Exercise Solution: P and NP

Prove their NP membership for the following problems (or their decision version).

1. **Partition.** Given a set S of n integers $a_1, a_2, \cdots, a_n$, is there a way to partition S into two subsets such that the sums in each subset are the same?

   First, for any yes-instance (that is, a set of numbers with an equal-sum bipartition), we define a certificate C as a subset of elements in one of the two parties. Note that the size of $\langle C \rangle$ is upper bounded by the size of $\langle S \rangle$ since C is a subset of S.

   Next, we design a polynomial-time verifier for PARTITION.

   V = "On input $\langle \langle S \rangle, C \rangle$:

   1. Check if C is a subset of S. If it is not true, *reject*.

   2. Check if the sum of elements in C is equal to the sum of elements in $S \setminus C$.

   If it is not true, *reject*

   3. If both Steps 1 and 2 pass, *accept*."

   Finally, we show that V can verify PARTITION in polynomial time. For Step 1, V scans the input string for at most $\min\{|C|, |S|\}$ times. For Step 2, V scans the input string twice. Hence, the total running time of V is polynomial in the input string size.

2. **Conjunctive normal form satisfiability.** A Boolean formula is in *Conjunctive Normal Form (CNF)* if it is a conjunction of clauses, where a clause is a disjunction of literals. Given a CNF formula, is it satisfiable?

   First, for any yes-instance (that is, a satisfiable CNF Boolean formula), we define a certificate C as a truth assignment of the variables that makes the Boolean formula evaluate to 1. Note that the size of $\langle C \rangle$ is upper bounded by the size of $\langle \phi \rangle$.

   Next, we design a polynomial-time verifier for 3SAT.

   V = "On input $\langle \langle \phi \rangle, C \rangle$:

   1. Assign the truth value to the literals according to C.

   2. Evaluate the value of ϕ. If it is true, *accept*; Otherwise, *reject*."

   Finally, we show that V can verify 3SAT in polynomial time. For Step 1, V scans the input string for at most $|\phi| \cdot n$ times, where n is the number of variables and $|\phi|$ denotes the total number of literals in ϕ (since each variable appears at most one in the input, $n \leq |\phi|$). For Step 2, V scans the input string once. Hence, the total running time of V is polynomial in the input string size.

3. **Minimum vertex cover.** Given graph $G = (V, E)$, a *vertex cover* is a subset of vertices $C \subseteq V$ such that for all edge $(u, v) \in E$, u or v is in C. The minimum vertex cover problem asks about finding the minimum vertex cover in the given graph.

   We show the NP-membership of the decision version of the minimum vertex cover problem: Given a graph G and an integer k, is there a vertex cover in G with size at most k?

   For any yes-instance (that is, a graph G with a vertex cover with a size of k), we define a certificate C as a set of k vertices that covers all the edges in G. Note that the size of $\langle C \rangle$ is upper bounded by the size of $\langle G \rangle$ since C is a subset of vertices in G.

   Next, we design a polynomial-time verifier for the decision problem.

   V = "On input $\langle \langle G, k \rangle, C \rangle$:
   Check if each edge in G has at least one endpoint in C.
   If it is true, *accept*; Otherwise, *reject*."

   Finally, since each edge in G needs to scan the input $\langle \langle G, k \rangle, C \rangle$ at most once, the total running time of V is polynomial in the input string size.

4. **Maximum independent set.** An *independent set* is a set of vertices where each pair of vertices is not adjacent to each other. The *maximum independent set problem* is: given a graph G, what is the size of the maximum independent set in G?

   We show the NP-membership of the decision version of the maximum independent set problem: Given a graph G and an integer k, is there an independent set in G with size at least k?

   For any yes-instance (that is, a graph G with an independent set with a size of k), we define a certificate C as a set of k vertices that are mutually nonadjacent in G. Note that the size of $\langle C \rangle$ is upper bounded by the size of $\langle G \rangle$ since C is a subset of vertices in G.

   Next, we design a polynomial-time verifier for the decision problem.

   V = "On input $\langle \langle G, k \rangle, C \rangle$:
   Check if each pair of vertices in C are non-adjacent in G.
   If it is true, *accept*; Otherwise, *reject*."

   Finally, since each pair in C needs to scan the input $\langle \langle G, k \rangle, C \rangle$ at most once, and there are at most k^2 pairs of vertices in C. Hence, the total running time of V is polynomial in the input string size.

5. **Bin packing.** Given n items, where each item i with size $s_i \in (0, 1]$, the goal is packing the items into the minimum number of capacity-1 bins.

   We show the NP-membership of the decision version of the bin packing problem: Given a set of n items with sizes $s_1, s_2, \cdots, s_n$ and an integer k, is it possible to pack the items into k capacity-1 bins?

   For any yes-instance, we define a certificate C as an assignment of n items into k bins by attaching each item an index of the bin it is assigned to. The size of $\langle C \rangle$ is $O(n \log k)$, which is polynomial in the length of the input $\langle s_1, s_2, \cdots, s_n, k \rangle$.

   Next, we design a polynomial-time verifier for the decision problem.

   V = "On input $\langle \langle s_1, s_2, \cdots, s_n, k \rangle, C \rangle$:
   For $i = 1, 2, \cdots, n$, check if the total size of all items assign to the same bin with item i is at most 1. If it is true, *accept*; Otherwise, *reject*."

   Finally, since each round needs to scan the input $\langle \langle s_1, s_2, \cdots, s_n, k \rangle, C \rangle$ at most once and there are n rounds, the total running time of V is polynomial in the input string size.

6. **Knapsack.** There are n items, each with positive integral *weight* w_j $(j = 1, \cdots, n)$ and positive integral *value* c_j $(j = 1, \cdots, n)$ and an integer b. The question is to find a subset of the items with total weight at most b and the maximal total value.

   We show the NP-membership of the decision version of the knapsack problem: Given a set of n items with positive integral *weight* w_j $(j = 1, \cdots, n)$ and positive integral *value* c_j $(j = 1, \cdots, n)$, a budget b, and a value k, is there a set of items with total weight of at most b and total value of at least k?

   For any yes-instance, we define a certificate C as a subset of items that has total weight at most b and total value at least k The size of $\langle C \rangle$ is $O(n \log n)$, which is polynomial in the length of the input $\langle w_1, s_2, \cdots, w_n, c_1, c_2, \cdots, c_n, b, k \rangle$ since C is a subset of the items.

   Next, we design a polynomial-time verifier for the decision problem.

   V = "On input $\langle \langle w_1, s_2, \cdots, w_n, c_1, c_2, \cdots, c_n, b, k \rangle, C \rangle$:
   Check if the total weight of the items in C is at most b and if the total value is at least k.
   If it is true, *accept*; Otherwise, *reject*."

   Finally, since summing up the weights and the values needs to scan the input $\langle \langle w_1, s_2, \cdots, w_n, c_1, c_2, \cdots, c_n, b, k \rangle, C \rangle$ at most once per item and there are at most n items in C, the total running time of V is polynomial in the input string size.