Homework 5

Problem 2.1:
Base Case: For $k = 5$:

$$\text{LHS: } 2^{5+1} - 1 = 63$$

$$\text{RHS: } 2(5^2) + 2(5) + 1 = 61$$

Clearly, $63 > 61$, so the statement holds true for the base case.
Inductive Hypothesis: Assume the inequality holds for some arbitrary $k = n$:

$$2^{n+1} - 1 > 2n^2 + 2n + 1$$

Inductive Step: For $k = n + 1$:
Starting with the LHS:

$$2^{n+2} - 1 = 2(2^{n+1}) - 1$$

Using the inductive hypothesis:

$$2^{n+1} > 2n^2 + 2n + 1$$

We get:

$$\begin{aligned} 2(2^{n+1}) - 1 &> 2(2n^2 + 2n + 1) - 1 \\ &= 4n^2 + 4n \end{aligned}$$

For the RHS:

$$\begin{aligned} &2(n+1)^2 + 2(n+1) + 1 \\ &= 2n^2 + 6n + 5 \end{aligned}$$

The inequality:

$$2^{n+2} - 1 > 2(n+1)^2 + 2(n+1) + 1$$

is equivalent to:

$$\begin{aligned} 4n^2 + 4n &> 2n^2 + 6n + 5 \\ \Rightarrow 2n^2 - 2n &> 5 \end{aligned}$$

This is true for all $n > 4$.
By the principle of induction, the statement $2^{k+1} - 1 > 2k^2 + 2k + 1$ holds for all $k > 4$.
Thus, a binary tree of depth $k > 4$ cannot be embedded on a 2D-mesh with load-1, dilation-1, and congestion-free properties.

Problem 2.2:
(a)

(i) For all-to-all broadcast, each processor sends its m words to every other processor. This means every processor receives m words from every other processor. Given P processors, the time is $T_1 = P \cdot (ts + m \cdot tw)$. We're not multiplying $P - 1$ since every processor sends its data to itself too, in all all-to-all scenario. After receiving all the words, local computation to sum up the elements of the array takes $O(m)$ time. Therefore, $T_{total(i)} = T_1$

(ii) Accumulation at a single node. Each of the P processors sends its m words to a single node: $T_2 = (P-1) \cdot (ts + m \cdot tw)$. With one-to-all broadcast by the root, the root sends the accumulated results to all processors including itself: $T_3 = P \cdot (ts + m \cdot tw)$. Therefore, $T_{total(ii)} = T_2 + T3$.

(iii) Just like in the all-to-all broadcast, but since we're only adding numbers, every processors sends its sum (1 word) to every other processor, and each processor keeps a running sum: $T_4 = P \cdot (ts + tw)$. Therefore, $T_{total(iii)} = T_4$.

(b) For $ts = 100$, $tw = 1$, and m is very large:

(i) $T_{total(i)} \approx P \cdot (100 + m)$

(ii) $T_{total(ii)} \approx P \cdot (100 + m) + P \cdot (100 + m)$

(iii) $T_{total(iii)} \approx P \cdot 101$

For very large m, $T_{total(iii)}$ is the smallest since it's not dependent on m.

(c) For $ts = 100$, $tw = 1$, and $m = 1$.

(i) $T_{total(i)} \approx P \cdot 101$

(ii) $T_{total(ii)} \approx P \cdot 101 + P \cdot 101$

(iii) $T_{total(iii)} \approx P \cdot 101$

In summary: (a) whichever is better cannot be determined. (b) $T_{total(iii)}$ is better. (c) Either $T_{total(i)}$ or $T_{total(iii)}$ is better.