Algorithms and Data Structures

Luke Nigro - Homework 1

1. **Divide:** Divide the problem into two subproblems, one containing $[1, ..., n-1]$ and the other with only element n. This takes $\Theta(1)$ time.
   Conquer: Solve each of the subproblems recursively, which will takes $T(n-1)$ in the worst-case scenario when solving the array $[1, ...n-1]$.
   Combine: Insert element n into the sorted now sorted array that contained elements $[1, ...n-1]$. This will take $\Theta(n)$ time in the worst-case scenario where the element n happens to be the minimum value (or maximum value depending on how you're sorting).
   Therefore,

$$T(n) = \begin{cases} \Theta(1), & n = 1 \\ T(n-1) + \Theta(n), & n > 1 \end{cases} \tag{1}$$

2. The first step in this problem, is sending the two boys across the river and then sending one of them back across, this is 2 trips. Then a soldier should cross the river, and send the other boy back across the river. We will now have both of the boys on the original side of the river, as well as $n-1$ soldiers. When we repeat this process $4n$ times, we will end up with all soldiers across the river and both of the 12-year-old boys in joint possession of the boat.

3. $\Omega(g(n,m)) = \{\int(n,m)$: there exists positive constants c, n_0, and m_0 such that $cg(n,m) \leq \int(n,m)$ for all $n > n_0$ and $m > m_0\}$.
   $\Theta(g(n,m)) = \{\int(n,m)$: there exists positive constants c_1, c_2, n_0, and m_0 such that $c_1 g(n,m) \leq \int(n,m) \leq c_2 g(n,m)$ for all $n > n_0$ and $m > m_0\}$.

4. I believe that the fastest way produce the top 1000 numbers in a disordered list of one billion numbers would be to first, start by placing

the first 1000 numbers in an array and finding the minimum value of that array. We now have two distinct arrays. Then you would compare the minimum value of the length 1000 array with a value of the larger array. If the minimum of the other array is larger, discard the number from the larger array and keep repeating the process. If the other number is larger, replace the array's current minimum with the number from the larger array. We will then have to find the new minimum of the array with 1000 elements. This process will have to be repeated one-billion minus 1000 times, and in the worst case scenario, we will have to recalculate the minimum the same number of times.