San Skulrattanakulchai
Gustavus Adolphus College Nov 13, 2015

Subset Sums (Variant) and Knapsacks

GT: Ch 12.6

Subset Sum Problem (Variant) Let there be given n items with positive weights $w_1, w_2, \ldots, w_n$. Also, let there be given an upper bound weight W. For any subset S of the items, define $wt(S)$ to be the sum of the weights of all items in it, i.e., $wt(S) = \sum\{ w_i : i \in S \}$. We wish to find a subset S^* of maximum weight not exceeding W, i.e.,

$$wt(S^*) = \max\{ wt(S) : S \text{ is a subset of the given items and } wt(S) \leq W \}.$$

Dynamic Programming Solution

For all $1 \leq i \leq n$ and $0 \leq w \leq W$, define $m(i, w)$ to be the maximum weight, not exceeding w, achievable by choosing some subset of the first i weights $w_1, w_2, \ldots, w_i$.

We seek $m(n, W)$.

Optimal Substructure Property

Fix i and w and let S^* be a subset of the first i items whose weight sum equals $m(i, w)$. There are 2 cases to consider.

Case 1: $w_i > w$. Then item i does not belong to S^*, for otherwise we would have $wt(S^*) \geq w_i > w$, a contradiction. So in Case 1 we have $m(i, w) = m(i-1, w)$.

Case 2: $w_i \leq w$. There are two subcases: either item i does or does not belong to S^*. If item i does belong to S^*, we have $m(i, w) = w_i + m(i-1, w-w_i)$. If item i does not belong to S^*, we have $m(i, w) = m(i-1, w)$. Exactly one of these cases must happen in the optimal packing. So in Case 2 we have $m(i, w) = \max\{ w_i + m(i-1, w-w_i),\ m(i-1, w) \}$.

Either Case 1 or Case 2 occurs.

Recurrence The above reasoning leads us to the recurrence

$$m(i, w) = \begin{cases} 0 & \text{if } i = 0 \text{ or } w = 0 \\ m(i-1, w) & \text{if } i \geq 1,\ w > 0,\ w_i > w \\ \max\{ w_i + m(i-1, w-w_i),\ m(i-1, w) \} & \text{if } i \geq 1,\ w > 0,\ w_i \leq w. \end{cases}$$

Running Time
In Pass 1, we fill in $O(nW)$ table entries, in $O(1)$ time per entry, $O(nW)$ time total.
In Pass 2, we trace backwards through $O(n+W)$ table entries, in $O(1)$ time per entry, $O(n+W)$ time total.

Remark
Algorithm does not prove that $P = NP$.

Knapsack Problem Let n items be given with values $v_1, v_2, \ldots, v_n$, and positive weights $w_1, w_2, \ldots, w_n$, respectively. Also let an upper bound weight W be given. For any subset S of the given set of items, define $v(S)$ to be the sum of the values of all items in it, i.e., $v(S) = \sum\{\, v_i : i \in S \,\}$. (We also define $wt(S)$ to be the sum of the weights of all items in it, i.e., $wt(S) = \sum\{\, w_i : i \in S \,\}$.) We wish to find a subset S^* of maximum value whose total weight $wt(S)$ does not exceed W, i.e.,

$$v(S^*) = \max\{\, v(S) : S \text{ is a subset of the set of given items and } wt(S) \le W \,\}.$$

Dynamic Programming Solution
This problem can be solved in essentially the same way as the above Subset Sum Problem. For $1 \le i \le n$ and $0 \le w \le W$, define $m(i, w)$ to be the maximum value achievable by choosing some subset of the first i items subject to its total weight not exceeding w.

We seek $m(n, W)$.

We can show this problem has Optimal Substructure Property. The recurrence for $m(i, w)$ is

$$m(i,w) = \begin{cases} 0 & \text{if } i = 0 \text{ or } w = 0 \\ m(i-1, w) & \text{if } i \ge 1,\ w > 0,\ w_i > w \\ \max\{\, v_i + m(i-1, w - w_i),\ m(i-1, w) \,\} & \text{if } i \ge 1,\ w > 0,\ w_i \le w. \end{cases}$$

Running Time
Same as the above Subset Sum Problem.

Exercise
Show that this handout's variant of the Subset Sum Problem reduces to the Knapsack Problem. In fact, the Knapsack Problem is a generalization of the Subset Sum Problem.