Dynamic Programming for Number Problems

K. Subramani
Department of Computer Science and Electrical Engineering,
West Virginia University,
Morgantown, WV
ksmani@csee.wvu.edu

1 Introduction

Dynamic Programming is a general technique to systematically compute solutions to problems, which are characterized by a large number of decision variables. The following issues are key in dynamic programming:

1. Existence of decision variables - Typically, there are \( n \) binary (0/1) decision variables;
2. Existence of a “State” - The state of the system at a particular point in time is characterized by the decisions taken on the decision variables at that time and the subsequent change in problem parameters;
3. Existence of optimal sub-structure - Let us say that decisions on some variables \( x_1, x_2, \ldots, x_i \) have been made to reach the current state \( A \). As a result the original problem is transformed into a new problem with changes in its parameters. The optimal sub-structure property states that the rest of the decisions i.e. decisions on the the variables \( x_{i+1}, \ldots, x_n \) still have to constitute an optimal decision with respect to the new problem.

We now observe these features in 2 problems that are commonly tackled through Dynamic Programming. Section §2 discusses the Partitioning problem, while Section §3 discusses the 0/1 Knapsack problem.

2 Partitioning

Partitioning is concerned with splitting a set into 2 parts with equal sums. In general, the addition operator, can be replaced by any other group operator.

2.1 Statement of Problem

Given a set \( A = \{a_1, a_2, \ldots, a_n\} \), such that \( a_1 \geq 0 \), is there a subset \( S \subseteq A \), such that

\[
\sum_{a \in S} a = \sum_{a \in A-S} a
\]

Observe that if \( \sum_{i=1}^{n} a_i = M \) is an odd number the answer is immediately “no”, since an odd number cannot be broken into two integral parts. In fact, the sum of the elements in the two subsets \( S \) and \( A - S \) must equal \( \frac{M}{2} \).

2.2 Casting as a Dynamic Program

Associate a decision variable \( x_i \) with each \( a_i \), where

\[
x_i = \begin{cases} 
1, & \text{if } a_i \in S \\
0, & \text{if } a_i \notin S 
\end{cases}
\]
Thus, a sequence of decisions have to be made on variables $x_1$ through $x_n$. The state of the system is characterized by $M = \sum_{a_i \in S} a_i$ i.e. the space available for moving new numbers into $S$.

We define $m[i, j]$ to be $T$ (true), if some subset of the elements in $\{a_1, a_2, \ldots, a_i\}$ has elements that add up to $j$. In this notation, $m[n, \frac{M}{2}]$ is the answer to our question, i.e. the answer to the input problem is “yes” if and only if $m[n, \frac{M}{2}]$ is $T$.

The key observation is that $m[i, j]$ can be true if and only if one of the following holds:

- $m[i - 1, j]$ is $T$. Clearly if there is a subset of the first $i - 1$ elements that sums to $j$, the same subset can be used as the subset of the first $i$ elements that sums to $j$. This corresponds to the case of assigning $x_i = 0$;
- $m[i - 1, j - a_i]$ is $T$. If $a_i$ is to be included in the subset of $\{a_1, a_2, \ldots, a_i\}$ that sums to $j$, then there must exist some subset of the first $i - 1$ elements that sums to $j - a_i$. This corresponds to the case of assigning $x_i = 1$.

Proceeding this way, we can build a table $m[1..n, 0..\frac{M}{2}]$ and check whether $m[n, \frac{M}{2}]$ is $T$.

3 0/1 Knapsack

3.1 Statement of Problem

Given a set of objects $O = \{o_1, o_2, \ldots, o_n\}$, with associated profits $\{p_1, p_2, \ldots, p_n\}$ and weights $\{w_1, w_2, \ldots, w_n\}$ and a knapsack of capacity $M$, decide which objects are to be placed in the knapsack, so as to maximize the profit, while respecting the capacity constraint.

3.2 Casting as a Dynamic Program

We associate a decision variable $x_i$ with object $o_i$, such that

$$
\begin{align*}
x_i &= 1, \text{ if object } o_i \text{ is in the knapsack} \\
&= 0, \text{ otherwise.}
\end{align*}
$$

Once again a sequence of decisions have to be made on the variables. Observe that if $x_1 = 0$, then the new subproblem is characterized by the set $\{o_2, o_3, \ldots, o_n\}, M, 0$ while if $x_1 = 1$, the new subproblem is $\{o_2, o_3, \ldots, o_n\}, M, p_1$ i.e. the profit has increased by $p_1$, while the available space has decreased by $M - p_1$.

Once again, we define $m[i, j]$ to be the maximum profit that can be realized by packing some subset of the first $i$ objects, into a knapsack of capacity $M$. Using this notation, clearly the entry $m[n, M]$ is what we seek. The crucial observation is that

$$
m[i, j] = \max\{m[i - 1, j], m[i - 1, j - w_i] + p_i\}
$$

The point is that if the decision on $o_i$ is to exclude it, then we solve a sub-problem $m[i - 1, j]$; if we choose to include it, then our profit increases by $p_i$, but now the available capacity has decreased by $w_i$.

Proceeding thus, we build the table $m[1..n, 0..M]$ and output $m[n, M]$.

4 Conclusion

Also read the solution to Quiz II.