Question 1 Page 1

Christian Ayala
Algorithms HW 4

Question 1ai)

$t_n = t_{n-1} + 6$ for $n > 1$, $t_1 = 2$

We can rewrite the recurrence relation as such:

$$T(n) = \begin{cases} 2 & \text{if } n = 1 \\ T(n-1) + 6 & \text{if } n > 1 \end{cases}$$

The base case is $n = 1$, which we can insert into our candidate solution as: $T(1) = 6(1) - 4 = 2$. For the inductive step, we assume $n > 1$, and $T(n-1) = 6(n-1) - 4$. With our recurrence, we substitute:

$$\begin{aligned} T(n) &= T(n-1) + 6 \\ &= (6(n-1) - 4) + 6 \\ &= 6n - 4 \end{aligned}$$

Verified

Question 1aii)

$t_n = t_{n-1} + n$

Rewritten:

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

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

Inserting the base case into our candidate solution: $T(1) = ((1+1)^2 - (1+1))/2 = 1$. For the inductive step, we assume $T(n-1) = (n^2 - n)/2$

$$\begin{aligned} T(n) &= T(n-1) + n \\ &= (n^2 - n)/2 + n \\ &= n\left(\tfrac{1}{2}(n+1)\right) = \tfrac{1}{2} n(n+1) \end{aligned}$$

Note that our candidate solution $\frac{(n+1)^2 - (n+1)}{2}$ can be simplified to the exact same formula, so we have verified the candidate solution.

Question 1 Page 2

Question 1bi)

$t_n = t_{n-1} + n^2$ for $n > 1$, $t_1 = 2$

The first few terms are:

$t_n = t_{n-1} + n^2$
$= t_{n-2} + 2n^2$
$= t_{n-3} + 3n^2$
$= t_{n-4} + 4n^2$
$= t_{n-n} + n(n^2) = t_0 + n^3$

Based on this recurrence relation, I estimate some form of kn^3

$T(n) \leq T(n-1) + n^2$

$\leq k(n-1)^3 + n^2 \leq kn^3$

$k(n^3 - 3n^2 + 3n - 1) + n^2 \leq kn^3$

$kn^3 - 3kn^2 + 3kn - k + n^2 \leq kn^3$

$-3kn^2 + 3kn - k + n^2 \leq 0$

$-3kn^2 + 3kn - k \leq -n^2$

$k(3n^2 - 3n + 1) \geq n^2$

$$k \geq \frac{n^2}{3n^2 - 3n + 1}$$

For any value $n \geq 1$, the largest value of the right hand side of the inequality is ≤ 1, so k can be any positive number ≥ 1. Therefore, as long as $n > 1$, $T(n)$ is bounded by $O(n^3)$, thus this recurrence is $O(n^3)$

Question 1 Page 3

Question 1bii)

$t_n = 3t_{n-1} + 2^n$ for $n > 1$, $t_1 = 1$

The first few terms are:

$t_n = 3t_{n-1} + 2^n$

$= 3(3t_{n-2} + 2^n) + 2^n = 9t_{n-2} + 4(2^n) = 9t_{n-2} + 2^{n+2}$

Question 2 Page 1

Question 2a)

$T(n) = aT(n/b) + cn$, (a,b,c) are all positive integers

Since the degree of the latter term is 1, we can simplify the Master Theorem to the following 3 cases:

$$T(n) = \begin{cases} \Theta(n) & \text{if } a<b \\ \Theta(n\log n) & \text{if } a=b \\ \Theta(n^{\log_b a}) & \text{if } a>b \end{cases}$$

Question 2bi)

$T(n) = 5T(n-5) + 1 \quad T(0) = T(1) = T(2) = T(3) = T(4) = 1$

$T(5) = T(6) = T(7) = T(8) = T(9) = 6$

$T(10) = T(11) = T(12) = T(13) = T(14) = 31$

$T(15)$ thru $T(20) = 156$

$$\begin{aligned} T(n) &= 5T(n-5) + 1 \\ &= 5(5T(n-10)+1) + 1 = 25T(n-10) + 5 + 1 \\ &= 125T(n-15) + 25 + 5 + 1 \\ &= 625T(n-20) + 125 + 25 + 5 + 1 \\ &= 5^{n/5}T(0) + \underbrace{\sum_{i=0}^{\lfloor n/5 \rfloor} 5^i} \end{aligned}$$

