4: Divide & Conquer

In this chapter offers 3 methods for solving recurrences — that is, for obtaining asymptotic 'Θ' or 'O' bounds on the solution:

① Substitution Method: we guess a bound & then use mathematical induction to prove our guess correct.

② Recurrence-tree method: converts the recurrence into a tree whose nodes represent the costs incurred at various levels of the recursion. We use techniques for bounding summations to solve the recurrence.

③ Master method: provides bounds for recurrence of the form

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

where $a \geq 1$, $b > 1$ & $f(n)$ is a given function.

4.1: The maximum-subarray problem

A brute-force solution: we can easily devise a brute-force solutn. to this problem: just try every possible pair of buy & sell dates in which the buy date precedes the sell date. A period of n days has $\binom{n}{2}$ such pairs of days.

Since $\binom{n}{2}$ is $\Theta(n^2)$, the best we can hope for is to evaluate each pair of dates in constant time, this approach would take $\Omega(n^2)$ time. Can we do better?

↓ maximum subarray.

Exercises

4.1-1: What does FIND-MAXIMUM-SUBARRAY return when all elements of A are negative?

If the index of greatest element of A is i, then it returns (i, i, A[i]).

4.1-2: Write pseudocode for the brute-force method of solving the maximum-subarray problem. Your procedure should run in $\Theta(n^2)$ time.

```
MAX-SUBARRAY-BRUTE-FORCE(A)
    n = A.length
    max-so-far = -∞
    for l = 1 to n
        sum = 0
        for h = l to n
            sum = sum + A[h]
            if sum > max-so-far
                max-so-far = sum
                low = l
                high = h
    return (low, high).
```

4.1-3: Implement both the brute-force & recursive algorithms for the maximum-subarray problem on your own computer. What problem size n_0 gives the crossover point at which the recursive algorithm beats the brute-force algorithm? Then, change the base case of the recursive algo. to use the brute-force algo. whenever the problem size is less than n_0. Does that change the crosspoints?

On my computer, n_0 is 37,

If the algo. is modified to used divide & conquer when $n \geq 37$ & the brute-force approach when n is less, the performance at the crossover point almost doubles. The performance at $n_0 - 1$ stays the same, though (or even gets worse, because of the ~~brute~~ added overhead).

What I find interesting is that if we set $n_0 = 20$ & used the mixed approach to start 40 elements, it is still faster than both.

4.1-4: Suppose we ~~change~~ change the definition of the maximum-subarray problem to allow the result to be an empty subarray where the sum of the values of an empty subarray is 0. How would you change any of the algorithms that do not allow empty subarrays to permit an empty subarray to be the result?

If the original algorithm returns a negative sum, returning an empty subarray instead.

4.1-5: Use the following ideas to develop a nonrecursive, linear-time algorithm for the maximum-subarray problem. Start at the left end of the array & progress toward the right, keeping track of the maximum subarray seen so far. Knowing a maximum subarray $A[1..j]$, extend the answer to find maximum subarray ending at index $j+1$ by using the following observation: a maximum subarray $A[i..j+1]$ is either a maximum subarray of $A[1..j]$ or a subarray $A[i..j+1]$ for some $1 \leq i \leq j+1$.

Determine the maximum subarray of the form $A[i..j+1]$ in constant time based on knowing a maximum subarray ending at index j.

```
MAX-SUBARRAY-LINEAR(A)
  n = A.length
  max-sum = -∞
  ending-here-sum = -∞
  for j = 1 to n ⟶ O(n) ⇒ Time complexity.
      ending-here-high = j
      if ending-here-sum > 0
          ending-here-sum = ending-here-sum + A[j]
      else ending-here-low = j
          ending-here-sum = A[j]
      if ending-here-sum > max-sum
          max-sum = ending-here-sum
          low  = ending-here-low
          high = ending-here-high
  return (low, high, max-sum)
```

4.2 : Strassen's algorithm for matrix multiplication

→ The key to strassen's method is to make the recursive tree less bushy.

→ Instead of performing 8 recursive multiplication of $n/2 \times n/2$ matrices, it performs only seven.

→ It has four steps:

① Divide the I/p matrices A & B & output matrix C into $n/2 \times n/2$ submatrices. ⟶ $\Theta(1)$

② Create 10 matrices $S_1, S_2, \ldots, S_{10}$, each of which is $n/2 \times n/2$ & is the sum or difference of 2 matrices created in step① ⟶ $\Theta(n^2)$

③ Using the submatrices created in step① & 10 matrices created in step②, recursively compute seven matrix products $P_1, P_2, \ldots, P_7$. Each matrix P_i is $n/2 \times n/2$.

④ Compute the desired submatrices $C_{11}, C_{12}, C_{21}, C_{22}$ of the result matrix C by adding & subtracting various combinations of the P_i matrices. ⟶ $\Theta(n^2)$ times.

$$T(n) = \begin{cases} \Theta(1) & \text{if } n=1, \\ 7T(n/2) + \Theta(n^2) & \text{if } n>1. \end{cases}$$

Analysis: $T(n) = 7T(n/2) + \Theta(n^2)$

$a=7,\ b=2,\ k=2,\ p=0$

Master method;

$T(n) = aT(n/b) + \Theta(n^k \lg^p n)$

$a \geq 1,\ b>1,\ k \geq 0,\ p \in \mathbb{R}$

$a > b^k$
$7 > 2^2$
$7 > 4$

} Use case 1:

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

$$T(n) = \Theta(n^{\log_2 7})$$

$$\boxed{T(n) = \Theta(n^{\lg 7})}$$ time complexity

$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}$, $B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}$ & $C = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix}$

Step(2): We create the following 10 matrices:

$S_1 = B_{12} - B_{22}$, $S_2 = A_{11} + A_{12}$,

$S_3 = A_{21} + A_{22}$, $S_4 = B_{21} - B_{11}$,

$S_5 = A_{11} + A_{22}$, $S_6 = B_{11} + B_{22}$

$S_7 = A_{12} - A_{22}$, $S_8 = B_{21} + B_{22}$

$S_9 = A_{11} - A_{21}$, $S_{10} = B_{11} + B_{12}$.

Step(3): We recursively multiply $n/2 \times n/2$ matrices seven times to compute the following $n/2 \times n/2$ matrices, each of which is the sum or difference of products of A & B submatrices:

$P_1 = A_{11} \cdot S_1 = A_{11} \cdot B_{11} - A_{11} \cdot B_{22}$,

$P_2 = S_2 \cdot B_{22} = A_{11} \cdot B_{22} + A_{12} \cdot B_{22}$,

$P_3 = S_3 \cdot B_{11} = A_{21} \cdot B_{11} + A_{22} \cdot B_{11}$,

$P_4 = A_{22} \cdot S_4 = A_{22} \cdot B_{21} - A_{22} \cdot B_{11}$,

$P_5 = S_5 \cdot S_6 = A_{11} \cdot B_{11} + A_{11} \cdot B_{22} + A_{22} \cdot B_{11} + A_{22} \cdot B_{22}$

$P_6 = S_7 \cdot S_8 = A_{12} \cdot B_{21} + A_{22} \cdot B_{22} - A_{22} \cdot B_{21} - A_{22} \cdot B_{22}$

$P_7 = S_9 \cdot S_{10} = A_{11} \cdot B_{11} + A_{11} \cdot B_{12} - A_{21} \cdot B_{11} - A_{21} \cdot B_{12}$.

Step(4): adds & subtracts the P_i matrices created in step (3) to construct the 4 $n/2 \times n/2$ submatrices of the product C.

$C_{11} = P_5 + P_4 - P_2 + P_6$

$C_{12} = P_1 + P_2$

$C_{21} = P_3 + P_4$

$C_{22} = P_5 + P_1 - P_3 - P_7$.

Exercises

4.2-1: Use strassen's algorithm to compute the matrix product $\begin{pmatrix} 1 & 3 \\ 7 & 5 \end{pmatrix} \begin{pmatrix} 6 & 8 \\ 4 & 2 \end{pmatrix}$

step(1): $A = \begin{pmatrix} 1 & 3 \\ 7 & 5 \end{pmatrix}$. $B = \begin{pmatrix} 6 & 8 \\ 4 & 2 \end{pmatrix}$

step(2): $S_1 = B_{12} - B_{22} = 8 - 2 = 6$

$S_2 = A_{11} + A_{12} = 1 + 3 = 4$

$S_3 = A_{21} + A_{22} = 7 + 5 = 12$

$S_4 = B_{21} - B_{11} = 4 - 6 = -2$

$S_5 = A_{11} + A_{22} = 1 + 5 = 6$

$S_6 = B_{11} + B_{22} = 6 + 2 = 8$

$S_7 = A_{12} - A_{22} = 3 - 5 = -2$

$S_8 = B_{21} + B_{22} = 4 + 2 = 6$

& B.

A+

A+

-B

A+

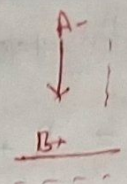
B+

A-

B+

$S_9 = A_{11} - A_{21} = 1 - 7 = -6$

$S_{10} = B_{11} + B_{12} = 6 + 8 = 14$



Step(3):

~~$P_1 = P_5 + P_4 - P_2 + P_6$~~

$P_1 = A_{11} \cdot S_1 = 1 * 6 = 6$

$P_2 = S_2 \cdot B_{22} = 4 * 2 = 8$

$P_3 = S_3 \cdot B_{11} = 12 * 6 = 72$

$P_4 = A_{22} \cdot S_4 = 5 * (-2) = -10$

$P_5 = S_5 \cdot S_6 = 8 * 6 = 48$

$P_6 = S_7 \cdot S_8 = (-2) * 6 = -12$

$P_7 = S_9 \cdot S_{10} = (-6 * 14) = -84$

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

Ans

Step(4): $C_{11} = P_5 + P_4 - P_2 + P_6 = 48 - 10 - 8 + (-12)$

$= 18$

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

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

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

$= 66$

4.2-2: Write pseudocode for Strassen's algo.

```
STRASSEN(A,B)
    n = A.rows
    if n == 1
        return a[1,1] * b[1,1]
    let C be a new nxn matrix
    A[1,1] = A[1..n/2][1..n/2]
    A[1,2] = A[1..n/2][n/2+1..n]
    A[2,1] = A[n/2+1..n][1..n/2]
    A[2,2] = A[n/2+1..n][n/2+1..n]
    B[1,1] = B[1..n/2][1..n/2]
    B[1,2] = B[1..n/2][n/2+1..n]
    B[2,1] = B[n/2+1..n][1..n/2]
    B[2,2] = B[n/2+1..n][n/2+1..n]
    S[1] = B[1,2] - B[2,2]
    S[2] = A[1,1] + A[1,2]
    S[3] = A[2,1] + A[2,2]
    S[4] = B[2,1] - B[1,1]
    S[5] = A[1,1] + A[2,2]
    S[6] = B[1,1] + B[2,2]
    S[7] = A[1,2] - A[2,2]
```

S[8] = B[2,1] + B[2,2]

S[9] = A[1,1] − A[2,1]

S[10] = B[1,1] + B[1,2]

P[1] = STRASSEN (A[1,1], S[1])

P[2] = STRASSEN(S[2], B[2,2])

P[3] = STRASSEN (S[3], B[1,1])

P[4] = STRASSEN (A[2,2], S[4])

P[5] = STRASSEN (S[5], S[6])

P[6] = STRASSEN (S[7], S[8])

P[7] = STRASSEN (S[9], S[10])

C[1..n/2][1..n/2] = P[5] + P[4] − P[2] + P[6]

C[1..n/2][n/2+1..n] = P[1] + P[2]

C[n/2+1..n][1..n/2] = P[3] + P[4]

C[n/2+1..n][n/2+1..n] = P[5] + P[1] − P[3] − P[7]

return C.

