# Assignment 3

Benjamin Sorenson

October 24, 2015

**Question 1:** There is no comparison sort whose running time is linear for at least half of the $n!$ inputs. From the proof of theorem 8.1, $\lg\left(\frac{n!}{2}\right)$ is still $\Omega(n \lg n)$. Similarly, $\lg((n-1)!)$ is also still $\Omega(n \lg n)$.

**Question 2:**

(a) By using this algorithm we can gaurentee that we avoid the worst case for `QUICKSORT`. We can add a call to `SELECT` to select the pivot corresponding to the $i^{th}$ order statistic (the median for example) for the input array $A$, avoiding the worst-case scenario for `QUICKSORT`—the case when there are $n-1$ elements on one side of the pivot and 0 on the other. Since `PARTITION` and `SELECT` run in $\Theta(n)$ time this means that we never have to worry about the worst case ($\emptyset(n^2)$), and we don't increase in the asymptotic running time of `PARTITION`.

(b) This bound is preferred because it is a guarantee not a probabilistic expectation.

**Question 3:**

(a)

$$m[i,j] = \begin{cases} 0 & \text{if } i = j; \\ \max\limits_{i \leq k < j} \{m[i,k] + m[k+1,j] + p_{i-1}p_kp_j\} & \text{if } i < j. \end{cases}$$

(b) *Proof.* For the sake of contradiction, assume that there is some parenthesization of the first $k$ $(A_i \ldots A_k)$ and the last $n-k$ $(A_{k+1} \ldots A_n)$ arrays such that $m[1,n] = m[1,k] + m[k+1,n] + p_0p_kp_n$ is the maximum number of possible scalar operations, and there exists some $m[1,k]' > m[1,k]$ then $m[1,n] < m[1,k]' + m[k+1,n] + p_0p_kp_n$ which contradicts the fact that $m[1,n]$ is the maximum number of scalar operations. By the same reasoning, a contradiction also arises if there exists an $m[k+1,n]' > m[k+1,n]$. $\square$

**Question 4:**

```
function MEMOIZED-LCS-LENGTH(X, Y)
    m ← length[X]
    n ← length[Y]
    for i ← 0 to m do
        for j ← 0 to n do
            c[i, j] ← −∞
        end for
    end for
    return LOOKUP-LCS-LENGTH(X, Y, n, m)
end function
function LOOKUP-LCS-LENGTH(X, Y, i, j)
    if c[i, j] > −∞ then
        return c[i, j]
    end if
    if i = 0 ∨ j = 0 then
        c[i, j] ← 0
    else if X[i] = Y[j] then
        c[i, j] ← LOOKUP-LCS-LENGTH(X, Y, i − 1, j − 1)+1
    else
        left ← LOOKUP-LCS-LENGTH(X, Y, i, j − 1)
        up ← LOOKUP-LCS-LENGTH(X, Y, i − 1, j)
        c[i, j] ← max{left, up}
    end if
    return c[i, j]
end function
```

**Question 5:**

```
procedure PRINT-LCS(c, X, i, j)
    if i = 0 ∨ j = 0 then
        return
    end if
    if c[i − 1, j] = c[i, j − 1] = c[i − 1, j − 1] then
        PRINT-LCS(c, X, i − 1, j − 1)
        print X[i]
    else if c[i − 1, j] ≥ c[i, j − 1] then
        PRINT-LCS(c, X, i − 1, j)
    else
        PRINT-LCS(c, X, i, j − 1)
    end if
end procedure
```

**Question 6:** (a) A brute force solution would be to iterate over every possible combination of exchanges of currencies starting with $c_1$ and ending with $c_n$ of lengths $1 \ldots n-1$. Since there are $n-1$ exchange rates in each sequence, the running time of this approach is given by the equation $\sum_{k=1}^{n-1} \binom{n-1}{k}$. Letting $m = n-1$, this becomes $\sum_{k=1}^{m} \binom{m}{k} = \sum_{k=0}^{m} \binom{m}{k} - 1 = 2^m - 1 = 2^{n-1} - 1 = \Theta\left(2^n\right)$. This approach doesn't take into account that each exchange must happen in order (that is, $c_i$ must be exchanged with $c_{i+1} \ldots c_n$), but an approach that did would still be exponential.

(b)

$$P\left(i\right) = \begin{cases} 1 & \text{if } i = 1; \\ \max\limits_{1 \le k < i} \left\{P\left(i-k\right) r_{i-k,i}\right\} & \text{if } i > 1. \end{cases}$$

(c) Suppose there was an optimal sequence of $m$ exchanges such $P(n) = P(k)r_{kn} - \sum_{i=1}^{m} C_i$ where $C_i$ is the commission charged at exchange $i$ yielded the maximum exchange rate. Consider $P(k)' > P(k)$. We cannot guarantee that substituting $P(k)'$ for $P(k)$ would result in a higher return since we don't know the commission charged for $P(k)'$.

x