Algorithm Engineering

Homework #4

Spencer Kaiser - 28750375

1. [Recurrences]

ai) $t_n = t_{n-1} + 6$ for $n > 1$ $\quad t_1 = 2$

Candidate Solution: $t_n = 6n - 4$

The above recurrence equation can be rewritten as follows:

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

This indicates the base case for the equation is when $\underline{n=1}$. This can be used with the candidate solution as follows:

$$T(1) = (6(1) - 4)$$
$$= (6 - 4)$$
$$= 2$$

$T(1) = 2$ ✓

Inductive Step

It is assumed that $n > 1$ and that $T(n-1) = 6(n-1) - 4$.

Given this, the following substitution can be made:

$$T(n) = T(n-1) + 6$$
$$T(n) = (6(n-1) - 4) + 6$$
$$= 6n - 6 - 4 + 6$$
$$= 6n - 4$$ ✓

Given this result, the recurrence equation is verified.

1.a: Cont Also find the solution if the recurrence is $t_n = t_{n-1} + 8$ given $n > 1, t_1 = 2$

The given recurrence equation can be rewritten as such:

$$T(n) = \begin{cases} 2 & \text{if } n = 1 \\ T(n-1) + 8 & \text{if } n > 1 \end{cases}$$

In order to find the candidate solution, $T(n)$ will be calculated for a small value of n:

$T(1) = 2$
$T(2) = T(1) + 8 = 2 + 8 = 10$ (+8)
$T(3) = T(2) + 8 = 10 + 8 = 18$ (+8)
$T(4) = T(3) + 8 = 18 + 8 = 26$ (+8)
$T(5) = T(4) + 8 = 26 + 8 = 34$

The following pattern between n & $T(n)$ can be identified

$$T(n) = 8n - 6$$

Next, this candidate solution will be tested against the recurrence equation:

$T(1) = 8(1) - 6$
$= 8 - 6$
$= 2$ ✓

Inductive Step

Assuming $n > 1$ and that $T(n-1) = 8(n-1) - 6$, the following substitution can be made:

$T(n) = t(n-1) + 8$
$= (8(n-1) - 6) + 8$
$= 8n - 8 - 6 + 8$
$= 8n - 6$ ✓

Given this result, the recurrence equation is verified.

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

Candidate Solution: $t_n = ((n+1)^2 - (n+1))/2$

The above recurrence equation can be rewritten as follows:

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

First, the candidate solution will be rewritten:

$$\begin{aligned} t_n &= ((n+1)^2 - (n+1))/2 \\ &= (n^2 + 2n + 1 - n - 1)/2 \\ &= (n^2 + n)/2 \\ &= \frac{n^2+n}{2} \end{aligned}$$

Next, the candidate solution will be tested against the recurrence equation:

$$\begin{aligned} T(1) &= \frac{(1)^2 + 1}{2} \\ &= \frac{2}{2} \\ &= 1 \checkmark \end{aligned}$$

Inductive Step

Assuming $n > 1$ and that $T(n-1) = \frac{(n-1)^2 + (n-1)}{2}$, the following substitution can be made:

$$\begin{aligned} T(n) &= T(n-1) + n \\ &= \frac{(n-1)^2 + (n-1)}{2} + n \\ &= \frac{n^2 - 2n + 1 + n - 1}{2} + n \\ &= \frac{n^2 - n}{2} + \frac{2n}{2} \\ &= \frac{n^2 - n + 2n}{2} \\ &= \frac{n^2 + n}{2} \checkmark \end{aligned}$$

Given this result, the recurrence equation is verified.

1-bi Solve the following recurrence equations using substitution

$$t_n = t_{n-1} + n^2 \quad \text{for } n > 1 \qquad t_1 = 2$$

This recurrence equation can be rewritten as the following:

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

In order to find a candidate solution, a small subset of n values will be calculated:

$$T(1) = 2$$
$$T(2) = T(1) + 2^2 = 2 + 2^2 = 6$$
$$T(3) = T(2) + 3^2 = 2 + 2^2 + 3^2 = 15$$
$$T(4) = T(3) + 4^2 = 2 + 2^2 + 3^2 + 4^2 = 31$$
$$T(5) = T(4) + 2 + 2^2 + 3^2 + 4^2 + 5^2 = 56$$
$$T(6) = T(5) + 2 + 2^2 + 3^2 + 4^2 + 5^2 + 6^2 = 92$$