This term grows faster, so we use it for our big-Oh notation

We see that $\sum_{i=0}^{\lfloor n/5 \rfloor} 5^i \leq 5^{n/5+1} \Rightarrow 5^{6n/5}$,

So our relation is $\underline{O(5^{6n/5})}$

Question 2 Page 2

Question 2bii)

$T(n) = 3T(\lfloor \frac{n}{4} \rfloor) + 2n$, $T(0) = T(1) = 1$

By the master theorem, $a = 3$ and $b = 4$. Since $a < b$, and c is constant, complexity is $\Theta(n)$.
The next 5 terms are:

$T(2) = 3*1 + 4 = 7$
$T(3) = 3*1 + 6 = 9$
$T(4) = 3*1 + 8 = 11$
$T(5) = 3*1 + 10 = 13$
$T(6) = 3*1 + 12 = 15$

Question 2biii)

$T(n) = 4T(\lfloor \frac{n}{2} \rfloor) + 2n^2$, $T(1) = 1$

Since the last term is $\Theta(n^2)$, and $a=4$ and $b=2$, we can also apply the master theorem to this relation:

if $a = b^i$, then $T(n) = \Theta(n^i \log_b n)$

Thus, relation has the complexity:

$T(n) = \Theta(n^2 \log_2 n)$

$T(2) = 4+8 = 12$
$T(3) = 4+18 = 22$
$T(4) = 48+32 = 80$
$T(5) = 48+50 = 98$
$T(6) = 88+72 = 160$

Question 2 Page 3

Question 2biv)

$T(n) = \frac{1}{n} + T(n-1)$, $T(1) = 1$

$$T(n) = \frac{1}{n} + \frac{1}{n-1} + T(n-2)$$
$$= \frac{1}{n} + \frac{1}{n-1} + \frac{1}{n-2} + T(n-3)$$
$$= \frac{1}{n} + \frac{1}{n-1} + \frac{1}{n-2} + \frac{1}{n-3} + T(n-4)$$
$$= \sum_{i=1}^{n} \frac{1}{n} + T(1)$$
$$= \sum_{i=1}^{n} \frac{1}{n}$$ ← This is the harmonic series

This series is well known to be bounded by the function $\ln(n)$. Thus, this recurrence relation is bounded by $\Theta(\ln(n))$.

Question 3 Page 1

Question 3a)

$$T(n) = (n-1) + \frac{1}{n}\sum_{i=0}^{n-1}\left(T(i) + T(n-1-i)\right), \quad T(0) = 0$$

$$S(n) = \sum_{i=0}^{n-1}\left(T(i) + T(n-1-i)\right)$$

We can express $T(n)$ in terms of $S(n)$ simply through substitution:

$$T(n) = (n-1) + \frac{1}{n} S(n)$$

Question 3 Page 2

Question 3b)

$S(1) = 0$	$S(7) = 52.4$	$T(1) = 0$	$T(7) = 13.486$
$S(2) = 0$	$S(8) = 79.371$	$T(2) = 1$	$T(8) = 16.922$
$S(3) = 2$	$S(9) = 113.214$	$T(3) = 2.667$	$T(9) = 20.579$
$S(4) = 7.333$	$S(10) = 154.373$	$T(4) = 4.833$	$T(10) = 24.437$
$S(5) = 17$	$S(11) = 203.248$	$T(5) = 7.4$	$T(11) = 28.477$
$S(6) = 31.8$	$S(12) = 260.202$	$T(6) = 10.3$	$T(12) = 32.684$

Question 4 Page 1

Question 4a) Part a)

In standard multiplication, we require 4 multiplications of coefficients:

$$(ax+b)(cx+d) = acx^2 + (ad+bc)x + bd$$

However, looking at the hint provided, we see that:

$$(a+b)(c+d) = \underbrace{ad+bc}+ac+bd$$

These two are the middle term for the above multiplication, and the other two terms are the remaining terms in the original multiplication. So we can rewrite the overall calculation as such:

$$(ax+b)(cx+d) = acx^2 + ((a+b)(c+d) - ac - bd)x + bd$$

In this case the three multiplications are (ac), (bd), and $((a+b)(c+d))$. We simply reuse the first two multiplications.

