

Algorithms

Homework 1 Due October 24 (Friday), 2025

物理三 黃紹凱 B12202004

October 24, 2025

Exercise 1.

(i) (5 pts) Please order the following functions asymptotically:

$$n, \log_3 n, 64\sqrt{n}, \log_{10} n, \log(n!), n\sqrt{n}, \log(3n), n \log^2 n, 2^n$$

(ii) (5 pts each) Please prove or disprove the following statements:

(a) $n^{1/2} = O(n^{1/3})$

(b) $3^n = \Omega(27^{\sqrt{n}})$

Solution 1. (No collaborators.)

(a) Notice that $\log(n!) = \Theta(n \log n)$, because of the following bounds:

$$\begin{aligned} \log(n!) &= \log 1 + \log 2 + \cdots + \log n \\ &\leq \log n + \log n + \cdots + \log n \quad (\text{n times}) = n \log n, \\ \log(n!) &= \log 1 + \log 2 + \cdots + \log \left(\frac{n}{2}\right) + \cdots + \log n \\ &\geq \log \left(\frac{n}{2}\right) + \log \left(\frac{n}{2} + 1\right) + \cdots + \log n \\ &\geq \frac{n}{2} \log \left(\frac{n}{2}\right). \end{aligned}$$

Therefore, the correct order is

$$\log_{10} n \sim \log_3 n \sim \log(3n) < 64\sqrt{n} < n < \log(n!) < n \log^2 n < n\sqrt{n} < 2^n.$$

(b) (i) Suppose $n^{1/2} \leq Cn^{1/3}$ as $n \rightarrow \infty$ for some positive C . Then raising to the sixth power and dividing by n^2 gives $n \leq C^6$, a contradiction. Hence this statement is false.

(ii) Note that $3^n = 27^{n/3}$. Since $n/3 \geq \sqrt{n}$ for sufficiently large n , we have $3^n \geq 27^{\sqrt{n}}$ for sufficiently large n . Hence this statement is true.

Exercise 2. Analyze the time complexity of the following code:

(a) (5 pts)

```
for (int i = 1; i <= n; i = i + 1) {
    for (int j = 1; j <= sqrt(i); j = j + 1) {
        ;
    }
}
```

(b) (5 pts)

```
for (int i = n; i >= 1; i = i - 1) {
    int j = i;
    while (j >= 2) {
```

```

        j = sqrt(j);
    }
}

```

Solution 2. (No collaborators.)

- (a) The outer loop runs n times, and the inner loop runs \sqrt{i} times for each i . Therefore, the total time complexity is

$$\sum_{i=1}^n \sqrt{i} = \Theta\left(\int_1^n du \sqrt{u}\right) = \Theta(n^{3/2}).$$

- (b) The outer loop runs n times, and the inner loop runs $\Theta(\log \log n)$ times, since $j^{1/2^m} = 2 \implies m = \log \log j$. Therefore, the total time complexity is

$$\Theta(n \log \log n).$$

Exercise 3. Analyze the following recursive functions asymptotically:

- (a) (5 pts)

$$T(n) = \begin{cases} T\left(\frac{n}{6}\right) + T\left(\frac{n}{4}\right) + n^2, & n > 1, \\ 1, & n = 1 \end{cases}$$

- (b) (5 pts)

$$T(n) = \begin{cases} 2T\left(\frac{n}{2}\right) + n \log n, & n > 1, \\ 1, & n = 1 \end{cases}$$

- (c) (5 pts)

$$T(n) = \begin{cases} 4T\left(n^{1/4}\right) + \log^2 n, & n > 4, \\ 2, & n \leq 4 \end{cases}$$

Solution 3. (No collaborators. References: Cormen Thomas H. et al. *Introduction to Algorithms* 4e)

- (a) Recall the following theorem from Cormen's Introduction to Algorithms:

Theorem 1 (Akra-Bazzi method). Given a recurrence formula of the form

$$T(n) = g(n) + \sum_{i=1}^k a_i T(b_i n + h_i(n)), \quad (1)$$

where $a_i > 0$, $0 < b_i < 1$, and $|g(n)| = O(n^c)$ for some constant c . Suppose that $h_i(n)$ are functions satisfying $h_i(n) = O\left(\frac{n}{(\log n)^2}\right)$. Let p be the unique solution to the equation

$$\sum_{i=1}^k a_i b_i^p = 1. \quad (2)$$

Then

$$T(n) = \Theta\left(n^p \left(1 + \int_1^n du \frac{g(u)}{u^{p+1}}\right)\right). \quad (3)$$