The pattern above is easily recognized as the sum of squares. As such, the recurrence equation can be rewritten as the following:

$$T(n) = 1 + \sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6} + 1$$

Next, this candidate solution is tested with the base case:

$$T(1) = 1 + (1)^2 = 2 \checkmark$$

Inductive Step - assuming $n > 1$ and $T(n-1) = \frac{(n-1)(n-1+1)(2(n-1)+1)}{6} + 1$

$$T(n) = T(n-1) + n^2$$
$$= \frac{(n-1)(n-1+1)(2(n-1)+1)}{6} + 1 + n^2$$
$$= \frac{(n-1)(n)(2n-1)}{6} + 1 + n^2$$
$$= \frac{(n^2-n)(2n-1)}{6} + 1 + n^2$$
$$= \frac{2n^3 - n^2 - 2n^2 + n + 6n^2}{6} + 1$$
$$= \frac{2n^3 + 3n^2 + n}{6} + 1$$
$$= \frac{n(2n^2+3n+1)}{6} + 1$$
$$= \frac{n(n+1)(2n+1)}{6} + 1 \checkmark$$

Given this result, the recurrence is verified.

1-bi cont. $t_n = t_{n-1} + n^2$ for $n > 1$ $t_1 = 2$

Using the substitution method, the recurrence can be solved:

Guess: $O(n^2)$ → Prove $T(n) \leq cn^2$

Using this guess, we substitute this value into the original recurrence.

Base Case:

$T(1) \leq c(1)^2$

$2 \leq c$ — True if $c < 2$ ✓

Inductive Step

$$\begin{aligned} T(n) &= T(n-1) + n^2 \\ &\leq c(n-1)^2 + n^2 \\ &= c(n^2 - 2n + 1) + n^2 \\ &= cn^2 + n^2 - 2n + c \\ &\leq cn^2 + n^2 \\ &\leq cn^2 \end{aligned}$$

$$\begin{aligned} T(2) &= T(1) + 2^2 \\ &= 2 + 4 \\ &= 6 \end{aligned}$$

$$\begin{aligned} T(2) &= T(1) + 2^2 \\ &\leq c(1)^2 + 2^2 \\ &\leq c + 4 \end{aligned}$$

True if $c \geq 2$

This substitution shows that $T(n) \leq cn^2$. As long as $c \geq 2$, the substitution holds, resulting in a bound of $O(n^2)$.

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

The recurrence equation can be rewritten as the following:

$$T(n) = \begin{cases} 1 & \text{if } n = 1 \\ 3T(n-1) + 2^n & \text{if } n > 1 \end{cases}$$

In order to find a candidate solution a small subset of n values will be calculated:

$T(1) = 1$
$T(2) = 3T(0) + 2^2 = 3(1) + 2^2 = 7$
$T(3) = 3T(2) + 2^3 = 3 + 2^2 + 2^3 = 15$
$T(4) = 3T(3) + 2^4 = 3 + 2^2 + 2^3 + 2^4 = 31$
$T(5) = 3T(4) + 2^5 = 3 + 2^2 + 2^3 + 2^4 + 2^5 = 63$
$T(6) = 3T(5) + 2^6 = 3 + 2^2 + 2^3 + 2^4 + 2^5 + 2^6 = 127$

Given the pattern in the calculations above, the recurrence can be represented by the following equation:

$$T(n) = \sum_{i=1}^{n} 2^n$$

In order to solve the recurrence, substitution must be used:

Guess: 2^n
Prove: $T(n) \leq C2^n$

Base Case: $T(1) \leq C2^1$
$T(1) \leq 2C$
$1 \leq 2C$
True if $C \geq .5$

Induction Step

$T(n) = 3T(n-1) + 2^n$
$T(n) \leq 3(C2^{n-1}) + 2^n$
$\leq 3(C(2^n \cdot 2^{-1})) + 2^n$
$\leq C(2^n \cdot \frac{1}{2}) + 2^n$
$\leq C(2^n) + 2^n$
$\leq C2^n + 2^n$
$\leq C2^n$

