CS101 Algorithms and Data Structures
Fall 2023
Homework 2

Due date: 23:59, October 22, 2023

1. Please write your solutions in English.
2. Submit your solutions to gradescope.com.
3. Set your FULL name to your Chinese name and your STUDENT ID correctly in Account Settings.
4. If you want to submit a handwritten version, scan it clearly. `CamScanner` is recommended.
5. When submitting, match your solutions to the problems correctly.
6. No late submission will be accepted.
7. Violations to any of the above may result in zero points.

1. (12 points) Multiple Choices

Each question has **one or more** correct answer(s). Select all the correct answer(s). For each question, you will get 0 points if you select one or more wrong answers, but you will get 1 point if you select a non-empty subset of the correct answers.

Write your answers in the following table.

(a)	(b)	(c)	(d)	(e)	(f)
AC	BC	AB	BC	D	A

(a) (2') Which of the following scenarios are appropriate for using hash tables?

A. For each user login, the website should verify whether the username (a string) exists.

B. While you are writing C++ codes, the IDE (for example, VS Code) is checking whether all brackets match correctly.

C. The Domain Name System (DNS) translates domain names (like www.shanghaitech.edu.cn) to IPv4 addresses (like 11.16.44.165).

D. A playlist where music files are played sequentially.

Solution:

B. Stack.

D. Array or list.

(b) (2') We have a hash table of size M with a uniformly distributed hash function. n elements are stored into the hash table. Which of the following statements are true?

A. If $n \leq M$, then the probability that there will be a hash collision is about $\frac{n}{M}$.

B. If collisions are resolved by chaining and the load factor is $\lambda = 0.9$, the average time complexity of a successful search (accessing an element which exists in the hash table) is $\Theta(1)$.

C. If collisions are resolved by linear probing, when there are many erase operations, lazy erasing is usually less time-consuming than actual erasing (attempting to fill the empty bin by moving other elements).

D. If collisions are resolved by quadratic probing, when the hash table is fully filled ($n = M$), you can reallocate a new space of size $2M$ and copy the M bins of the old space to the first M bins of the new space (like `std::vector`) in order to hold more elements.

Solution:

A. The probability is $1 - \frac{M(M-1)...(M-n+1)}{M^n}$, which grows much faster than $\frac{n}{M}$ as n increases from 0.

B. Suppose the elements $e_1, e_2, ..., e_n$ are inserted into the hash table sequentially.

Then e_i's expected order in the $h(e_i)$-th linked list is

$$
\begin{aligned}
&\mathbb{E}(e_i\text{'s average-case order in the } h(e_i)\text{-th linked list})\\
=&\sum_{j=1}^{M}\mathbb{P}(h(e_i)=j)(1+\text{the size of the } j\text{-th linked list})\\
=&\sum_{j=1}^{M}\frac{1}{M}(1+\text{the size of the } j\text{-th linked list})\\
=&1+\frac{1}{M}\sum_{j=1}^{M}(\text{the size of the } j\text{-th linked list})\\
=&1+\frac{i-1}{M}\\
=&1+0.9\frac{i-1}{n}\leq 1.9
\end{aligned}
$$

Then the average time of a successful search is $\Theta(1)$+the expected order of a randomly chosen element, which is also less than a constant time.

C. The only advantage of actual erasing is that the following search operations may probe less times than lazy erasing. However this is a minor effect if there are many erase operations.

D. A new hash function that maps elements to $0 \sim 2M-1$ should also be adapted. Then reassign all elements according to the new hash values.

(c) (2') Consider a table of capacity 7 using open addressing with hash function k mod 7 and linear probing. After inserting 6 values into an empty hash table, the table is below. Which of the following choices give a possible order of the key insertion sequence?

Index	0	1	2	3	4	5	6
Keys	47	15	49	24		26	61

A. 26, 61, 47, 15, 24, 49
B. 26, 24, 15, 61, 47, 49
C. 24, 61, 26, 47, 15, 49
D. 26, 61, 15, 49, 24, 47

Solution: Just verify all the choices.

(d) (2') Which of the following statements is(are) **NOT** correct?

A. $n! = o(n^n)$.
B. $(\log n)^2 = \omega(\sqrt{n})$.
C. $n^{\log(n^2)} = O(n^2 \log(n^2))$.
D. $n + \log n = \Omega(n + \log\log n)$.

Solution:

A. $n! \sim \sqrt{2\pi n}\left(\frac{n}{e}\right)^n = o(n^n)$

B. $(\log n)^k = o(n^\varepsilon), \forall k \in \mathbb{Z}^+, \varepsilon \in \mathbb{R}^+$

C. $n^{\log(n^2)} = \omega(n^3) = \omega(n^2 \log(n^2))$

D. $n \sim (n + \log n) \sim (n + \log\log n)$

(e) (2') Consider the recurrence relation

$$T(n) = \begin{cases} T(n-1) + 3n, & n > 0 \\ 2, & n = 0 \end{cases}$$

Which of the following statements are true?

