Missouri University of Science & Technology — Department of Computer Science

Spring 2023 — **CS 2500: Algorithms (Sec: 102)**

Solutions to Homework 1a

Name: *Sid Nadendla* **Email:** *nadendla@mst.edu*

Problem 1 Asymptotic Analysis *30 points*

1. Prove the following statements:

   (i) For any $0 < x < 1$, $e^x = \sum_{i=0}^{m-1} \frac{x^i}{i!} + \Theta(x^m)$, for all $m = 1, 2, \cdots$. **(10pts)**

   SOLUTION:

   Note that the input x is a continuous variable, which is not consistent with the traditional Θ-definition discussed in the class. Therefore, we make a minor adaptation to the definition of Θ-notation as shown below:

   Definition 1 (Θ-Notation for Real Functions)**.** *Consider two functions $f : \mathcal{X} \to \mathbb{R}$ and $g : \mathcal{X} \to \mathbb{R}$, on some set $\mathcal{X} \subset \mathbb{R}$. Then, we say $f = \Theta(g)$, if there exists two positive numbers c_L and c_U such that, for all $x \in \mathcal{X}$, we have*

$$\sum_{i=0}^{m-1} \frac{x^i}{i!} + c_L x^m \ \le\ e^x \ \le\ c_U \cdot g(x).$$

   In this problem, we have $\mathcal{X} = (0, 1)$. Then, we want to prove that, there exists two positive numbers c_L and c_U such that, for all $x \in \mathcal{X}$, then we have

$$\sum_{i=0}^{m-1} \frac{x^i}{i!} + c_L \cdot x^m \ \le\ e^x \ \le\ \sum_{i=0}^{m-1} \frac{x^i}{i!} + c_U \cdot x^m, \text{ for all } m = 1, 2, \cdots .$$

   Substituting the Taylor's series expansion of e^x at $x = 0$, for all $m = 1, 2, \cdots$, we want to find two positive constants c_L and c_U such that

$$c_L \cdot x^m \ \le\ \sum_{i=m}^{\infty} \frac{x^i}{i!} \ \le\ c_U \cdot x^m.$$

   Without any loss of generality, divide x^m on all sides to reduce the problem to the following:

$$\text{Find two positive constants } c_L \text{ and } c_U \text{ such that } c_L \le \sum_{i=m}^{\infty} \frac{x^{i-m}}{i!} \le c_U.$$

Note that since every term in the summation is non-negative for all $x \in (0,1)$, we ignore all the other terms except for the first term in the summation to pick c_L, i.e. $c_L = \frac{1}{m!}$.

On the other hand, if we expand the summation upon substituting the inequality $x \leq 1$, we get

$$\begin{aligned} \sum_{i=m}^{\infty} \frac{x^{i-m}}{i!} &\leq \sum_{i=m}^{\infty} \frac{1}{i!} \\ &< \sum_{i=0}^{m-1} \frac{1}{i!} + \sum_{i=m}^{\infty} \frac{1}{i!} = e. \end{aligned}$$

Therefore, we can pick $c_U = e$.

Given that there exists two positive constants $c_L = \frac{1}{m!}$ and $c_U = e$ for which the inequality in Definition 1, for any $x \in (0,1)$, we have

$$e^x = \sum_{i=0}^{m-1} \frac{x^i}{i!} + \Theta(x^m), \text{ for all } m = 1, 2, \cdots .$$

□

2. Solve the following problems in *CLRS*.

   (a) Problems 3.1(a,b,c) (Page 71 in 4th edition). **(9pts)**

   SOLUTION:

   Solution to Prob. 3.1(a):

   Given $p(n) = \sum_{i=0}^{d} a_i\, n^i$ with $a_d > 0$, we need to prove that $p(n) = O(n^k)$ for all $k \geq d$.