$T(2) = 3T(1) + 2^2$
$= 3(1) + 4$
$= 7$
$T(2) \leq 3(C2^1) + 2^2$
$\leq 3(C2) + 4$
$7 \leq 6C + 4$
True if $C \geq .5$

This substitution proves to be true, therefore $T(n) \leq Cn^2$, therefore the bound is $O(2^n)$

2 [Recurrences]

a) Simplify the master theorem for solving recurrences of the form (where a, b, c are positive integers greater than unity: $T(n) = aT(n/b) + cn$

The master theorem can be rewritten in terms of $f(n)$ as such:

$$T(n) = aT(n/b) + f(n)$$

$$f(n) = T(n) - aT(n/b)$$

Given this representation, there are three possible cases for the asymptotic bounds of $f(n)$, assuming $f(n)$ is a positively bounded function:

1) $f(n) = O(n^{\log_b a - \epsilon})$ where constant $\epsilon > 0$

2) $f(n) = \Theta(n^{\log_b a} \log^k n)$ where $k \geq 0$

3) $f(n) = \Omega(n^{\log_b a + \epsilon})$ where $\epsilon > 0$

Depending on the cases above, respective bounds of $T(n)$ are the following:

1) $T(n) = \Theta(n^{\log_b a})$

2) $T(n) = \Theta(n^{\log_b a} \log^{k+1} n)$

3) $T(n) = \Theta(f(n))$

Given that c is a positive integer, the Master Theorem can be simplified to the three following cases:

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

2-b1 Give an asymptotic formula for each of the following recurrence equations, using big "Θ" rather than just big "O" for better results when possible. Show the values of the next 5 values of $T(n)$ in each case

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

$T(5) = 4T(5-4) + 1 = 4T(1) + 1 = 4(2) + 1 = 9$

$T(6) = 4T(6-4) + 1$
$T(7) = 4T(7-4) + 1$
$T(8) = 4T(8-4) + 1$
$= 4(2) + 1 = 9$

$T(9) = 4T(9-4) + 1 = 4T(5) + 1 = 4(9) + 1 = 37$

The recurrence can also be represented as follows:

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

$= 4(4T(n-8) + 1) + 1 = 16T(n-8) + 1 + 4$

$= 16(4T(n-12) + 1) + 5 = 64T(n-12) + 1 + 4 + 16$

$= 64(4T(n-16) + 1) + 21 = 256T(n-16) + 1 + 4 + 16 + 64$

In order to solve the recurrence, the substitution method must be used.

Guess: $T(n) = O(2^n)$
Prove: $T(n) \leq c2^n$

Base Case:
$T(1) \leq c2^{(1)}$
$T(1) \leq c2$
$2 \leq c2$
} true if $c \leq 1$

$T(n) = 4T(n-4) + 1$
$\leq 4(c2^{n-1}) + 1$
$\leq 4(c(2^n \cdot 2^{-1})) + 1$
$\leq 4(c(2^n \cdot \frac{1}{2}))$
$\leq 4(c2^n)$
$\leq c2^n$

$T(5) = 4T(1) + 1$
$= 4(2) + 1$
$= 7$
$T(5) = 4(c2^1) + 1$
$= 4(2c) + 1$
$= 8c + 1$
True if $c \geq .875$

The substitution proves $T(n) \leq c2^n$, therefore, the bound of the recurrence is $O(2^n)$

2-b2 $T(n) = 3T(\lfloor n/4 \rfloor) + 2n \quad T(0) = T(1) = 1$

Because the recurrence equation follows the pattern of the master theorem, it can be used to determine the asymptotic bound.

$T(n) = 3T(\lfloor n/4 \rfloor) + 2n$ (a = 3, b = 4)

Because $a < b$, the bound of this reccurrence equation is $\Theta(n)$.

