University of Nevada, Las Vegas Computer Science 477/677 Spring 2026

Assignment 3: Remarks

There are no new problems added to homework3, but read this anyway.

The knapsack problem, as given in that homework3, is too hard for the exam. It is still part of the homework3, though.

A Coin Row Problem. An easier dynamic programming problem, which could be on the exam, is the following variation of the coin-row problem. You are given a row of n coins $c_1, c_2, \ldots c_n$, each of positive value. Write an algorithm which selects a subset of those coins of maximum total value, subject to the condition that the selected set contains no three consecutive coins.

Let $x_1, \ldots x_n$ be the values of the coins, where x_i is the value of the i^{th} coin. We define $A]i]$ to be the value of S_i, the maximum weigth *legal* subset of those coins; that is, S_i cannot contain any three consecutive coins. We initialize the program by writing $A[0] = 0$, $A[1] = x_1$, and $A[2] = x[1] + x[2]$. We then dynamically compute $A[i]$ for all $i \geq 3$.

Since $S[i]$ cannot contain c_i, c_{i-1}, and c_{i-2}, We can naturally break the problem into three cases.

Case I: $c_i \notin S_i$. In this case, S_i must be a the maximum weight legal subset of the first $i-1$ coins, hence $S_i = S_{i-1}$, and we have $A[i] = A[i-1]$.

Case II: $c_{i-1} \notin S_i$. Without loss of generality, $c_i \in S_i$, and $S_i = \{c_i\} \cup S_{i-2}$. Thus $A[i] = x_i + A[i-2]$.

Case III: $c_{i-2} \notin S_i$. Without loss of generality, $c_i, c_{i-1} \in S_i$. Thus, $A[i] = x_i + x_{i-1} + A[i-3]$

$A[i]$ is the maximum of those three values, that is, $A[i] = \max\{A[i-1], x_i + A[i-2], x_i + x_{i-1} + A[i-3]\}$.

The solution to that problem instance is $A[n]$.

Pseudo-Polynomial Time Algorithm for Subset Sum. The knapsack problem is based on the subset sum problem. Given a sequence of n positive numbers $x_1, x_2, \ldots x_n$, and a number K, does there exist a subsequence whose terms total K? This problem is known to be $\mathcal{NP}$-complete, but there is a well-known pseudo-polynomial, actually $O(nK)$ time, algorithm as given below. That algorithm is not polynomial time unless K is small. Here is a description of that algorithm.

We can assume, without loss of generality, that no term x_i is larger than K.

We initialize a Boolean array A to be false for all $0 \leq m \leq K$. The meaning of $A[m]$ is that a subsequence whose total is m has been found.

```
For all i from 1 to n
    A[x_i] = 1 \\ (true)
    For all m from K − 1 to 1 \\ (why backwards?)
        If (m + x_i <= K)
            A[m + x_i] = 1
If (A[K]) return 1
Else return 0
```

Knapsack Problem. We are given a list of objects each of which has a value and a weight, and a knapsack which can only hold a given maximum weight. Write an algorithm which finds the maximum value of any set of those objects whose total weight does not exceed the capacity of the knapsack. The knapsack problem is clearly at least as hard as the subset sum problem. We extend the algorithm given above for subset sum to knapsack.

Let $c_1, \ldots c_n$ be the objects, and let w_i be the weight of c_i, and v_i its value. We assume all w_i are positive integers, and all v_i are positive. Let K be the capacity of the knapsack. Without loss of generality, $w_i \leq K$ for all i.

We compute an array $V[n+1][K+1]$, where $V[i][m]$ is the maximum value of any subset of $\{c_1, \ldots c_i\}$ whose total weight is less than or equal to m. The maximum value of any knapsack of objects is $V[n][K]$.

We first set $V[0][m] = 0$ for all $0 \leq m \leq K$, then compute values of $V[i][m]$ for $i > 0$ and for all m by dynamic programming.

$V[0][m] = 0$ for all m from 0 to K.
For all i from 1 to n
    For all m from 0 to $w_i - 1$
        $V[i][m] = V[i-1][m]$ \\ since c_i is not used in that range
    For all m from w_i to K
        $tmp = v_i + V[i-1][m - w_i]$ \\ value of a set which includes c_i
        If $tmp > V[i][m]$ \\ if that value is greater than the previous value
            $V[i][m] = tmp$
Return $V[n][K]$

Note that $V[i-1, m] \leq V[i, m]$, and $V[i, m] \leq V[i][m+1]$.