ShanghaiTech University

CS101 Algorithms and Data Structures
Fall 2024

Homework 1

Due date: October 9, 2024, at 23:59

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

1. (7 points) Academic Integrity

Please determine who violated academic integrity or committed plagiarism in the following situations. Tick (√) for the correct answers.

(a) (1') Alice posted one of the homework questions on Stack Overflow and copied the answer from others to her homework with some trivial modifications.

[✓ **Alice**] ◯ Bob ◯ Both Alice and Bob ◯ Neither Alice nor Bob

(b) (1') Alice entered the question description into ChatGPT, then slightly modified ChatGPT's response and submitted it as her own answer.

[✓ **Alice**] ◯ Bob ◯ Both Alice and Bob ◯ Neither Alice nor Bob

(c) (1') Bob lent Alice his laptop to play Genshin. Alice found Bob's solution to Homework 1 on the desktop and copied it using USB drive sneakily.

◯ Alice ◯ Bob [✓ **Both Alice and Bob**] ◯ Neither Alice nor Bob

(d) (1') Alice and Bob are good friends. Alice asked if Bob could send her some code snippets so that she could have some ideas of what is going on in this programming assignment (promising, of course, not copying). Bob agreed and shared his code repository with Alice. Alice copied Bob's code, changed some variable names, and submitted the code to CS101 Online Judge.

◯ Alice ◯ Bob [✓ **Both Alice and Bob**] ◯ Neither Alice nor Bob

(e) (1') To boost efficiency and save precious time, Bob used Copilot as an assistant to write code in his programming assignments.

◯ Alice [✓ **Bob**] ◯ Both Alice and Bob ◯ Neither Alice nor Bob

(f) (1') Bob is Alice's boyfriend. In order to enhance their romantic relationship, Alice and Bob studied together in ShanghaiTech Library and collaborated on one homework question. They discussed about some algorithm using a whiteboard, without giving any exact solution.

◯ Alice ◯ Bob ◯ Both Alice and Bob [✓ **Neither Alice nor Bob**]

(g) (1') Alice completed the programming assignment herself on Bob's computer and accidentally submitted her code to CS101 Online Judge using Bob's account. After that, Alice and Bob resubmitted their assignments using their own accounts respectively.

◯ Alice ◯ Bob [✓ **Both Alice and Bob**] ◯ Neither Alice nor Bob

2. (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)
ADE	D	AD	A

(a) (3') Which of the following statements about arrays and linked lists are true?

A. A doubly linked list consumes more memory than a (singly) linked list of the same length.

B. Inserting an element into the middle of an array takes constant time.

C. Reversing a singly linked list takes constant time.

D. Given a pointer to some node in a doubly linked list, we are able to gain access to every node of it.

E. If we implement a queue using the circular array, the minimal memory we need is related to the maximal possible numbers of elements in the queue.

Solution:

B. $O(n)$ time.

C. $\Theta(n)$ time.

(b) (3') Suppose we use a circular array with an index range from 0 to $N-1$ to implement a queue, and currently Front is pointing at index m. We will know that if this queue is full with N elements, the last element in it should be stored at index = _____. (Options below are mapped to **Integers Modulo N**: $\mathbb{Z}_N = \{0, 1, 2, ..., N-1\}$)

A. 0

B. m

C. $N-1$

D. $m-1$

Solution: It is a circular array, so the last element is just before the first element.

(c) (3') Which of the following statements is(are) correct?

A. $n! = o(n^n)$**.**

B. $n^n = O(n!)$

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

D. $n + \log n = \Omega(n + \log\log n)$**.**

E. $n^{\log(n^2)} = O(n^2 \log(n^2))$.

Solution:

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

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

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

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

(d) (3') 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 is 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, which may be helpful for you to strictly prove whether each choice is correct:

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

Solution: To quickly judge whether such diverge function is $\Theta/O/\Omega/o/\omega$ of another function, we don't really need to strictly prove by the definition of Landau Symbols without using the limit.

For example, for $f(n) + g(n)$ which diverges into $\Theta(n^{1.5})$ and $\Theta(n \log n)$, these are correct:

$$\begin{aligned} f(n)+g(n) &= o(n^2) \\ f(n)+g(n) &= O(n^{1.5}) \\ f(n)+g(n) &= \Omega(n\log n) \\ f(n)+g(n) &= \omega(n) \end{aligned}$$

But these are wrong:

$$\begin{aligned} f(n)+g(n) &= \Theta(n^{1.5}) \\ f(n)+g(n) &= o(n^{1.5}) \\ f(n)+g(n) &= \omega(n\log n) \\ f(n)+g(n) &= \Theta(n\log n) \end{aligned}$$

Anyway, here is the strict proof by the definition of Landau Symbols without using the limit.