Question 4a) Part b) High/Low Halves

First, we denote the two input polynomials as P and Q. When examining one of these polynomials, we denote its form as:

$$P = p_n + p_{n-1}x + p_{n-2}x^2 + p_{n-3}x^3 + \dots + p_1 x^{n-1}$$

Where n is the degree of the polynomial.

Similarly, we can notate q using the following form:

$$Q = q_n + q_{n-1}x + q_{n-2}x^2 + q_{n-3}x^3 + \dots + q_1 x^{n-1}$$

Where n is the degree of the polynomial.

Question 4 Page 2

Using standard polynomial multiplications, we can describe the multiplication of P and Q as:

$$P * Q = (p_1 X^{n-1} + p_2 X^{n-2} + \ldots + p_n) * (q_1 x^{n-1} + q_2 x^{n-2} + \ldots + q_n)$$

With direct multiplication, this requires $O(n^2)$ multiplications. However, we can divide a polynomial into two parts: a high-degree part and a low-degree part. We choose the boundary as: $m = \lceil \frac{n}{2} \rceil$. We can factor out x^m from all terms of degree $\geq m$. As an example on P:

$$P = p_n + p_{n-1}X + p_{n-2}X^2 + p_{n-3}X^3 + \ldots + p_1 x^{n-1}$$
$$P = (p_n + p_{n-1}X + \ldots + p_{n-m-1}X^{n-m-1}) + (p_{n-m}X^{n-m} + p_{n-m+1}X^{n-m+1} + \ldots + p_1 X^{n-1})$$
$$P = \underbrace{(p_n + p_{n-1}X + \ldots + p_{n-m-1}X^{n-m-1})}_{\text{We call this } B} + X^m \underbrace{(p_{n-m} + p_{n-m+1}X + \ldots + p_1 X^{m-1})}_{\text{We call this } A}$$

We can describe Q in a very similar way:

$$Q = \underbrace{(q_n + q_{n-1}X + \ldots + q_{n-m-1}X^{n-m-1})}_{\text{We call this } D} + X^m \underbrace{(q_{n-m} + q_{n-m+1}X + \ldots + q_1 X^{m-1})}_{\text{We call this } C}$$

With our A, B, C, D notation, we can rewrite $P * Q$:

$$P * Q = (Ax^m + B)(Cx^m + D)$$

We note that this has the same form as the problem in part A, so we know we need 3 multiplication operations at each step.

Question 4 Page 3

With our newly written multiplicative form, we see that:

$$P*Q = (Ax^m + B)(Cx^m + D)$$

And that we need to multiply $(A+B)(C+D)$, (AC) and (BD). We can continuously recur with these three multiplications until each multiplicative step is multiplying a polynomial of degree 0 (as in, its just a single multiplication). Since we are dividing the problem in half, with 3 multiplications at each iteration (and a summation), we form the following recurrence relation:

$$T(n) = 3T\left(\frac{n}{2}\right) + O(n)$$

We can apply the Master Theorem to this relation, which results in $\Theta(n^{\lg 3})$. Since $O(n)$ is bounded by $\Theta(n^{\lg 3})$, our overall complexity is $\Theta(n^{\lg 3})$

Part B: Odd or Even

This method is actually very similar to the previous one, just set up a little differently. We can divide a polynomial into two approximately equal parts: even and odd degrees, as such.

$$P = p_n + p_{n-1}x + p_{n-2}x^2 + p_{n-3}x^3 + \dots + p_1 x^{n-1}$$

$$P = (p_{n-1}x + p_{n-3}x^3 + \dots + p_1 x^{n-1}) + (p_n + p_{n-2}x^2 + \dots + p_2 x^{n-2})$$

factor out an x from odds

$$P = x\underbrace{(p_{n-1} + p_{n-3}x^2 + \dots + p_1 x^{n-2})}_{\text{We call this } A} + \underbrace{(p_n + p_{n-2}x^2 + \dots + p_2 x^{n-2})}_{\text{We call this } B}$$

Question 4 Page 4

Similarly, we can apply the same transformation to Q:

$$Q = x\underbrace{\left(q_{n-1} + q_{n-3}x^2 + \dots + q_1 x^{n-2}\right)}_{\text{We call this } C} + \underbrace{\left(q_n + q_{n-2}x^2 + \dots + q_2 x^{n-2}\right)}_{\text{We call this } D}$$

Thus, multiplying P and Q leads to the following form.

$$P * Q = (Ax + B)(Cx + D)$$

Once again, this is the exact same structure as before, where we recursively divide the problem in half with 3 operations per division. Using the master theorem thus leads us to this operation being bounded by $\Theta(n^{\log 3})$.

