Problem Set 1

All parts are due Thursday, September 18 at 11:59PM. Please download the .zip archive for this problem set, and refer to the `README.txt` file for instructions on preparing your solutions. Remember, your goal is to communicate. Full credit will be given only to a correct solution which is described clearly. Convoluted and obtuse descriptions might receive low marks, even when they are correct. Also, aim for concise solutions, as it will save you time spent on write-ups, and also help you conceptualize the key idea of the problem.

Name: Rishad Rahman

Collaborators: None

Part A

Problem 1-1.

Answer: $\frac{1}{n} \to \frac{1}{5} \to \log(\log n) \to \log n \to n^{\frac{1}{100}} \to 10^{100}n \to \log(n!) \to n \log n \to \binom{n}{100} \leftrightarrow n^{100} \to 3^{\sqrt{n}} \to 2^n \leftrightarrow 2^{n+1} \to n2^n \to 3^n \to 2^{2n} \leftrightarrow 4^n \to n!$ where $f(n) \to g(n)$ means $f(n) = O(g(n))$ while $f(n) \leftrightarrow g(n)$ means $f(n) = \Theta(g(n))$.

Problem 1-2.

(a) Answer: $\boxed{\Theta(n^2)}$

Solution: Level k, $k \in \{0, 1, 2..., \lg n\}$, of the recursion tree has 4^k elements all equal to $\lg(\frac{n}{2^k})$. Summing all the work done, we get $n^2T(1) = \Theta(n^2)$ in addition to $\sum_{k=0}^{\lg n} 4^k \lg(\frac{n}{2^k})$ which is bounded above by

$$\sum_{k=0}^{\lg n} 4^k \frac{n}{2^k} = n \sum_{k=0}^{\lg n} 2^k = \Theta(n^2)$$

and bounded below by

$$\sum_{k=0}^{\lg n} 4^k = \Theta(n^2)$$

(b) Answer: $\boxed{\Theta(n^2 \log_3 n)}$

Solution: In a similar fashion, each level k has 9^k elements all equal to $(\frac{n}{3^k})^2$. Summing the work done in addition to $9^{\log_3 n}T(1) = n^2T(1) = \Theta(n^2)$ gives $\sum_{k=0}^{\log_3 n} n^2 = \Theta(n^2 \log_3 n)$.

(c) Answer: $\boxed{\Theta(n)}$

Solution: We have $T(n) = T(\frac{n}{2}) + 2n$ which we can continuously expand until we get $T(2) = \Theta(1)$ and we get $T(n) = 2n + n + \frac{n}{2} + ... + \frac{n}{2^{\lg n - 3}} + \Theta(1) = \Theta(n)$

(d) Answer: $\boxed{\Theta(n \lg n)}$

Solution: We have $S(x, y) = \underbrace{x + x + ... + x}_{\Theta(\lg y)} + \Theta(x) = \Theta(x \lg y)$.

Problem 1-3.

(a) Solution: Rotate the circle until a_0 is a global minimum; the actual labeling doesn't matter as the differences only rely on the geometry of the circle which is symmetric under rotations. Define a function $f : \mathbb{Z} \to \mathbb{R}$ such that $f(m) = a_{m \bmod n}$. If $0 \leq f(\frac{n}{2}) - f(0) \leq 1$, we are done. Suppose $f(\frac{n}{2}) - f(0) > 1$. Note that $f(n) - f(\frac{n}{2}) = f(0) - f(\frac{n}{2}) < 0$ but $[f(x + \frac{n}{2} + 1) - f(x + 1)] - [f(x + \frac{n}{2}) - f(x)] = [f(x + \frac{n}{2} + 1) - f(x + \frac{n}{2})] - [f(x + 1) - f(x)] \in \{-2, -1, 0, 1, 2\}$ which follows from the condition given in the problem. This implies to go from $f(\frac{n}{2}) - f(0) > 1$ to $f(n) - f(\frac{n}{2}) < 0$ with $f(x) - f(x - \frac{n}{2})$ as x goes from $\frac{n}{2} \to n$ in increments of 1, we must hit $x = p$ such that $-1 \leq f(p) - f(p - \frac{n}{2}) \leq 1$ because if $p - 1$ is the last index such that the $f(p - 1) - f(p - 1 - \frac{n}{2}) > 1$ then $-1 < f(p) - f(p - \frac{n}{2}) < 1$ since the difference can't dip more than 2 as we increment x by 1. Therefore $|D(p)| \leq 1$.

(b) Solution: The first steps define the procedure on $a_0 \to a_{\frac{n}{2}}$ and then we will follow with a recursion. If $-1 \leq a_0 - a_{\frac{n}{2}} \leq 1$, we are done. Otherwise let $s = \operatorname{sgn}(a_0 - a_{\frac{n}{2}})$. Take $s' = \operatorname{sgn}(a_{\lfloor \frac{n}{4} \rfloor} - a_{\lfloor \frac{n}{4} \rfloor + \frac{n}{2}})$. If $s' = 0$ we are done. Otherwise if $ss' = -1$, recurse this procedure on $a_1 \to a_{\lfloor \frac{n}{4} \rfloor}$, and if $ss' = 1$ then recurse on $a_{\lfloor \frac{n}{4} \rfloor + 1} \to a_{\frac{n}{2}}$. If we have one index i left, we are done, take $(i, i + \frac{n}{2})$. If there are two elements left in the set to recurse on, compute differences and pick the pair $(i, j = i + \frac{n}{2})$ such that $|a_i - a_j| \leq 1$. Each base case takes constant time so $T(1), T(2) = \Theta(1)$. In each step of the algorithm we take constant time computing differences and we are halving the size of the problem if we didn't arrive at a solution. So there are a total of $O(\lg n)$ steps taking constant time each and hence the running time is $\boxed{O(\lg n)}$.

(c) Solution: We have to prove that the algorithm terminates with a returned value and that the returned value solves the problem. The algorithm does terminate since we either terminate or halve the problem size at each step which will end at a base cases of 1 or 2 which the algorithm covers. At each step of the algorithm

where we don't terminate i.e. when we don't have an absolute bound of 1 nor $ss' = 0$, we recurse so that the subproblem consisting of $a_k \to a_l$ has the property $\operatorname{sgn}(D(k) \cdot D(l)) = -1$. This is obviously true for $a_0 \to a_{\frac{n}{2}}$. Suppose that wlog $D(k) > 0$ and that $D(k') > 0$ where k' is the calculated midpoint between k and l. According to the algorithm we then recurse on $k' \to l$ and $\operatorname{sgn}(D(k') \cdot D(l)) = -1$ since $D(k')$ has the same sign as $D(k)$. If $D(k') < 0$ we recurse on $k \to k'$ and $\operatorname{sgn}(D(k) \cdot D(k')) < 0$ so $\operatorname{sgn}(D(k) \cdot D(l))$ is invariant. The fact that we must have at least one i such that $|D(i)| \leq 1$ at each step of the recursion follows from the argument in (a) i.e. by using the difference in sign and finding where $D(i)$ changes sign. Therefore our final bases cases must contain i such that $|D(i)| \leq 1$ which the algorithm returns. And obviously if $|D(i)| \leq 1$ for the base case, $|D(i)| \leq 1$ for the whole problem so we are done.