Solution:**

This statement is true.

Let TM $M$ decide $L$ in time $p(n)$ (a polynomial).

Given $x$ of length $n$, we want to know if $x \in L^*$. We could look at every way to break $x$ up into substrings. That would not give a poly time algorithm since there are lots of ways to break up $x$ (exercise: how many?).

We will actually solve a "harder" problem: given $x$ of length $n$, determine for ALL prefixes of $x$, are they in $L^*$. This is helpful since when we are trying to determine if, say, $x_1 \cdots x_i \in L^*$, we already know the answers to

$$\varepsilon \in L^*, \quad x_1 \in L^*, \quad x_1x_2 \in L^*, \quad \cdots \cdots, \quad x_1x_2 \cdots x_{i-1} \in L^*.$$

*Intuition:* $x_1 \cdots x_i \in L^*$ if and only if it can be broken into two pieces, the first one in $L^*$, and the second in $L$.

We now present the algorithm that will determine if $x \in L^*$. The array `A[i]` will store if $x_1 \cdots x_i \in L^*$.

```
input x of length n
A[1] = A[2] = ... = A[n] = FALSE
A[0] = TRUE
for i = 1 to n do
    for j = 0 to n-1 do
        # Use machine M to test for membership in L
        if A[j] and (x_j, ..., x_{i-1}) in L then
            A[i] = TRUE
        endif
    endfor
endfor
output A[n]
```

*What is the runtime of the above algorithm?* The only time that matters is the calls to $M$. There are $O(n^2)$ calls to $M$, all on inputs of length $\leq n$, hence the runtime is bounded by $O(n^2 p(n))$. Since $p(n)$ is a polynomial, $n^2 p(n)$ is also a polynomial.

**(d)** Prove/Disprove: The class NP is closed under concatenation, that is, if $L_1, L_2 \in \text{NP}$, then $L_1 L_2 \in \text{NP}$. **(6)**

**Solution:**

This statement is true.

For any two NP-languages $L_1$ and $L_2$, let $M_1$ and $M_2$ be the NTMs that decide them in polynomial time. We construct a NTM $M'$ that decides the concatenation of $L_1$ and $L_2$ in polynomial time.

$M'$ ="On input $w$,

(1) Nondeterministically cut $w$ into two substrings, $w = w_1 w_2$.

(2) Run $M_1$ on $w_1$.

(3) Run $M_2$ on $w_2$.

(4) If both accepts, accept.
Otherwise, continue with the next choice of $w_1$ and $w_2$."

In both steps, $M'$ uses its non-determinism when the machine is being run. $M'$ accepts $w$ if and only if $w$ can be expressed as $w_1 w_2$ such that $M_1$ accepts $w_1$ and $M_2$ accepts $w_2$. Therefore, $M'$ decides the concatenation of $L_1$ and $L_2$.

Since Steps 2 and 3 runs in polynomial time and is repeated for at most $O(n)$ time, the algorithm runs in polynomial time.

**(e)** Prove/Disprove: The class NP-complete is closed under union, that is, if $L_1$, $L_2$ are NP-complete, then so is $L_1 \cup L_2$. **(6)**

**Solution:**

This statement is false.

We construct a counter example using the following two NP-complete sets.

$$\mathsf{HAMPATH} = \{G \mid G \text{ is an undirected graph with a Hamiltonian path}\}$$

$$\mathsf{SAT} = \{\phi \mid \phi \text{ is a satisfiable CNF formula}\}$$

Let

$$L_1 = \{(G,\phi) \mid G \in \mathsf{HAMPATH},\ \phi \text{ is any CNF formula}\},$$

and

$$L_2 = \{(G,\phi) \mid G \text{ is an undirected graph},\ \phi \in \mathsf{SAT}\}.$$

Clearly, $L_1, L_2$ are both NP-complete, whereas

$$L_1 \cup L_2 = \{(G,\phi) \mid G \text{ is an undirected graph},\ \phi \text{ is a CNF formula}\}$$

is not. It can be decided in polynomial time.