A. $T(n) = 2^{n-1}$

B. $T(n) = O(n)$

C. $T(n) = \Omega\left(3^n\right)$

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

Solution:

$$T(n) = 2 + \sum_{i=1}^{n} 3i = 2 + \frac{3n(n+1)}{2} = \Theta(n^2)$$

(f) (2') Your two magic algorithms run in

$$f(n) = n\lceil\sqrt{n}\rceil(1 + (-1)^n) + 1$$

$$g(n) = n\lfloor\log n\rfloor$$

time, where n is the input size. Which of the following statements are true?

A. $f(n) = o\left(n^2\right)$.

B. $f(n) = \Omega\left(n\right)$.

C. $f(n) + g(n) = \Theta\left(n^{1.5}\right)$.

D. $f(n) + g(n) = \omega\left(n \log\left(1.5n\right)\right)$.

Here is the definition of Landau Symbols without using the limit:

$$f(n) = \Theta(g(n)) : \exists c_1, c_2 \in \mathbb{R}^+, \exists n_0, \forall n > n_0, 0 \le c_1 \cdot g(n) \le f(n) \le c_2 \cdot g(n).$$
$$f(n) = O(g(n)) : \exists c \in \mathbb{R}^+, \exists n_0, \forall n > n_0, 0 \le f(n) \le c \cdot g(n).$$
$$f(n) = \Omega(g(n)) : \exists c \in \mathbb{R}^+, \exists n_0, \forall n > n_0, 0 \le g(n) \le c \cdot f(n).$$
$$f(n) = o(g(n)) : \forall c \in \mathbb{R}^+, \exists n_0, \forall n > n_0, 0 \le f(n) < c \cdot g(n).$$
$$f(n) = \omega(g(n)) : \forall c \in \mathbb{R}^+, \exists n_0, \forall n > n_0, 0 \le g(n) < c \cdot f(n).$$

Solution:

A. $f(n) \le (2n\lceil\sqrt{n}\rceil + 1) \sim n^{1.5} = o(n^2)$

B.

$$\begin{aligned} & f(n) \neq \Omega(n) \\ \iff & \neg(\exists c \in \mathbb{R}^+, \exists n_0, \forall n > n_0, 0 \le n \le c \cdot f(n)) \\ \iff & \forall c \in \mathbb{R}^+, \forall n_0, \exists n > n_0, (n < 0) \vee (n > c \cdot f(n)) \end{aligned}$$

This holds because you can always find such odd n that $n > c \cdot f(n) = c$.

C.

$$\begin{aligned} & f(n) + g(n) \neq \Omega(n^{1.5}) \\ \iff & \neg(\exists c \in \mathbb{R}^+, \exists n_0, \forall n > n_0, 0 \le n^{1.5} \le c \cdot (f(n) + g(n)) \\ \iff & \forall c \in \mathbb{R}^+, \forall n_0, \exists n > n_0, (n^{1.5} < 0) \vee (n^{1.5} > c \cdot (f(n) + g(n))) \end{aligned}$$

This holds because you can always find such odd n that
$n^{1.5} > c \cdot (f(n) + g(n)) = c + c \cdot g(n)$, because $n^{1.5} = \omega(c + c \cdot g(n))$.
And then because

$$(f(n) + g(n) = O(n^{1.5})) \wedge (f(n) + g(n) = \Omega(n^{1.5})) \iff (f(n) + g(n) = \Theta(n^{1.5}))$$

we can get

$$(f(n) + g(n) \neq O(n^{1.5})) \vee (f(n) + g(n) \neq \Omega(n^{1.5})) \iff (f(n) + g(n) \neq \Theta(n^{1.5}))$$

D.

$$\begin{aligned} & f(n) + g(n) \neq \omega(n \log(1.5n)) \\ \iff & \neg(\forall c \in \mathbb{R}^+, \exists n_0, \forall n > n_0, 0 \le n \log(1.5n) \le c \cdot (f(n) + g(n)) \\ \iff & \exists c \in \mathbb{R}^+, \forall n_0, \exists n > n_0, (n \log(1.5n) < 0) \vee (n \log(1.5n) > c \cdot (f(n) + g(n))) \end{aligned}$$

