CSCI2100 Asm1

Q1. [10 marks] Consider the algorithm TWOSUM shown below. What is the worst case time complexity (using **Big-Oh**) of this algorithm? Justify your answer.

```
Algorithm TWOSUM(arr, n, target)
1:  numPairs = 0
2:  for i = 0 to n − 2
3:      for j = i + 1 to n − 1
4:          if arr[i] + arr[j] == target
5:              numPairs = numPairs + 1
6:  return numPairs
```

The number of basic operations:

1. $O(1)$
2. $O(n)$
3. $O(n^2)$
4. $O(n^2)$
5. $O(n^2)$
6. $O(1)$

By the sum property, the total time complexity of the above algorithm

$= O(max(1, n, n^2, n^2, n^2, 1)) = O(n^2)$

Q2. [24 marks] Answer the following questions related to $O(\cdot)$, $\Omega(\cdot)$ and $\Theta(\cdot)$.

- **(i). [5 marks]** What is the asymptotic (**Big-Oh**) complexity of the function $g(n) = (n^3 + n^2 \log n) \cdot (n^2 + n\sqrt{n})$?
- **(ii). [5 marks]** What is the asymptotic (**Big-Theta**) complexity of the function $g(n) = (n^4 + 12n^2 + 5n) \cdot (3n + 6)$?
- **(iii). [7 marks]** Using the basic definitions, prove that $\sum_{i=1}^{n} i^n = O(n^{n+1})$.
- **(iv). [7 marks]** Let $f(n)$ be an non-negative function. Using the basic definitions, prove that $max(f(n), O(f(n))) = \Theta(f(n))$.

(i). Let $g_1(n) = n^3 + n^2 logn$ and $g_2(n) = n^2 + n\sqrt{n}$

Then, $g_1(n) = O(n^3)$ and $g_2(n) = O(n^2)$

Then, $g(n) = O(n^3 \cdot n^2) = O(n^5)$

(ii). Let $g_1(n) = n^4 + 12^2 + 5n$ and $g_2(n) = 3n + 6$

Then, $g_1(n) = \Theta(n^4)$ and $g_2(n) = \Theta(n)$

Then, $g(n) = \Theta(n^4 \cdot n) = \Theta(n^5)$

(iii). Let $g(n) = \sum_{i=1}^{n} i^n$

When $n \geq 1$, $g(n) = i^n + g(n-1) = i^n + (i-1)^n + \cdots + 1^n$

$\leq n^n + n^n + \cdots + n^n$

$\leq n^{n+1}$

$= O(n^{n+1})$

Therefore, $g(n) = \sum_{i=1}^{n} i^n = O(n^{n+1})$

(iv). First, $max(f(n), O(f(n))) \leq f(n) + O(f(n))$

$\leq O(f(n) + O(f(n)))$

$\leq O(f(n))$

Also, $max(f(n), O(f(n))) \geq \frac{1}{2}[f(n) + O(f(n))]$

$\geq \Omega(\frac{1}{2}[f(n) + O(f(n))])$

$\geq \Omega(f(n))$

Therefore, $max(f(n), O(f(n))) = \Theta(f(n))$.

Q3. [15 marks] Use the master method to give asymptotic (**Big-Oh**) bounds for the following recurrences.

- **(i). [5 marks]** $g(1) = c_0$, $g(n) \leq 3g(\lceil n/2 \rceil) + 2n^2$.
- **(ii). [5 marks]** $g(1) = c_0$, $g(n) \leq 8g(\lceil n/2 \rceil) + 2n^3$.
- **(iii). [5 marks]** $g(1) = c_0$, $g(n) \leq 2g(\lceil n/4 \rceil) + 4$.

(i). $g(n) \leq 3 \cdot g(\lceil \frac{n}{2} \rceil) + O(n^2)$

Note that $a = 3, b = 2$ and $\lambda = 2$.

$\because log_b a = log_2 3 \approx 1.58 < 2 = \lambda$

$\therefore g(n) = O(n^2)$

(ii). $g(n) \leq 8 \cdot g(\lceil \frac{n}{2} \rceil) + O(n^3)$

Note that $a = 8, b = 2$ and $\lambda = 3$.

$\because log_b a = log_2 8 = 3 = \lambda$

$\therefore g(n) = O(n^3 \cdot logn)$

(iii). $g(n) \leq 2 \cdot g(\lceil \frac{n}{4} \rceil) + O(n^0)$

Note that $a = 2, b = 4$ and $\lambda = 0$.

