1. CSE 6220: HW2

> **Note:** This assignment is submitted by: **Somdut Roy (GTID: sroy86)**

1.1

(a) $op(a,b) = 2a+b, \Rightarrow a_i = 2a_{i-1} + a_{i-2}$.
Say, $V_i = \begin{bmatrix} a_i & a_{i-1} \end{bmatrix}, M = \begin{bmatrix} 2 & 1 \\ 1 & 0 \end{bmatrix}$.
It is evident that $V_i = V_{i-1} * M$.
Therefore, parallel prefix alogorithm can be used in this case.

(b) $op(a,b) = \sqrt{a^2+b^2}, \Rightarrow a_i^2 = a_{i-1}^2 + a_{i-2}^2$.
Say, $V_i = \begin{bmatrix} a_i^2 & a_{i-1}^2 \end{bmatrix}, M = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$.
It is evident that $V_i = V_{i-1} * M$.
Therefore, parallel prefix alogorithm can be used in this case as well.

1.2

The criteria for the viewer to see a particular mountain peak is that the angle of inclination w.r.t the viewer has to be higher than all other peak in front of it. Firstly we calculate the tan of angle of inclination (since, tangent is a monotonically increasing function) for each peak. To that array we use max operation and then another logical operation to see if the tan of angle of inclination is equal to maxima function or not.

1. $T_i = \frac{h_i - h_0}{d_i}$.
2. $X_i = max(T_i, X_{i-1})$, assuming $X_1 = T_1$. Here we can use parallel prefix algorithm.

3. $S_i = \begin{cases} 1, & \text{if } X_i = T_i \\ 0, & \text{otherwise.} \end{cases}$ where 1 and 0 mean that the peak can be seen and can't be seen respectively.

Run time: Steps 1 and 3 will each take $O(\frac{n}{p})$ time and step 2 will take $O(\frac{n}{p} + log(p))$ time.

1.3

Steps:

1. $A_i = \begin{cases} 1, & \text{if ith parathesis is 'open'} \\ -1, & \text{if ith parathesis is 'closed'.} \end{cases}$
2. $X_i = X_{i-1} + A_i$, given that $X_1 = A_1$. Here we can use parallel prefix algorithm.
3. $L_i = L_{i-1} * (X_i \geq 0)$, given that $L_0 = 1$. Here we can use parallel prefix algorithm.

If $L_n = 1$, it is well informed, else if $L_n = 0$, it is not well informed.

Run time: Step 1 will take $O(\frac{n}{p})$ time whereas steps 2 and 3 each will take $O(\frac{n}{p} + log(p))$ time.

1.4

Steps:

1. $M_{ij} = \begin{cases} 1, & \text{if } L[i] = j \\ 0, & \text{otherwise} \end{cases} \forall\ i = 1, 2, ..., k \text{ and } j = 1, 2, ..., n$
2. $N_{ij} = \begin{cases} M_{ij}, & \text{if i=1 and j=1} \\ N_{i(j-1)} + M_{ij}, & \text{if j>1} \\ N_{(i-1)n} + M_{ij}, & \text{i>1 and j=1} \end{cases}$
   Here we can use parallel prefix algorithm.
3. $L_{ij} = M_{ij} * N_{ij}$.
4. $R_i = \sum_{m=1}^{n} N_{im}$.

Run time: Steps 1, 3 and 4 each will take $O(\frac{n}{p})$ time whereas step 2 will take $O(\frac{n}{p} + log(p))$ time.

1.5

We assume that the sequence of numbers start with 1. Otherwise for all 0's till the first 1, the problem can not return any value. Therefore $A_1 = 1$. Solution B_i is given by:

$B_i = B_{i-1} * (1 - A_i) + (1 - A_i)$.

To solve this in parallel, the task is to be divided into the following three steps:

1. $V_i = \begin{bmatrix} B_i & 1 \end{bmatrix}$.
2. $M_i = \begin{bmatrix} 1 - A_i & 0 \\ 1 - A_i & 1 \end{bmatrix}$.
3. Evidently, $V_i = V_{i-1} * M_i$. We can use parallel prefix algorithm to solve this.

Run time: Steps 1 and 2 each will take $O(\frac{n}{p})$ time whereas step 3 will take $O(\frac{n}{p} + log(p))$ time.