Intro to Algorithms, COMP-160, Homework #8

Benjamin Tanen, 03/31/2016

1. **You've just acquired a whole lot of candy. Specifically, you have n unit pieces. Now you want to ship it all to your favorite charity. The post office gives you a lost of k different box sizes, each of which can be packed with some integer amount of candy. In order words, for $1 \leq i \leq k$, box i fits b_i pieces of candy, where b_i is an integer. You observe that $b_1 = 1$. Purchasing and shipping a box costs a dollar, regardless of size. There is an unlimited number of boxes available, for each size.**

   Every box you use must be fully packed, to prevent damage. Figure out how to make the delivery, paying as little as possible. What is the time and space complexity of finding the optimal solution?

   Say we have n pieces of candy and we need to pack all of it into boxes of varying sizes, minimizing the total number of boxes. We have k different types of boxes where B is the set of all boxes. We can describe the cost of packing all of n pieces of our candy as $C(n) =$ the number of boxes used.

   By using the approach of dynamic programming, we can solve for $C(n)$ by first solving the problem for 1 to $n-1$ pieces of candy. We know that $C(1) = 1$ because we have a box that holds $b_1 = 1$ pieces of candy and that $C(0) = 0$ because we need zero boxes to pack zero pieces of candy.

   From here, we can consider $C(i)$ where $1 \leq i \leq n$ and where we have already calculated $C(j)$ for all $j < i$. To do this, we can consider each box b in B that holds b pieces of candy (NOTE: every box b that we consider from B must hold less than or equal to i pieces of candy - otherwise we are wasting space). We want to minimize the total number of boxes that we will use so we want to find the box b that minimizes $C(i-b)$. Once we have found the box b that minimizes the value of $C(i-b)$, we know there exists one box that can hold b pieces of candy. This means we only need one more box than the number of boxes necessary for $C(i-b)$. Therefore, we can then calculate $C(n) = 1 + \min_{b \in B}\{C(n-b)\}$.

   In order to calculate any problem (or subproblem), we must loop over all k of our boxes b where $b \leq n$ and query $C(i-b)$ (this takes $\Theta(k)$ time per problem). This also requires that we first calculate each subproblem $C(j)$ (where $j < i$). Therefore, the time complexity cost of calculating $C(i)$ is $\Theta(ik)$. Thus the time complexity of calculating $C(n)$ is $O(nk)$.

   Finally, since we can store the sizes of our k boxes and the results of each iteration $C(i)$ (for $0 \leq i \leq n$) in two one-dimensional array of lengths k and n respectively, our space complexity for this problem is $\Theta(n+k)$.