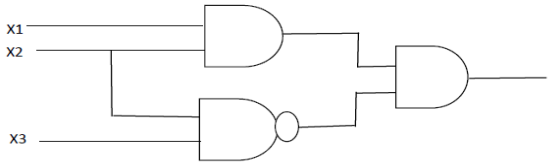
CSE 4131: ALGORITHM DESIGN 2 APRIL, 2024

Assignment 2: Solutions

CO 2: Distinguish between computationally tractable and intractable problems, define and relate class P, NP, NPH, NPC, PSPACE and PSPACE complete and extension limit of Tractibility.

Sl. No.	Questions & Solutions
1.	a. Define the class NPH (NP-Hard). How does it differ from NPC? b. Provide examples of NP-Hard problems that are not necessarily in NP. Explain why they are classified as NP-Hard. c. Discuss the relationship between NP-Hard problems and computational hardness. How are they relevant in practical computing scenarios?
Ans: 1a.	NP-Hard={X : all problem Y∈NP can be reduce to X i.e $Y \leq_P X$. }. A subset of NP-Hard problems that are both NP-Hard and in NP. They are among the hardest problems in NP.
Ans: 1b.	NP-Hard problem: Vertex cover optimization problem. The Vertex Cover Optimization Problem is NP-Hard because it is in NP and can be used to solve other NP-Hard problems, and it lacks a known polynomial-time algorithm to solve it for all instances.
Ans: 1c.	Take an example Job Scheduling Problem/ Load Balancing problem. **NP-Hardness**: Job scheduling problems fall into the category of NP-Hard problems due to their combinatorial nature and the need to search through a vast solution space. For example, consider the problem of scheduling jobs on the machines to minimize the makespan (the time when all jobs are completed). This problem is known as the Parallel Machine Scheduling problem and is NP-Hard. **Computational Hardness**: As the number of jobs or machines increases, the number of possible schedules grows exponentially, making it computationally hard to find the optimal solution. For large instances, search becomes impractical due to the time required to explore all possible schedules. Due to the computational hardness of NP-Hard scheduling problems, approximation algorithms are commonly used in practice. These algorithms may not guarantee the optimal solution.
2.	a. Define the class NPC (NP-Complete). What are the defining properties of an NP-Complete problem? b. Provide examples of well-known NP-Complete problems and explain why they are considered as such. c. Explain the concept of polynomial-time reduction. How is it used to prove a problem is NP-Complete?
Ans: 2a.	**Definition of NPC (NP-Complete)**: NP-Complete (NPC) is a class of decision problems in computational complexity theory. A decision problem is a yes/no question, and NP-Complete problems are among the most difficult problems in NP (nondeterministic polynomial time). The defining properties of an NP-Complete problem are: 1. **In NP**: An NP-Complete problem must be in the complexity class NP, meaning that given a potential solution; it can be verified in polynomial time. 2. **NP-Hardness**: An NP-Complete problem must be at least as hard as the hardest problems in NP. This means that every problem in NP can be polynomial-time reduced to an NP-Complete problem.
Ans: 2b.	**Boolean Satisfiability (SAT)**: • In SAT, given a Boolean formula in conjunctive normal form (CNF), the problem is to determine whether there exists an assignment of truth values to variables that satisfies the formula. • SAT is NP-Complete because it is in NP (verification of a satisfying assignment is polynomial-time verifiable), and it can be used to polynomial-time reduce any problem in NP-Hard to SAT • Another examples: Independent decision problem, Vertex cover decision problem, TSP decision problem.
Ans: 2c.	1. **Choose an existing NP-Complete problem**: Start with a problem that is already known to be NP-Complete, such as 3SAT or Hamiltonian Cycle. 2. **Define a mapping**: Define a mapping from instances of the known NP-Complete problem to instances