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

B.

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

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

C.

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

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

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

D.

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

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

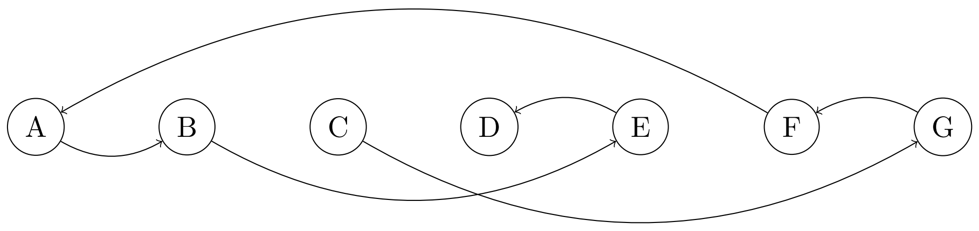
3. (10 points) Array Representation of Linked-List

Recall that we store a linked-list in an array to avoid memory allocation problems in the lecture. In this question, we use two arrays: `next` and `value` to implement a singly-linked-list.

The elements at each index in `next` and `value` are together to represent a node in the linked-list, where `next` represents the array index where the next node is located (-1 for already reaching the tail), and `value` represents the data stoblue in the node.

(a) (5') **Store a Linked-List in an Array**

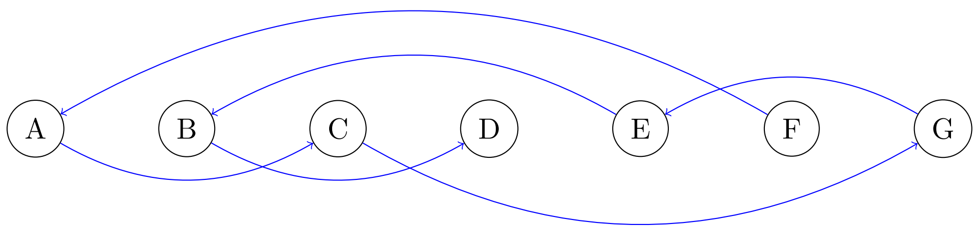
Fill in the table below to finish the array representation of the linked-list shown below. Fill in -1 to represent the end of the linked-list.



index	0	1	2	3	4	5	6
`value`	A	B	C	D	E	F	G
`next`	1	4	6	-1	3	0	5

(b) (5') **From Array to Linked-List**

Here are the arrays `next` and `value`. Please draw the linked-list below represented by these two arrays.



index	0	1	2	3	4	5	6
`value`	A	B	C	D	E	F	G
`next`	2	3	6	-1	1	0	4

4. (10 points) Vocabulary List

Alice is memorizing English words to enlarge her vocabulary in preparation for her approaching CET-4 test and she needs your help! Alice uses a `Node` to define a word and a linked list which contains the words that she is trying to memorize now. The structure of `Node` is as follows.

```
struct Node {
  string word;
  Node *next;
  Node(const string& _word, Node *_next)
      : word(_word), next(_next) {}
};
Node *head;
```

Each time she reviews her vocabulary list, she steps through the linked list starting from `head`. However, sometimes there is a word that she doesn't know, so she wants to grab the corresponding `Node` out and put it at the start of the linked list, so next time she can review it first.

For example, if the vocabulary list is

`algorithm` → `breath` → `capacity` → `degree`

and she doesn't know the word `capacity`, then she wants the list be modified to

`capacity` → `algorithm` → `breath` → `degree`

(a) (6') Alice needs your help to finish her function `move_to_first(Node *head, Node *p)` where `head` represents the first element of her non-empty vocabulary list and she wants to move the **next** word of `p` to the first. The function returns the **new** head pointer after this process. She can guarantee that `p` points to a `Node` in the linked list, but not the last one.

```
Node* move_to_first(Node *head, Node *p) {
    Node *to_move = p->next;
    // Write something below
    // ...
    // Write something above
}
```

Please help her complete the function.

Solution:

```
p->next = to_move->next;
to_move->next = head;
return to_move;
```

(b) (4') Now Alice wants to reverse the entire list using the `move_to_first` function. Please help her complete the reversion function `reverse_list(Node *head)` where `head` is the head pointer of the vocabulary list to be reversed. The function returns the **new** head pointer after this process. She can guarantee that the list is non-empty.

```
Node* reverse_list(Node *head){
    Node *p = head;
    while(/* (1) */){
        /* (2) */ = move_to_first(/* (3) */, /* (4) */);
    }
    return head;
}
```

Please help her complete this function by filling your code into blank (1) to (4). Each blank should be just an expression or a variable.