Question 4 Page 5

Question 4 Part b)

$$\begin{array}{c} \\ 1 \\ 2 \end{array}\begin{array}{cc} 1 & 2 \\ \multicolumn{2}{c}{\begin{bmatrix} 1 & 3 \\ 5 & 7 \end{bmatrix}} \end{array}\begin{array}{cc} 1 & 2 \\ \multicolumn{2}{c}{\begin{bmatrix} 6 & 8 \\ 4 & 2 \end{bmatrix}} \end{array}\begin{array}{c} \\ 1 \\ 2 \end{array}$$

$$\quad\quad A \quad\quad\quad B$$

First we compute the S's:

$S_1 = B_{12} - B_{22} = 6$
$S_2 = A_{11} + A_{12} = 4$
$S_3 = A_{21} + A_{22} = 12$
$S_4 = B_{21} - B_{11} = -2$
$S_5 = A_{11} + A_{22} = 8$
$S_6 = B_{11} + B_{22} = 8$
$S_7 = A_{12} - A_{22} = -4$
$S_8 = B_{21} + B_{22} = 6$
$S_9 = A_{11} - A_{21} = -4$
$S_{10} = B_{11} + B_{12} = 14$

Next, we compute the P's:

$P_1 = A_{11} \cdot S_1 = 6$
$P_2 = S_2 \cdot B_{22} = 8$
$P_3 = S_3 \cdot B_{11} = 72$
$P_4 = A_{22} \cdot S_4 = -14$
$P_5 = S_5 \cdot S_6 = 64$
$P_6 = S_7 \cdot S_8 = -24$
$P_7 = S_9 \cdot S_{10} = -56$

Finally, we compute the C's

$C_{11} = P_5 + P_4 - P_2 + P_6 = 18$
$C_{12} = P_1 + P_2 = 14$
$C_{21} = P_3 + P_4 = 58$
$C_{22} = P_5 + P_1 - P_3 - P_7 = 54$

So our final matrix is:

$$\begin{bmatrix} 1 & 3 \\ 5 & 7 \end{bmatrix}\begin{bmatrix} 6 & 8 \\ 4 & 2 \end{bmatrix} = \begin{bmatrix} 18 & 14 \\ 58 & 54 \end{bmatrix}$$

Question 4 Page 6

Question 4 Part C)

A is a $kn \times n$ matrix, and B is an $n \times kn$ matrix
We can divide A and B into k submatrices. The resulting matrices look like:

$$A = \begin{bmatrix} A_1 \\ A_2 \\ A_3 \\ \vdots \\ A_k \end{bmatrix} \qquad B = \begin{bmatrix} B_1 & B_2 & B_3 & \cdots & B_k \end{bmatrix}$$

Each A_i and B_i is an $n \times n$ matrix

Multiplying this column vector of matrices by a row of matrices leads to a matrix of the following form:

$$\begin{bmatrix} A_1B_1 & A_1B_2 & \cdots & A_1B_k \\ A_2B_1 & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ A_kB_1 & \cdots & \cdots & A_kB_k \end{bmatrix}$$

Since each multiplication above takes $\Theta(n^{\log 7})$ time with Strassen's subroutine, and there are k^2 entries in the matrix, we can perform this calculation in $\underline{\Theta(k^2 n^{\log 7})}$

Question 4 Page 7

A is an $n \times kn$ matrix, B is a $kn \times n$ matrix.
Using the same tactic, we can represent A & B as such:

$$A = \begin{bmatrix} A_1 & A_2 & \dots & A_k \end{bmatrix} \qquad B = \begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ B_k \end{bmatrix}$$

