Homework 1: Solutiuon

CS 4364/5364
Spring 2021

Due: 10 February 2021

1. **(20 points) We talked in class about writing the knapsack problem as an integer linear program. Remember for the knapsack problem (without replacement) you are given the following:**

   - w_i – the weight of item i for $i = 1 \ldots n$,
   - v_i – the value of each item i for $i = 1 \ldots n$, and
   - m – the maximum weight you can cary in your knapsack.

   The goal is to find a set of items with maximum value such that the total weight of the items is less than or equal to the maximum weight you can cary.

   Your task for this homework assignment is to formalize the ILP for the 0/1 knapsack problem, and provide an explanation to how each component (each constraint or group of constraints and the objective equation) contributes to solving the problem. The definition should be in the same form as the example below. The explanation should be in paragraph form following the solution. The *variables* in your ILP will be the set x_i of binary values (i.e. $x_i \in \{1, 0\}$) that states if item i is in (1) or not in (0) your knapsack.

   The solution:

$$\begin{aligned} \text{maximize} \quad & \sum_{i=1\ldots n} v_i \, x_i \\ \text{subject to} \quad & \sum_{i=1\ldots n} w_i \, x_i \leq m, \\ & x_i \in \{0, 1\}, \qquad i = 1 \ldots n \end{aligned}$$

   The explanation:

   In the linear program above we have a single group of binary variables x_i, one for

each possible item we can put in our knapsack. The last line in our program restricts these variable to either being 1 (meaning the item is in our knapsack) or 0 (meaning it is not).

Once we know the x variables have to be 0 or 1 we can move on to the objective function, where we trying to maximize the function $\sum_{i=1...n} v_i\ x_i$. Because x_i is 0 or 1 the value of the item is only added into the sum if that item is chosen. Therefore this function represents the total value of all of the items included in the solution. Because the problem definition asks for the maximum value, we're going to try and maximize this function.

Finally, we are restricted to a certain weight. Just as we did with the total value, we use the fact that the x values are 0 or 1 to calculate a total weight of items in our knapsack. We then bound that by the total weight we can carry. Without this constraint the solution would be to include all items (this would maximize the objective function) but it violates the weight condition in the problem.

2. **(20 points) Give an algorithm that takes in two string α and β, of length n and m, and finds the longest suffix of α that exactly matches a prefix of β. The algorithm should run in $O(n+m)$ time. (Hint: your algorithm will likely rely on the fact that the $Z_i(S)$ values can be computed in time thats linear with respect to the length of a string S).**

The algorithm:

(a) construct a new string $S = \beta\$\alpha$ where the character \$ is not found in either α or β.

(b) calculate $Z_i(S), \forall i = 1....|\alpha| + |\beta| + 1$.

(c) starting with $i = |\beta| + 1$ scan in increasing values of i

- for each i check if $Z_i(S) = |\alpha|+|\beta|+1-i$, if it is return the $\beta[1...|\alpha|+|\beta|+1-i]$ (which is also $\alpha[i - |\beta| - 1...|\alpha|]$).

(d) if the condition is never met return the empty string.

Correctness:

We know that the Z value tells us the length of the substring starting at a given position that exactly matches a prefix of the same string. Since we are trying to match the prefix of β we need to put it first when constructing our special string. Then when we calculate all of the Z values, if the length of the matching substring is the length thats left in the string we know its a suffix of α. By scanning from the left most character of α in S, the first position we find that matches this criteria will

be the longest, if there was a longer one it would have had to have started earlier in α.

Complexity:

Step (a) takes $O(n + m)$ time (really $m + n + 1$), linear time to construct a new string. We know from class that we can calculate all of the Z values for a string in time linear with respect to the string length, which in this case is $O(m + n)$. The loop in (c) runs at most $O(n)$ times, and the work done inside (check and return) are constant time operations. The final step is constant time and may never be reached. Therefore the total running time is $O(m + n)$ (2 items that are $O(m + n)$, an $O(n)$ loop, and a $O(1)$ operation).