

CSCE 5150 – Analysis of Computer Algorithms

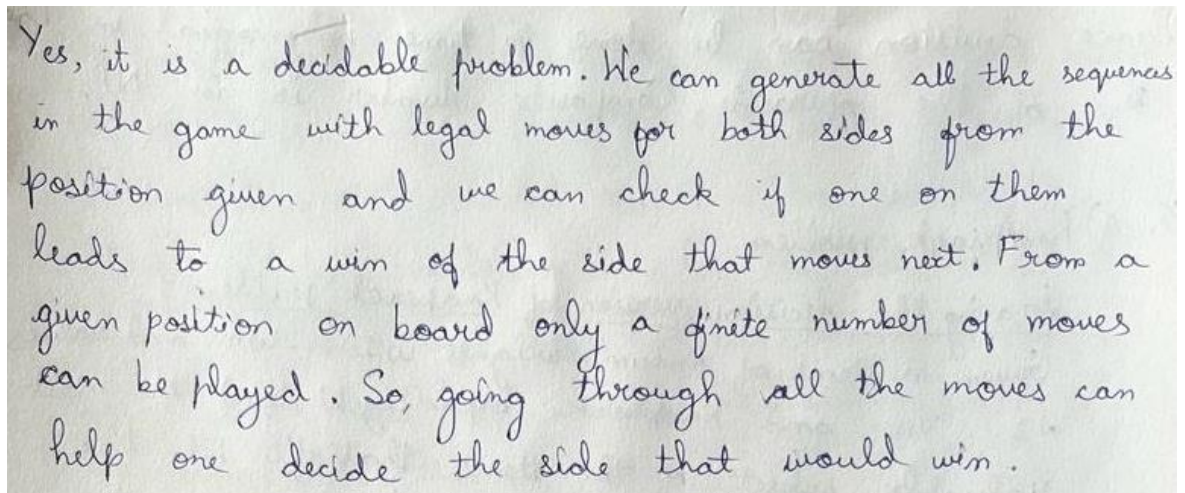
Homework No. 7 – NP-Completeness

Neha Goud Baddam

11519516

A game of chess can be posed as the following decision problem: given a legal positioning of chess pieces and information about which side is to move, determine whether that side can win. Is this decision problem decidable?

1.



Yes, it is a decidable problem. We can generate all the sequences in the game with legal moves for both sides from the position given and we can check if one of them leads to a win of the side that moves next. From a given position on board only a finite number of moves can be played. So, going through all the moves can help one decide the side that would win.

Consider the following brute-force algorithm for solving the composite number problem: Check successive integers from 2 to $\lfloor n/2 \rfloor$ as possible divisors of n . If one of them divides n evenly, return yes (i.e., the number is composite), if none of them does, return no. Why does this algorithm not put the problem in class P ?

2.

CSCE 5150 – Analysis of Computer Algorithms

Homework No. 7 – NP-Completeness

Neha Goud Baddam

11519516

Let P be a class of decision problem L for which we can find a solution in polynomial time, i.e. there is a polynomial time function $f(x)$ where x is string and $f(x) = \text{True}$, if and only if x is in L .

→ NP is the class in decision problem L such that there is a polynomial time function $f(x, c)$, $f(x, c) = \text{True}$ if x is in L . Here x is string & c is certificate.

For the given problem of Composite Number, we can consider all instances for which n is composite as L .

We can look at this as a language L by simply coding n in $\log n$ bits as a binary number. So every binary composite number is in L and nothing else. We can show that problem is in NP by providing a polynomial time function $f(x, c)$

- c cannot be bigger than n , c can be binary encoding of non-trivial factor of n , the size of c would be polynomial in size of n .
- The function f checks if c divides n evenly, if it does then n is proved to be composite and f returns True.

Since division can be done in time polynomial in the size of the operands, Composite Number is in NP.

State the decision version for each of the following problems and outline a polynomial-time algorithm that verifies whether or not a proposed solution solves the problem. (You may assume that a proposed solution represents a legitimate input to your verification algorithm.)

- knapsack problem
- bin packing problem

3.

CSCE 5150 – Analysis of Computer Algorithms

Homework No. 7 – NP-Completeness

Neha Goud Baddam

11519516

A) Knapsack problem.