4.2-3: How would you modify Strassen's algo. to multiply $n \times n$ matrices in which n is not an exact power of 2. Show that the resulting algo. runs in time $\Theta(n^{\lg 7})$.

We can just extend it to an $n \times n$ matrix & pad it with zeros. It's obviously $\Theta(n^{\lg 7})$.

4.2-4: What is the largest k such that if you can multiply 3×3 matrices using k multiplications (not assuming commutativity of multiplication), then you can multiply $n \times n$ matrices in time $o(n^{\lg 7})$? What would the running time of this algorithm be?

If you can multiply 3×3 matrices using k multiplications, then you can multiply $n \times n$ matrices by recursively multiplying $n/3 \times n/3$ matrices, in time

$a = k$, $b = 3$

$$T(n) = k\,T(n/3) + \Theta(n^2).$$

Using master method to solve this recurrence, consider the ratio of $n^{\log_3 k}$ & n^2:

$a \quad b^k$

$\log_b a$

→ If $\log_3 k = 2$, case 2 applies & $T(n) = \Theta(n^2 \lg n)$.

In this case $k = 9$ & $T(n) = \Theta(n^{\lg 7})$.

→ If $\log_3 k < 2$, case 3 applies & $T(n) = \Theta(n^2)$

In this case, $k < 9$ & $T(n) = \Theta(n^{\lg 7})$

→ If $\log_3 k > 2$, case 1 applies & $T(n) = \Theta(n^{\log_3 k})$.

In this case, $k > 9$ & $T(n) = \Theta(n^{\lg 7})$ when

$\log_3 k < \lg 7$ i.e. $k < 3^{\lg 7} \approx 21.85$.

The largest such integer k is 21.

Thus, $k = 21$ & the running times is $\Theta(n^{\log_3 k}) = \Theta(n^{\log_3 21}) = O(n^{2.80})$ (since $\log_3 21 \approx 2.77$).

4.2-5: V. Pan has discovered a way of multiplying 68×68 matrices using 132464 multiplications, a way of multiplying 70×70 matrices using 143640 multiplications, & a way of multiplying 72×72 matrices using 155424 multiplications. Which method yields the best asymptotic running time when used in a divide-and-conquer matrix-multiplication algorithm? How does it compare to Strassen's algorithm?

Using what we know from the last exercise, we need to pick the smallest of the following

$\log_{68} 132464 \approx 2.795128$

$\log_{70} 143640 \approx 2.795122$

$\log_{72} 155424 \approx 2.795147.$

The fastest one asymptotically is 70×70 using 143640.

4.2-6: How quickly can you multiply a $kn \times n$ matrix by $n \times kn$ matrix, using Strassen's algo. as a subroutine? Answer the same question with the order of the i/p matrices reversed.

→ $(kn \times n)(n \times kn)$ produces a $kn \times kn$ matrix.
This produces k^2 multiplications of $n \times n$ matrices

→ $(n \times kn)(kn \times n)$ produces an $n \times n$ matrix.
This produces k multiplications of $k-1$ additions.

4.2-7: Show how to multiply the complex numbers $a+bi$ & $c+di$ using only three multiplications of real numbers. The algorithms should be take a, b, c & d as i/p & produce the real component $ac-bd$ & the imaginary component $ad+bc$ separately.

The three matrices are

$A = (a+b)(c+d) = ac + ad + bc + bd$

$B = ac$

$C = bd$

The result is $(B-C) + (A-B-C)i$.

4.3 : The substitution method for solving recurrences

Exercises:

4.3-1: Show that the solution of $T(n) = T(n-1) + n$ is $O(n^2)$.

We guess $T(n) \le cn^2$ for some constant $c > 0$.

We have

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

The last quality is less than or equal to cn^2 if $c(1-2n)+n \le 0$ or equivalently, $c \ge \frac{n}{(2n-1)}$. This last condition holds $\forall\, n \ge 1$ & $c \ge 1$.

For the boundary condition, we set $T(1) = 1$ & so $T(1) = 1 \le c \cdot 1^2$. Thus we can choose $n_0 = 1$ & $c = 1$.

4.3-2: Show that the solution of $T(n) = T(\lceil n/2 \rceil) + 1$ is $O(\lg n)$.

We guess $T(n) \le c \lg(n-a)$,

$$\begin{aligned} T(n) &\le c \lg(\lceil n/2 \rceil - a) + 1 \\ &\le c \lg((n+1)/2 - a) + 1 \\ &= c \lg((n+1-2a)/2) + 1 \\ &= c \lg(n+1-2a) - c \lg 2 + 1 \qquad (c \ge 1) \end{aligned}$$

4.3-3: We saw that the solution of $T(n) = 2T(\lfloor n/2 \rfloor) + n$ is $O(n \lg n)$. Show that the sol^n of this recurrence is also $\Omega(n \lg n)$. Conclude that the sol^n is $\Theta(n \lg n)$.

First, we guess $T(n) \le cn \lg n$,

$$\begin{aligned} T(n) &\le 2c\lfloor n/2 \rfloor \lg \lfloor n/2 \rfloor + n \\ &\le cn \lg(n/2) + n \\ &= cn \lg n - cn \lg 2 + n \\ &= cn \lg n + (1-c)n \\ &\le cn \lg n. \end{aligned} \qquad (\because c \ge 1)$$

Next, we guess $T(n) \ge c(n+a)\lg(n+a)$,

$$\begin{aligned} T(n) &\ge 2c(\lfloor n/2 \rfloor + a)(\lg \lfloor n/2 \rfloor + a) + n \\ &\ge 2c((n-1)/2 + a)(\lg((n-1)/2) + a) + n \\ &= 2c \cdot \frac{n-1+2a}{2} \lg \frac{n-1+2a}{2} + n \\ &= c(n-1+2a)\lg(n-1+2a) - c(n-1+2a)\lg 2 + n \\ &= c(n-1+2a)\lg(n-1+2a) + (1-c)n - (2a-1)c \end{aligned}$$

$$\left[\because 0 \le c < 1,\ n \ge \frac{(2a-1)c}{1-c}\right]$$

$$\begin{aligned} &\ge c(n-1+2a)\lg(n-1+2a) \qquad (a \ge 1) \\ &\ge c(n+a)\lg(n+a). \end{aligned}$$

4.3-4: Show that by making a different inductive hypothesis, we can overcome the difficulty with the boundary condition $T(1)=1$ for recurrence (4.19) without adjusting the boundary conditions for the inductive proof.

$$T(n) = 2T(\lfloor n/2 \rfloor) + n \qquad (4.19)$$

We guess $T(n) \le n\lg n + n$

$$\begin{aligned} T(n) &\le 2\left(c\lfloor n/2\rfloor \lg\lfloor n/2\rfloor\right) + \lfloor n/2\rfloor) + n \\ &\le 2c\left(n/2 \lg(n/2)\right) + 2(n/2) + n \\ &= cn\lg(n/2) + 2n \\ &= cn\lg n - cn\lg 2 + 2n \\ &= cn\lg n + (2-c)n \\ &\le cn\lg n + n. \qquad [\because c \ge 1] \end{aligned}$$

This time, the boundary condition is

$$T(1) = 1 \le cn\lg n + n = 0 + 1 = 1.$$

4.3-5: Show that $\Theta(n\lg n)$ is the sol". to the "exact" recurrance (4.3 for merge sort.

The recurrance is

$$T(n) = T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + \Theta(n)$$

To show Θ bound, separately show O & Ω bounds

→ For $O(n\lg n)$, we guess $T(n) \le c(n-2)\lg(n-2) - 2c$

$$\begin{aligned} T(n) &\le c\left(\lceil n/2\rceil - 2\right)\lg\left(\lceil n/2\rceil - 2\right) + c\left(\lfloor n/2\rfloor - 2\right)\lg\left(\lfloor n/2\rfloor - 2\right) \\ &\quad + dn \\ &\le c\left(n/2 + 1 - 2\right)\lg\left(n/2 + 1 - 2\right) - 2c + c\left(n/2 - 2\right) \\ &\quad \lg\left(n/2 - 2\right) - 2c + dn \\ &\le c\left(n/2 - 1\right)\lg\left(n/2 - 1\right) + c\left(n/2 - 1\right)\lg\left(n/2 - 1\right) + dn \\ &= c\frac{(n-2)}{2}\lg\frac{(n-2)}{2} + c\frac{n-2}{2}\lg\frac{n-2}{2} - 4c \\ &\quad + dn \\ &= c(n-2)\lg\frac{n-2}{2} - 4c + dn \\ &= c(n-2)\lg(n-2) - c(n-2) - 4c + dn \\ &= c(n-2)\lg(n-2) + (d-c)n + 2c - 4c \\ &\le c(n-2)\lg(n-2) - 2c \qquad [\forall\ c > d]. \end{aligned}$$

→ For $\Omega(n\lg n)$, we guess $T(n) \ge c(n+2)\lg(n+2) + 2c$

$$\begin{aligned} T(n) &\ge c\left(\lceil n/2\rceil + 2\right)\lg\left(\lceil n/2\rceil + 2\right) + c\left(\lfloor n/2\rfloor + 2\right)\lg\left(\lfloor n/2\rfloor + 2\right) \\ &\quad + dn \\ &\ge c\left(n/2 + 2\right)\lg\left(n/2 + 2\right) + 2c + c\left(n/2 - 1 + 2\right)\lg\left(n/2 - 1 + 2\right) \\ &\quad + 2c + dn \\ &\ge c\left(n/2 + 1\right)\lg\left(n/2 + 1\right) + c\left(n/2 + 1\right)\lg\left(n/2 + 1\right) + 4c + dn \\ &= c\frac{(n+2)}{2}\lg\frac{(n+2)}{2} + c\frac{(n+2)}{2}\lg\frac{(n+2)}{2} + 4c + dn \end{aligned}$$

$T(n) = c(n+2)\lg(n+2) - c(n+2) + 4c + dn$

$\leq (n+2)\lg(n+2) + (d-c)\cdot n - 2c + 4c$

$\geq c(n+2)\lg(n+2) + 2c.$ $[\forall\ d > c]$

4.3-6: Show that the sol'n to $T(n) = 2T(\lfloor n/2 \rfloor + 17) + n$ is $O(n \lg n)$.