By theorem 1, we have $a_1 = a_2 = 1$, $b_1 = \frac{1}{6}$, $b_2 = \frac{1}{4}$, and $g(n) = n^2$. Solving for p yields $p \approx 0.439$. But we do not need the exact value of p here, since

$$n^p \left(1 + \int_1^n du \frac{u^2}{u^{p+1}} \right) = n^p \left(1 + \frac{n^{2-p}}{2-p} \right).$$

Therefore,

$$T(n) = \Theta \left(n^p \left(1 + \int_1^n du \frac{u^2}{u^{p+1}} \right) \right) = \Theta(n^2).$$

- (b) By the Master Theorem, we have $a = 2$, $b = 2$, and $f(n) = n \log n$. Note that $n^{\log_b a} = n^{\log_2 2} = n$. Since $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some $\epsilon > 0$, f is asymptotically positive, and regularity condition holds, we have

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

- (c) Consider the change of variables $m = \log \log n$, and let $S(m) = T(2^{2^m})$. Then the recurrence becomes

$$S(m) = \begin{cases} 4S(m-2), & m > 1, \\ 2, & m \leq 1. \end{cases}$$

This is a *linear* non-homogeneous difference equation with constant coefficients. Try $S_h(m) = r^m$ for some $r > 0$ as the homogeneous solution, then $r = \pm 2$, and $S_h(m) = c_1 2^m + c_2 (-2)^m$ for some constants c_1 and c_2 . Try $S_p = Am + B$ as the particular solution, then

$$Am + B = 4(4(m-2) + B) + m \implies A = -\frac{1}{3}, \quad B = -\frac{8}{9}.$$

With the initial conditions $S(0) = S(1) = 2$, we have

$$c_1 = \frac{9}{4}, \quad c_2 = \frac{23}{36},$$

then, grouping by parity gives

$$\begin{aligned} S(2k) &= \frac{26}{9} 4^k - \frac{2}{3} k - \frac{8}{9}, \\ S(2k+1) &= \frac{29}{9} 4^k - \frac{2}{3} k - \frac{11}{9}. \end{aligned}$$

Then

$$S(m) = \Theta(2^{2^m}) \implies T(n) = \Theta(2^{2 \log \log n}) = \Theta((\log n)^2).$$

Exercise 4. (15 pts) Analyze the expected number of comparisons involving the smallest element during the execution of QuickSort on n distinct numbers.

Hint: You may use a recursive function or random variables.

Solution 4. (No collaborators.)

Index the input by their ranks $1 < 2 < \dots < n$, and let X_{ij} be the indicator random variable that is 1 if elements i and j are compared during the execution of QuickSort, and 0 otherwise. Let's focus on the smallest element, i.e., $i = 1$. Then the total number of comparisons involving the smallest element can be expressed as

$$\mathbb{E}[X] = \sum_{j=2}^n \mathbb{P}[X_{1j} = 1] = \sum_{j=2}^n \frac{2}{j} = \Theta(\log n),$$

since the smallest element is compared with element j if and only if one of them is chosen as the pivot before any other element in $\{1, 2, \dots, j\}$, and the pivot is uniform within that set. Hence, the expected number of comparisons involving the smallest element during the execution of QuickSort on n distinct numbers is $\Theta(\log n)$.

Remark. We can also analyze by recursion: as above let X be the total number of comparisons. Then if the pivot rank $r = 1$, we have $n - 1$ comparisons. Otherwise, we have 1 comparison with the pivot and a left subproblem with $r - 1$ elements. Therefore,

$$\mathbb{E}[X^{(n)}] = \frac{1}{n} \left[(n-1) + \sum_{r=2}^n (1 + \mathbb{E}(X^{(r-1)})) \right], \quad \mathbb{E}[X^{(1)}] = 0,$$

and hence $\mathbb{E}[X^{(n)}] = 2(H_n - 1) = O(\log n)$.

Exercise 5. (10 pts) Consider a set S of n integers in the range $[0, n^{\log_2 \log_2 n} - 1]$. Describe how to sort S efficiently and analyze the time complexity. Note that your method must have time complexity $o(n \log n)$.

Solution 5. (No collaborators.)

Use RadixSort with base $b = n$. Since the number of passes is at most $d = \lceil \log_b(n^{\log_2 \log_2 n}) \rceil = \lceil \log_2 \log_2 n \rceil$ digits in base b . Each counting sort costs $O(n + b) = O(n + n) = O(n)$, so the total time complexity would be

$$O(dn) = O(n \log \log n) = o(n \log n).$$

Exercise 6. (10 pts) Analyze the expected time complexity of QuickSelect using random variables.

Hint: When analyzing the probability of comparing i and j , consider the position of k (the target) relative to i and j : $i < k < j$, $i < j < k$, or $k < i < j$. Determine under what situations i and j will be compared.

