Due: 1 pm on Friday, February 28

Assignment 1

1 Homework Guidelines

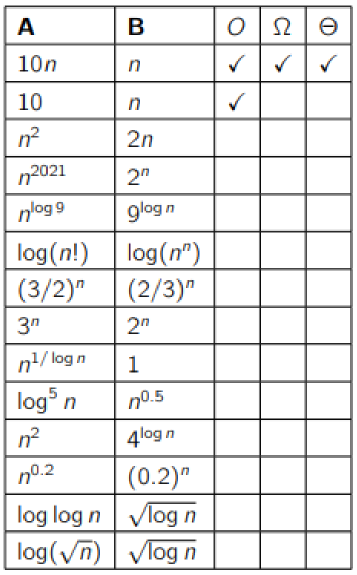
- Welcome to Algorithm Design and Analysis (Honor Track)!
- You can write your solutions in either English or Chinese with equal treatment. Just take it easy :)
- You are encouraged to discuss assignment problems with your peers, with the TAs, and with the course instructor. However, your solutions should be based on your own understanding and should be written **independently**. For each assignment, if you discussed the problems with other students in the class, you should include a list of the students' names.
- We encourage the use of LLMs to assist with learning, but we hope you use them to deepen your understanding of the material rather than relying on LLM-generated content to complete assignments without comprehension.
- Please remember to pay attention to the assignment deadline, as late submissions will NOT be accepted.

2 Textbook Exercises

1.19(1)(3)(5)(7), 1.21

3 Complexity Bounds

For each blank, indicate whether Ai is in O, Ω, or Θ of Bi. More than one space per row can be valid.



A	**B**	O	Ω	Θ
$10n$	n	✓	✓	✓
10	n	✓		
n^2	$2n$			
n^{2021}	2^n			
$n^{\log 9}$	$9^{\log n}$			
$\log(n!)$	$\log(n^n)$			
$(3/2)^n$	$(2/3)^n$			
3^n	2^n			
$n^{1/\log n}$	1			
$\log^5 n$	$n^{0.5}$			
n^2	$4^{\log n}$			
$n^{0.2}$	$(0.2)^n$			
$\log\log n$	$\sqrt{\log n}$			
$\log(\sqrt{n})$	$\sqrt{\log n}$			

4 Partial Sum of an 1D Array

Given an array `A` consisting of `n` integers `A[1],A[2],...,A[n]`. You want to obtain a 2D $n \times n$ array `B` where `B[i][j]` (for `i<j`) contains the sum of array entries `A[i]` through `A[j]`, i.e., `A[i]+A[i+1]+...+A[j]`. The value of array entry `B[i][j]` is left unspecified whenever `i⩾j`.

Here's a basic algorithm to solve this problem:

```
For $i = 1,\ 2, \ldots, n$
  For $j = i + 1,\ i + 2, \ldots, n$
    Add up array entries $A[i]$ through $A[j]$
    Store the result in $B[i, j]$
  Endfor
Endfor
```

(a) For some function f you may choose, give a bound of $O(f(n))$ on the runtime of this algorithm.

(b) For this same function f, prove that the runtime of the algorithm is also $\Omega(f(n))$ (This shows an asymptotically tight bound of $\Theta(f(n))$ on the runtime).

(c) Although the algorithm you analysis in (a)(b) is the most intuitive way to solve the problem, after all, it contains some highly unnecessary sources of inefficiency. Give a different algorithm to solve the problem, with an asymptotically better runtime. In other words, you should develop an algorithm with runtime $O(g(n))$, where $\lim_{n\to\infty} g(n)/f(n) = 0$.