	of the problem in question. 3. **Show polynomial-time reduction**: Show that this mapping can be done in polynomial time. This means that for any instance of the known NP-Complete problem, you can transform it into an instance of the new problem in polynomial time.
3.	Given a decision vertex-cover problem instances $G(V, E, k)$, where $V = \{1,2,3,4,5,6,7\}$ and $E = \{(1,2), (1,3), (2,3), (1,4), (2,5), (3,6), (4,6), (4,5), (5,7), (6,7)\}$, $k = 3$. Find the decision set-cover problem instances (i.e., V, subsets of V and k). And give a possible solution.
Ans:	The set of edges $U = \{(1,2), (1,3), (2,3), (1,4), (2,5), (3,6), (4,6), (4,5), (5,7), (6,7)\}$ Now the subsets of Edge set U are following: S_1={(1,2), (1,3), (1,4)} , S_2={(1,2),(2,3),(2,5)}, S_3={(1,3),(2,3),(3,6)}, S_4={(1,4),(4,6),(4,5)}, S_5={(2,5),(5,7),(4,5)}, S_6={(3,6),(4,6),(6,7)} One possible set cover is U=$S_1 \cup S_2 \cup S_5 \cup S_6$. And the vertex cover (VC) set S={1,2,5,6} of size 4. Because we cannot find a VC of size 3 since number of edges 10 $\nleq k * d$=9.[See the concept of Q5.]
4.	A new telecom company is trying to establish its offices in a large city. The company wants to open offices at various (traffic) junctions of the road to cover maximum percentage of consumers. The local govt. authority has put a restriction that the company cannot open offices at adjacent junctions and has to leave at least two junctions gap between two offices. Now, our goal is to find the maximum number of offices that can be opened by the company in the given city. a. Define the decision version of the problem b. Show that, the given problem is a NP-Complete problem. (Hint: Show the reduction of a NP complete problem to the given problem and justify the hardness).
Ans: 4a.	**Decision Version of the Problem:** Given a city represented as a graph where each vertex represents a junction and each edge represents a road between two junctions, and an integer k, the decision problem can be stated as follows: Is it possible for the telecom company to open k offices in the city such that no two offices are adjacent (i.e., there is at least a distance of two junctions between any two offices)?
	To show that the given problem is NP-complete, we will reduce the Independent Set (IS) problem to it. **Independent Set (IS) Problem:** Given a graph G and an integer k, the Independent Set problem whether there exists a set of k vertices in G such that no two vertices in the set are adjacent. **Reduction:** Vertex Cover decision problem $\leq_P$ Independent Set Decision problem (Already done in the class)
5.	Draw a graph $G(V, E)$ from the given information where V is the set of all vertices, $E = \{(v_1, v_2), (v_1, v_9), (v_2, v_3), (v_2, v_8), (v_3, v_5), (v_4, v_5), (v_5, v_6), (v_6, v_7), (v_7, v_8), (v_7, v_9)\}$ set of all edges. a. Can a Vertex Cover of size 3 possible for the given graph? Justify your answer. b. Can a Vertex Cover of size 4 possible for the given graph? Justify your answer. c. Why can't an undirected Complete Graph with 8 vertices have a vertex cover of size 3?
Ans: 5a.	No. The graph has 9 nodes. The maximum degree of any node is 3. To have a vertex cover of size 3, there must be at most 3*3=9 edges. But the graph has 10 edges.
Ans: 5b.	Yes. The graph has 9 nodes. The maximum degree of any node is 3. To have a vertex cover of size 4, there must be at most 4*3=12 edges. But the graph has 10 edges.
Ans: 5c.	The undirected Complete Graph with 8 vertices will have (8*7)/2=28 edges. Degree of each vertex will be 7. Considering the Vertex cover of size 3, there has to be at most 3*7=21 edges. But there are 28 edges.
6.	Given a Boolean circuit as follows