Solution 6. (No collaborators. Proof method is inspired by https://courses.grainger.illinois.edu/cs574/sp2022/lec/notes/03_quick_sort.pdf)

Index the input by their ranks $1 < 2 < \dots < n$, and let k be the target rank. Fix two indices $i < j$, and let X_{ij} be the indicator random variable that is 1 if elements i and j are compared during the execution of QuickSelect, and 0 otherwise. Then the total number of comparisons can be expressed as

$$X = \sum_{1 \leq i < j \leq n} X_{ij}, \quad \mathbb{E}[X] = \sum_{1 \leq i < j \leq n} \mathbb{P}[X_{ij}].$$

- (i) $i < j < k$: The i and j elements are compared if and only if one of them is chosen as the pivot before any other element in $\{i, i+1, \dots, j\}$, so $\mathbb{P}[X_{ij}] = \frac{2}{j-i+1}$. Then

$$E_1 = \mathbb{E} \left[\sum_{i < j < k} X_{ij} \right] = \sum_{i=1}^{k-2} \sum_{j=i+1}^{k-1} \frac{2}{k-i+1} = 2 \sum_{i=1}^{k-2} \frac{k-i-1}{k-i+1} \leq 2(k-2).$$

- (ii) $k < i < j$: By symmetry, we have

$$E_2 = \mathbb{E} \left[\sum_{k < i < j} X_{ij} \right] = 2 \sum_{i=k+1}^n \frac{j-k-1}{j-k+1} \leq 2(n-k).$$

- (iii) $i < k < j$: The i and j elements are compared if and only if the first nonzero indicator variable among $X_{i,i+1}, X_{i,i+2}, \dots, X_{j-1,j}$ is X_{ij} , which happens with probability $\frac{2}{j-i+1}$. Then

$$E_3 = \mathbb{E} \left[\sum_{i=1}^{k-1} \sum_{j=k+1}^n X_{ij} \right] = \sum_{i=1}^{k-1} \sum_{j=k+1}^n \frac{2}{j-i+1} \leq 2n.$$

(iv) $i = k$: Similarly, $\mathbb{P}[X_{ij} = 1] = \frac{2}{j-k+1}$, and

$$E_4 = \mathbb{E} \left[\sum_{j=k+1}^n X_{ij} \right] = \sum_{j=k+1}^n \frac{2}{j-k+1} \leq 2 \log n.$$

(v) $j = k$: Similarly, $\mathbb{P}[X_{ij} = 1] = \frac{2}{k-i+1}$, and

$$E_5 = \mathbb{E} \left[\sum_{i=1}^{k-1} X_{ij} \right] = \sum_{i=1}^{k-1} \frac{2}{k-i+1} \leq 2 \log k.$$

Combining all the cases, we have

$$\mathbb{E}[X] = E_1 + E_2 + E_3 + E_4 + E_5 \leq 2(k-2) + 2(n-k) + 2n + 2 \log n + 2 \log k = O(n).$$

Exercise 7. (15 pts) Given two sequences X (of length n) and Y (of length m), develop an algorithm to compute the length of a shortest common supersequence between X and Y .

Example: If $X = \langle A, T, C, G, T \rangle$ and $Y = \langle T, G, A, C \rangle$, then one shortest common supersequence is $\langle A, T, C, G, A, C, T \rangle$, and its length is 7.

Note: You are not allowed to apply the longest common subsequence algorithm. An $\omega(nm)$ -time algorithm will not receive full points.

Solution 7. (No collaborators. ChapGPT is used to polish the pseudocode and format pseudocode box in L^AT_EX. Wikipedia - Shortest common supersequence is referenced for the remark at the end.)

Let $d[i, j]$ be the length of a shortest common supersequence of $X[1..i]$ and $Y[1..j]$. The base cases are $d[0, j] = j$ and $d[i, 0] = i$, since the shortest common supersequence is just the longer sequence. For $i, j \geq 1$, there are two cases: If the i th character of X and the j th character of Y coincide, then we can append that character to the shortest common supersequence of $X[1..i-1]$ and $Y[1..j-1]$, so $d[i, j] = d[i-1, j-1] + 1$. Otherwise, we have two options:

- (i) Append $Y[j]$ to the shortest common supersequence of $X[1..i]$ and $Y[1..j-1]$: $d[i, j] = d[i, j-1] + 1$
- (ii) Append $X[i]$ to the shortest common supersequence of $X[1..i-1]$ and $Y[1..j]$: $d[i, j] = d[i-1, j] + 1$.