The next five terms are as follows

$T(2) = 3T(\lfloor 2/4 \rfloor) + 2(2) = 3T(0) + 2(2) = 3(1) + 4 = 7$

$T(3) = 3T(\lfloor 3/4 \rfloor) + 2(3) = 3T(0) + 2(3) = 3(1) + 6 = 9$

$T(4) = 3T(\lfloor 4/4 \rfloor) + 2(4) = 3T(1) + 2(4) = 3(1) + 8 = 11$

$T(5) = 3T(\lfloor 5/4 \rfloor) + 2(5) = 3T(1) + 2(5) = 3(1) + 10 = 13$

$T(6) = 3T(\lfloor 6/4 \rfloor) + 2(6) = 3T(1) + 2(6) = 3(1) + 12 = 15$

2-b3 $T(n) = 4T(\lfloor n/2 \rfloor) + 2n^2 \quad T(1) = 1$

Because this recurrence equation also follows the master theorem, the master theorem can be used to determine the asymptotic bound:

$a = 4 \quad b = 2 \quad f(n) = 2n^2$

$2n^2 \neq O(n^{\log_2 4 - \epsilon}) = O(n^{2-\epsilon})$

$2n^2 = O(n^{\log_2 4}) = O(n^2)$ ✓

After comparing $f(n)$ against the cases declared by the Master theorem, the following bound applies to this recurrence:

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

$T(2) = 4T(\lfloor 2/2 \rfloor) + 2(2)^2 = 4T(1) + 2(4) = 4(1) + 8 = 12$

$T(3) = 4T(\lfloor 3/2 \rfloor) + 2(3)^2 = 4T(1) + 2(9) = 4(1) + 18 = 22$

$T(4) = 4T(\lfloor 4/2 \rfloor) + 2(4)^2 = 4T(2) + 2(16) = 4(12) + 32 = 80$

$T(5) = 4T(\lfloor 5/2 \rfloor) + 2(5)^2 = 4T(2) + 2(25) = 4(12) + 50 = 98$

$T(6) = 4T(\lfloor 6/2 \rfloor) + 2(6)^2 = 4T(3) + 2(36) = 4(22) + 72 = 160$

2-b4 $T(n) = 1/n + T(n-1) \quad T(1) = 2$

$T(2) = \frac{1}{2} + 2 = 5/2 = 2.5$

$T(3) = \frac{1}{3} + \frac{1}{2} + 2 = \frac{2}{6} + \frac{15}{6} = 17/6 = 2.83$

$T(4) = \frac{1}{4} + \frac{1}{3} + \frac{1}{2} + 2 = \frac{6}{24} + \frac{66}{24} = 74/24 = 37/12 = 3.08$

$T(5) = \frac{1}{5} + \frac{1}{4} + \frac{1}{3} + \frac{1}{2} + 2 = 3.28$

$T(6) = \frac{1}{6} + \frac{1}{5} + \frac{1}{4} + \frac{1}{3} + \frac{1}{2} + 2 = 3.45$

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

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

$= \frac{1}{n} + \frac{1}{n-1} + \frac{1}{n-2} + T(n-3)$ Base case

$\frac{1}{n} + \frac{1}{n-1} + \frac{1}{n-2} \cdots + \frac{1}{n-n} + T(1)$

$T(n) = \sum_{i=2}^{n} \frac{1}{n} + T(1)$

This equation is a common equation known as a "harmonic series" which is bound by $O(\lg n)$, however, because the recurrence requires a summation from 2 to n, the overall asymptotic bound is $O(n)$.

Question #3

3a

$$T(n) = (n-1) + \frac{1}{n}\sum_{i=0}^{n-1}(T(i)+T(n-1-i)) \qquad T(0) = 0$$

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

T(n) in terms of S(n):

Through substitution, the function for T(n) can be easily represented in terms of S(n) as follows:

$$T(n) = (n-1) + \frac{1}{n}\underbrace{\sum_{i=0}^{n-1}(T(i)+T(n-1-i))}_{S(n)} \rightarrow \boxed{T(n) = (n-1) + \frac{1}{n}S(n)}$$

S(n) in terms of S(n-1) and T(n-1)

After recognizing a pattern while evaluating S(n) for 1, 2, ... 10, S(n) can be described with the following formula:

$$S(n) = \sum_{i=0}^{n-1} 2T(n-1)$$

Using this formula and after further examining the patterns it creates, S(n) can also be represented as follows:

$$\boxed{S(n) = S(n-1) + 2T(n-1)}$$

Kaiser - 287503375

3b $T(n)$ for $n = 1, 2, \ldots 10$

$$T(n) = (n-1) + \frac{1}{n}\sum_{i=0}^{n-1}\underbrace{(T(i) + T(n-1-i))}_{=2T(n-1)}$$