We guess $T(n) \leq c(n-a)\lg(n-a)$,

$T(n) \leq 2c(\lfloor n/2 \rfloor + 17 - a)\lg(\lfloor n/2 \rfloor + 17 - a) + n$

$\leq 2c(n/2 + 17 - a)\lg(n/2 + 17 - a) + n$

$= c(n + 34 - 2a)\lg\frac{n+34-2a}{2} + n$

$= c(n+34-2a)\lg(n+34-2a) - c(n+34-2a) + n$ $[\because c > 1,\ n > na = f(a)]$

$\leq c(n+34-2a)\lg(n+34-2a)$ $[a \geq 34]$

$\leq c(n-a)\lg(n-a).$

4.3-7: Using the master method in section 4.5, you can show that the solution to the recurrence $T(n) = 4T(n/3) + n$ is $T(n) = \Theta(n^{\log_3 4})$. Show that a substitution proof with the assumption $T(n) \leq cn^{\log_3 4}$ fails. Then show how to subtract off a lower-order term to make the substitution proof work.

We guess $T(n) \leq cn^{\log_3 4}$ first,

$T(n) \leq 4c(n/3)^{\log_3 4} + n$

$= cn^{\log_3 4} + n.$

We stuck there.

We guess $T(n) \leq cn^{\log_3 4} - dn$ again,

$T(n) \leq 4(c(n/3)^{\log_3 4} - d(n/3)) + n$

$= 4(cn^{\log_3 4}/4 - dn/3) + n$

$= cn^{\log_3 4} - \frac{4}{3}dn + n$

$\leq cn^{\log_3 4} - dn.$ $[\because d \geq 3]$

4.3-8: Using the master method in section 4.5, you can show that the solution to the recurrence $T(n) = 4T(n/2) + n$ is $T(n) = \Theta(n^2)$. Show that a substitution proof with the assumption $T(n) \leq cn^2$ fails. Then show how to subtract off a lower-order term to make the substitution proof work.

First, let's try the guess $T(n) \leq cn^2$, Then we get

$T(n) = 4T(n/2) + n$

$\leq 4c(n/2)^2 + n$

$= cn^2 + n$

we cannot proceed any further from the inequality above to conclude $T(n) \le cn^2$.

Alternatively, let us try the guess

$$T(n) \le cn^2 - cn,$$

which subtracts off a lower-order term. Now we have

$$\begin{aligned} T(n) &= 4T(n/2) + n \\ &\le 4(c(n/2)^2 - c(n/2)) + n \\ &= 4c(n/2)^2 - 4c(n/2) + n \\ &= cn^2 + (1-2c)n \\ &\le n^2, \end{aligned} \qquad [\because c \ge \tfrac{1}{2}]$$

4.3-9: Solve the recurrence $T(n) = 3T(\sqrt{n}) + \lg n$ by making a change of variables. Your solution should be asymptotically tight. Do not worry about whether values are integral.

First, $T(n) = 3T(\sqrt{n}) + \lg n$ let $m = \lg n$

$$T(2^m) = 3T(2^{m/2}) + m$$

$$S(m) = 3S(m/2) + m.$$

Now, we guess $S(m) \le cm^{\lg 3} + dm$,

$$\begin{aligned} S(m) &\le 3(c(m/2)^{\lg 3} + d(m/2)) + m \\ &\le cm^{\lg 3} + \left(\tfrac{3}{2}d + 1\right)m \qquad (d \le -2) \\ &\le cm^{\lg 3} + dm \end{aligned}$$

The we guess, $S(m) \ge cm^{\lg 3} + dm$,

$$\begin{aligned} S(m) &\ge 3(c(m/2)^{\lg 3} + d(m/2)) + m \\ &\ge cm^{\lg 3} + \left(\tfrac{3}{2}d + 1\right)m \qquad (d \ge -2) \\ &\ge cm^{\lg 3} + dm \end{aligned}$$

Thus, $S(m) = \Theta(m^{\lg 3})$

$$= \Theta(\lg^{\lg 3} n)$$

4.4: The recursion-tree Method for solving recurrences

4.4-1: Use a recursion tree to determine a good asymptotic upper bound on the recurrence $T(n) = 3T(\lfloor n/2 \rfloor) + n$. Use the substitution method to verify your answer.

The subproblem size for a node at depth i is $n/2^i$.

Thus, the tree has $\lg n + 1$ levels & $3^{\lg n} = n^{\lg 3}$ leaves.

The total cost over all nodes at depth i, for $i = 0, 1, 2, \ldots, \lg n - 1$ is $3^i(n/2^i) = (3/2)^i n$.

$$\begin{aligned} T(n) &= n + \tfrac{3}{2}n + \left(\tfrac{3}{2}\right)^2 n + \cdots + \left(\tfrac{3}{2}\right)^{\lg n - 1} n + \Theta(n^{\lg 3}) \\ &= \sum_{i=0}^{\lg n - 1} \left(\tfrac{3}{2}\right)^i n + \Theta(n^{\lg 3}). \end{aligned}$$

$$T(n) = \frac{(3/2)^{\lg n} - 1}{(3/2) - 1} n + \Theta(n^{\lg 3})$$

$$= 2\left[(3/2)^{\lg n} - 1\right] n + \Theta(n^{\lg 3})$$

$$= 2\left[n^{\lg(3/2)} - 1\right] n + \Theta(n^{\lg 3})$$

$$= 2\left[n^{\lg 3 - \lg 2} - 1\right] n + \Theta(n^{\lg 3})$$

$$= 2\left[n^{\lg 3 - 1 + 1} - n\right] + \Theta(n^{\lg 3})$$

$$= O(n^{\lg 3}).$$

→ We guess $T(n) \le cn^{\lg 3} - dn$,

$$T(n) = 3T(\lfloor n/2 \rfloor) + n$$

$$\le 3 \cdot (c(n/2)^{\lg 3} - d(n/2)) + n$$

$$= (3/2^{\lg 3})\, cn^{\lg 3} - (3d/2)n + n$$

$$= cn^{\lg 3} + (1 - 3d/2)n$$

$$\le cn^{\lg 3} - dn \qquad [d \ge 2]$$

4.4-2: Use a recursion tree to determine a good asymptotic upper bound on the recurrence $T(n) = T(n/2) + n^2$. Use the substitution method to verify your answer.

→ The subproblem size for a node at depth i is $n/2^i$.

Thus, the tree has $\lg n + 1$ levels & $1^{\lg n} = 1$ leaf.

The total cost over all nodes at depth i, for $i = 0, 1, 2, \ldots, \lg n - 1$, is $1^i (n/2^i)^2 = (1/4)^i n^2$.

$$T(n) = \sum_{i=0}^{\lg n - 1} \left(\frac{1}{4}\right)^i n^2 + \Theta(1)$$

$$< \sum_{i=0}^{\infty} \left(\frac{1}{4}\right)^i n^2 + \Theta(1)$$

$$= \frac{1}{1 - (1/4)} n^2 + \Theta(1)$$

$$= \Theta(n^2).$$

→ We guess $T(n) \le cn^2$,

$$T(n) \le c(n/2)^2 + n^2$$

$$= cn^2/4 + n^2$$

$$= \left(\frac{c}{4} + 1\right) n^2$$

$$\le cn^2, \qquad \left[c \ge 4/3\right].$$

4.4-3: Use a recursion tree to determine a good asymptotic upper bound on the recurrence $T(n) = 4T(n/2 + 2) + n$. Use the substitution method to verify your answer.

→ The subproblem size for a node at depth i is $n/2^i$.

Thus, the tree has $\lg n + 1$ levels & $4^{\lg n} = n^2$ leaves.

The total cost over all nodes at depth i, for $i = 0, 1, 2, \ldots, \lg n - 1$, is $4^i (n/2^i + 2) = 2^i n + 2 \cdot 4^i$.

$$T(n) = \sum_{i=0}^{\lg n-1} (2^i n + 2\cdot 4^i) + \Theta(n^2)$$

$$= \sum_{i=0}^{\lg n-1} 2^i n + \sum_{i=0}^{\lg n-1} 2\cdot 4^i + \Theta(n^2)$$

$$= \frac{2^{\lg n}-1}{2-1} n + 2\cdot\frac{4^{\lg n}-1}{4-1} + \Theta(n^2)$$

$$= (2^{\lg n}-1)\, n + \frac{2}{3}(4^{\lg n}-1) + \Theta(n^2)$$

$$= (n-1)n + \frac{2}{3}(n^2-1) + \Theta(n^2)$$

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

→ We guess $T(n) \le c(n^2 - dn)$,

$$T(n) = 4T(n/2+2) + n$$

$$\le 4c[(n/2+2)^2 - d(n/2+2)] + n$$

$$\le 4c(n^2/4 + 2n + 4 - dn/2 - 2d) + n$$

$$\le cn^2 + 8cn + 16c - 2cdn - 8cd + n$$

$$= cn^2 - cdn + 8cn + 16c - cdn - 8cd + n$$

$$= c(n^2 - dn) - (cd - 8c - 1)n - (d-2)\cdot 8c$$

$$\le c(n^2 - dn), \quad [\because cd - 8c - 1 \ge 0]$$

4.4-4: Use a recursion tree to determine a good asymptotic upper bound on the recurrence $T(n) = 2T(n-1) + 1$. Use the substitution method to verify your answer.