Stating the decision version of Knapsack problem:

Given n items of known weights $w_1 \dots w_n$ and values $v_1 \dots v_n$ and knapsack capacity W , find the most valuable subset of the items that fit into the knapsack. (objects cannot be broken)

Solution:

Given n items of known weights $w_1 \dots w_n$, values $v_1 \dots v_n$ and knapsack capacity W and given a positive integer K , decide whether there is a subset of items that fits into the knapsack whose value is greater or equal to K , (objects cannot be broken).

Polynomial-time algorithm to verify proposed solution (P_i) :

The algorithm is given subset S of objects. It checks

1. Check if $P_i \in S$ $w_i < W$
2. Check that $P_i \in S$ $v_i \geq K$.

If above conditions are true, then proposed solution (certificate) is the solution.

CSCE 5150 – Analysis of Computer Algorithms

Homework No. 7 – NP-Completeness

Neha Goud Baddam

11519516

B) We have to determine whether one can place a given set of items into no more than m bins. A proposed solution does not exceed m , can be verified by checking the number of bins in the solution and that the sum of sizes of items assigned to the same bin does not exceed the bin capacity of the bins. The time efficiency is $O(n)$ when n is number of items in the instance.

Determine whether the following problem is *NP*-complete. Given several sequences of uppercase and lowercase letters, is it possible to select a letter from each sequence without selecting both the upper- and lowercase versions of any letter? For example, if the sequences are *Abc*, *BC*, *aB*, and *ac*, it is possible to choose *A* from the first sequence, *B* from the second and third, and *c* from the fourth. An example where there is no way to make the required selections is given by the four sequences *AB*, *Ab*, *aB*, and *ab*. [Kar86]

4.

CSCE 5150 – Analysis of Computer Algorithms

Homework No. 7 – NP-Completeness

Neha Goud Baddam

11519516

A problem is said to be NP complete if it is NP hard and belongs to NP.

Firstly, let us look at the given problem

1. Every instance has solution as yes/no. So it is decision problem
2. The no. of solutions is dependent on the length of the sequences, So, it is finite.
3. We can say, if given a sequence and answer if we verify its correctness in polynomial time. So, it is polynomially verifiable.

Hence we can say that it is NP problem.

Now, we need to prove that it is NP hard or not.

CSCE 5150 – Analysis of Computer Algorithms

Homework No. 7 – NP-Completeness

Neha Goud Baddam

11519516

A problem is considered NP hard if we cannot prove whether polynomial solution exists or not.

For given problem, we must check all combination to find solution.

- Remove all duplicate letters from each sequence.
- Now ~~maximum~~ maximum width = 52 (26 lower & 26 upper)
- From sequence 1, select first letter & delete its opposite case from each sequence. It takes $n * 52 \log 52$ steps.
- From sequence 2, select the first letter and continue the above step repeatedly.
- Follow the same process for all the sequences.
Now the total complexity = $O(n^2)$.
- This may not be the solution. So, we have to non deterministically select all possible combinations. We have 52 ways of selecting a letter from each sequence. We have a total of n sequences. Total complexity = 52^n .

Therefore, there is no polynomial algorithm, it is a NP hard problem.

As it is both NP and NP-Hard problem, it is said to be NP-complete problem.

CSCE 5150 – Analysis of Computer Algorithms

Homework No. 7 – NP-Completeness

Neha Goud Baddam

11519516

King Arthur expects 150 knights for an annual dinner at Camelot. Unfortunately, some of the knights quarrel with each other, and Arthur knows who quarrels with whom. Arthur wants to seat his guests around a table so that no two quarreling knights sit next to each other.

- a. Which standard problem can be used to model King Arthur's task?
 - b. As a research project, find a proof that Arthur's problem has a solution if each knight does not quarrel with at least 75 other knights.
- 5.

a) We can create a graph where vertices represent the knights and an edge connecting two vertices if two knights can sit next to each other. Then the solution to King Arthur's problem exists if the graph has a Hamiltonian Circuit. As Hamiltonian cycle is NP complete, it works by visiting each vertex of graph exactly once to find if there exists a simple cycle for graph. The problem can be solved using Hamiltonian undirected cycle matching technique.

CSCE 5150 – Analysis of Computer Algorithms

Homework No. 7 – NP-Completeness

Neha Goud Baddam

11519516

b) According to Dirac's theorem, a graph with vertices greater than 3 i.e. $n \geq 3$ has a Hamiltonian cycle if the degree of each vertex is greater than or equal to $n/2$. Hence, it is easy to form a cyclic group so the king can arrange them in round table. If more than 75 quarrel it is not possible. Hence, it is possible to arrange 75 kings if each knight does not quarrel at least 75 other knights.