← determined while performing $S(n)$ calculations

$T(1) = (1-1) + \frac{1}{1}(0+0) = 0$

$T(2) = (2-1) + \frac{1}{2}(\underset{T(0)}{0} + \underset{T(1)}{0} + \underset{T(0)}{0} + \underset{T(0)}{0}) = 1$

$T(3) = (3-1) + \frac{1}{3}(\underset{T(0)}{0} + \underset{T(2)}{1} + \underset{T(1)}{0} + \underset{T(1)}{0} + \underset{T(2)}{1} + \underset{T(0)}{0}) = 2 + 2/3 \approx 2.667$

$T(4) = (4-1) + \frac{1}{4}(2T(0) + 2T(1) + 2T(2) + 2T(3))$

$= 3 + \frac{1}{4}(0 + 0 + 2 + (4 + 4/3)) \approx 4.83$

$T(5) = (5-1) + \frac{1}{5}(2T(0) + 2T(1) + 2T(2) + 2T(3) + 2T(4))$

$= 4 + \frac{1}{5}(17) \approx 7.4$

$T(6) = (6-1) + \frac{1}{6}(2T(0) + 2T(1) + 2T(2) + 2T(3) + 2T(4) + 2T(5))$

$= 5 + \frac{1}{6}(31.8) \approx 10.3$

$T(7) = (7-1) + \frac{1}{7}(2T(0) + 2T(1) + 2T(2) + 2T(3) + 2T(4) + 2T(5) + 2T(6))$

$= 6 + \frac{1}{7}(52.4) \approx 13.49$

$T(8) = (8-1) + \frac{1}{8}(2T(0) + 2T(1) + 2T(2) + 2T(3) + 2T(4) + 2T(5) + 2T(6) + 2T(7))$

$= 7 + \frac{1}{8}(79.3) \approx 16.92$

$T(9) = (9-1) + \frac{1}{9}(2T(0) + 2T(1) + 2T(2) + 2T(3) + 2T(4) + 2T(5) + 2T(6) + 2T(7) + 2T(8))$

$= 8 + \frac{1}{9}(113.2) \approx 20.58$

$T(10) = (10-1) + \frac{1}{10}(2T(0) + 2T(1) + 2T(2) + 2T(3) + 2T(4) + 2T(5) + 2T(6) + 2T(7) + 2T(8) + 2T(9))$

$= 9 + \frac{1}{10}(154.3) \approx 24.44$

3b $S(n)$ for $n = 1, 2, \ldots 10$ $\quad S(n) = \sum_{i=0}^{n-1} (T(i) + T(n-1-i))$

$S(1) = T(0) + T(0)$

$S(2) = T(0) + T(1) + T(1) + T(0)$

$S(3) = T(0) + T(2) + T(1) + T(1) + T(2) + T(0)$

$S(4) = T(0) + T(3) + T(1) + T(2) + T(2) + T(1) + T(3) + T(0)$

$= 2T(0) + 2T(1) + 2T(2) + 2T(3)$

$S(5) = T(0) + T(4) + T(1) + T(3) + T(2) + T(2) + T(3) + T(1) + T(4) + T(0)$

$= 2T(0) + 2T(1) + 2T(2) + 2T(3) + 2T(4)$

Recognizing the pattern above...

$S(6) = 2T(0) + 2T(1) + 2T(2) + 2T(3) + 2T(4) + 2T(5)$

$S(7) = 2T(0) + 2T(1) + 2T(2) + 2T(3) + 2T(4) + 2T(5) + 2T(6)$

$S(8) = 2T(0) + 2T(1) + 2T(2) + 2T(3) + 2T(4) + 2T(5) + 2T(6) + 2T(7)$

$S(9) = 2T(0) + 2T(1) + 2T(2) + 2T(3) + 2T(4) + 2T(5) + 2T(6) + 2T(7)$

$+ 2T(8)$

$S(10) = 2T(0) + 2T(1) + 2T(2) + 2T(3) + 2T(4) + 2T(5) + 2T(6) + 2T(7)$

$+ 2T(8) + 2T(9)$

$S(n)$ can be summarized by the following equation:

$$S(n) = \sum_{i=0}^{n-1} 2T(n-1)$$

$S(1) = 0$

$S(2) = 0$

$S(3) = 2$

$S(4) = 7.33$

$S(5) = 17$

$S(6) = 31.8$

$S(7) = 52.4$

$S(8) = 79.37$

$S(9) = 113.21$

$S(10) = 154.37$

3c

What are the time & space requirements for computing T(n)?