Convert the following circuit to CNF (Reduction of Circuit Satisfiability Problem to CNF SAT)

$$A \leftrightarrow B = (\bar{A} + B)(A + \bar{B}) \quad (1)$$

$$\textbf{DeMorgan's Law} : \overline{A.B} = \bar{A} + \bar{B}; \overline{A + B} = \bar{A}.\bar{B} \quad (2)$$

$$\textbf{Distributive Law} : A.(B + C) = A.B + A.C; A + (B.C) = (A + B).(A + C) \quad (3)$$

$$\begin{aligned} \phi_1 &= (y_0 \leftrightarrow x_1.x_2) \\ &= (y_0 + \overline{x_1.x_2}).(\bar{y_0} + x_1.x_2)[from(1)] \\ &= (y_0 + \bar{x_1} + \bar{x_2}).(\bar{y_0} + x_1.x_2)[from(2)] \\ &= (y_0 + \bar{x_1} + \bar{x_2}).(\bar{y_0} + x_1).(\bar{y_0} + x_2)[from(3)] \end{aligned} \quad (4)$$

$$\begin{aligned} \phi_2 &= (y_1 \leftrightarrow \overline{x_2.x_3}) \\ &= (y_1 + \overline{\overline{x_2.x_3}}).(\bar{y_1} + \overline{x_2.x_3})[from(1)] \\ &= (y_1 + x_2.x_3).(\bar{y_1} + \overline{x_2.x_3}) \\ &= (y_1 + x_2)(y_1 + x3).(\bar{y_1} + \bar{x_2}).(\bar{y_1} + \bar{x_3})[from(3)] \end{aligned} \quad (5)$$

$$\begin{aligned} \phi_3 &= (y_2 \leftrightarrow y_0.y_1) \\ &= (y_2 + \overline{y_0.y_1}).(\bar{y_2} + y_0.y_1)[from(1)] \\ &= (y_2 + \bar{y_0} + \bar{y_1}).(\bar{y_2} + y_0.y_1)[from(2)] \\ &= (\bar{y_0} + \bar{y_1} + y_2).(\bar{y_2} + y_0).(\bar{y_2} + y_1)[from(3)] \end{aligned} \quad (6)$$

Finally,

$$\phi = \phi_1.\phi_2.\phi_3 \quad (7)$$

7. Considering the context of planning problems,

1	2	3
	4	6
7	5	8

(Initial State)

1	2	3
4	5	6
7	8	

(Goal State)

a. Represent the Initial Configuration of the given instance of 8-puzzle game as a set of conditions.
b. Show the sequence configurations with feasible operations that can reach the Goal Configuration from the Initial Configuration.
c. Why are planning problems like the 15-puzzle game said to be in PSPACE?