**Q6.** *Prove* or *disprove* the following assertions (Parts (a)-(b)). Give clear justifications. No credits will be given for incorrect/incomplete/missing explanations, even though the truth value is correctly assigned.

You may make use of the assumption that NP $\neq$ co-NP, if necessary. Clearly indicate where you require this assumption.

**(a)** Prove/Disprove: The following language is NP-complete.

$$\mathsf{SMALLCYCLE} = \left\{ \langle G \rangle \mid \text{The longest cycle in the directed graph } G \text{ is of length } \leq \left\lfloor \frac{n(G)}{2} \right\rfloor \right\}$$

(Here, $n(G)$ denotes the number of vertices in $G$ and $\lfloor\ \rfloor$ is the floor function.) **(6)**

**Solution:**

This assertion is false (under the assumption NP $\neq$ co-NP).

Let us first show that $\mathsf{SMALLCYCLE} \in$ co-NP. If $\langle G \rangle$ is not in $\mathsf{SMALLCYCLE}$, then we can convince one about this fact either by indicating that $G$ is acyclic or by explicitly providing a cycle in $G$ of length $> \left\lfloor \frac{n(G)}{2} \right\rfloor$. That is a succinct and polynomial time verifiable disqualification for $G$. Now, if $\mathsf{SMALLCYCLE}$ were NP-complete, we would have NP = co-NP.

**(b)** Prove/Disprove: The following language is NP-complete.

$$\mathsf{BIGCYCLE} = \left\{ \langle G \rangle \mid G \text{ is a directed graph having a cycle of length } \geq \left\lfloor \frac{n(G)}{2} \right\rfloor \right\}$$

(Here, $n(G)$ denotes the number of vertices in $G$ and $\lfloor\ \rfloor$ is the floor function.) **(6)**

**Solution:**

This assertion is true.

Description of a cycle in G of length $\geq \left\lfloor \frac{n(G)}{2} \right\rfloor$ is a succinct certificate for $\langle G \rangle$ to be in $\mathsf{BIGCYCLE}$, that is, $\mathsf{BIGCYCLE} \in \text{NP}$.

For proving the NP-hardness, we can show $\mathsf{HAMCYCLE} \leq_P \mathsf{BIGCYCLE}$, where

$$\mathsf{HAMCYCLE} = \{G \mid G \text{ is an undirected graph with a Hamiltonian cycle}\}.$$

Let $G$ be an instance for $\mathsf{HAMCYCLE}$ with $m$ vertices. Add (exactly) $m$ isolated vertices to $G$, thereby obtaining a graph $G'$ on $2m$ vertices. It is evident that $G'$ has a cycle of length $m$ (that is, $\geq \left\lfloor \frac{n(G')}{2} \right\rfloor$), if and only if $G$ has a Hamiltonian cycle.

**Q7.** An undirected graph is $k$-colorable if the vertices of the graph can be assigned a colour from a given fixed set of $k$ colors such that no two adjacent vertices (sharing a common edge in between) receive the same color. Prove that, the language, $\mathsf{3COLOR} = \{\langle G \rangle \mid \text{The undirected graph } G \text{ is 3-colorable}\}$, is PSPACE-complete if and only if PSPACE = NP.

(Note that, 3COLOR is known to be NP-complete.) **(5)**

**Solution:**

We already know that 3COLOR is NP-complete. Because we have used the same reduction mechanism (polynomial time) for defining completeness in both NP and PSPACE, PSPACE = NP implies that 3COLOR is PSPACE-complete.

For proving the converse, assume that 3COLOR is PSPACE-complete. Take any $L \in$ PSPACE. By definition, we then have $L \leq_P$ 3COLOR. But then, this reduction followed by an NP algorithm for 3COLOR solves $L$ in a nondeterministic polynomial time, implying that $L \in$ NP, that is, PSPACE $\subseteq$ NP. The reverse inclusion, NP $\subseteq$ PSPACE, is well-known. Both these together imply, PSPACE = NP.

— Question Paper Ends Here —