Time Complexity

When computing T(n), one calculation must be made for all values between the base case and n. In order to calculate T(n) with only a single calculation for all values between the base case and n, two variables must be used. For each T(i) calculation, the value of T(i-1) must be saved, & the value of S(i-1). Using these values, S(i) can be computed by using $S(i-1) + 2T(i-1)$, which is the exact equation discussed in 3a for representing S(n) in terms of S(n-1) and T(n-1). Because each step of the calculation only requires 2 values from the step before it, the overall space complexity is $O(n)$.

Space Complexity

Because the solution discussed above only requires the use of two variables, the overall space complexity of this solution is $O(1)$.

4 [Polynomial and Matrix Multiplication]

4 a-a Show how to multiply two linear polynomials $ax+b$ and $cx+d$ using only three multiplications.

Performing the multiplication of $(ax+b)(cx+d)$ results in the following:

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

looking at the result, it becomes apparent that there are 4 sets of coefficient multiplications. However, utilizing the provided hint, the following can be identified:

$$(a+b)(c+d) = ad + bc + ab + cd$$

of this result, both ad and bc are terms in the middle term of the original result. Using this finding, the original result can be rewritten as the following:

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

This result only involves three multiplications: ac, bd, and $(a+b)(c+d)$.

4 a-b Give two divide and conquer algorithms for multiplying two polynomials of degree-bound n in $\Theta(n^{\lg 3})$ time. The first algorithm should divide the input polynomial coefficients into a high half and a low half, and the second algorithm should divide them according to whether their index is odd or even.

The first algorithm, known as the "High/Low Algorithm" divides the input polynomial coefficients into a "high half" and a "low half". This algorithm employs a divide and conquer approach which begins by splitting the P and Q polynomials, which belong to degree-bound n, into individual polynomials A, B, C, and D, which are degree-bound m.

Next, the algorithm calculates the expression $(Ax^m + B)((Cx^m + D)$ by utilizing recursion for the multiplications.

One final component of this algorithm involves shifting the bits of the resulting polynomials. This is required to ensure accurate powers of X.

4 a-b cont.

The second algorithm also employs a divide and conquer approach to solve the problem, however this algorithm divides elements based on whether the index of the elements are odd or even.

The primary difference between the High/Low algorithm and the Odd/Even algorithm is what happens in the "Conquer" component of the execution. In the Odd Even algorithm, powers of x are shifted 2 and 1 as opposed to n and m respectively.

40 Use Strassen's algorithm to compute the following matrix product:

$$\begin{pmatrix} 1 & 3 \\ 7 & 5 \end{pmatrix}\begin{pmatrix} 6 & 8 \\ 4 & 2 \end{pmatrix}$$

To begin the process, the S-values are calculated:

$S_1 = B_{12} - B_{22} = 8 - 2 = 6$
$S_2 = A_{11} + A_{12} = 1 + 3 = 4$
$S_3 = A_{21} + A_{22} = 7 + 5 = 12$
$S_4 = B_{21} - B_{11} = 4 - 6 = -2$
$S_5 = A_{11} + A_{22} = 1 + 5 = 6$
$S_6 = B_{11} + B_{22} = 6 + 2 = 8$
$S_7 = A_{12} - A_{22} = 3 - 5 = -2$
$S_8 = B_{21} + B_{22} = 4 + 2 = 6$
$S_9 = A_{11} - A_{21} = 1 - 7 = -6$
$S_{10} = B_{11} + B_{12} = 6 + 8 = 14$

Next, the P values are calculated

$P_1 = A_{11} \cdot S_1 = 1 \cdot 6 = 6$
$P_2 = S_2 \cdot B_{22} = 4 \cdot 2 = 8$
$P_3 = S_3 \cdot B_{11} = 12 \cdot 6 = 72$
$P_4 = A_{22} \cdot S_4 = 5 \cdot -2 = -10$
$P_5 = S_5 \cdot S_6 = 6 \cdot 8 = 48$
$P_6 = S_7 \cdot S_8 = -2 \cdot 6 = -12$
$P_7 = S_9 \cdot S_{10} = -6 \cdot 14 = -84$