   In other words, we need to find two constants N_0 and $c > 0$ such that, for all nN_0, we have

$$\sum_{i=0}^{d} a_i\, n^i \leq c\, n^k. \tag{1}$$

   Note that, by dividing $c\, n^k$ on both sides, Equation (1) can be equivalently written as

$$\frac{1}{c} \sum_{i=0}^{d} a_i\, n^{i-k} \leq 1. \tag{2}$$

   If $k \geq d$, the exponent on n is always less than or equal to zero. Therefore, as $n \to \infty$, we have

$$\frac{1}{c} \sum_{i=0}^{d} a_i\, n^{i-k} \longrightarrow \begin{cases} 0, & \text{if } k > d, \\ \frac{a_d}{c}, & \text{if } k = d. \end{cases} \tag{3}$$

In other words, if choose $c = a_d > 0$, for some large N_0, the inequality in Equation (2) holds true. $\square$

Solution to Prob. 3.1(b):

For $k \le d$, as $n \to \infty$, we will have

$$\frac{1}{c}\sum_{i=0}^{d} a_i\, n^{i-k} \longrightarrow \begin{cases} > 0, & \text{if } k < d, \\ \dfrac{a_d}{c}, & \text{if } k = d. \end{cases} \tag{4}$$

This is because, for $k < d$ and any choice of $c > 0$ and N_0, the absolute value of $a_d n^{d-k}$ grows faster than all other terms, which is also known to be positive. Similarly, for $k = d$, we can pick $c = a_d$ to prove that $p(n) = \Omega(n^k)$, for all $k \le d$.

$\square$

Solution to Prob. 3.1(c):

When $k = d$, the polynomial $p(n)$ converges to $\dfrac{a_d}{c}$ as $N \to \infty$. Therefore, we can always find a large enough N_0 and appropriate constants c_1 and c_2 such that

$$c_1 \le \sum_{i=0}^{d} a_i\, n^{i-k} \le c_2. \tag{5}$$

For example, we could choose $c_2 = 2a_d$, $c_1 = \frac{1}{2}a_d$ and N_0 is some large number that guarantees both the inequalities. Such a number exists since $\dfrac{1}{a_d}\displaystyle\sum_{i=0}^{d} a_i\, n^{i-k} \to 1$ as $n \to \infty$, when $k = d$. $\square$

(b) Problems 4.1(g) (Page 119 in 4th edition) and 4.4(j) (Page 121 in 4th edition) using recursion trees method, and use Master's theorem to verify your result. State your base case clearly and articulate your solution. **(6pts)**

SOLUTION:

Solution to Prob. 4.1(g): Given the recursion $T(n) = 2T(n/4) + \sqrt{n}$, we obtain the recursion tree shown in Figure 1. If the recursion tree has K levels in total, the final term will be $T\left(\dfrac{n}{4^K}\right) = T(1)$. In other words, $K = \log_4 n$.

Since $T(n)$ is the sum of all terms in the recursion tree in Figure 1, we have

$$\begin{aligned} T(n) &= (K-1)\sqrt{n} + 2^K T(1) \\ &= (\log_4 n - 1)\sqrt{n} + \sqrt{n}T(1) \;=\; \Theta(\sqrt{n}\log n) \end{aligned} \tag{6}$$

From Case 2 in Master's theorem, since $f(n) = n^{\log_4 2} = \sqrt{n}$, we have

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

$\square$

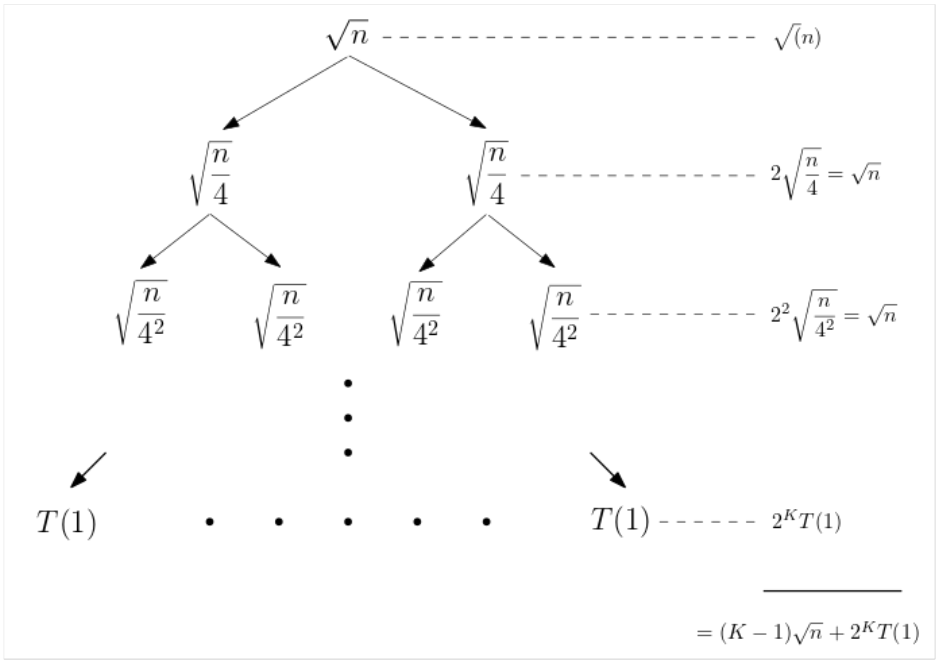
$\sqrt{n}$
$\sqrt{(n)}$
$\sqrt{\frac{n}{4}}$
$2\sqrt{\frac{n}{4}} = \sqrt{n}$
$\sqrt{\frac{n}{4^2}}$
$2^2\sqrt{\frac{n}{4^2}} = \sqrt{n}$
$T(1)$
$2^K T(1)$
$= (K-1)\sqrt{n} + 2^K T(1)$