Solution:

1. `p->next != nullptr`
2. `head`
3. `head`
4. `p`

5. (10 points) Postfix Expression

Reverse Polish notation (RPN) is a mathematical notation in which operators follow their operands. Using a stack, we can evaluate postfix notation equations easily.

A postfix expression (Reverse-Polish Notation) with single-digit operands is shown below: (where ˆ is the exponentiation operator)

$$8\ 2\ 3\ \hat{}\ /\ 2\ 3\ *\ +\ 5\ 1\ *\ -$$

(a) (1') What's the in-fix expression of the expression above?

Solution:

$$8\ /\ 2\ \hat{}\ 3\ +\ 2\ *\ 3\ -\ 5\ *\ 1$$

(b) (4') The changing of the stack to calculate the final result is:

		3				3				1		
	2	2	8		2	2	6		5	5	5	
8	8	8	8	1	1	1	1	7	7	7	7	2

(c) (3') Try to convert the following in-fix expression into post-fix expression: (You don't need to calculate them)

(1) $1 * 2 + 3$

Solution: 1 2 * 3 +

(2) $1 + 2 * 3 + (4 * 5 + 6) * 7$

Solution: 1 2 3 * + 4 5 * 6 + 7 * +

(3) $1 + (2 * 3\hat{}4)/(5 + 6) + 7$

Solution: 1 2 3 4 ˆ * 5 6 + / + 7 +

(d) (2') Please judge whether the following post-fix expression is legal. Tick (√) for the legal expressions.

○ 1 2 ∗ + 3 5 + [✓ 4 5 6 / ∗ 1 /] ○ 1 + 2 − 3 + 4 [✓ 7 8 9 1 + − ∗]

6. (11 points) Compare and Prove

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

Note 1: Give your answer in the most precise form.

Note 2: You'd better try to prove them by calculating limits.

(a) (3') $f(n) = 1.01^n$ and $g(n) = n^{10}$

Solution: $f(n) = \omega(g(n))$.

Show by limits:

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{1.01^n}{n^{10}} = \lim_{n\to\infty} \frac{1.01^n \ln^{10}(1.01)}{10!} = +\infty \Rightarrow f(n) = \omega(g(n))$$

(b) (4') $f(n) = \log(n!)$ and $g(n) = \log(n^n)$

Solution: $f(n) = \Theta(g(n))$

Method 1: $f(n) = \Theta(g(n)) \Leftrightarrow f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$

Step 1: $f(n) = O(g(n))$ since $f(n) \le g(n)$ (trivial).

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

$\exists c_1, c_2 \in \mathbb{R}^+, \exists n_0, \forall n > n_0, 0 \le c_1 \cdot g(n) \le f(n) \le c_2 \cdot g(n) \Rightarrow f(n) = \Theta(g(n))$.

We will show that $\frac{1}{2} g(n) \le f(n) \le g(n)$.

Step 1: $f(n) \le g(n)$ since $e^{f(n)} = n! < n^n = e^{g(n)}$.

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

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

Since $i(n-i+1) \ge n$ for $i = 1, 2, ..., n$.

Method 3: Stirling Formula

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

(c) (4') $f(n) = n^{1+\varepsilon}, \varepsilon \in \mathbb{R}^+$ and $g(n) = n(\log n)^k, k \in \mathbb{Z}^+$

Solution: $f(n) = \omega(g(n))$

Method 1:

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

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

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

Method 2: L'Hopital's rule

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

7. (11 points) Problem Analysis

Consider the following basic problem:

You're given an array A consisting of n integers $A[1], A[2], ..., A[n]$. You'd like to output a two-dimensional n-by-n array B in which $B[i, j]$ (for $i < j$) contains the sum of array entries $A[i]$ through $A[j]$ – that is, the sum $A[i] + A[i+1] + ... + A[j]$. (the value of array entry $B[i, j]$ is left unspecified whenever $i \geq j$.)

Here's the pseudocode of a simple algorithm to solve this problem. (Initially $B[i, j] = 0$ for all valid i, j)

```
for $i = 1$ to $n$ do
    for $j = i + 1$ to $n$ do
        for $k = i$ to $j$ do
            $B[i, j] \leftarrow B[i, j] + A[k]$
        end for
    end for
end for
```

For the following questions,

- Define $T(n)$ as the total running time of the algorithm.
- Suppose each operation like $B[i, j] \leftarrow B[i, j] + A[k]$ costs a constant time c.
- Ignore the time that `for` loop iterations take.

(a) (1') Give out a function f satisfying that the running time of the algorithm is $\Theta(f(n))$. That is, $T(n) = \Theta($ n^3 $)$. (Give your answer in the most simplified form.)

