Fundamental Algorithms - Spring 2018
Homework 7

Daniel Rivera Ruiz
Department of Computer Science
New York University
drr342@nyu.edu

1. Given $X = 10010101$ and $Y = 010110110$ we use the definitions from class to build the matrices c and B:

$$c(i,j) = \begin{cases} 0 & \text{, if } i = 0 \text{ or } j = 0 \\ c(i-1, j-1) + 1 & \text{, if } x_i = y_j \\ \max[c(i-1, j), c(i, j-1)] & \text{, otherwise} \end{cases}$$

$$B(i,j) = \begin{cases} Stop & \text{, if } i = 0 \text{ or } j = 0 \\ \nwarrow & \text{, if } x_i = y_j \\ \leftarrow & \text{, if } x_i \neq y_j \text{ and } c(i, j-1) \geq c(i-1, j) \\ \uparrow & \text{, otherwise} \end{cases}$$

The matrix c looks as follows:

$i_\downarrow / j_\rightarrow$	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	1	1	1	1	1	1	1
2	0	1	1	2	2	2	2	2	2	2
3	0	1	1	2	2	2	3	3	3	3
4	0	1	2	2	3	3	3	4	4	4
5	0	1	2	3	3	3	4	4	4	5
6	0	1	2	3	4	4	4	5	5	5
7	0	1	2	3	4	4	5	5	5	6
8	0	1	2	3	4	5	5	6	6	6

And the matrix B as follows:

$i_\downarrow / j_\rightarrow$	0	1	2	3	4	5	6	7	8	9
0	S	S	S	S	S	S	S	S	S	S
1	S	$\leftarrow$	$\nwarrow$	$\leftarrow$	$\nwarrow$	$\nwarrow$	$\leftarrow$	$\nwarrow$	$\nwarrow$	$\leftarrow$
2	S	$\nwarrow$	$\leftarrow$	$\nwarrow$	$\leftarrow$	$\leftarrow$	$\nwarrow$	$\leftarrow$	$\leftarrow$	$\nwarrow$
3	S	$\nwarrow$	$\leftarrow$	$\nwarrow$	$\leftarrow$	$\leftarrow$	$\nwarrow$	$\leftarrow$	$\leftarrow$	$\nwarrow$
4	S	$\uparrow$	$\nwarrow$	$\leftarrow$	$\nwarrow$	$\nwarrow$	$\leftarrow$	$\nwarrow$	$\nwarrow$	$\leftarrow$
5	S	$\nwarrow$	$\uparrow$	$\nwarrow$	$\leftarrow$	$\leftarrow$	$\nwarrow$	$\leftarrow$	$\leftarrow$	$\nwarrow$
6	S	$\uparrow$	$\nwarrow$	$\uparrow$	$\nwarrow$	$\nwarrow$	$\leftarrow$	$\nwarrow$	$\nwarrow$	$\leftarrow$
7	S	$\nwarrow$	$\uparrow$	$\nwarrow$	$\uparrow$	$\leftarrow$	$\nwarrow$	$\leftarrow$	$\leftarrow$	$\nwarrow$
8	S	$\uparrow$	$\nwarrow$	$\uparrow$	$\nwarrow$	$\nwarrow$	$\leftarrow$	$\nwarrow$	$\nwarrow$	$\leftarrow$

Using B we start from the bottom right corner and follow a path that takes us to an S, looking for the diagonal arrows $\nwarrow$, which are associated to the values present in the LCS. For the current problem we get the following path, where the underlined positions correspond to a $\nwarrow$:

$$\textit{Path} = B(8,9) - \underline{B(8,8)} - B(7,7) - \underline{B(7,6)} - \underline{B(6,5)} - B(5,4) - \underline{B(5,3)} - \underline{B(4,2)} - \underline{B(3,1)}$$

Finally, printing the underlined values of the path in reverse order we get the LCS:

$$\textit{LCS} = 010101$$

2. The following table shows all the parenthesizations for $ABCDE$ according to the recursive formula for $P(n)$ with $n = 5$:

i	$P(i)$	$P(n-i)$	Total	Parenthesizations
1	1	5	5	$A(B(C(DE)))$, $A(B((CD)E))$, $A((BC)(DE))$ $A(((BC)D)E)$, $A((B(CD))E)$
2	1	2	2	$(AB)(C(DE))$, $(AB)((CD)E)$
3	2	1	2	$(A(BC))(DE)$, $((AB)C)(DE)$
4	5	1	5	$(A(B(CD)))E$, $(A((BC)D))E$, $((AB)(CD))E$ $(((AB)C)D)E$, $((A(BC))D)E$
			14	

3. (a) The following algorithm will find all the values $INC[i]$ for $1 \leq i \leq n$ in $O(n^2)$ time:

```
INC[i] ← 1
for i = 2 to n :
    max ← 0
    for j = 1 to i :
        if (x_j < x_i and INC[j] > max) :
            max ← INC[j]
        end if
    end for
    INC[i] ← max + 1
end for
```

(b) The following algorithm will find LIS and DIS in $O(n)$ time, assuming that the values $INC[i]$ and $DEC[i]$ are given for all $1 \leq i \leq n$:

```
LIS ← 1
DIS ← 1
for i = 2 to n :
    if (INC[i] > LIS) :
        LIS ← INC[i]
    end if
    if (DEC[i] > DIS) :
        DIS ← DEC[i]
    end if
end for
```

(c) Since all the x_i are different, there are only two possibilities to consider:

- $x_i < x_j$. In this case we know that $INC[j]$ will at least be equals to $INC[i] + 1$ (in the extreme case where the longest increasing subsequence is formed by the LIS up to x_i plus the x_j element). Therefore, we will always have $INC[i] \neq INC[j]$.
- $x_i > x_j$. In this case we know that $DEC[j]$ will at least be equals to $DEC[i] + 1$ (in the extreme case where the longest decreasing subsequence is formed by the DIS up to x_i plus the x_j element). Therefore, we will always have $DEC[i] \neq DEC[j]$.