Figure 1: Recursion Tree for Problem 4.1(g)

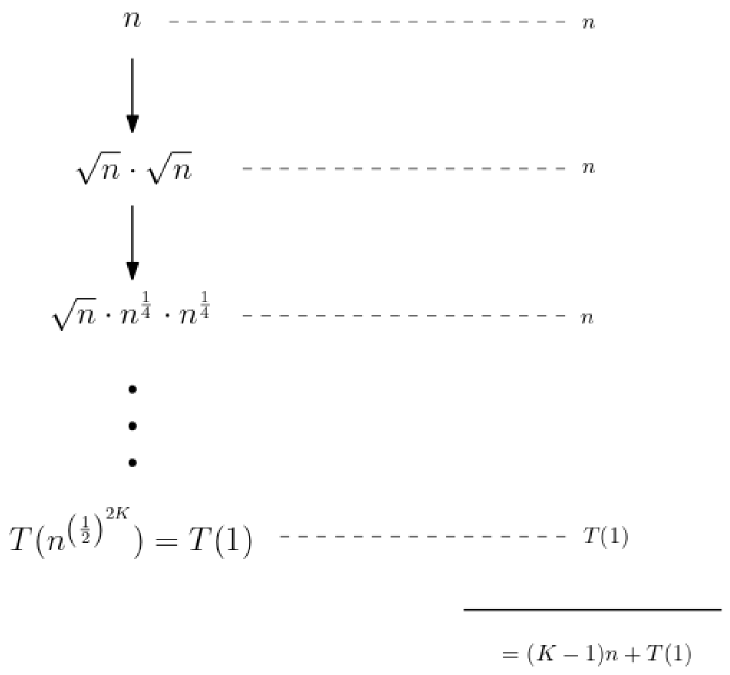
n
$\sqrt{n} \cdot \sqrt{n}$
$\sqrt{n} \cdot n^{\frac{1}{4}} \cdot n^{\frac{1}{4}}$
$T(n^{(\frac{1}{2})^{2K}}) = T(1)$
$T(1)$
$= (K-1)n + T(1)$

Figure 2: Recursion Tree for Problem 4.4(h)

Solution to Prob. 4.4(j): Given the recursion $T(n) = \sqrt{n}T(\sqrt{n}) + n$, we obtain the recursion tree shown in Figure 2. If the recursion tree has K levels in total, the final term will be $T\left(n^{\left(\frac{1}{2}\right)^{2K}}\right) = T(1)$. In other words, $K \to \infty$. Such a recursion leads to an infinite loop that never terminates. Consequently, we have $T(n) \to \infty$. $\square$

(c) Problems 4.1(h) (Page 119 in 4th edition) and 4.4(h) (Page 121 in 4th edition) using substitution method. State your base case clearly and articulate your solution. **(5pts)**

SOLUTION:

Solution to Prob. 4.1(h): Given the recursion $T(n) = T(n-2) + n^2$, let us test if this recursion reduces to $O(n^3)$ using induction principles.

Base case: When $n = 2$, we have $T(2) = T(0) + 2^2 = T(0) + 4 \leq 2^3$, assuming $T(0) = 0$.

