Question-9 Solution

Part A:

Knapsack: Dynamic programming II (Knapsack II) is an optimization problem where for given set S of n items, with w_i & v_i as weights & values(profit). We seek to find subset $S' \subseteq S$ such that weight of subset is minimized. Which has $OPT(i,v)$ as:-

$$OPT(i,v) = \begin{cases} 0 & \text{if } v = 0 \\ \infty & \text{if } i = 0, v > 0 \\ OPT(i-1, v) & \text{if } v_i > v \\ \min\{OPT(i-1,v),\ w_i + OPT(i-1, v - v_i)\} & \text{otherwise} \end{cases}$$

If, above problem has to written in Decision problem format, it can be written as:- Given a finite set S, non-negative weights w_i and non-negative profit (value) v_i, if there a subset $S' \subseteq S$ such that sum of profit/values obtains arbitrary integer V and weights of subset is minimized.

i.e

$$\sum_{S_i \in S'} w_i \leq W \ (\text{minimized})$$

$$\sum_{S_i \in S'} v_i = V$$

- Subset-SUM problem:- Given a finite set Z of k integers and an arbitrary integer X, is there exists a subset $Z' \subseteq Z$ such that sum of elements in subset equals to X.

i.e

$$\sum_{Z_i \in Z'} Z_i = X$$

According to the question:- SUBSET-SUM problem is NP-Complete

* I would like proceed proof to show decision version of knapsack-II is NP-Complete using specific to Generic case reduction

I claim that we can reduce SUBSET-SUM problem to Knapsack problem

i.e $\quad$ SUBSET-SUM $\leq_P$ Knapsack

Proof :- Given for instance of subset $(z_1, z_2 \ldots z_k, \cancel{X})$, let us create Knapsack instance as

$$\begin{cases} v_i = w_i = z_i \\ V = W = \cancel{X} \end{cases}$$

Based on how professors proves the reduction in the lecture.

Based on ~~problem~~ definitions that we have defined in the start of the ~~problem~~ problem:- for knapsack

$$\begin{cases} \sum_{s_i \in S'} w_i \le W \Rightarrow \sum_{z_i \in S'} z_i \le \cancel{X} \\ \sum_{s_i \in S'} v_i = V \Rightarrow \sum_{z_i \in S'} z_i = X \end{cases} \Rightarrow \sum_{z_i \in Z'} z_i = \cancel{X}$$

which ends up as the same solution of Subset Sum problem.

This proves that one should be able to reduce instance of subset problem to instance of knapsack problem in polynomial time.

Hence :- <u>SUBSET-SUM $\le_p$ knapsack</u>

Based on establish intractability, when $X \le_p Y$ and if X is NP-complete, Y is also NP-Complete

Since, subset $\le_p$ knapsack, we can derive that

<u>Decision Version of knapsack problem is NP-Complete</u>

→ As mentioned, specific to generic is used here as I wanted to prove if one is able to knapsack problem [which specific to weights and value], ~~the~~ subset-sum problem can also be solved. [which is the generic version of Knapsack problem with subset-sum equals to constant given value]

Moreover, the above reduction was not an example of equivalent or not an example of encoding with gadgets.

Part B:-

Optimization version of knapsack II indicates the Dynamic programming version of it. As defined in the part A, dynamic version of knapsack-II is defined as

$$OPT(i,v) = \begin{cases} 0 & \text{if } v=0 \\ \infty & \text{if } i=0, v>0 \\ OPT(i-1,v) & \text{if } v_i > v \\ \min\{OPT(i-1,v),\ w_i + OPT(i-1, v-v_i)\} & \text{otherwise.} \end{cases}$$

As mentioned in the class / lecture, the Runtime complexity of knapsack II is $O(nV^*) = \underline{O(n^2 V_{max})}$ n.

$\boxed{V^* \leq V_{max}}$ $V^* \rightarrow$ max value of Profit/Value

The above algorithm is NON-POLYNOMIAL on input.

→ Hence optimization version of knapsack-II is $\boxed{\text{NOT}}$ NP-complete. It is Non-polynomial.

Part C:-

$\frac{1}{2}$ approximation algorithm of knapsack (I)

optimal knapsack I is defined as:-

$$OPT(i,w) = \begin{cases} 0 & \text{for } i=0 \\ OPT(i-1,w) & \text{for } w_i > w \\ \max\{OPT(i-1,w),\ v_i + OPT(i-1, w-w_i)\} & \text{otherwise} \end{cases}$$

$\frac{1}{2}$ approximation indicates that approximation constant to be $1/2$:

which implies :- Approximated Sol'n $= \frac{1}{2}$ Optimised Alg.

This implies there exists no optimization solution in specific but Approximated Alg using Greedy Alg techniques.

which makes

Approximated Alg $=$ $(1-\epsilon)$ Optimised Alg

$$1 - \epsilon = 1/2$$

$$\underline{\epsilon = 1/2}$$

This can be proved correct using the theorem given in the lecture by professor :-

that is: If S is a solution found by our Algorithm and S^* is any other feasible solution then Approximate $\geq$ better feasible solution.

Based on the lecture:-

the runtime of Alg would be:-

$$\boxed{O(n^3/\epsilon)} \rightarrow O(n^3/_{1/2})$$

$$\rightarrow \boxed{O(2n^3)}$$

Above Algorithm works based on the Rounding & Scaling where

$$\bar{v}_i = \left\lceil \frac{v_i}{\theta} \right\rceil \theta , \quad \hat{v}_i = \left\lceil \frac{v_i}{\theta} \right\rceil$$

$V_{max} \rightarrow$ largest value of value/profit

$\epsilon \rightarrow$ precision parameter

$\theta \rightarrow$ Scaling factor $= \frac{\epsilon V_{max}}{n}$

$$\theta = \frac{\epsilon V_{max}}{n}$$