(d) Given the sequence of length $m = ab + 1$, we label each number x_i in the sequence with the pair $(INC[i], DEC[i])$. According to the previous result from $3(c)$, each two numbers in the sequence are labeled with a different pair: if $i < j$ and $x_i \leq x_j$ then $INC[i] < INC[j]$, and on the other hand if $x_i \geq x_j$ then $DEC[i] < DEC[j]$. If we assume that all the $INC[i]$ values are at most a and all the $DEC[i]$ values are at most b, then we only have ab possible values to label the elements of the sequence. However, we know that the length m of the sequence is $ab + 1$, so there must be an element x_i for which $INC[i]$ is greater than a or alternatively $DEC[i]$ is greater than b. In the first case the resulting LIS will be of length at least $a + 1$, and in the second the DIS will have length at least $b + 1$. ($Q.E.D.$)

4. The following code written in Java and based on the algorithm proposed in the textbook will print the minimum cost for the matrix-chain product, as well as the optimal parenthesization associated to it:

```java

public class MatrixChainMultiplication {

    public static void printParens(int[][] s, int i, int j) {
        if (i == j)
            System.out.print("A" + i);
        else {
            System.out.print("(");
            printParens(s, i, s[i-1][j-2]);
            printParens(s, s[i-1][j-2] + 1, j);
            System.out.print(")");
        }
    }

    public static void main(String[] args) {
        int[] p = {5, 10, 3, 12, 5, 50, 6};
        int n = p.length - 1;
        int[][] m = new int[n][n];
        int[][] s = new int[n-1][n-1];

        for (int i = 1; i <= n; i++) {
            m[i-1][i-1] = 0;
        }

        for (int d = 2; d <= n; d++) {
            for (int i = 1; i <= n - d + 1; i++) {
                int j = i + d - 1;
                m[i-1][j-1] = Integer.MAX_VALUE;
                for (int k = i; k <= j - 1; k++) {
                    int q = m[i-1][k-1] + m[k][j-1] + p[i-1] * p[k] * p[j];
                    if (q < m[i-1][j-1]) {
                        m[i-1][j-1] = q;
                        s[i-1][j-2] = k;
                    }
                }
            }
        }

        System.out.println("min cost = " + m[0][n-1]);
        printParens(s, 1, n);

    }

}

```

After running the code, we get a minimum cost value of 2,010 and the following parenthesization for the six matrices: $((A_1A_2)((A_3A_4)(A_5A_6)))$.

5. (a)

$$\log_2\left(\frac{4^n}{\sqrt{n}}\right) = \log_2\left(4^n\right) - \log_2\left(\sqrt{n}\right) = n \cdot \log_2(4) - \frac{1}{2}\log_2(n) = 2n - \frac{1}{2}\log_2(n)$$

If we look at the simplified expression, the dominant factor is $2n$ and therefore the asymptotic value is $O(n)$.

(b)

$$\begin{aligned}
5^{313340} &< 7^{271251} \\
\ln\left(5^{313340}\right) &< \ln\left(7^{271251}\right) \\
313340 \ln(5) &< 271251 \ln(7) \\
313340 \cdot 1.6094 &< 271251 \cdot 1.9459 \\
504,301.2755 &< 527,830.0738
\end{aligned}$$

(c)

$$\begin{gathered}
n^2 \log_2\left(n^2\right) = 2n^2 \log_2(n) \\
\log_2^2\left(n^3\right) = \left[3 \log_2(n)\right]^2 = 9 \log_2^2(n)
\end{gathered}$$

(d)

$$\begin{aligned}
e^{-\frac{x^2}{2}} &= \frac{1}{n} \\
-\frac{x^2}{2} &= \ln\left(\frac{1}{n}\right) \\
x^2 &= (-2)(-\ln(n)) \\
x &= \sqrt{2\ln(n)}
\end{aligned}$$

(e)

$$\begin{gathered}
\log_n\left(2^n\right) = n \log_n(2) \\
\log_n\left(n^2\right) = 2 \log_n(n) = 2
\end{gathered}$$

(f)

$$\log_2(n) = \frac{\log_3(n)}{\log_3(2)}$$

(g) If we double k times the value of i we get $2^k i$. So we are looking for the smallest k such that $2^k i \geq n$ and therefore $k \geq \log_2\left(\frac{n}{i}\right)$.

(h)

$$\log_2\left[n^n e^{-n} \sqrt{2\pi n}\right] = n \log_2(n) - n \log_2(e) + \frac{1}{2} \log_2(2\pi) + \frac{1}{2} \log_2(n)$$

We observe that the second term of the sum is in the order of $O(n)$ (since $\log_2(e)$ is a constant), the third term is constant so it is $O(1)$ and the last term is in the order of $O(\log_2(n))$. Therefore, the leading term of the expression will be the first one, whose asymptotic value is $O(n \log_2(n))$.

The bracketed expression inside the logarithm is interesting because it corresponds to the approximation for $n!$ according to Stirling's formula:

$$n! \sim n^n e^{-n} \sqrt{2\pi n}$$