→ The subproblem size for a node at depth i is $n-i$, Thus, the tree has $n+1$ levels ($i = 0, 1, 2, \ldots, n$) of 2^n leaves.

The total cost over all nodes at depth i for $i = 0, 1, \ldots, n-1$ is 2^i.

The n^{th} level has 2^n leaves each with cost $\Theta(1)$, so that total cost of n^{th} level is $\Theta(2^n)$.

Adding the costs of all levels of the recursion tree we get the following:

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

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

$$= 2^n - 1 + \Theta(2^n)$$

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

→ We guess $T(n) \le c2^n - d$,

$$T(n) \le 2(c2^{n-1} - d) + 1$$

$$= c2^n - 2d + 1$$

$$\le c2^n - d \quad [\because d \ge 1]$$

Thus $T(n) = O(2^n)$

4.4-5: Use a recursion tree to determine a good asymptotic bound on the recurrence $T(n) = T(n-1) + T(n/2) + n$. Use the substitution method to verify your answer.

The trees makes it look like it is exponential in the worst case. The tree is not full (not a complete binary tree of height n), but it is not polynomial either. This easy to show $O(2^n)$ & $\Omega(2^n)$.

To justify that this is a pretty tight upper bound, we'll show that we can't have any other choice, If we have that $T(n) \le cn^k$, when we substitute into the recurrence, the new coefficient for n^k can be a high as $c\left(1 + \frac{1}{2^k}\right)$ which is bigger than c regardless of how we choose the value c.

→ We guess $T(n) \le c2^n - 4n$,

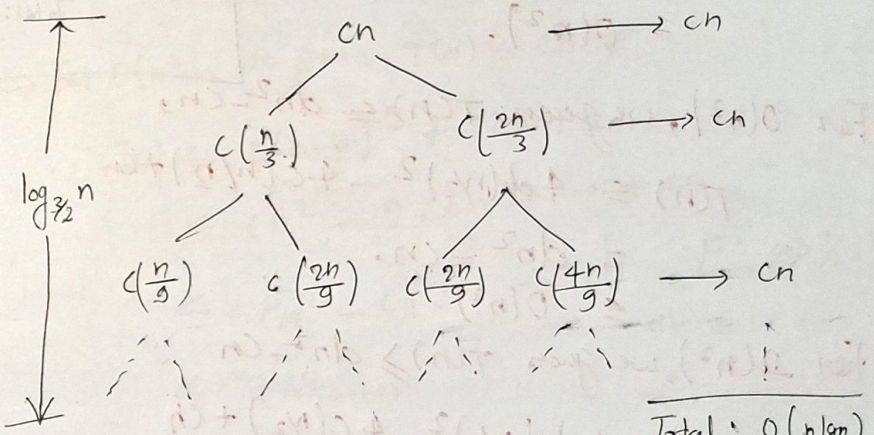
$$\begin{aligned} T(n) &\le c2^{n-1} - 4(n-1) + c2^{n/2} - 4n/2 + n \\ &= c(2^{n-1} + 2^{n/2}) - 3n + 4 \qquad (n \ge 1/4) \\ &\le c(2^{n-1} + 2^{n/2}) - 4n \qquad (n \ge 2) \\ &= c(2^{n-1} + 2^{n-1}) - 4n \\ &\le c2^n - 4n \\ &= O(2^n). \end{aligned}$$

→ We guess $T(n) \ge cn^2$,

$$\begin{aligned} T(n) &\ge c(n-1)^2 + c(n/2)^2 + n \\ &= cn^2 - 2cn + c + cn^2/4 + n \\ &= \left(\tfrac{5}{4}\right)cn^2 + (1-2c)n + c \\ &\ge cn^2 + (1-2c)n + c \qquad [\because c \le 1/2] \\ &= \Omega(n^2). \end{aligned}$$

4.4-6: Argue that the solution to the recurrence $T(n) = T(n/3) + T(2n/3) + cn$, where c is a constant, is $\Omega(n \lg n)$ by appealing to the recursion tree.

We know that the cost of each level of the tree is cn by examining the tree in fig. 4.6. To find a lower bound on the cost of the algo., we need a lower bound on the height of the tree.



The shortest simple path from root to leaf is found by following the leftest child at each node. Since we divide by 3 at each step, we see that this path has length $\log_3 n$. Therefore, the cost of the algo is:

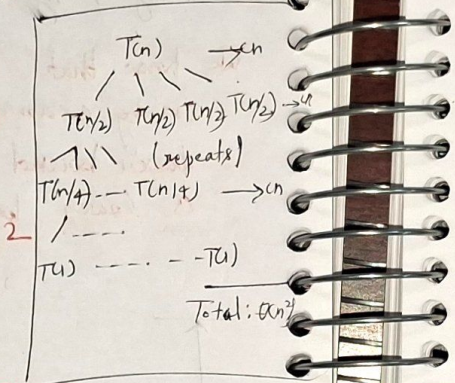
$$cn(\log_3 n + 1) \ge cn\log_3 n = \frac{c}{\lg 3} n \lg n = \Omega(n \lg n).$$

4.4-7: Draw the recursion tree for $T(n) = 4T(\lfloor n/2 \rfloor) + cn$, where c is a constant, & provide a tight asymptotic bound on its solution. Verify your answer with the substitution method.

→ The subproblem size for a node at depth i is $n/2^i$.

Thus, the tree has $\lg n + 1$ levels & $4^{\lg n} = n^{\lg 4} = n^2$ leaves.

The total cost overall nodes at depth i, for $i = 0, 1, \ldots, \lg n - 1$ is $4^i (cn/2^i) = 2^i cn$.



$$T(n) = \sum_{i=0}^{\lg n - 1} 2^i cn + \Theta(n^2)$$

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

$$= \Theta(n^2).$$

→ For $O(n^2)$, we guess $T(n) \le dn^2 - cn$,

$$T(n) \le 4d(n/2)^2 - 4c(n/2) + cn$$

$$= dn^2 - cn.$$

$$\le O(n^2)$$

→ For $\Omega(n^2)$, we guess $T(n) \ge dn^2 - cn$

$$T(n) \ge 4d(n/2)^2 - 4c(n/2) + cn$$

$$= dn^2 - cn$$

$$\ge dn^2$$

$$\ge \Omega(n^2)$$

4.4-8: Use a recursion tree to give an asymptotic tight solution to the recurrence $T(n) = T(n-a) + T(a) + cn$, where $a \ge 1$ & $c > 0$ are constants.

→ The tree has $n/a + 1$ levels

The total cost overall nodes at depth i, for $i = 0, 1, 2, \ldots, n/a - 1$, is : $c(n - ia)$.

$$T(n) = \sum_{i=0}^{n/a} c(n - ia) + (n/a)ca$$

$$= \sum_{i=0}^{n/a} cn - \sum_{i=0}^{n/a} cia + (n/a)ca$$

$$= cn^2/a - \Theta(n) + \Theta(n)$$

$$= \Theta(n^2).$$

→ For $O(n^2)$, we guess $T(n) \le cn^2$,

$$T(n) \le c(n-a)^2 + ca + cn$$

$$\le cn^2 - 2can + ca + cn$$

$$\le cn^2 - c(2an - a - n) \quad [\because a \ge 1/2, n \ge 2a]$$

$$\le cn^2 - cn$$

$$\le cn^2$$

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

$$= O(n^2).$$

→ For $\Omega(n^2)$, we guess $T(n) \geq cn^2$,

$$T(n) \geq c(n-a)^2 + ca + cn$$
$$\geq cn^2 - 2acn + ca + cn$$
$$\geq cn^2 - c(2an - a - n) \quad [a < 1/2, n > 2a]$$
$$\geq cn^2 + cn$$
$$\geq cn^2$$

$$\boxed{T(n) = \Omega(n^2)}$$

4.4-9: We can use a recursion tree to give an asymptotically tight soln to the recurrence $T(n) = T(\alpha n) + T((1-\alpha)n) + cn$, where α is a constant in the range $0 < \alpha < 1$ & $c > 0$ is also a constant.

We can assume that $0 < \alpha \leq 1/2$, since otherwise we can let $\beta = 1 - \alpha$ & solve it for β.

Thus, the depth of the tree is $\log_{1/\alpha} n$ & each level cost cn. And let's guess that the leaves are $\Theta(n)$,

$$T(n) = \sum_{i=0}^{\log_{1/\alpha} n} cn + \Theta(n)$$
$$= cn \log_{1/\alpha} n + \Theta(n)$$
$$= \Theta(n \lg n).$$

We can also show $T(n) = \Theta(n \lg n)$ by substitution.

To prove the upper bound, we guess $T(n) \leq dn \lg n$ for a constant $d > 0$

$$T(n) = T(\alpha n) + T((1-\alpha)n) + cn$$
$$\leq d\alpha n \lg(\alpha n) + d(1-\alpha)n \lg((1-\alpha)n) + cn$$
$$= d\alpha n \lg \alpha + d\alpha n \lg n + d(1-\alpha)n \lg(1-\alpha) + d(1-\alpha)n \lg n + cn$$
$$= dn \lg n + dn(\alpha \lg \alpha + (1-\alpha)\lg(1-\alpha)) + cn$$
$$\leq dn \lg n$$

$$\left[\because d \geq \frac{-c}{\alpha \lg \alpha + (1-\alpha)\lg(1-\alpha)}\right]$$

We can achieve this result by solving the inequality

$$dn \lg n + dn(\alpha \lg \alpha + (1-\alpha)\lg(1-\alpha)) + cn \leq dn \lg n$$
$$\Rightarrow dn(\alpha \lg \alpha + (1-\alpha)\lg(1-\alpha)) + cn \leq 0$$
$$\Rightarrow d(\alpha \lg \alpha + (1-\alpha)\lg(1-\alpha)) \leq -c$$
$$\Rightarrow d \geq \frac{-c}{\alpha \lg \alpha + (1-\alpha)\lg(1-\alpha)},$$

To prove the lower bound, we guess that $T(n) \geq dn \lg n$ for constant $d > 0$,

$$T(n) = T(\alpha n) + T((1-\alpha)n) + cn$$
$$\geq d\alpha n \lg(\alpha n) + d(1-\alpha)n \lg((1-\alpha)n) + cn$$

$T(n) = d\alpha n \lg \alpha + d\alpha n \lg n + d(1-\alpha) n \lg (1-\alpha) + d(1-\alpha) n \lg n + cn$

$= dn \lg n + cn (\alpha \lg \alpha + (1-\alpha) \lg (1-\alpha)) + cn$

$\geq dn \lg n,$ i.e. $0 < d \leq \frac{-c}{\alpha \lg \alpha + (1-\alpha) \lg (1-\alpha)}$.