$\because log_b a = log_4 2 = 0.5 > 0 = \lambda$

$\therefore g(n) = O(n^{log_b a}) = O(n^{0.5})$

Q4. [10 marks] For the following functions, sort them in **nonincreasing** order of the growth rate. (Hint. If $f_1 = O(f_2)$, f_1 grows no faster than f_2.)

$$f_1(n) = n^{1.4}\log n, \quad f_2(n) = (\log n)^{10}, \quad f_3(n) = 1.0001^n, \quad f_4(n) = n^{1.5}.$$

$$\lim_{n\to\infty} \frac{n^{1.5}}{n^{1.4}logn} = \lim_{n\to\infty} \frac{n^{0.1}}{logn}$$

$$= 0.1 \cdot \lim_{n\to\infty} n^{0.1} \to \infty$$

$$\therefore f_1 \leq f_4$$

$$\lim_{n\to\infty} \frac{n^{1.4}logn}{(logn)^{10}} = \lim_{n\to\infty} \frac{n^{1.4}}{(logn)^{9}}$$

$$= \lim_{n\to\infty} \frac{1.4n^{1.4}}{9(logn)^{8}} \to \infty$$

$$= \frac{(1.4)^9}{9!} \cdot \lim_{n\to\infty} n^{1.4} \to \infty$$

$$\therefore f_2 \leq f_1$$

Note that a polynomial function grows no faster than an exponential function;

$$\therefore\ f_4 \leq f_3$$

$\therefore$ Growth rate with non-increasing order: f_2, f_1, f_4, f_3

Q5. [16 marks] Given that $T(1) = 1$, answer the following questions.

(i). [8 marks] Show that the solution of $T(n) = T(n-1) + 2n$ is $O(n^2)$.

(ii). [8 marks] Show that the solution of $T(n) = n \cdot T(n-1)$ is $O(n^n)$.

(i).
$$\begin{aligned} T(n) &= T(n-1) + 2n \\ &= T(n-2) + 2n + 2(n-1) \\ &= T(1) + 2\sum_{k=2}^{n} k \\ &= 1 + 2 \cdot \frac{(2+n)(n-1)}{2} \\ &= 1 + n^2 + n - 2 \\ &= O(n^2) \end{aligned}$$

(ii).
$$\begin{aligned} T(n) &= n \cdot T(n-1) \\ &= n \cdot (n-1) \cdot T(n-2) \\ &= n! \cdot T(1) \\ &= n! \\ &\leq n^n \\ &= O(n^n) \end{aligned}$$

Q6. [15 marks] Answer the following questions about the linked list.

(i). [5 marks] Assume that we have a linked list: $head \leftrightarrow 1 \leftrightarrow 7 \leftrightarrow 4 \leftrightarrow 9 \leftrightarrow tail$ as shown in Fig. 1. Fill the values in the question marks.

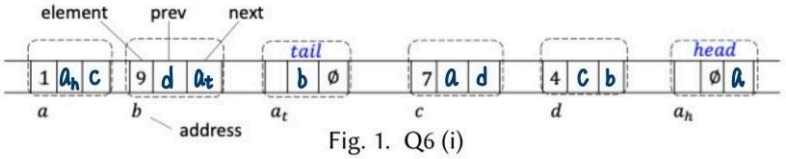

Fig. 1. Q6 (i)

(ii). [5 marks] We have a linked list: $head \leftrightarrow 6 \leftrightarrow 2 \leftrightarrow 4 \leftrightarrow tail$ as shown in Fig. 2. Assume that we insert element 5 to the tail of the list, that is, after 4, and the address of the node is a. Update the values after the insertion.

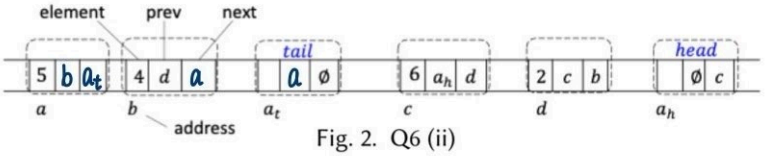

Fig. 2. Q6 (ii)

(iii). [5 marks] We have a linked list: $head \leftrightarrow 4 \leftrightarrow 3 \leftrightarrow 6 \leftrightarrow 9 \leftrightarrow tail$ as shown in Fig. 3. Update the values after deleting the node storing element 9 assuming we know its address b.

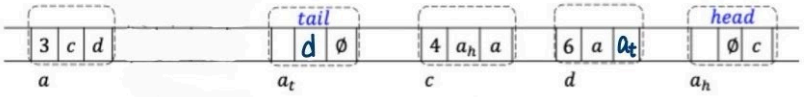

Fig. 3. Q6 (iii)

Q7. [10 marks] Answer the following questions about stacks. Initially, the stacks are empty.

- **(i). [3 marks]** Assume the sequence of pushed-in elements is $[1, 3, 2, 4, 5, 6]$, and the sequence of stack operations is [PUSH, PUSH, POP, PUSH, PUSH, POP, POP, PUSH, PUSH, POP, POP, POP]. What is the sequence of popped-out elements?
  - **Example.** Consider the sequence of pushed-in elements $[3, 1, 2]$, and the sequence of stack operations [PUSH, PUSH, POP, PUSH, POP, POP]. That means the stack operations are [PUSH(3), PUSH(1), POP( ), PUSH(2), POP( ), POP( )]. Then the relative sequence of popped-out elements, which consists of the popped elements of each POP operation in order, will be $[1, 2, 3]$.