Each matrix A_i and B_i is an $n \times n$ matrix. In this case, the resulting matrix is a single $n \times n$ matrix, and we calculate it by sequentially multiplying $A_i B_i$ and adding them all up. This has the summative form:

$$\sum_{i=1}^{k} A_i B_i$$

Since we are doing k multiplications, each of which uses Strassen's algorithm, our total complexity is $\Theta(kn^{\log 7})$

Question 5 Page 1

Question 5a)

For a given set of numbers $x_1, x_2, \dots, x_n$, we know that the median x_k is an element such that there are $n/2$ elements $\leq x_k$, and $n/2$ elements $\geq x_k$. Thus, if we assign each number x_i a weight $w_i = 1/n$, then by summing the weights of all x's less than x_k

$$\sum_{i=1}^{k} w_i = \sum_{i=1}^{k} \frac{1}{n}$$

$$= \frac{1}{n} \sum_{i=1}^{k} 1$$

n is independent of k, so pull it out of the summation

The summation of $\sum_{i=1}^{k} 1 = k$, which, since x_k is the median, is $\leq \frac{n}{2}$

$$= \frac{1}{n} \sum_{i=1}^{k} 1 \leq \frac{1}{n} \cdot \frac{n}{2} = \frac{1}{2}$$

Similarly, we can sum the weights of all values $> x_k$

$$\sum_{i=k+1}^{n} w_i = \sum_{i=k+1}^{n} \frac{1}{n}$$

$$= \frac{1}{n} \sum_{i=k+1}^{n} 1$$

Once again, $\sum_{i=k+1}^{n} 1$ is the number of elements greater than x_k, which again is $\leq \frac{n}{2}$.

$$= \frac{1}{n} \sum_{i=k+1}^{n} 1 \leq \frac{n}{2} \cdot \frac{1}{n} = \frac{1}{2}$$

Thus x_k is the weighted median of the set $x_1, \dots, x_n$, with weights $w_i = 1/n$

Question 5 Page 2

Question 5b)

Given a set of n elements, we can first sort the set using a heapsort ($\Theta(n\log n)$). Then, starting at x_1, we perform a linear traversal of the x's, adding the weights until the sum of the weights is greater than $\frac{1}{2}$. The element x_k that we stop at is our weighted median.

We can see that this is the case because of the following points:

- If all x's are sorted based on weights, and x_k is the first element that causes the weight sum to be $\geq \frac{1}{2}$, then elements $x_1, \ldots, x_{k-1}$ have a sum weight $< \frac{1}{2}$.
- Similarly, if $x_1, \ldots, x_k$ have sum weights $\geq \frac{1}{2}$, then all elements $x_{k+1}, \ldots, x_n$ have sum weights $< \frac{1}{2}$.

Thus, x_k is our weighted median. Since the sort is $\Theta(n\lg n)$, and our linear scan is $\Theta(n)$, our overall complexity is $\Theta(n\lg n)$.

Question 5 Page 3

Question 5c)

For this question, I will assume that the process of finding the non-weighted median of n numbers can be done in $\Theta(n)$ time, through the SELECT algorithm mentioned in the book.

The strategy is to find the non-weighted median of the set $x_1, \dots, x_n$, and partition the array into two halves around x_m (median). Then we look at the weight sums of the halves: if they are both $< \frac{1}{2}$, then we have found the weighted median. Otherwise, we know the weighted median is in the half with the larger weight sum, so we recur.

We can look at the algorithm with the following steps, where X is the set of elements:

```
WMedian(X)
    if X.length == 1    return X
    if X.length == 2    return x_i with higher weight
    x_k <- find non-weighted median in X
    Partition X into two halves around x_k
    w_l <- sum of all weights for x_1 .... x_{k-1}
    w_g <- sum of all weights for x_{k+1} .... x_n              done
    if w_l and w_g are both < 1/2, return x_k
    if w_l > 1/2
        Weight of x_k <- w_k + w_g
        return WMedian(x_1 .... x_k)
    else
        Weight of x_k <- w_k + w_l
        return WMedian(x_k .... x_n)
```

Question 5 Page 4

Examing the algorithm line by line, both 1 and 2 are $\Theta(1)$ operations. Line 3 (finding the non-weighted median) was shown in the book as $\Theta(n)$, using the SELECT algorithm. Partitioning X can be done through index partitioning, which is $\Theta(1)$. Lines 5 and 6 combined perform at most n additions (for the first recursive call), which is another $\Theta(n)$ function. The rest of the algorithm is constant time. Therefore, at each call of the function, we have $\Theta(n)$ work.