This holds because let $c = 1$ and you can always find such odd n that
$n \log(1.5n) > c \cdot (f(n) + g(n)) = 1 + n\lfloor \log n \rfloor$ because

$$\begin{aligned} & n \log(1.5n) - (1 + n\lfloor \log n \rfloor) \\ = & (n \log n - n\lfloor \log n \rfloor) + (n \log 1.5 - 1) \end{aligned}$$

where $n \log n \ge n\lfloor \log n \rfloor$ and $n \log 1.5 = \omega(1)$

2. (11 points) Hash Table Insertions and Deletions

Consider an empty hash table of capacity 7 and with hash function $h(k) = (3k + 6) \mod 7$. Collisions are resolved by quadratic probing with the probing function $H_i(k) = (h(k) + i^2) \mod 7$, paired with lazy erasing. We will give three kinds of instructions, which are among the set {`Insert`, `Delete`, `Search`}. For `Insert/Delete` instructions, you need to fill the hash table after each instruction. For `Search` instructions, write down probing sequence (index). Use '`D`' to indicate that the bin has been marked as deleted.

(a) (1') `Insert 9`

Index	0	1	2	3	4	5	6
Key Value						9	

(b) (1') `Insert 17`

Index	0	1	2	3	4	5	6
Key Value		17				9	

(c) (1') `Insert 32`

Index	0	1	2	3	4	5	6
Key Value		17			32	9	

(d) (1') `Insert 24`

Index	0	1	2	3	4	5	6
Key Value		17	24		32	9	

(e) (1') `Insert 18`

Index	0	1	2	3	4	5	6
Key Value		17	24		32	9	18

(f) (1') `Search 18`

Solution: 4, 5, 1, 6

(g) (1') `Delete 32`

Index	0	1	2	3	4	5	6
Key Value		17	24		D	9	18

(h) (1') `Insert 25`

Index	0	1	2	3	4	5	6
Key Value		17	24		25	9	18

(i) (3') Suppose that the collisions are resolved by linear probing.

i. Write down the content of the hash table after `Insert 9, 17, 32, 24, 18`.

Index	0	1	2	3	4	5	6
Key Value		17	24		32	9	18

ii. What is the load factor λ ?

Solution:

$$\lambda = \frac{5}{7}.$$

3. (4 points) Analysing the Time Complexity of a C++ Function

What is the time complexity of `fun` (in the form of $\Theta(f(n))$, where n is the size of the `vector a`)?

```
void fun(std::vector<int> a) {
    int n = a.size();
    for (int i = 1; i < n; i *= 2) {
        // do O(1) operations
        for (int j = 0; j < n; j += i * 2) {
            // do O(1) operations
            for (int k = 0; k < i; ++k) {
                // do O(1) operations
            }
        }
    }
}
```

NOTE: Please clearly demonstrate your complexity analysis: you should give the complexity of the basic parts of an algorithm first, and then analyse the complexity of larger parts. The answer of the total complexity alone only accounts for 1pt.

Solution: The total time complexity is $\Theta(n \log n)$. (1pt)

The time complexity of the inner loop is $\Theta(i)$. (1pt)

The time complexity of the middle loop is $\Theta(n)$, because the middle loop will run $\left\lceil \frac{n}{2i} \right\rceil$ times, $\left\lceil \frac{n}{2i} \right\rceil \Theta(i) = \Theta(n)$. (1pt)

The time complexity of the outer loop is $\Theta(n \log n)$, because the outer loop will run $\lceil \log_2 n \rceil$ times, $\lceil \log_2 n \rceil \Theta(n) = \Theta(n \log n)$. (1pt)

4. (6 points) Compare and Prove

For each pair of functions $f(n)$ and $g(n)$, give your answer whether $f(n) = o(g(n))$, $f(n) = \omega(g(n))$ or $f(n) = \Theta(g(n))$. Give a **proof** of your answers.

(a) (3')

$$f(n) = \log(n!)$$
$$g(n) = \log(n^n)$$

Solution:

Method 1: $\Theta \impliedby O \wedge \Omega$

$f(n) = \Theta(g(n))$ because $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$.

Step 1: $f(n) = O(g(n))$ because obviously $f(n) \leq g(n)$.

Step 2: Prove $f(n) = \Omega(g(n))$:

$$\begin{aligned} f(n) =& \sum_{i=1}^{n} \log i \\ >& \sum_{i=\lfloor \frac{n}{2} \rfloor + 1}^{n} \log i \\ >& \sum_{i=\lfloor \frac{n}{2} \rfloor + 1}^{n} \log \frac{n}{2} \\ =& \frac{n}{2} \log \frac{n}{2} = \Theta(n \log n) = \Theta(\log(n^n)) \end{aligned}$$

Method 2: By the definition of Θ

$f(n) = \Theta(g(n))$ because $\frac{1}{2} g(n) \leq f(n) \leq g(n)$.