We can achieve this result by solving the inequality

$dn \lg n + dn(\alpha \lg \alpha + (1-\alpha) \lg (1-\alpha)) + cn \geq dn \lg n$

$\Rightarrow dn(\alpha \lg \alpha + (1-\alpha) \lg (1-\alpha)) + cn \geq 0$

$\Rightarrow d(\alpha \lg \alpha + (1-\alpha) \lg (1-\alpha)) \geq -c$

$\Rightarrow 0 < d \leq \frac{-c}{\alpha \lg \alpha + (1-\alpha) \lg (1-\alpha)}$

Therefore, $T(n) = \Theta(n \lg n)$.

4.5: The master method for solving recurrences

Theorem 4.1: (Master theorem)

Let $a \geq 1$ & $b > 1$ be constants, Let $f(n)$ be a function, & let $T(n)$ be defined on the nonnegative integers by the recurrence

$$T(n) = a\,T(n/b) + f(n).$$

Where we interpret n/b to mean either $\lfloor n/b \rfloor$ or $\lceil n/b \rceil$. Then $T(n)$ has the following asymptotic bounds:

1. If $f(n) = O(n^{\log_b a - \epsilon})$ for some $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$.
2. If $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} \log n)$.
3. If $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some constant $\epsilon > 0$ & if $af(n/b) \leq cf(n)$ for some constant $c < 1$ & all sufficiently large n, then $T(n) = \Theta(f(n))$.

Excercises:

4.5-1: Use the master method to give tight asymptotic bounds for the following recurrences:

(a) $T(n) = 2T(n/4) + 1$; $a = 2, b = 4$ then $f(n) \leq n^{\log_b a} = n^{\log_4 2}$

Sol^{n}.

$T(n) = a\,T(n/b) + \Theta(n^k \log^p n)$

$a = 2, b = 4, k = 0, p = 0$

$a > b^k$; $2 > 4^0$; $2 > 1$ } then using case 1: $T(n) = \Theta(n^{\log_b a})$

$\Rightarrow T(n) = \Theta(n^{\log_4 2})$

$\Rightarrow T(n) = \Theta(n^{1/2})$

$T(n) = \Theta(\sqrt{n})$.

(b) $T(n) = 2T(n/4) + \sqrt{n}$

$a = 2, b = 4, k = 1/2, p = 0$

$a = b^k$ and $2 = 4^{1/2}$ } Case 2: $T(n) = \Theta(n^{\log_b a} \log^{p+1} n)$

$\Rightarrow T(n) = \Theta(n^{\log_4 2} \log n) = \Theta(\sqrt{n} \log n)$ Ans.

(c) $T(n) = 2T(n/4) + n$

$a = 2, b = 4, k = 1, p = 0$

$a < b^k$ and $2 < 4$ } Case 3: $T(n) = \Theta(n^k \log^p n)$

$T(n) = \Theta(n \log^0 n) = \Theta(n)$ Ans.

(d) $T(n) = 2T(n/4) + n^2$

Case 3: $\boxed{T(n) = \Theta(n^2).}$ Ans.

4.5-2: Professor Caesar wishes to develop a matrix-multiplication that is asymptotically faster than Strassen's algorithm. His algorithm will use the divide-and-conquer method, dividing each matrix into pieces of size $n/4 \times n/4$, & the divide & combine steps together ~~create in order to beat~~ will take $\Theta(n^2)$ time. He needs to determine how many subproblems his algorithm has to create in order to beat Strassen's algo. If his algo creates a subproblems, then the recurrence for the running time $T(n)$ becomes $T(n) = aT(n/4) + \Theta(n^2)$. What is the largest integer values of a for which professor Caesar's algorithm would be asymptotically faster than strassen's algorithm?

Strassen's algorithm has running time of $\Theta(n^{\lg 7})$.

The largest integer a such that $\log_4 a < \lg 7$ is

$\boxed{a = 48}$.

$\Rightarrow \frac{\log_2 a}{\log_2 4} < \frac{\log_2 7}{\log_2 2^2} \Rightarrow \log_2 a < \lg 7 \times 2$

$\Rightarrow \boxed{a \leq 48}$

4.5-3: Use the master method to show that the solution to the binary-search recurrence $T(n) = T(n/2) + \Theta(1)$ is $T(n) = \Theta(\lg n)$.

$a = 1, b = 2, k = 0, p = 0$

$a \neq b^k$
$1 \neq 2^0$
$1 = 1$ } Case 2: $T(n) = \Theta(n^k \log^{p+1} n)$

$T(n) = \Theta(n^0 \log^{0+1} n)$

$\boxed{T(n) = \Theta(\log n).}$ Ans.

4.5-4: Can the master method be applied to the recurrence $T(n) = 4T(n/2) + n^2 \lg n$? why or why not? Give an asymptotically upper bound for this recurrence.

$a = 4, b = 2$, we have $f(n) = n^2 \lg n \neq \Omega(n^{2-\epsilon}) \neq \Omega(n^{2+\epsilon})$, so we cannot apply the master method.

but use of $\boxed{T(n) = aT(n/b) + \Theta(n^k \log^p n)}$ basis,

$a = b^k$
$4 = 2^2$ } Case 2: $T(n) = \Theta(n^k \log^{p+1} n)$

$\boxed{T(n) = \Theta(n^2 \log^2 n)}$ Ans.

but for the asymptotic upper bound,
We guess $T(n) \le cn^2 \lg^2 n$, substituting
$T(n/2) \le c(n/2)^2 \lg^2(n/2)$ into recurrence yields

$$\begin{aligned} T(n) &= 4T(n/2) + n^2 \lg n \\ &\le 4c(n/2)^2 \lg^2(n/2) + n^2 \lg n \\ &= cn^2 \lg(n/2) \lg n - cn^2 \lg(n/2) \lg 2 + n^2 \lg n \\ &= cn^2 \lg^2 n - cn^2 \lg n \lg 2 - cn^2 \lg(n/2) \lg 2 + n^2 \lg n \\ &= cn^2 \lg^2 n + (1 - c \lg 2) n^2 \lg n - cn^2 \lg(n/2) \lg 2 \\ &\le cn^2 \lg^2 n - cn^2 \lg(n/2) \lg 2 \qquad [\because c \ge 1/\lg 2] \\ &\le cn^2 \lg^2 n. \end{aligned}$$

(Exercise 4.6-2 is the general case of this)

*4.5-5: Consider the regularity condition $af(n/b) \le cf(n)$ for some constant $c < 1$, which is part of case 3 of the master theorem. Give an example of constants $a \ge 1$ & $b > 1$ & a func $f(n)$ that satisfies all the conditions in case 3 of the master theorem, except the regularity condition.

$a = 1$, $b = 2$ & $f(n) = n(2 - \cos n)$.

If we try to prove it,

$$\frac{n}{2}(2 - \cos\tfrac{n}{2}) < cn$$

$$\frac{1 - \cos(n/2)}{2} < c$$

$$\Rightarrow \quad 1 - \frac{\cos(n/2)}{2} \le c$$

Since, $\min \cos(n/2) = -1$, this implies that $c \ge 3/2$
But $c < 1$.

4.6: Proof of the master theorem

*4.6-1: Give a simple & exact expression for n_j in eqn. (4.27) for the case in which b is a positive integer instead of an arbitrary real number.

We state that $\forall j \ge 0$, $n_j = \lceil \frac{n}{b^j} \rceil$.

Indeed, for $j = 0$ we have from the recurrence's base case that $n_0 = n = \lceil \frac{n}{b^0} \rceil$.

Now, suppose $n_{j-1} = \lceil \frac{n}{b^{j-1}} \rceil$ for some $j > 0$. By definition,

$$n_j = \left\lceil \frac{n_{j-1}}{b} \right\rceil.$$

It follows from the induction hypothesis that

$$n_j = \left\lceil \frac{\lceil \frac{n}{b^{j-1}} \rceil}{b} \right\rceil.$$

Since b is a positive integer, eqn (3.4) implies that

$$\left\lceil \frac{\lceil \frac{n}{b^{j-1}} \rceil}{b} \right\rceil = \left\lceil \frac{n}{b^j} \right\rceil \Rightarrow \quad n_j = \left\lceil \frac{n}{b^j} \right\rceil.$$

P.S. n_j is obtained by shifting the base b representation of n, j positions to the right & adding 1 if any of the j least significant positions are non-zero.

*4.6-2: Show that if $f(n) = \Theta(n^{\log_b a} \lg^k n)$, where $k \geq 0$ then the master recurrence has solution $T(n) = \Theta(n^{\log_b a} \lg^{k+1} n)$. For simplicity, confine your analysis to exact powers of b.

$$g(n) = \sum_{j=0}^{\log_b n - 1} a^j f(n/b^j)$$

$$f(n/b^j) = \Theta\left((n/b^j)^{\log_b a} \lg^k (n/b^j)\right)$$

$$g(n) = \Theta\left(\sum_{j=0}^{\log_b n - 1} a^j \left(\frac{n}{b^j}\right)^{\log_b a} \lg^k \left(\frac{n}{b^j}\right)\right)$$

$$= \Theta(A)$$

$$A = \sum_{j=0}^{\log_b n - 1} a^j \left(\frac{n}{b^j}\right)^{\log_b a} \lg^k \frac{n}{b^j}$$

$$= n^{\log_b a} \sum_{j=1}^{\log_b n - 1} \left(\frac{a}{b^{\log_b a}}\right)^j \lg^k \frac{n}{b^j}$$

$$= n^{\log_b a} \cdot \sum_{j=0}^{\log_b n - 1} \lg^k \frac{n}{b^j}$$

$$= n^{\log_b a} \cdot B.$$

$$\lg^k \frac{n}{d} = (\lg n - \lg d)^k = \lg^k n - \Theta(\lg^k n)$$

$$B = \sum_{j=0}^{\log_b n - 1} \lg^k \frac{n}{b^j} = \sum_{j=0}^{\log_b n - 1} \left(\lg^k n - \Theta(\lg^k n)\right)$$

$$= \log_b n \lg^k n + \log_b n \cdot \Theta(\lg^k n)$$

$$= \Theta(\log_b n \lg^k n)$$

$$= \Theta(\lg^{k+1} n)$$

$$g(n) = \Theta(A) = \Theta(n^{\log_b a} B)$$

$$\Rightarrow \boxed{g(n) = \Theta(n^{\log_b a} \lg^{k+1} n)}$$ Hence proved.