(b) (3') For this function chosen f, show that the running time of the algorithm on an input of size n is upper bounded by $f(n)$. That is, prove $T(n) = O(f(n))$.

Solution:

$$T(n) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} \sum_{k=i}^{j} c \leq \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} c = cn^3 \Rightarrow T(n) = O(n^3)$$

(c) (4') For the same function f, show that the running time of the algorithm on an input of size n is also lower bounded by $f(n)$. That is, prove $T(n) = \Omega(f(n))$.

Solution: Method 1:

$$
\begin{aligned}
T(n) &= \sum_{i=1}^{n} \sum_{j=i+1}^{n} \sum_{k=i}^{j} c \\
&\geq \sum_{i=1}^{\frac{1}{3}n} \sum_{j=\frac{2}{3}n}^{n} \sum_{k=i}^{j} c \\
&\geq \sum_{i=1}^{\frac{1}{3}n} \sum_{j=\frac{2}{3}n}^{n} \sum_{k=\frac{1}{3}n}^{\frac{2}{3}n} c = \frac{c}{27} n^3 \Rightarrow T(n) = \Omega(n^3)
\end{aligned}
$$

Method 2: The outer loop goes through n times.

The total iterations of the inner loop can be expressed as:

$$(n-1)+(n-2)+\ldots+1+0 = \frac{1}{2}(n-1)(n+1-1) = \frac{1}{2}(n-1)n = \left(\frac{1}{2}n - \frac{1}{2}\right)n = \frac{1}{2}n^2 - \frac{1}{2}n.$$

The number of operations for adding during n iterations of the inner loop is calculated as follows:

- -1 operation for storing the result in $B[i, j]$.
- -1 operation to add entries $A[t]$ and $A[t+1]$ for each combination $A[t], A[t+1]$ from $A[i]$ through $A[j]$.

For each value of j:

- For $j = i + 1$, there is 1 operation.
- For $j = i + 2$, there are 2 operations.
- Continuing this way, for $j = n$, there are $n - i$ operations.

Calculating the number of addition operations for specific values of i:

- For $i = 1$:

$$\# \text{ of addition operations} = 1 + 2 + \ldots + (n-1) = \frac{1}{2}(n-1)n = \frac{1}{2}n^2 - \frac{1}{2}n.$$

- For $i = 2$:

$$\# \text{ of addition operations} = 1 + 2 + \ldots + (n-2) = \frac{1}{2}(n-2)(n-1).$$

- For $i = 3$:

$$\# \text{ of addition operations} = 1 + 2 + \ldots + (n-3) = \frac{1}{2}(n-3)(n-2).$$

- For $i = k$:

$$\# \text{ of addition operations} = 1 + 2 + \ldots + (n-k) = \frac{1}{2}(n-k)(n-k+1).$$

- For $i = n-2$:

$$\# \text{ of addition operations} = 1 + (n-(n-2)) = 1 + 2.$$

- For $i = \frac{n}{2}$:

$$\# \text{ of addition operations} = 1+2+\ldots+\left(n-\frac{n}{2}\right) = \frac{1}{2}\left(n-\frac{n}{2}\right)\left(n-\frac{n}{2}+1\right) \geq \frac{n^2}{8}$$

All $i \leq \frac{n}{2}$ have at least as many addition operations as $i = \frac{n}{2}$, hence they also have at least $\frac{n^2}{8}$ addition operations.
There are $\frac{n}{2}$ iterations where $i \leq \frac{n}{2}$, and each iteration has at least $\frac{n^2}{8}$ addition operations. Therefore, the total number of addition operations is:

$$\geq \left(\frac{n}{2}\right)\left(\frac{n^2}{8}\right) = \frac{n^3}{16}.$$

Thus, for all $n \geq 2$, we conclude:

$$T(n) \geq \frac{n^3}{16} \quad \Rightarrow \quad T(n) \geq \frac{1}{16}n^3 \text{ for all } n_0 > 1,$$

indicating that the algorithm is lower bound by n^3.

(d) (3') Although the algorithm you just analyzed is the most natural way to solve the problem—after all, it just iterates through the relevant entries of the array B, filling in a value for each—it contains some highly unnecessary sources of inefficiency. Give a different algorithm to solve the problem, with an asymptotically better running time. In other words, you should design another algorithm with running time $\Theta(g(n))$, where $\lim_{n\to\infty} \frac{g(n)}{f(n)} = 0$.
Just write the pseudocode below.

Solution:

```
1: for $i = 1$ to $n$ do
2:     $B[i,i] \leftarrow A[i]$
3:     for $j = i+1$ to $n$ do
4:         $B[i,j] \leftarrow B[i,j-1] + A[j]$
5:     end for
6: end for
```