CSCI3160: Regular Exercise Set 6

Prepared by Yufei Tao

Problem 1*. Let A be an array of n integers. Define a function $f(x)$ — where $x \geq 0$ is an integer — as follows:

$$f(x) = \begin{cases} 0 & \text{if } x = 0 \\ \max_{i=1}^{x}(A[i] + f(x-i)) & \text{otherwise} \end{cases}$$

Consider the following algorithm for calculating $f(x)$:

algorithm $f(x)$
1. **if** $x = 0$ **then return** 0
2. $max = -\infty$
3. **for** $i = 1$ **to** x
4. $\quad v = A[i] + f(x-i)$
5. $\quad$ **if** $v > max$ **then** $max = v$
6. **return** max

Prove: the above algorithm takes $\Omega(2^n)$ time to calculate $f(n)$.

Problem 2. Consider once again Problem 1. Design an algorithm to calculate $f(n)$ in $O(n^2)$ time.

Problem 3. Recall that, on the optimal BST problem, we have explained in the class how to calculate $optavg(1, n)$ using dynamic programming in $O(n^3)$ time where function $optavg(a, b)$ is recursively defined as

$$optavg(a,b) = \begin{cases} 0 & \text{if } a > b \\ \sum_{i=a}^{b} W[i] + \min_{r=a}^{b}\{optavg(a, r-1) + optavg(r+1, b)\} & \text{otherwise} \end{cases}$$

However, we have not yet explained how to build in an optimal BST. Describe an algorithm to do so in $O(n^3)$ time (in fact, you can build the tree in $O(n)$ time after having computed $optavg(1, n)$, but you will need to modify what we did in dynamic programming slightly).

Problem 4 (Rod-Cutting; Section 15.1 of the Textbook). Let A be an array of n integers. Let us define an *n-sum sequence* as a sequence of integers $x_1, x_2, ..., x_t$ (where t can be any integer at least 1) satisfying both conditions below:

- $1 \leq x_i \leq n$ for all $i \in [1, t]$
- $\sum_{i=1}^{t} x_i = n$.

Define the *cost* of the above n-sum sequence as $\sum_{i=1}^{t} A[x_i]$. Give an algorithm to produce an n-sum sequence with the largest cost in $O(n^2)$ time.