*4.6-3 Show that case 3 of the master method is overstated, in the sense that the regularity condition $af(n/b) \leq cf(n)$ for some constant $c < 1$ implies that there exists a constant $\epsilon > 0$ such that $f(n) = \Omega(n^{\log_b a + \epsilon})$.

$$af(n/b) \leq cf(n)$$

$$\Rightarrow f(n/b) \leq \frac{c}{a} f(n)$$

$$\Rightarrow f(n) \leq \frac{c}{a} f(bn)$$

$$= \frac{c}{a}\left(\frac{c}{a} f(b^2 a)\right)$$

$$= \frac{c}{a}\left(\frac{c}{a}\left(\frac{c}{a} f(b^3 n)\right)\right)$$

$$= \left(\frac{c}{a}\right)^i f(b^i n)$$

$\Rightarrow f(b^i n) \geq \left(\frac{a}{c}\right)^i f(n).$

Let $n=1$, then we have

$$f(b^i) \geq \left(\frac{a}{c}\right)^i f(1) \rightarrow (*).$$

Let $b^i = n \Rightarrow i = \log_b n$, then substitute back to eqn $(*)$,

$$f(n) \geq \left(\frac{a}{c}\right)^{\log_b n} f(1)$$

$$\geq n^{\log_b \frac{a}{c}} f(1)$$

$$\geq n^{\log_b a + \epsilon} \quad [\because \epsilon > 0 \text{ because } \tfrac{a}{c} > a \text{ i.e. } c < 1]$$

$$\boxed{f(n) = \Omega(n^{\log_b a + \epsilon})}$$

Hence proved.

4 : Problems

4-1: Recurrence examples

Give asymptotic upper & lower bound for $T(n)$ in each of the following recurrence. Assume that $T(n)$ is constant for $n \leq 2$. Make your bounds as tight as possible & justify your answers.

ⓐ $T(n) = 2T(n/2) + n^4$

By master theorem, $a=2$, $b=2$, $k=4$, $p=0$

$$T(n) = a\,T(n/b) + \Theta(n^k \log^p n)$$

$a < b^k$, $2 < 2^4$ } Case 3: $T(n) = \Theta(n^k \log^p n)$

$\boxed{T(n) = \Theta(n^4)}$ Ans.

ⓑ $T(n) = T(7n/10) + n$

$a=1$, $b=10/7$, $k=1$ & $p=0$

$a < b^k$ } Case 3: $\boxed{T(n) = \Theta(n)}$ Ans.

ⓒ $T(n) = \underset{a}{16}\,T(n/\underset{b}{4}) + n^{2k}$

$a = b^k$, $16 = 4^2$ } case 2: $T(n) = \Theta(n^{\log_b a} \log^{p+1} n)$

$T(n) = \Theta(n^{\log_4 16} \log^{0+1} n)$

$\boxed{T(n) = \Theta(n^2 \log n)}$ Ans.

ⓓ $T(n) = \underset{a}{7}\,T(n/\underset{b}{2}) + n^{2k}$

$a < b^k$, $7 < 3^2$ } Case 3: $T(n) = \Theta(n^k \log^p n)$

$\boxed{T(n) = \Theta(n^2)}$ Ans.

(e) $T(n) = \underset{a}{7}\, T(n/\underset{b}{2}) + n^2$

$a > b^k$, $7 > 2^2$ } Case 1: $T(n) = \Theta(n^{\log_b a})$

$\boxed{T(n) = \Theta(n^{\log_2 7})}$ Ans.

(f) $T(n) = \underset{a}{2}\, T(n/\underset{b}{4}) + \sqrt{n}^{k=1/2}$

$a = b^k$, $2 = 4^{1/2}$ } Case 2: $\boxed{T(n) = \Theta(\sqrt{n}\log n)}$ Ans.

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

Let $d = n \bmod 2$

$$T(n) = \sum_{j=1}^{j=n/2} (2j+d)^2$$

$$= \sum_{j=1}^{n/2} 4j^2 + 4jd + d^2$$

$$= \frac{n(n+2)(n+1)}{6} + \frac{n(n+2)d}{2} + \frac{d^2 n}{2}$$

$\boxed{T(n) = \Theta(n^3)}$ Ans.

4-2: Parameter-passing costs

Soln: (a) (1) $T(n) = T(n/2) + c = \Theta(\lg n)$ [Master theorem].

(2) $\Theta(n \lg n)$.

$$T(n) = T(n/2) + cN$$
$$= 2cN + T(n/4)$$
$$= 3cN + T(n/8)$$
$$= \sum_{i=0}^{\lg n - 1} (2^i cN/2^i)$$
$$= cN \lg n$$
$$= \Theta(n \lg n).$$

(3) $T(n) = T(n/2) + cn = \Theta(n)$ [Master theorem].

Soln: (b) (1) $T(n) = 2T(n/2) + cn = \Theta(n \lg n)$ [Master Method]

(2) $\Theta(n^2)$ $T(n) = 2T(n/2) + cn + 2N$

$$= 4N + cn + 2c(n/2) + 4T(n/4)$$
$$= 8N + 2cn + 4c(n/4) + 8T(n/8)$$
$$= \sum_{i=0}^{\lg n - 1} (cn + 2^i N)$$
$$= \sum_{i=0}^{\lg n - 1} cn + N \sum_{i=0}^{\lg n - 1} 2^i$$
$$= cn \lg n + N \frac{2^{\lg n} - 1}{2 - 1}$$
$$= cn \lg n + 2N - N = \Theta(nN)$$
$$= \Theta(n^2)$$

③ $\Theta(n \lg n \lg n)$

$$T(n) = 2T(n/2) + cn + 2n/2$$
$$= 2T(n/2) + (c+1)n$$
$$= \Theta(n \lg n).$$

4-3: More recurrence examples

(a) $T(n) = 4T(n/3) + n \lg n$

By master theorem,

$$T(n) = a\,T(n/b) + \Theta(n^k \lg^p n)$$
$$a = 4,\ b = 3$$

$a > b^k$, $4 > 3^1$ } Case 1: $T(n) = \Theta(n^{\log_b a})$

$$\boxed{T(n) = \Theta(n^{\log_3 4})}$$ Ans.

(b) $T(n) = 3T(n/3) + n/\lg n$

$a = 3, b = 3, k = 1, p = -1$

$a = b^k$ } Case 2: If $p = -1$; $T(n) = \Theta(n^{\log_b a} \log_b \log_b n)$

$$T(n) = \Theta(n^{\log_3 3} \log_3 \log_3 n)$$
$$\boxed{T(n) = \Theta(n \log_3 \log_3 n)}$$ Ans.

{OR}

By the recurrence-tree method, we guess $T(n) = \Theta(n \log_3 \log_3 n)$.

We starting by proving upper bound.

Suppose $k < n \Rightarrow T(k) \le ck \log_3 \log_3 k - k$, where we subtract a lower bound term to strengthen our induction hypothesis.

It follows that

$$T(n) \le 3\left(c\frac{n}{3}\log_3 \log_3 \frac{n}{3} - \frac{n}{3}\right) + \frac{n}{\lg n}$$
$$\le cn \log_3 \log_3 n - n + \frac{n}{\lg n}$$
$$\le cn \log_3 \log_3 n,$$

if n is sufficiently large.

The lower bound can proved analogously.

(c) $T(n) = 4T(n/2) + n^2\sqrt{n}$

$a = 4, b = 2, k = 5/2$ & $p = 0$

$a < b^k$, $4 < 2^{5/2}$ } case 3:

$$T(n) = \Theta(n^{5/2})$$
$$T(n) = \Theta(n^{2.5})$$
$$T(n) = \Theta(n^2\sqrt{n})$$ Ans.

(d) $T(n) = 3T(n/3 - 2) + n/2$

By master theorem, $T(n) \approx 3T(n/3) + n/2$

$a = 3, b = 3, k = 1, p = 0$

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

$$T(n) = \Theta(n^{\log_3 3} \log^{(1)} n)$$
$$\boxed{T(n) = \Theta(n \lg n)}$$ Ans. {OR}

At level k:

→ There are 3^k calls to $T\left(\frac{n}{3^k} - 2k\right)$.

→ Cost for each call is $\frac{1}{2}\left(\frac{n}{3^k} - 2k\right)$

→ Total cost at this level:

$$3^k \cdot \frac{1}{2}\left(\frac{n}{3^k} - 2k\right) = \frac{n}{2} - k \cdot 3^k$$

i.e $$T(n) = \sum_{k=0}^{\log_3(n)} \left(\frac{n}{2} - k \cdot 3^k\right)$$

$$= \sum_{k=0}^{\log_3(n)} \frac{n}{2} - \sum_{k=0}^{\log_3(n)} k \cdot 3^k$$

$$\leq \frac{n}{2}\log_3(n)$$

$$\boxed{T(n) \leq O(n \lg n)}$$ Ans.

(e) $T(n) = 2 \cdot T(n/2) + n/\lg n$

$a = 2,\ b = 2,\ k = 1,\ p = -1$

$a = b^k$ } case 2: if $p = -1$, then $T(n) = \theta\left(n^{\log_b a} \log_b \log_b n\right)$

$$T(n) = \theta\left(n^{\log_2 2} \log_2 \log_2 n\right)$$

$$\boxed{T(n) = \theta(n \lg \lg n)}$$ Ans.

(f) $T(n) = T(n/2) + T(n/4) + T(n/8) + n$

We guess $T(n) \leq cn$,

$$T(n) \leq T(n/2) + T(n/4) + T(n/8) + n$$

$$\leq \frac{7}{8}cn + n \leq cn. \qquad [\because c \geq 8]$$

(g) $T(n) = T(n-1) + 1/n$.

This recurrence corresponds to the harmonic series.

So that $T(n) = H_n$, where $H_n = 1/1 + 1/2 + 1/3 + \cdots 1/n$.

→ For the base case, we have $T(1) = 1 = H_1$.

