CSC 226 - Assignment 1

Sample Solution

1 Evaluating Polynomials

a) The naive approach $p(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + \ldots + a_nx^n$

$val = a_0$	
$for\ i = 1\ to\ n\ do:$	n iterations
$\quad val\ += a_ix^i$	i multiplications and 1 addition
$return\ val$	

The ith term, a_ix^i, has i multiplications, thus $1+2+3+\ldots+n = \frac{n(n+1)}{2}$ total multiplications. There are also n additions in the polynomial, hence running time for this method is $T(n) = \frac{n(n+1)}{2} + n \in O(n^2)$.

b) using the nested form: $p(x) = a_0 + x(a_1 + x(a_2 + x(a_3 + \ldots + x(a_{n-1} + xa_n)\ldots)))$

$val = a_{n-1} + xa_n$	1 add and 1 multiply
$for\ i = n-2\ to\ 1\ do:$	$n-2$ iterations
$\quad val\ += a_i$	1 addition
$\quad val\ *= x$	1 multiply
$val += a_0$	1 addition
$return\ val$	

Using this algorithm, it is easy to see there are n additions and $n-1$ multiplications occurring in the evaluation of $p(x)$ and thus $T(n) = n - 1 + n \in O(n)$

2 The 3SUM Problem

Given an array of n integers, determine if there are 3 integers in the array that sum to zero.

sort(array)	$O(nlogn)$
$for\ i = 0\ to\ n-1\ do:$	n iterations
$\quad j = i+1$	
$\quad k = n-1$	
$\quad while\ j < k\ do:$	$n-i-3$ iterations
$\quad\quad val = array[i] + array[j] + array[k]$	
$\quad\quad if\ val > 0:$	
$\quad\quad\quad k = k-1$	
$\quad\quad else\ if\ val < 0:$	
$\quad\quad\quad j = j+1$	
$\quad\quad else:$	
$\quad\quad\quad return\ TRUE$	
$return\ FALSE$	

Asymptotic Analysis:
In a worst-case scenario, there will be no triple that sums to zero. That is, the algorithm with have to check every possible combination before returning FALSE. The for loop will do n iterations and on each iteration, the while loop will also do $n-i-3$ iterations.

$$\sum_{i=0}^{n-1} n - i - 3 = \sum_{i=0}^{n-1} n - \sum_{i=0}^{n-1} i - \sum_{i=0}^{n-1} 3 = n^2 - \frac{n(n+1)}{2} - 3n \quad \therefore \quad T(n) \in O(n^2)$$

3 linearSelect - grouping by 3

Modify linearSelect such that pickCleverPivot groups by 3.

step 1: divide array into $\frac{n}{3}$ groups of size 3, takes $O(1)$ time
step 2: sort each group of size 3 (at most 3 comparisons), takes $\frac{n}{3} \times 3 = n$ time
step 3: determine the median of each group and gather medians, takes n time
step 4: use linearSelect recursively to determine the median of medians
$\Rightarrow$ if linearSelect is $T(n)$, then this step is $T(\frac{n}{3})$

$\therefore$ pickCleverPivot is $2n + T(\frac{n}{3})$.

By selecting the pivot this way, we guarantee that $2 \times \frac{n/3}{2} = \frac{n}{3}$ elements are less than the pivot, and thus $n - \frac{n}{3} = \frac{2n}{3}$ elements are greater than the pivot (or vice versa). This means, in the worst-case scenario, that the conquer step is $T(\frac{2n}{3})$.

$\therefore\ T(n) = 3n + T(\frac{n}{3}) + T(\frac{2n}{3})$

Now, I will show $T(n) \notin O(n)$:

Suppose for contradiction that $T(n) \leq cn$ for some constant c, then
$T(n) = 3n + T(\frac{n}{3}) + T(\frac{2n}{3})$
$\leq 3n + \frac{cn}{3} + \frac{2cn}{3}$
$= 3n + cn$
this implies $3n + cn \leq cn \ \Leftrightarrow\ 3n \leq 0$ which is impossible

Therefore, this modified version of linearSelect does not run in $O(n)$ time.

4 The Master Theorem

a) $T(n) = 16T(\frac{n}{4}) + n^4$

Since $log_b(a) = log_4(16) = 2 < 4 = c$, we have case (c) of The Master Theorem $\Rightarrow\ T(n) \in \Theta(n^4)$

b) $T(n) = 125T(\frac{n}{5}) + n^2$

Since $log_b(a) = log_5(125) = 3 > 2 = c$, we have case (a) of The Master Theorem $\Rightarrow\ T(n) \in \Theta(n^3)$

c) $T(n) = 64T(\frac{n}{8}) + n^2$

Since $log_b(a) = log_8(64) = 2 = c$, we have case (b) of The Master Theorem $\Rightarrow\ T(n) \in \Theta(n^2 logn)$

5 Matrix Multiplication

Strassen's algorithm: $T(n) = 7T(\frac{n}{2}) + \Theta(n^2) \in O(n^{2.81})$ which is case (a) of the Master Theorem.

General divide-and-conquer algorithm: $T(n) = aT(\frac{n}{b}) + \Theta(n^c)$

If this algorithm is going to do better than Strassen's method, the value of b (the factor by which the subproblem size decreases) must increase. This way, we stay in case (a) of the Master Theorem and the value of $log_b(a)$ will decrease and thus having a better runtime than Strassen's.

Suppose $T(n) = aT(\frac{n}{3}) + \Theta(n^2)$, then in order to beat Strassen's method we need

$$log_3(a) \leq log_2(7)$$

$$\Leftrightarrow a \leq 7^{1/log_3(2)} \approx 21.85$$

Therefore, a (the number of subproblems) must be less than or equal to 21 in order to beat Strassen's method.