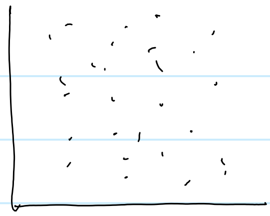
Looking at the recurrence relation, we see that we are dividing the input X in half every step, but only recurring on one of those halves. Thus, our recurrence relation looks like:

$$T(n) = T(n/2) + \Theta(n)$$

Using the master theorem from problem 2, we see that this function has an overall complexity of $\Theta(n)$.

Question 6 Page 1

Question 6)



Consider this random set of vertices:
When trying to find the convex hull, one approach is to use divide-and-conquer. In this strategy, we divide the graph into two subgraphs, find the hull of the two halves, then combine the two. We can show that this can be done in total $O(n\lg n)$ complexity.

The first thing we do is sort the vertices based on their horizontal (x) coordinates. Using heapsort, we can do this in $\Theta(n\lg n)$ time. Then, we follow these steps:

ConvexHull(V)

1 If $|V| <= 3$, compute the hull with brute force and return it
2 Else, divide V into two sets, V_ℓ and V_g, where V_ℓ is the set of vertices with x coordinates less than or equal to the median x coordinates, and V_g is all other points
3 Compute $H_\ell \leftarrow$ ConvexHull(V_ℓ)
4 Compute $H_g \leftarrow$ ConvexHull(V_g)
5 Merge H_ℓ and H_g

Line 5 (the merge) can be done in $\Theta(n)$ time, which I will describe shortly.

Question 6 Page 2

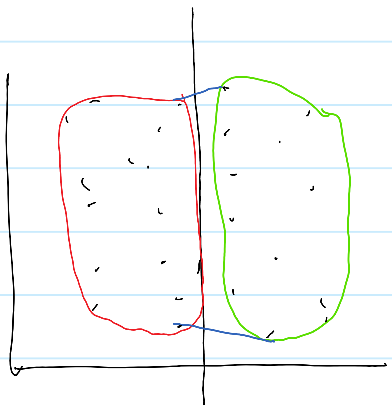
Assuming that the merge step is indeed $\Theta(n)$, we can see that by dividing the set into 2 parts and recurring with each one, we have the following recurrence relation:

$$T(n) = 2T(n/2) + \Theta(n)$$

This is the exact same relation as with merge sort, and using the master theorem we see a total complexity of $\Theta(n \log n)$.

Now that we have demonstrated overall complexity, let us examine the steps and complexity for the merge step, which I claim is $\Theta(n)$.

Once we have the hull for two halves of our graph, we need to find the upper and lower tangents that connect the two (crudely shown in blue here). We then remove all points that lie within these two tangents.



The strategy is to start with the two points with the closest x-coordinates. We then rotate around the two hulls until we find the tangent. On the next page is the algorithm

Question 6 Page 3

FindLowerTangent(H_e, H_g)
    Let A be the rightmost point of H_e
    Let B be the leftmost point of H_g
    While $\overline{AB}$ is not the lower tangent of (H_e, H_g)
        While $\overline{AB}$ is not the lower tangent from H_e
            $A \leftarrow A+1$ //next clockwise vertex in H_e
        While $\overline{AB}$ is not the lower tangent from H_G
            $B \leftarrow B+$ //next counterclockwise vertex in H_G
    Return $\overline{AB}$.

As A and B move around their respective hulls, we toss out the vertices that are visited, thus leaving only the lower tangent. The upper tangent is symmetric, and when both are determined, the hulls have been merged.

What we see is that, in each case, at most K vertices are checked, where $k = |H_e| + |H_G|$. In the worst case, $K = n$, meaning this has complexity $\Theta(n)$.

Thus, we have shown that overall, the recurrence relation is $T(n) = 2T(n/2) + \Theta(n)$, which can be expressed as $\underline{O(n \log n)}$.