We will take the optimal solution among these options, so together we have

$$d[i, j] = \begin{cases} d[i-1, j-1] + 1, & \text{if } X[i] = Y[j]; \\ \min\{d[i, j-1] + 1, d[i-1, j] + 1\}, & \text{if } X[i] \neq Y[j]. \end{cases}$$

The time complexity of this algorithm is $\Theta(nm)$, since it fills up an $n \times m$ table with $O(1)$ work per entry, plus $O(n+m)$ time for initialization. The pseudocode is as follows:

```
SCS-LENGTH( $X[1..n], Y[1..m]$ )
1. create array  $dp[0..n][0..m]$ 
2. for  $i = 0$  to  $n$  do  $dp[i][0] \leftarrow i$ 
3. for  $j = 0$  to  $m$  do  $dp[0][j] \leftarrow j$ 
4. for  $i = 1$  to  $n$  do
5.   for  $j = 1$  to  $m$  do
6.     if  $X[i] = Y[j]$  then
7.        $dp[i][j] \leftarrow 1 + dp[i-1][j-1]$ 
8.     else
9.        $dp[i][j] \leftarrow 1 + \min\{dp[i-1][j], dp[i][j-1]\}$ 
10. return  $dp[n][m]$ 
```

Remark. This solution does not use the LCS algorithm, even though for two input strings, the relation

$$\text{LCS}(X, Y) + \text{SCS}(X, Y) = n + m$$

is true. However, there are no similar results for more than two sequences. I.e. LCS and SCS are not dual problems.

Exercise 8. (10 pts) Given sequences X (length n) and Y (length m), you can perform the following operations:

- **Insert:** insert any character at any position (cost = 2)
- **Delete:** delete a character (cost = 2)
- **Replace:** replace a character with another (cost = 3)

Develop an algorithm to compute the minimum cost of converting X into Y . An $\omega(nm)$ -time algorithm will not receive full points.

Solution 8. (No collaborators. ChapGPT is used to polish the pseudocode and format pseudocode box in \LaTeX and give refinement suggestions on time complexity analysis.)

This is the Edit Distance Problem with given operational costs. Let $d[i, j]$ be the minimum cost of converting the first i characters of X ($X[1..i]$) into the first j characters of Y ($Y[1..j]$). The base cases are: $d[0, j] = 2j$, $d[i, 0] = 2i$, since we are inserting and deleting i, j characters, respectively. For $i, j \geq 1$, there are two cases: If the i th character of X and the j th character of Y coincide, then no operation is needed, and $d[i, j] = d[i - 1, j - 1]$. Otherwise, we have three options:

- (i) Insert $Y[j]$ after processing $X[1..i]$: $d[i, j] = d[i - 1, j] + 2$
- (ii) Delete $X[i]$ after processing $X[1..i - 1]$: $d[i, j] = d[i - 1, j] + 2$
- (iii) Replace $X[i]$ with $Y[j]$: $d[i, j] = d[i - 1, j - 1] + 3$.

We will take the optimal among these options, so together we have

$$d[i, j] = \begin{cases} d[i - 1, j - 1], & \text{if } X[i] = Y[j], \\ \min\{d[i - 1, j] + 2, d[i, j - 1] + 2, d[i - 1, j - 1] + 3\}, & \text{if } X[i] \neq Y[j]. \end{cases}$$

The time complexity of this algorithm is $\Theta(nm)$, since it fills up an $n \times m$ table with $O(1)$ work per entry, plus $O(n + m)$ time for initialization. The pseudocode is as follows:

```

MIN-COST-EDIT( $X[1..n]$ ,  $Y[1..m]$ )
  # Insert/Delete cost = 2, Replace cost = 3
  1. create array  $d[0..n, 0..m]$ 
  2. for  $i = 0$  to  $n$  do  $d[i, 0] \leftarrow 2i$ 
  3. for  $j = 0$  to  $m$  do  $d[0, j] \leftarrow 2j$ 
  4. for  $i = 1$  to  $n$  do
    5. for  $j = 1$  to  $m$  do
      6. if  $X[i] = Y[j]$  then
        7.  $d[i, j] \leftarrow d[i - 1, j - 1]$ 
      8. else
        9.  $d[i, j] \leftarrow \min\{d[i, j - 1] + 2, d[i - 1, j] + 2, d[i - 1, j - 1] + 3\}$ 
    10. return  $d[n, m]$ 

```

Recommended Exercises:

- Chapter 3: P3-2, P3-3(a), P3-4
- Chapter 4: E4.3-1, E4.4-1, E4.4-4, E4.5-1, P4-4
- Chapter 6: E6.1-8, E6.3-4, P6-1
- Chapter 7: E7.2-5, E7.3-2, E7.4-4, P7-4
- Chapter 8: E8.2-6, E8.3-5, E8.4-2, P8-2
- Chapter 9: E9.1-2, E9.2-3, E9.3-3, E9.3-6, P9-1