Name: Suhail Basalama

Homework 1

1)

$$T(n) = T(n-1) + 2$$

$$= (T(n-2) + 2) + 2$$

$$= (T(n-3) + 2) + 2 \times 2$$

$$= T(n-3) + 3 \times 2$$

$$\dots$$

$$= \big(T\big(n-(n-1)\big) + 2\big) + (n-2) \times 2$$

$$= T(1) + (n-1) \times 2$$

$$= 2 + 2n - 2$$

$$T(n) = 2n$$

2)

$$T(n) = T(n-1) + 4n - 3$$

$$= (T(n-2) + 4(n-1) - 3) + 4n - 3$$

$$= T(n-2) + 2 \times 4n - 2 \times 3 - 4$$

$$= (T(n-3) + 4(n-2) - 3) + 2 \times 4n - 2 \times 3 - 4$$

$$= T(n-3) + 3 \times 4n - 3 \times 3 - (4 + 8)$$

$$\dots$$

$$= T\big(n-(n-1)\big) + (n-1) \times 4n - (n-1) \times 3 - (4 + 8 + 12 + \cdots)$$

$$= T(1) + (n-1) \times 4n - (n-1) \times 3 - 2(n-1)(n-2)$$

$$T(n) = 2n^2 - n + 1$$

3)

$$T(n) = 2T(n-1) - 1$$

$$= 2(2T(n-2) - 1) - 1$$

$$= 2^2T(n-2) - 2 - 1$$

$$= 2^2(2T(n-3) - 1) - 2) - 1$$

$$= 2^3T(n-3) - 2^2 - 2^1 - 2^0$$

$$\dots$$

$$= 2^{n-1}T(n-(n-1)) - (2^0 + 2^1 + 2^2 + \cdots + 2^{n-1})$$

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

$$= 2^{n-1} \times 2 - 2^{n-1} + 1$$

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

4)

$$T(m) = 2T(m-1) + m - 1$$

$$= 2(2T(m-2) + m - 2) + m - 1$$

$$= 2^2T(m-2) + 2m + 2^0m - (2 \times 2 + 1 \times 2^0)$$

$$= 2^3T(m-3) + (2^2m + 2^1m + 2^0m) - (3 \times 2^2 + 2 \times 2^1 + 1 \times 2^0)$$

$$\dots$$

$$= 2^{(m-1)}T\big(m-(m-1)\big) + (2^{(m-2)}m + \cdots + 2^2m + 2^1m + 2^0m) \\ - ((m-1)2^{(m-2)} + \cdots + 3 \times 2^2 + 2 \times 2^1 + 1 \times 2^0)$$

$$= 0 + m\big(2^{(m-1)} - 1\big) - \big(2^{(m-1)}(m-2) + 1\big)$$

$$T(m) = 2^m - m - 1$$

5)

$$n = 2^m - 1$$

$$n + 1 = 2^m$$

$$\log_2(n + 1) = \log_2(2^m)$$

$$m = \log_2(n + 1)$$

$$T(m) = 2^m - m - 1$$

$$T(n) = 2^{\log_2(n+1)} - \log_2(n + 1) - 1$$

$$T(n) = n + 1 - \log_2(n + 1) - 1$$

$$T(n) = n - \log_2(n + 1)$$

2-1

a. Since the time complexity of an insertion sort for a list of length k is $\Theta(k^2)$, sorting n/k sub-lists takes time $\Theta(\frac{n}{k}k^2) = \Theta(nk)$.

b. If we have coarseness k, then we can use the usual merging steps, but we will start it at the level where each array has size at most k. Thus, the depth of the merge tree is $lg(n) - lg(k) = lg(n/k)$. Each level of merging is still time cn; therefore, the merging takes time $\Theta(nlg(n/k))$.

c. Putting k as a function of n, as long as $k(n) \in O(lg(n))$, it has the same running time. For any constant choice of k, the asymptotics are the same.

d. We can optimize the previous expression to get k as follow:

$$c1n - nc2/k = 0$$

Where $c1$ and $c2$ are the coefficients of nk and $nlg(n/k)$ hidden by the asymptotic notation. Specifically, a constant choice of k is optimal. In practice we could find the best choice of this k by just trying and timing for various values for sufficiently large n.