Sabancı University
Faculty of Engineering and Natural Sciences

CS301 – Algorithms

Homework 1

Due: March 6, 2024 @ 23.55
(upload to SUCourse)

PLEASE NOTE:

- Provide only the requested information and nothing more. Unreadable, unintelligible, and irrelevant answers will not be considered.
- Submit only a PDF file. (-20 pts penalty for any other format)
- Not every question of this homework will be graded. We will announce the question(s) that will be graded after the submission.
- You can collaborate with your `TA/INSTRUCTOR ONLY` and discuss the solutions of the problems. However, you have to write down the solutions on your own.
- Plagiarism will not be tolerated.

Late Submission Policy:

- Your homework grade will be decided by multiplying what you normally get from your answers by a "submission time factor (STF)".
- If you submit on time (i.e. before the deadline), your STF is 1. So, you don't lose anything.
- If you submit late, you will lose 0.01 of your STF for every 5 mins of delay.
- We will not accept any homework later than 500 mins after the deadline.
- SUCourse's timestamp will be used for STF computation.
- If you submit multiple times, the last submission time will be used.

Question 1

The recurrence relation of a recursive divide and conquer algorithm is given. Explain this recurrence, verbally, in terms of the size of each sub-problem, the cost of dividing the problem, and combining solutions.

$$T(n) = 3T(\tfrac{n}{4}) + 2n + n^3$$

ANSWER:

$T(n) = 3T(\frac{n}{2}) + 2n + n^3$

Tan Ufuk Çelik
ID: 28285

Answer:

→ In terms of the size of each problem;
The given recursion relation shows that we divide each problem of size n into 3 smaller subproblems of size n/4. This is expressed by the term 3T(n/4).

→ The cost of dividing problem;
In this recursion relationship, the cost associated with dividing the problem is 2n. This is linear cost associated with breaking the original problem into sub-problems

→ Combining solutions;
The cost of combining solutions is n^3. This is cubic cost and with merging or combining the solutions of the sub-problems.
In divide and conquer algorithms, after doing the sub-problems, these solutions should be combined to get the solution of the original problem.

Therefore, as a summary;

$\frac{n}{4}$ → the size of each problem (n → input size)

2n → cost of the dividing

n^3 → cost of combining

Question 2

Find an asymptotically tight lower bound for the following recurrence by using the substitution method.

$$T(n) = T(\tfrac{n}{3}) + T(\tfrac{2n}{3}) + \Theta(n)$$

ANSWER:

Find an asymptotically tight lower bound for the following recurrence by using the substitution method.

$$T(n) = T(\tfrac{n}{3}) + T(\tfrac{2n}{3}) + \Theta(n)$$

Tan Ufuk Çelik
ID: 28285

Answer

Using the substitution method, I will try a candidate lower bound function $T(n) = \Omega(n \log n)$

1) Inductive hypothesis:

Assume $T(k) \geq k \log k \ldots\ldots (k < n)$

2) Inductive step:

Then, let's prove $T(n) \geq n \log n$

$$\underbrace{T(n) = T(\tfrac{n}{3}) + T(\tfrac{2n}{3}) + \Theta(n)}$$

Using the inductive hypothesis,

$$\geq \left(\frac{n}{3} \cdot \log\left(\frac{n}{3}\right)\right) + \left(\frac{2n}{3} \log\left(\frac{2n}{3}\right)\right) + \Theta(n)$$

3) Simplifying the expression:

$$= \frac{n}{3}\log n - \cancel{\frac{n}{3}\log 3} + \frac{2n}{3}\log n - \cancel{\frac{2n}{3}\log\frac{3}{2}} + \Theta(n)$$

Combining the terms with $\Theta(n)$ and dropping the lower-order terms:

4) $\geq n \log n + \Theta(n)$ $\longrightarrow$ I dropped here

Therefore, $T(n) \geq n \log n$ is ok for the given recurrence relation.

Then, base case:

To show the base case, for simplicity let's assume that $T(1) = 1$

For $n = 1$, $g(n) = n \log n = 0$, which satisfies the base case.

So, $T(n)$ is asymptotically lower-bounded by $n \log n$.

5) It is $\boxed{\Omega(n \log n)}$ asymptotically tight lower bound.

RESULT ANSWER

Question 3

For the following recurrences, either solve it by using the master method or show that it cannot be solved with the master method.

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

ANSWER:

Tan Ufuk Çelik
ID:28285

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

Since the master theorem works with recurrences of the form

In this case; $a = 1$, $b = 2$, $f(n) = \theta(1)$

≥ 1 ; > 1 ; asymptotically positive ✓

→ We need $n^{\log_b^a}$, let's calculate it:

$$n^{\log_b^a} = n^{\log_2^1} = n^0 = 1$$

→ We need to compare $n^{\log_2^1}$ with $f(n)$

$$\underbrace{f(n) = \theta(1) = \theta(n^{\log_2^1})}_{\text{It is Case 2}} \implies T(n) = \theta(n^{\log_b^a} \log n)$$

$$= \theta(n^{\log_2^1} \log n)$$

$$= \theta(n^0 \log n)$$

$$\boxed{= \theta(\log n)}$$

(b) $T(n) = 3T(\frac{n}{4}) + n \lg n$

ANSWER:

Tan Ufuk Çelik
ID: 28285

$T(n) = 3T\left(\frac{n}{4}\right) + n \log n$

$a = 3, \quad b = 4, \quad f(n) = n \log n$

$a \geq 1$ ✓ $\quad b > 1$ ✓ $\quad f(n)$ is asymptoticly positive ✓

→ We need $n^{\log_b^a}$, let's calculate it:

$n^{\log_b^a} = n^{\log_4^3}$ and $\log_4^3 < 1$

→ We need to compare $n^{\log_b^a}$ with $f(n)$

$f(n) = n \log n = \Omega\left(n^{\log_4^{3+\varepsilon}}\right)$

→ Case 3 may apply, we need to check:

a. $f(n/b) \leq c \cdot f(n)$

$3 \cdot \frac{n}{4} \cdot \log \frac{n}{4} \leq c \cdot n \cdot \log n$

where $c \geq 3/4$ $\boxed{\text{and} \quad c < 1}$ } necessary condition (and for large n)

This finishes our necessary requirements and we have that

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

$\boxed{\text{Then, } T(n) = \Theta(n \log n)}$