Ans: 7a.	Cij = Value i Cell position j Initial Configuration is : <C11,C22,C33,C45,C58,C66,C77,C89,C94> [Empty cell=9] Final Configuration is : <C11,C22,C33,C44,C55,C66,C77,C88,C99>
Ans: 7b.	An operation makes some conditions FALSE in the current configuration and makes some new conditions TRUE in the next configuration. Initial Configuration is : <C11,C22,C33,**C45**,C58,C66,C77,C89,**C94**> *Operation1:* C94,C45 as FALSE . C44, C95 as TRUE Intermediate Configuration1: <C11,C22,C33,**C44**,C58,C66,C77,C89,**C95**> *Operation2:* C58,C95 as FLASE. C55,C89 as TRUE. Intermediate Configuration2: <C11,C22,C33,C44,**C55**,C66,C77,C89,**C98**> *Operation3:* C89,C98 as FLASE. C88,C99 as TRUE. Final Configuration is : <C11,C22,C33,C44,C55,C66,C77,**C88,C99**>
Ans: 7c.	For Planning problems, a systematic search using a cost function on the state space tree can help us reach the goal configuration. Considering the maximum branching factor as b and the maximum depth of the state space tree as m, with DFS, the space requirement will be $O(bm)$, which is assumed to be linear space. This can hint that planning problems are said to be in PSPACE.
8.	a. Write the 3-SAT expression for the given QSAT expression "$\exists x_1 \forall x_2 \exists x_3 \forall x_4 \exists x_5\ \varphi(x) = (x_1 \vee \neg x_3 \vee x_4) \wedge (\neg x_2 \vee x_3 \vee \neg x_5) \wedge (\neg x_1 \vee x_3 \vee \neg x_2)$" using appropriate quantifiers. b. Is there a satisfying assignment for the given Q-SAT "$\exists x_1 \forall x_2 \exists x_3\ \varphi(x) = (x_1 \vee \neg x_3 \vee x_2) \wedge (\neg x_2 \vee x_3 \vee \neg x_1) \wedge (\neg x_1 \vee \neg x_3 \vee \neg x_2)$. c. Why Q-SAT ∈ PSPACE?
Ans: 8a.	3-SAT uses only existential quantifiers. So the expression is "∃x1∃x2∃x3∃x4∃x5: (x1∨¬x3∨x4) ∧ (¬x2∨x3∨¬x5) ∧ (¬x1∨x3∨¬x2)"
Ans: 8b.	Yes. Drawing the recursion tree, it will be found that the Q-SAT is satisfiable (x1=0, x2=0 and 1, x3=0).
Ans: 8c.	Consider the recursive approach (Divide and Conquer) to evaluate all possible assignments. The depth of recursion will be n as there are n variables. Only 1 bit of information is required for each subproblem. Reusing the stack space, the total space requirement will be polynomial with respect to input size.
9.	An *Eulerian cycle* is a tour that visits every edge in a graph exactly once. An *Eulerian subgraph* is a subset of the edges and vertices of a graph that has an Eulerian cycle. Prove that the problem of finding the number of edges in the largest Eulerian subgraph of a graph is NP-hard.
	To prove that the problem of finding the number of edges in the largest Eulerian subgraph of a graph is NP-hard, we can reduce a known NP-hard problem to it such as the Hamiltonian cycle problem. 1. Given an instance of the Hamiltonian cycle problem, which involves finding a cycle that visits every vertex exactly once in a graph, we construct a new graph. 2. For each vertex in the original graph, we duplicate it in the new graph. This ensures that in the resulting Eulerian cycle, each original vertex is visited exactly twice (once for entering and once for leaving). Given a graph G, create a new graph G' as follows: • For each vertex v in G, create two vertices v_{in} and v_{out} in G'. • For each edge (u,v) in G, add an edge (u_{out}, v_{in}) in G'. 3. Connect each pair of corresponding duplicated vertices (i.e., vertices representing the same original vertex) with an edge in the new graph. This ensures that the resulting cycle will visit each duplicated vertex exactly once, forming a Hamiltonian cycle in the original graph. 4. Now, the largest Eulerian cycle problem in the new graph is to find the cycle that visits the maximum number of edges. Since each edge in the new graph corresponds to a vertex in the original graph, finding the largest Eulerian cycle in the new graph is equivalent to finding the Hamiltonian cycle in the original graph. By solving the largest Eulerian cycle problem in the constructed graph, we can find a solution to the original Hamiltonian cycle problem.

	This reduction demonstrates that if we can efficiently solve the largest Eulerian cycle problem, then we can efficiently solve the Hamiltonian cycle problem as well.
10.	a. Reduce a 3-SAT formula to an independent set problem. (Explain with an example). b. Suppose, the 3-SAT problem has k clauses, then it is satisfiable if and only if the corresponding graph has an independent set of size k. Now, give an example to show that 3-SAT is not satisfiable if and only if the corresponding graph does not have any independent set of size k.
Ans: 10.a	Already done in the class.
Ans: 10.b	$\varphi(x)=(\overline{x_1} \vee \overline{x_2} \vee \overline{x_3}) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee x_2 \vee \overline{x_3}) \wedge (\overline{x_1} \vee x_2 \vee x_3) \wedge (x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (x_1 \vee \overline{x_2} \vee x_3) \wedge (x_1 \vee x_2 \vee \overline{x_3}) \wedge (x_1 \vee x_2 \vee x_3)$, Here, φ is not satisfiable for all truth assignment. We cannot find independent set of size 8.