→ For the inductive step, we assume that $T(n-1) = H_{n-1}$ & we have

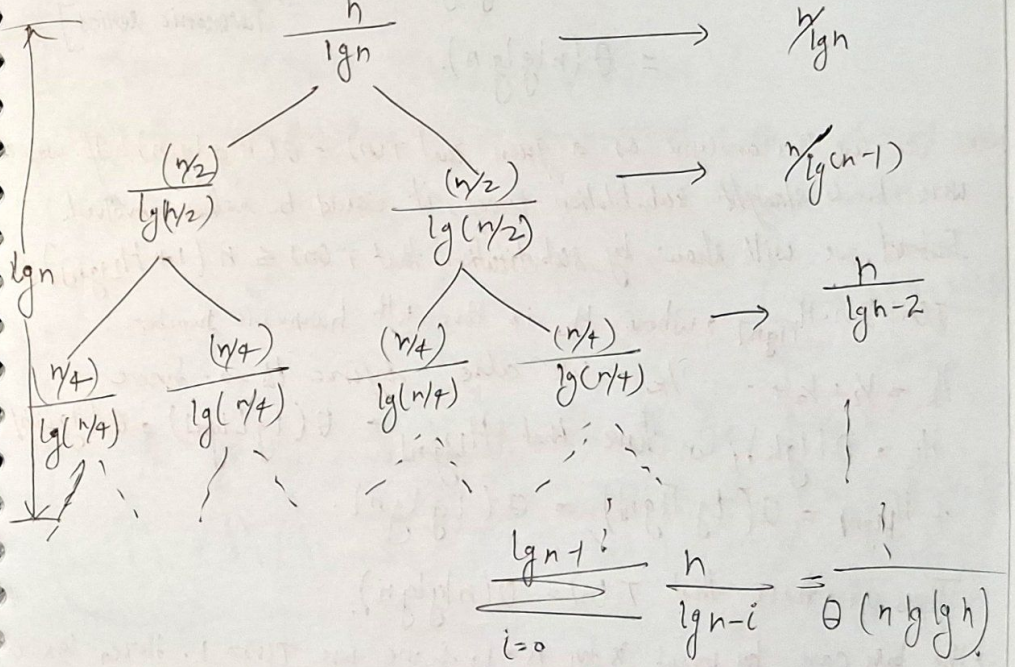
$$T(n) = T(n-1) + 1/n$$
$$= H_{n-1} + 1/n$$
$$= H_n.$$

Since, $H_n = \theta(\lg n)$ by equation (A.7), we have that $T(n) = \theta(\lg n)$.

2nd Method of sol:

(e) $2T(n) = 2T(n/2) + n/\lg n$

We can get a guess by means of a recursion tree:



We get the sum of each level by observing that at depth i, we have 2^i nodes, each with a numerator of $n/2^i$ & a denominator of $\lg(n/2^i) = \lg n - i$, so that the cost at depth i is

$$2^i \cdot \frac{n/2^i}{\lg n - i} = \frac{n}{\lg n - i}$$

The sum for all levels is

$$\sum_{i=0}^{\lg n - 1} \frac{n}{\lg n - i} = \sum_{i=1}^{\lg n} \frac{n}{i}$$

$$= n \sum_{i=1}^{\lg n} 1/i$$

$$= n \cdot \theta(\lg \lg n) \quad \text{[by equation (A.7), the harmonic series]}$$

$$= \theta(n \lg \lg n).$$

We can use this analysis as a guess that $T(n) = \theta(n \lg \lg n)$. If we were to do straight substitution proof, it would be rather involved. Instead, we will show by substitution that $T(n) \le n(1 + H_{\lfloor \lg n \rfloor})$ & $T(n) \ge n \cdot H_{\lceil \lg n \rceil}$, where H_k is the k^{th} harmonic number: $H_k = 1/1 + 1/2 + \cdots + 1/k$. We also define $H_0 = 0$. Since $H_k = \theta(\lg k)$, we have that $H_{\lfloor \lg n \rfloor} = \theta(\lg \lfloor \lg n \rfloor) = \theta(\lg \lg n)$ & $H_{\lceil \lg n \rceil} = \theta(\lg \lceil \lg n \rceil) = \theta(\lg \lg n)$.

Thus, we have that $T(n) = \theta(n \lg \lg n)$.

→ The base case for proof is for $n = 1$, & we use $T(1) = 1$. Here, $\lg n = 0$ so that $\lg n = \lfloor \lg n \rfloor = \lceil \lg n \rceil$. Since $H_0 = 0$, we have $T(1) = 1 \le (1 + H_0)$ & $T(1) = 1 \ge 0 = 1 \cdot H_0$.

→ For the upper bound of $T(n) \le n(1 + H_{\lfloor \lg n \rfloor})$, we have

$$\begin{aligned} T(n) &= 2T(n/2) + n/\lg n \\ &\le 2\left((n/2)(1 + H_{\lfloor \lg n \rfloor})\right) + n/\lg n \\ &= n(1 + H_{\lfloor \lg n \rfloor - 1}) + n/\lg n \\ &= n(1 + H_{\lfloor \lg n \rfloor - 1} + 1/\lg n) \\ &\le n(1 + H_{\lfloor \lg n \rfloor - 1} + 1/\lfloor \lg n \rfloor) \\ &= n(1 + H_{\lfloor \lg n \rfloor}), \end{aligned}$$

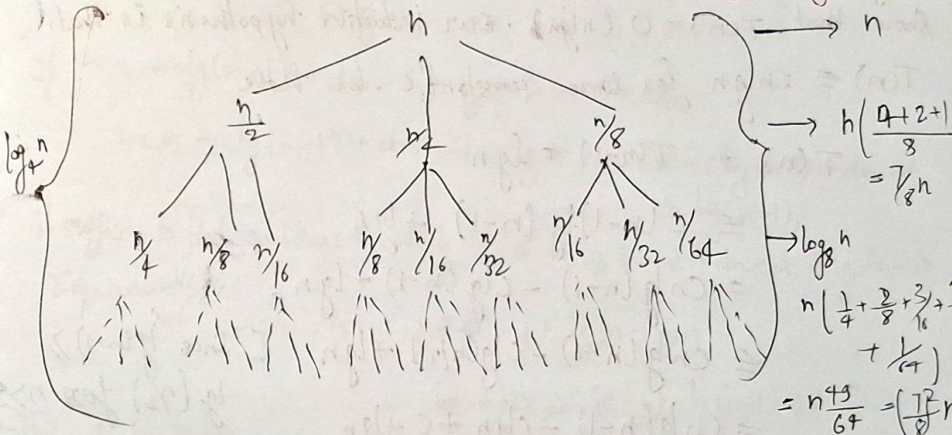
[∵ identity $H_k = H_{k-1} + 1/k$]

Thus, $T(n) = \theta(n \lg \lg n)$

2nd part:

(f) $T(n) = T(n/2) + T(n/4) + T(n/8) + n$

Using the recursion tree shown below, we get a guess of $T(n) = \theta(n)$



$$\sum_{i=1}^{\lg n} \left(\frac{7}{8}\right)^i n = \theta(n)$$

We use the substitution method to prove that $T(n) = O(n)$. Our inductive hypothesis is that $T(n) \le cn$ for some constant $c > 0$. We have

$$\begin{aligned} T(n) &= T(n/2) + T(n/4) + T(n/8) + n \\ &\le cn/2 + cn/4 + cn/8 + n \\ &= 7cn/8 + n \\ &= (1 + 7c/8)n \\ &\le cn \qquad [\because c \ge 8]. \end{aligned}$$

$\therefore \boxed{T(n) = O(n)}$

Similar for lower bound $\boxed{T(n) = \Omega(n)}$

Since $T(n) = O(n)$ & $T(n) = \Omega(n)$, we have that $\boxed{T(n) = \Theta(n)}$.

(b) $T(n) = T(n-1) + \lg n$

We guess that $T(n) = \Theta(n \lg n)$. To prove the upper bound, we will show that $T(n) = O(n \lg n)$. Our inductive hypothesis is that $T(n) \le cn \lg n$ for some constant c. We have

$$\begin{aligned} T(n) &= T(n-1) + \lg n \\ &\le c(n-1)\lg(n-1) + \lg n \\ &= cn\lg(n-1) - c\lg(n-1) + \lg n \\ &\le cn\lg(n-1) - c\lg(n/2) + \lg n \qquad [\text{Since } \lg(n-1) \ge \lg(n/2) \text{ for } n \ge 2] \\ &= cn\lg(n-1) - c\lg n + c + \lg n \\ &< cn\lg n - c\lg n + c + \lg n \\ &\le cn\lg n. \end{aligned}$$

if $-c\lg n + c + \lg n \le 0$. Equivalently,

$$\begin{aligned} -c\lg n + c + \lg n &\le 0 \\ c &\le (c-1)\lg n \\ \lg n &\ge c/(c-1) \end{aligned}$$

This works for $c = 2$ & all $n \ge 4$.

→ To prove the lower bound, we will show that $T(n) = \Omega(n \lg n)$. Our inductive hypothesis is that $T(n) \ge cn\lg n + dn$ for constants c & d. We have

$$\begin{aligned} T(n) &= T(n-1) + \lg n \\ &\ge cn(n-1)\lg(n-1) + d(n-1) + \lg n \\ &= cn\lg(n-1) - c\lg(n-1) + dn - d + \lg n \\ &\ge cn\lg(n/2) - c\lg(n-1) + dn - d + \lg n \\ &\quad (\text{since } \lg(n-1) \ge \lg(n/2) \ \forall n \ge 2) \\ &= cn\lg n - cn - c\lg(n-1) + dn - d + \lg n \\ &\ge cn\lg n \end{aligned}$$

if $-cn - c\lg(n-1) + dn - d + \lg n \ge 0$. Since

$$-cn - c\lg(n-1) + dn - d - \lg n > -cn - c\lg(n-1) + dn - d + \lg(n-1),$$

it suffices to find conditions in which $-cn - c\lg(n-1) + dn - d + \lg(n-1) \ge 0$. Equivalently,

$$\begin{aligned} -cn - c\lg(n-1) + dn - d + \lg(n-1) &\ge 0 \\ (d-c)n &\ge (c-1)\lg(n-1) + d \end{aligned}$$

This is works for $c = 1$, $d = 2$ & $\forall n \ge 2$.