Step 1: $f(n) \leq g(n)$ is obvious.

Step 2: Prove $\frac{1}{2} g(n) \leq f(n) \iff 2f(n) - g(n) \geq 0$:

$$\begin{aligned} 2f(n) - g(n) =& 2\sum_{i=1}^{n} \log i - n \log n \\ =& \sum_{i=1}^{n} \log i + \sum_{i=1}^{n} \log(n - i + 1) - \sum_{i=1}^{n} \log n \\ =& \sum_{i=1}^{n} (\log i + \log(n - i + 1) - \log n) \\ =& \sum_{i=1}^{n} \log \frac{i(n - i + 1)}{n} \geq 0 \end{aligned}$$

Because it is obvious that $i(n - i + 1) \geq n$ for $i = 1, 2, ..., n$.

Method 3: Stirling Formula

$$
\begin{aligned}
n! =& \sqrt{2\pi n}\left(\frac{n}{e}\right)^n e^{\frac{\theta_n}{12n}}, (0 < \theta_n < 1) \\
\log(n!) =& \frac{1}{2}\log n + \frac{1}{2}\log(2\pi) + n\log n + \frac{\theta_n}{12n}\log e \\
\lim_{n\to\infty}\frac{f(n)}{g(n)} =& \lim_{n\to\infty}\frac{\frac{1}{2}\log n + \frac{1}{2}\log(2\pi) + n\log n + \frac{\theta_n}{12n}\log e}{n\log n} \\
=& \lim_{n\to\infty}\left(\frac{1}{2n} + \frac{\log 2\pi}{2n\log n} + 1 + \frac{\theta_n \log e}{12n^2\log n}\right) \\
=& 0 + 0 + 1 + 0 = 1
\end{aligned}
$$

(b) (3')

$$f(n) = n^{1+\varepsilon}, \qquad (\varepsilon \in \mathbb{R}^+)$$
$$g(n) = n(\log n)^k, \qquad (k \in \mathbb{Z}^+)$$

Solution: $f(n) = \omega(g(n))$ because $\lim_{n\to\infty} \frac{f(n)}{g(n)} = +\infty$.

Method 1:

Step 1: Prove $\lim_{n\to\infty} \frac{n^c}{\log n} = +\infty, \forall c > 0$.

$$\begin{aligned}\lim_{n\to\infty} \frac{n^c}{\log n} &= \lim_{n\to\infty} \frac{cn^c}{\log n^c} \\ &= \lim_{n\to\infty} \frac{cn}{\log n} = +\infty\end{aligned}$$

because obviously $\lim_{n\to\infty} \frac{n}{\log n} = +\infty$.

Step 2: Prove $\lim_{n\to\infty} \frac{n^{1+\varepsilon}}{n(\log n)^k} = +\infty, \forall \varepsilon > 0, k \in \mathbb{Z}^+$.

$$\begin{aligned}\lim_{n\to\infty} \frac{n^{1+\varepsilon}}{n(\log n)^k} &= \lim_{n\to\infty} \frac{n^\varepsilon}{(\log n)^k} \\ &= \lim_{n\to\infty} \left(\frac{n^{\frac{\varepsilon}{k}}}{\log n}\right)^k = +\infty\end{aligned}$$

because $\lim_{n\to\infty} \frac{n^{\frac{\varepsilon}{k}}}{\log n} = +\infty$ which is implied by Step 1.

Method 2: L'Hopital's rule

$$\begin{aligned}\lim_{n\to\infty} \frac{n^{1+\varepsilon}}{n(\log n)^k} &= \lim_{n\to\infty} \frac{n^\varepsilon}{(\log n)^k} \\ &= \lim_{x\to+\infty} \frac{x^\varepsilon}{(\log x)^k} \\ &= \lim_{x\to+\infty} \frac{\varepsilon x^{\varepsilon-1}}{k(\log x)^{k-1}\frac{\log_2 e}{x}} \text{ (L'Hopital's rule)} \\ &= \lim_{x\to+\infty} \frac{\varepsilon x^\varepsilon \ln 2}{k(\log x)^{k-1}} \\ &= \lim_{x\to+\infty} \frac{\varepsilon^2 x^{\varepsilon-1} \ln 2}{k(k-1)(\log x)^{k-2}\frac{\log_2 e}{x}} \text{ (L'Hopital's rule)} \\ &= \lim_{x\to+\infty} \frac{\varepsilon^2 x^\varepsilon \ln^2 2}{k(k-1)(\log x)^{k-2}} \\ &= \ldots\ldots \\ &= \lim_{x\to+\infty} \frac{\varepsilon^k x^\varepsilon \ln^k 2}{k!} = +\infty\end{aligned}$$