Maintenance case: Assume that the recursion is true for $n = 2, 3, \cdots, k$. Then, we have

$$\begin{aligned} T(k+1) &= T(k-1) + k^2 \\ &\leq (k-1)^3 + k^2 \\ &< (k-1)^3 + 2^3 + 6(k-1)(k+1) \\ &= (k+1)^3. \end{aligned}$$

Termination case: Assuming that the recursion terminates for some finite value $n = N$, we have $T(N) = O(N^3)$. □

Solution to Prob. 4.4(h): Given the recursion $T(n) = T(n-1) + \log n$, let us test if this recursion reduces to $O(n \log n)$ using induction principles.

Base case: For $n = 1$, we have $T(1) = T(0) + \log 1 \leq 0 = 1 \log 1$.

Maintenance case: Assume that the recursion is true for $n = 2, 3, \cdots, k$. Then, we have

$$\begin{aligned} T(k+1) &= T(k) + \log(k+1) \\ &\leq k \log k + \log(k+1) \\ &< k \log(k+1) + \log(k+1) \\ &< (k+1) \log(k+1). \end{aligned}$$

Termination case: Assuming that the recursion terminates for some finite value $n = N$, we have $T(N) = O(N \log N)$. □

Problem 2 Validation: Theory and Experiments *20 points*

In the class, we studied different kinds of algorithms. This problem is about the theoretical analysis of the algorithms presented in the class and their validation using simulation experiments.

1. Compute the average running time of *Merge Sort* and the *maximum element* algorithms formally. **(10pts)**

   SOLUTION:

   Merge Sort (5pts): The run time of `MergeSort` is dictated by the recursion $T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n)$, which is independent of the input array itself. In other words, if the input array is picked randomly from a set $\{A_1, \cdots, A_m\}$ (each of same size n) with uniform probability distribution, we have

$$\begin{aligned}\mathbb{E}\left[T(n)\right] &= \sum_{i=1}^{m} \frac{1}{m} \cdot T(n) \\ &= \frac{1}{m} \cdot m\, T(n) \\ &= T(n) \;=\; \Theta(n \log n).\end{aligned}$$

   Find-Max (5pts): Let c_i denote the cost of executing the i^{th} line in `Find-Max` function given below.

```
FIND-MAX(A)
1  $Max = A[0]$
2  for $i = 1$ to $A.length$
3      if $A[i] > Max$
4          $Max = A[i]$
5  return $Max$
```

   Then, the average run-time for `Find-Max` can be computed as follows:

$$\mathbb{E}[T(n)] = c_1 \cdot 1 + c_2 \cdot n + c_3 \cdot (n-1) + c_4 \sum_{i=1}^{n} \mathbb{E}(t_i) + c_5 \cdot 1,$$

   where $\mathbb{E}(t_i)$ is the expected number of times Line 4 gets executed, when random inputs are fed into `Find-Max`.

   If $S_A(i)$ denotes the indicator function that returns a value '1' when $A[i] > Max$, then $\mathbb{E}(t_i) = \mathbb{E}[S_A(i)]$. Given any two real numbers $A[i]$ and Max, since there are only two possibilities: (i) $A[i] > Max$, and (ii) $A[i] \leq Max$, let us assume that these two cases occur with equal probability. Then, $\mathbb{E}(t_i) = \mathbb{E}[S_A(i)] = \frac{1}{2}$. Upon substituting this expression in the above equation, we obtain

$$\begin{aligned}\mathbb{E}[T(n)] &= c_1 \cdot 1 + c_2 \cdot n + c_3 \cdot (n-1) + c_4 \sum_{i=1}^{n} \frac{1}{2} + c_5 \cdot 1 \\ &= c_1 \cdot 1 + c_2 \cdot n + c_3 \cdot (n-1) + c_4 \cdot \frac{n}{2} + c_5 \cdot 1 \\ &= \Theta(n).\end{aligned}$$

□

2. Implement *Merge Sort* and the *maximum element* algorithms in Python, validate their correctness and simulate their average run-time performance empirically, for different input sizes. **(10pts)**
   (Hint: Random inputs can be generated using `random.sample()` function.)

   SOLUTION: Code is provided as a Jupyter notebook, along with these solutions.