Using the S and P values to the left, finally the C values will be calculated:

$C_{11} = P_5 + P_4 - P_2 + P_6$
$= 48 - 10 - 8 - 12$
$= 18$

$C_{12} = P_1 + P_2$
$= 6 + 8$
$= 14$

$C_{21} = P_3 + P_4$
$= 72 - 10$
$= 62$

$C_{22} = P_5 + P_1 - P_3 - P_7$
$= 48 + 6 - 72 + 84$
$= 66$

Using those values, the resulting matrix is as follows:

$$C = \begin{pmatrix} 18 & 14 \\ 62 & 66 \end{pmatrix}$$

4c How quickly can you multiply a $kn \times n$ matrix by an $n \times kn$ matrix, using Strassen's algorithm as a subroutine? Answer the same question with the order of the input matrices reversed.

Given the A & B matrices described above, they can be visually represented by the following:

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

Multiplying these matrices results in a new matrix with dimensions $kn \times kn$. This matrix is illustrated below:

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

Strassens subroutine performs in $\Theta(n^{\log 7})$ time and because the resulting matrix contains $k \times k$ (k^2) entries, the overall complexity is $\underline{\Theta(k^2 n^{\log 7})}$.

4c cont.

Reversing the process from the previous section involves multiplying an $n \times nk$ matrix with a $nk \times n$ matrix. The resulting matrix is of size $n \times n$.

$$A = [A_1 \; A_2 \; A_3 \cdots A_k]$$

$$B = \begin{bmatrix} B_1 \\ B_2 \\ B_3 \\ \vdots \\ B_k \end{bmatrix}$$

Because this method involves performing k calculations using Strassen's subroutine, the overall complexity is $\Theta(k n^{\log 7})$.

5 [Weighted Median]

For n distinct elements $x_1, x_2, \ldots, x_n$ with positive weights $w_1, w_2, \ldots, w_n$ such that $\sum_{i=1}^{n} w_i = 1$, the weighted (lower) median is the element x_k satisfying

$$\sum_{x_i < x_k} w_i < 1/2 \quad \text{and} \quad \sum_{x_i > x_k} w_i \leq 1/2$$

5a argue that the median of $x_1, x_2, \ldots, x_n$ is the weighted median of the x_i with weights $w_i = 1/n$ for $i = 1, 2, \ldots, n$

An element is considered to be the median of the set if the number of elements less than x_k is $\leq n/2$ and if the number of elements greater than x_k is $\leq n/2$. Furthermore, if each element x_i is assigned a weight of w_i where $w_i = \frac{1}{n}$, the sum of all weights of all elements less than x_k is $< 1/2$ and the weights of all elements greater than x_k is $\leq 1/2$.

Sum of weights of elements smaller than x_k

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

Assuming x_k is the median, $\frac{1}{n}$ will be summed $\frac{n}{2}$ times. In other words, the number of elements smaller than x_k can be represented as $\frac{n}{2} \cdot \frac{1}{n} = \frac{n}{2n} = \frac{1}{2}$

5a Cont.

Similarly, the sum of weights of all elements greater than X_k can be represented by the following:

$$\sum_{i=K+1}^{N} W_i = \sum_{i=K+1}^{n} \frac{1}{n}$$

Again, assuming X_k is the median, this summation will sum $1/n$ $n/2$ times, which can be represented as the following:

$$\frac{1}{n} \cdot \frac{n}{2} = \frac{n}{2n} = \frac{1}{2}$$

Given these two representations, it can be proved that the median of set X_i is the weighted median of the set.

5b Show how to compute the weighted median of n elements in $O(n \lg n)$ worst-case time using sorting.

In order to find the weighted median of n elements in $O(n \lg n)$ worst-case time, the data must first be sorted. By utilizing merge sort or heap sort, the elements can be sorted by value.