Since $T(n) = O(n \lg n)$ & $T(n) = \Omega(n \lg n)$, we conclude that $T(n) = \Theta(n \lg n)$.

① $T(n) = T(n-2) + 2\lg n$

→ We guess that $T(n) = \Theta(n\lg n)$. We show that the upper bound of $T(n) = O(n\lg n)$ by means of the inductive hypothesis $T(n) \le cn\lg n$ for some constant $c > 0$. We have

$$\begin{aligned} T(n) &= T(n-2) + 2\lg n \\ &\le cn(n-2)\lg(n-2) + 2\lg n \\ &\le c(n-2)\lg n + 2\lg n \\ &= (cn - 2c + 2)\lg n \\ &= cn\lg n + (2-2c)\lg n \\ &\le cn\lg n \qquad (\because c > 1) \end{aligned}$$

$\therefore \boxed{T(n) = O(n\lg n)}$

→ For lower bound of $T(n) = \Omega(n\lg n)$, we'll show that $T(n) \ge cn\lg n + dn$, for constants $c > 0, d > 0$ to be choosen. We assume that $n \ge 4$. which implies that

① $\lg(n-2) \ge \lg(n/2)$ ② $n/2 \ge \lg n$ & ③ $n/2 \ge 2$

we have,

$$\begin{aligned} T(n) &\ge c(n-2)\lg(n-2) + d(n-2) + 2\lg n \\ &= cn\lg(n-2) - 2c\lg(n-2) + dn - 2d + 2\lg n \\ &> cn\lg(n-2) - 2c\lg n + dn - 2d + 2\lg n \\ &\quad (\text{since } -\lg n < -\lg(n-2)) \\ &= cn\lg(n-2) - 2(c-1)\lg n + dn - 2d \\ &= cn\lg n - cn - 2(c-1)\lg n + dn - 2d \\ &\ge cn\lg n, \end{aligned}$$

if $-cn - 2(c-1)\lg n + dn - 2d \ge 0$ or, equivalently, $dn \ge cn + 2(c-1)\lg n + 2d$. Pick any constant $c > 1/2$ & then pick any constant d such that $d > 2(2c-1)$.

then

$$d/2 \ge 2c - 1 = c + (c-1),$$

& adding $d/2$ to both sides, we have

$$d \ge c + (c-1) + d/2$$

multiplying by n yields

$$dn \ge cn + (c-1)n + dn/2$$

& then both multiplying & dividing middle term by 2 gives

$$dn \ge cn + 2(c-1)n/2 + dn/2$$

Using inequalities ② & ③, we get

$$dn \ge cn + 2(c-1)\lg n + 2d,$$

which is what we needed to show. Thus $T(n) = \Omega(n\lg n)$.

Since $T(n) = O(n\lg n)$ & $T(n) = \Omega(n\lg n)$, we concluded that $T(n) = \Theta(n\lg n)$.

Again ① $T(n) = T(n-2) + 1/\lg n$

$$\boxed{T(n) = \Theta(n/\lg n)}$$

$$\begin{aligned} T(n) &= T(n-2) + \frac{1}{\lg(n)} \\ &= T(n-4) + \frac{1}{\lg(n)} + \frac{1}{\lg(n-2)} \\ &= T(n-6) + \frac{1}{\lg(n)} + \frac{1}{\lg(n-2)} + \frac{1}{\lg(n-4)} = \cdots \end{aligned}$$

After k steps, we get:

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

→ base case: when $n-2k = 1$

$$k = \frac{n-1}{2}$$

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

Since there are approximately $n/2$ terms in this sum, we can approximate it as:

$$\sum_{i=0}^{(n-1)/2-1} \frac{1}{\log(n-2i)} \approx \frac{n}{2} \cdot \frac{1}{\log n} = \frac{n}{2\log n}.$$

Finally,

$$T(n) \approx T(1) + \frac{n}{2\lg n}.$$

If we ignore the constant $T(1)$ for large n, we get:

$$\boxed{T(n) = \Theta\left(\frac{n}{\lg n}\right)}$$

(i) $T(n) = \sqrt{n}\,T(\sqrt{n}) + n$

We guess $T(n) \le cn\lg\lg n$,

$$T(n) \le \sqrt{n}\,c\sqrt{n}\lg\lg\sqrt{n} + n = cn\lg\lg\sqrt{n} + n$$
$$= cn\lg\frac{\lg n}{2} + n = cn\lg\lg n - cn\lg 2 + n$$
$$= cn\lg\lg n + (1-c)n \qquad (\because c > 1)$$
$$\le cn\lg\lg n = \Theta(n\lg\lg n).$$

{ OR }

Let i be the smallest i so that $n^{\frac{1}{2^i}} \le 2$. We recall from a previous problem (3-6.e) that this is $\lg\lg n$. Expanding the recurrence, we have that it is

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

$$\boxed{T(n) = \Theta(n\lg\lg n)}$$ Ans.

2nd Method of soln (i)

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

Recall the χ_A denotes the indicator funct^n of A. We see that the sum is

$$T(0) + \sum_{j=1}^{n} \frac{1}{j} = T(0) + \int_1^{n+1} \sum_{j=1}^{n+1} \frac{\chi_{j,j+1}(x)}{j}\,dx.$$

Since $\frac{1}{x}$ is monotonically decreasing, we have that for every $i \in \mathbb{Z}^+$,

$$\text{Supse } (i,i+1) \sum_{j=1}^{n+1} \frac{\chi_{j,j+1}(x)}{j} - \frac{1}{x} = \frac{1}{i} - \frac{1}{i+1} = \frac{1}{i(i+1)}$$

Our expression for $T(n)$ becomes

$$T(N) = T(0) + \int_1^{n+1} \left(\frac{1}{x} + O\left(\frac{1}{\lfloor x\rfloor(\lfloor x\rfloor+1)}\right)\right)dx$$

We deal with the error term by first chopping out the constant amount b/w 1 & 2 & then bound the error term by $O\left(\frac{1}{x(x-1)}\right)$ which has an anti-derivative (by method of partial fractions) that is $O\left(\frac{1}{n}\right)$,

$$T(N) = \int_1^{N+1} \frac{dx}{x} + O\left(\frac{1}{n}\right)$$

This gets us our final answer of $T(n) = \Theta(\lg n)$.

4.4: Fibonacci numbers

$$F(z) = \sum_{i=0}^{\infty} F_i z^i = 0 + z + z^2 + 2z^3 + 3z^4 + \cdots$$

where F_i is the ith Fibonacci number.

(a) Show that $F(z) = z + zF(z) + z^2 F$.

Proof:

$$z + zF(z) + z^2 F(z) = z + z\sum_{i=0}^{\infty} F_i z^i + z^2 \sum_{i=0}^{\infty} F_i z^i$$

$$= z + \sum_{i=1}^{\infty} F_{i-1} z^i + \sum_{i=2}^{\infty} F_{i-2} z^i$$

$$= z + F_1 z + \sum_{i=2}^{\infty} (F_{i-1} + F_{i-2}) z^i$$

$$= z + F_1 z + \sum_{i=2}^{\infty} F_i z^i$$

$$= F(z).$$

(b) Show that $F(z) = z/1 - z - z^2 = z/(1-\phi z)(1-\hat{\phi} z)$

$$= \frac{1}{\sqrt{5}} \left(\frac{1}{1-\phi z} - \frac{1}{1-\hat{\phi} z} \right).$$

where, $\phi = \frac{1+\sqrt{5}}{2} = 1.61803\ldots$ & $\hat{\phi} = \frac{1-\sqrt{5}}{2} = -0.61803\ldots$

Proof: Note that $\phi - \hat{\phi} = \sqrt{5}$, $\phi + \hat{\phi} = 1$ & $\phi \cdot \hat{\phi} = -1$

$$F(z) = \frac{F(z)(1-z-z^2)}{1-z-z^2} = \frac{F(z) - zF(z) - z^2 F(z) - z + z}{1+z-z^2}$$

$$= \frac{F(z) - F(z) + z}{1 - z - z^2} = \frac{z}{1 - z - z^2}$$ Hence proved

$$= \frac{z}{1 - (\phi + \hat{\phi})z + z^2(\phi\hat{\phi})} = \frac{z}{(1-\phi z)(1-\hat{\phi} z)}$$ Hence proved

$$= \frac{\sqrt{5} z}{\sqrt{5}(1-\phi z)(1-\hat{\phi} z)} = \frac{(\phi - \hat{\phi})z + 1 - 1}{\sqrt{5}(1-\phi z)(1-\hat{\phi} z)}$$

$$= \frac{(1-\hat{\phi} z) - (1-\phi z)}{\sqrt{5}(1-\phi z)(1-\hat{\phi} z)} = \frac{1}{\sqrt{5}} \left(\frac{1}{1-\phi z} - \frac{1}{1-\hat{\phi} z} \right)$$

Hence proved

(c) Show that $F(z) = \sum_{i=0}^{\infty} \frac{1}{\sqrt{5}} (\phi^i - \hat{\phi}^i) z^i$.

Proof:

We have $\frac{1}{1-x} = \sum_{k=0}^{\infty} x^k$, when $|x| < 1$, thus

$$F(z) = \frac{1}{\sqrt{5}} \left(\frac{1}{1-\phi z} - \frac{1}{1-\hat{\phi} z} \right) = \frac{1}{\sqrt{5}} \left(\sum_{i=0}^{\infty} \phi^i z^i - \sum_{i=0}^{\infty} \hat{\phi}^i z^i \right)$$

$$= \sum_{i=0}^{\infty} \frac{1}{\sqrt{5}} (\phi^i - \hat{\phi}^i) z^i$$

Hence proved

(d) Use part (c) to prove that $F_i = \phi^i/\sqrt{5}$ for $i > 0$, rounded to the nearest integer. (Hint: observe that $|\hat{\phi}| < 1$.)

Proof: $F(z) = \sum_{i=0}^{\infty} \alpha_i z^i$ where $\alpha_i = \frac{\phi^i - \hat{\phi}^i}{\sqrt{5}}$.

From this follows that $\alpha_i = F_i$, that is

$$F_i = \frac{\phi^i - \hat{\phi}^i}{\sqrt{5}} = \frac{\phi^i}{\sqrt{5}} - \frac{\hat{\phi}^i}{\sqrt{5}}$$

For $i = 1$, $\phi/\sqrt{5} = (\sqrt{5}+5)/10 > 0.5$. For $i > 2$, $|\hat{\phi}^i| < 0.5$