#2 Assignment - CMPT 405

Luiz Fernando Peres de Oliveira - 301288301 - lperesde@sfu.ca

October 5, 2018

#1

First, we draw the table with cost c of multiplying two matrices in the dimensions $\{1 \times 1, 1 \times d, d \times 1, d \times d\}$

m_1	m_2	m_{res}	cost
1×1	1×1	1×1	1
1×1	$1 \times d$	$1 \times d$	d
$1 \times d$	$d \times 1$	1×1	d
$1 \times d$	$d \times d$	$1 \times d$	d^2
$d \times 1$	1×1	$d \times 1$	d
$d \times 1$	$1 \times d$	$d \times d$	d^2
$d \times d$	$d \times 1$	$d \times 1$	d^2
$d \times d$	$d \times d$	$d \times d$	d^3

Intuition:
The algorithm should always choose the

#2

#3

Definition: Let A be an array with size $n+1$ and s be a sequence of integers. Initialize $A[0] = -\infty$ and for $1 \leq i \leq n$, define $A[i]$ as the largest contiguous subsequence sum in s after an iteration i. At the end, the largest possible sum will then be the highest element in A.

Recurrence:

$$A[i] = \begin{cases} -\infty & \text{if } i = 0 \\ \max\{A[i-1] + s[i], s[i]\} & \text{otherwise} \end{cases}$$

Algorithm:
Input: s, n

Make array A of size $n+1$
$A[0] \leftarrow -\infty$

insert *none* in the index 0 of s // *make* $|s| = |A|$ *for the loop*
$best_i \leftarrow 0$
for i **from** 1 **to** n **do**
    $A[i] \leftarrow \max\{A[i-1] + s[i], s[i]\}$
    if $A[i] > A[i-1]$ **then**
        $best_i \leftarrow i$
    end if
end for
$s' \leftarrow \emptyset$ // *find best subsequence index set* s'
while $A[best_i] = A[best_i - 1] + s[best_i]$ **do**
    $s' \leftarrow s' \cup \{best_i\}$
    $best_i \leftarrow best_i - 1$
end while
$s' \leftarrow s' \cup best_i$ // *add the lowest index of subsequence*
return s'

Demonstration:

$s =$	none	-2	11	-4	13	-5	-2
$A =$	$-\infty$	-2	11	7	**20**	15	13

$s' = \{2, 3, 4\}$

At the end, as s' demonstrates, we will have the range 2..4 comprising the largest possible sum of a contiguous subsequence in s.

Running time: The running time of the loops are $O(n)$. All operations on A (inside the loops) are constant ($O(1)$) and therefore the total running time of the algorithm is $O(n)$.

#4
The idea of the problem is to take any node on the tree T, consider it as a root and then do a **postorder traversal** applying a recurrence similar do the one on question #3. Consider all vertices are labelled $v_1, ..., v_i, ..., v_n$, $1 \leq i \leq n$. Let A be an array that keeps the highest sum of a subtree with root on vertex v. As we are doing a postorder traversal, when we compute $A[v_p]$, v_p being a parent vertex, we will have the sum of the children already computed. We only sum over $A[v_c]$, for all c children of v_p, if $A[v_c] > 0$ (call it $A[v_c+]$). The recurrence and algorithm would then be:

$$A[v_i] = \Big\{ \max \{ \ \textstyle\sum_{c^+ \in children(v_i)} A[v_c^+] + w(v_i), w(v_i) \ \}$$

Algorithm:
Input: T

Make array A of size n
$best_i \leftarrow 0$ // v_{best_i} *will be the root of* H
for each $v_i \in T_{postorder}$ **do**
    $A[v_i] \leftarrow \max\{\ \sum_{c^+ \in children(v_i)} A[v_c^+] + w(v_i), w(v_i)\ \}$
    if $A[v_i] > A[best_i]$ **then**
        $best_i \leftarrow i$
    end if
end for
$H \leftarrow$ Recursively do (v_i, all $v_c^+ \in$ children(v_i))} starting on root $A[v_{best_i}]$
return $H, A[v_{best_i}]$

#5