After sorting has finished, the weighted median can be found in $O(n)$ time. In order to do this, you need a single variable. A single pass through the elements must be made where the weights of each element are added to the stored variable. This process continues until the sum is $\geq \frac{1}{2}$. At this point, the weighted median is the current element. Because this element X_k caused the sum to become $\geq \frac{1}{2}$, the sum of all preceeding elements must be $\leq \frac{1}{2}$, which is the requirement for a weighted median. Furthermore, because the sum became $\geq \frac{1}{2}$, the sum of the remaining elements must be $\leq \frac{1}{2}$.

5c Show how to compute the weighted median in $\Theta(n)$ worst case time using a linear-time median algorithm.

In order to find the weighted median in $\Theta(n)$ time, the Select algorithm must be used. The algorithm resembles a binary search as the Select algorithm finds the median of the set and then recurses into the half that contains the weighted median. After the median of the set is found, the sums of the two halves created by splitting at the median determines the location of the weighted median. If both halves have a total weight of $\leq 1/2$, the median is the weighted median. If, however, one side has a sum of greater than $1/2$, the algorithm recurses into that half.

As described above, this is a modified Select algorithm which aims to find the weighted median. In order to account for this, the recursive function, which operates in $\Theta(n)$ time, accepts the number which partitions the weights. This process continues until the weights of the lower array and upper array are $\leq 1/2$, at which point the weighted median is found.

6 [Points in Space]

In order to solve the convex hull problem in $O(n \lg n)$ time, divide and conquer must be used to recursively split the nodes & then a clever, yet simple, algorithm to merge the sets.

First, the data must be sorted by X-position. In order to do this in $O(n \lg n)$ time, heapsort will be used. Once the data is sorted, the nodes are then split into 2 subsets. Each split is made in $\Theta(n)$ time with the resulting left & right subsets containing $\lceil n/2 \rceil$ and $\lfloor n/2 \rfloor$ points respectively. Each subset is recursively split until the base case, which is discussed next, is reached. Overall, the entire splitting process using divide and conquer can be described with the recurrence $T(n) = 2T(n/2) + O(n)$ and runs in $O(n \lg n)$ time.

The recursive splitting process continues splitting subsets until each subset contains i nodes where $3 \leq i < 6$. At this point, the special terminal case is reached and the problem becomes trivial. In order to solve each subproblem of creating a convex hull with subsets of 3, 4, or 5, Jarvis's March is used. This algorithm uses what is known as "Package Wrapping" to compute the convex hull of the subset in $O(nh)$, where h is the number of vertices in the convex hull being formed.

6 Cont

At this point, the graph has been recursively split and the merging algorithm begins. Thus far, all components of the divide and conquer approach have operated in $O(n \lg n)$ time, so it is up to the merging process to remain under this time complexity restriction.

The first step in the merging process involves making a pass through the left subset, S_L, and the right subset, S_R. This pass is tasked with determining the two nodes with the highest and lowest y-positions within each of the two subsets. Because each pass must iterate over, at worst, $n/2$ points, each pass operates in $O(n)$ time.

After determining each subset's highest and lowest points, the merging process must first remove what would become inner-paths within the convex hull which will be formed by the merging process. In other words, without this step, the convex hull would be formed with two paths within it; these paths, which represent the right and left portions of S_L and S_R respectively must be removed.

This removal process begins with the highest point of S_L. From this point, the path to the lowest point in S_L is found by identifying the first edge belonging to the highest point begining with the top-middle and moving clockwise. Once this path is found, every edge along the path is removed until the lowest point of the set is reached. The same process continues on the right subset, however, the process begins with the lowest node of

6 cont. 2

S_R and searches for the path by moving clockwise from the bottom-middle of that node. Once both paths have been removed, the process of actually merging the two subgraphs can begin. This process performs in $O(n)$.

The process of merging the two subgraphs is quite simple. In order to create one convex hull from S_L and S_R, an edge is created between the two highest points and another edge is created connecting the two lowest. Because each set of edges has been previously identified, creating the two edges can be done in $O(1)$ time.

After merging the two subsets, the recursive process continues merging adjacent subsets until all subsets have been merged. The final result is a convex hull for all n nodes which was created by utilizing a sorting algorithm which performs in $O(n \lg n)$ time, a divide and conquer, recursive splitting process which performs in $O(n \lg n)$ time, and a merging process which performs in $O(n)$ time. As a result, the entire implementation performs in $O(n \lg n)$ time.