(i). The stack operations are:

[PUSH(1), PUSH(3), POP(), PUSH(2), PUSH(4), POP(), POP(), PUSH(5), PUSH(6), POP(), POP(), POP()]

∴ The required sequence is: [3,4,2,6,5,1]

(ii). [2 marks] Assuming that the size of the stack is not infinite, an error will occur if the number of elements in the stack exceeds the size of the stack. In (i), what is the minimum possible stack size required to ensure that all operations are error-free? This means, that during the process, what is the maximum number of elements in the stack?

- **Example.** Consider the sequence of pushed-in elements $[3, 1, 2]$, and the sequence of stack operations [PUSH, PUSH, POP, PUSH, POP, POP]. That means:
  (1) After PUSH(3), the elements in the stack from bottom to top are $[3]$.
  (2) After PUSH(1), the elements in the stack from bottom to top are $[3, 1]$.
  (3) After POP( ), the elements in the stack from bottom to top are $[3]$.
  (4) After PUSH(2), the elements in the stack from bottom to top are $[3, 2]$.
  (5) After POP( ), the elements in the stack from bottom to top are $[3]$.
  (6) After POP( ), the elements in the stack from bottom to top are $[]$.
  During the entire process, the minimum number of elements in the stack is 2, so the size of the stack is at least 2.

(ii). The stack operations are:

(1) After PUSH(1), the element in the stack from bottom to top is [1].
(2) After PUSH(3), the elements in the stack from bottom to top are [1,3].
(3) After POP(), the element in the stack from bottom to top is [1].
(4) After PUSH(2), the elements in the stack from bottom to top are [1,2].
(5) After PUSH(4), the elements in the stack from bottom to top are [1,2,4].
(6) After POP(), the elements in the stack from bottom to top are [1,2].
(7) After POP(), the element in the stack from bottom to top is [1].
(8) After PUSH(5), the elements in the stack from bottom to top are [1,5].
(9) After PUSH(6), the elements in the stack from bottom to top are [1,5,6].
(10) After POP(), the elements in the stack from bottom to top are[1,5].
(11) After POP(), the element in the stack from bottom to top is [1].
(12) After POP(), the element in the stack from bottom to top is [].

∴ Maximum number of elements in the stack = 3

(iii). [3 marks] Assume the sequence of pushed-in elements is $[1, 3, 2, 5, 4, 6]$, and the sequence of popped-out elements is $[1, 2, 3, 4, 5, 6]$. What is the minimum possible size of the stack?

* **Example.** Consider the sequence of pushed-in elements $[1, 3, 2]$, and the sequence of popped-out elements $[1, 2, 3]$. There exists a sequence of stack operations [PUSH, POP, PUSH, PUSH, POP, POP], which means the stack operations are [PUSH(1), POP( ), PUSH(3), PUSH(2), POP( ), POP( )]. During the entire process, the minimum number of elements in the stack is 2, so the size of the stack is at least 2.

(iii). To match the pushed-in and popped-out elements, the stack operations are:
[PUSH(1), POP(), PUSH(3), PUSH(2), POP(), POP(), PUSH(5), PUSH(4), POP(), POP(), PUSH(6), POP()]

Note that the size of the stack is at maximum after PUSH(2) AND PUSH(4).
∴ Minimum size of the stack = 2

(iv). [2 mark] Assume the sequence of pushed-in elements is $[1, 2, 3, 4, 5]$, and the size of stack is 4. How many different sequences of operations will not result in an error(This means that during the execution of each operation sequence, the number of elements in the stack will not exceed 4)?*HINT: For a stack of unlimited size and a sequence of pushed-in elements of length n, the number of different sequences of operations is* $\frac{1}{n+1}\binom{2n}{n}$.

* **Example.** Consider the sequence of pushed-in elements $[1, 2, 3]$. If there is no limit to the size of the stack, then it can be directly concluded that the number of different operation sequences is $\frac{1}{3+1}\binom{2\times3}{3} = 5$. If the size of the stack is 2, then there are 4 possible operation sequences, which are [PUSH, POP, PUSH, POP, PUSH, POP], [PUSH, PUSH, POP, POP, PUSH, POP], [PUSH, POP, PUSH, PUSH, POP, POP] and [PUSH, PUSH, POP, PUSH, POP, POP] respectively, where the number of elements in the stack during each operation sequence will not exceed 2.

(iv). Note that when the size of the stack = 4,
the only stack operation that will lead to an error is:
[PUSH(), PUSH(), PUSH(), PUSH(), PUSH(), POP(), POP(), POP(), POP(), POP()].

∴ The number of different sequences of operations = $\frac{1}{5+1}\binom{2\cdot5}{5} - 1 = 41$.