Midterm

Graded

Student

Nick Zhu

Total Points

92 / 100 pts

Question 1

MCQ **40** / 40 pts

1.1 **MCQ1** **5** / 5 pts

✔ **– 0 pts** Correct

1.2 **MCQ2** **5** / 5 pts

✔ **– 0 pts** Correct

1.3 **MCQ3** **5** / 5 pts

✔ **– 0 pts** Correct

1.4 **MCQ4** **5** / 5 pts

✔ **– 0 pts** Correct

1.5 **MCQ5** **5** / 5 pts

✔ **– 0 pts** Correct

1.6 **MCQ6** **5** / 5 pts

✔ **– 0 pts** Correct

1.7 **MCQ7** **5** / 5 pts

✔ **– 0 pts** Correct

1.8 **MCQ8** **5** / 5 pts

✔ **– 0 pts** Correct

Question 2

Stock Trading **30** / 30 pts

2.1 **2(a)** **4** / 4 pts

✔ **– 0 pts** Correct

2.2 **2(b)** **8** / 8 pts

✔ **– 0 pts** Correct

2.3 **2(c)** **18** / 18 pts

✔ **– 0 pts** Correct

Question 3

Strategic Fishing **22** / 30 pts

3.1 **3(a)** **4** / 4 pts

✔ **– 0 pts** Correct

3.2 **3(b)** **18** / 20 pts

✔ **– 2 pts** Base Case incorrect

3.3 **3(c)** **0** / 6 pts

✔ **– 6 pts** No Answer

Basic Algorithms (Section 5) Midterm Exam
Instructor: Jiaxin Guan

March 12, 2025

1. Write your name and net ID below (and on the top of each detached page).
2. You have 75 minutes to complete this exam.
3. You may use a double-sided, letter-size "cheatsheet". The use of phones, computers, or other reference material during the exam is not permitted.
4. You may use any algorithm or theorem we saw in class (or homework) without proof, as long as you state it correctly. For all questions asking you to give algorithms, you do not have to give detailed pseudo-code, or even any pseudo-code. It is enough to give a clear description of your algorithm. You should additionally give a brief justification of the correctness and the claimed run time of your proposed algorithm.
5. Use the provided blue books to do any scratch work. The only work that will be graded is that written in the provided solution space on the exam. **Answers written on the back of the page will not be graded.** Both the blue books and midterms will be collected at the end of the exam.
6. Read each question before solving any and start with the problems you are most confident about solving.
7. This exam contains 12 pages (including this cover page). Good luck!!

Question	Points	Score
Multiple Choices	40	
Stock Trading	30	
Strategic Fishing	30	
Total:	100	

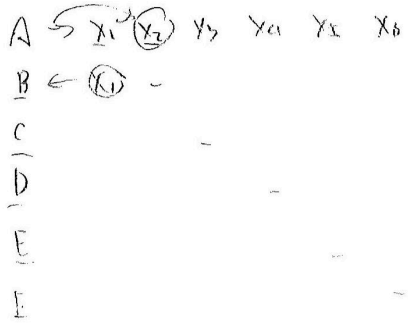
Write your name:	Write your Net ID (NYU email):
Nick Zhu	xz4687

1 Multiple Choices (40 points)

Choose the best answer for each of the questions below. No justification is needed.

1. Consider the stable matching problem with 6 students and 6 schools, where all students share the same preference list, and all schools share the same preference list. What is the maximum number of possible stable matchings in this case? .....(A)

   (A) 1.
   (B) 12.
   (C) 6.
   (D) 2.



2. Consider the following function: $3n^2+10n\log n+500\log n$

$$f(n) = 3n^2 + 10n\log n + 500\log(n^{10}).$$

   Which of the following asymptotic classifications of $f(n)$ is **incorrect**? .....(D)

   (A) $O(n^2)$ ✓
   (B) $\omega(\log(n^{10}))$ ✓
   (C) $o(n^2\log n)$ ✓
   (D) $\Theta(n\log n)$

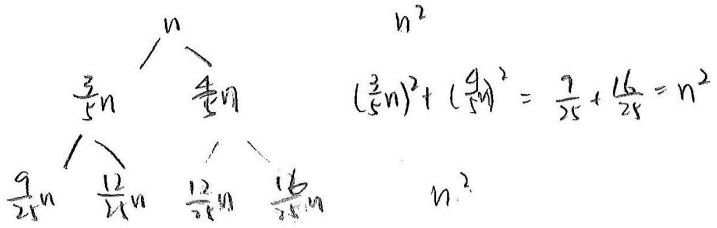
3. Let $f(n)$ and $g(n)$ be positive functions. Which of the following is **false**? ..... (C)

   (A) If $f(n) = O(g(n))$, then $f(n)\cdot g(n) = O(g(n)^2)$. ✓
   (B) If $f(n) = o(g(n))$, then $f(n)^2 = o(g(n)^2)$. ✓
   (C) If $f(n) = \Omega(g(n))$, then $2^{f(n)} = \Omega(2^{g(n)})$.
   (D) If $f(n) = \Theta(g(n))$, then $f(n)+g(n) = \Theta(g(n))$. ✓

4. Suppose we have $T(n) = T(3n/5) + T(4n/5) + n^2$ and $T(0) = T(1) = 1$, what is the asymptotic runtime of $T(n)$? ..... (A)

(A) $T(n) = \Theta(n^2 \log n)$
(B) $T(n) = \Theta(n)$
(C) $T(n) = \Theta(n^2 (\log n)^2)$
(D) $T(n) = \Theta(n \log n)$

$T = T(\frac{3}{5}n) + T(\frac{4}{5}n) + n^2$



5. Suppose we have $T(n) = 3T(n/4) + n^{3/2}$ and $T(0) = T(1) = 1$, what is the asymptotic runtime of $T(n)$? ..... (A)

(A) $T(n) = \Theta(n^{3/2})$
(B) $T(n) = \Theta(n^{3/2} \log n)$
(C) $T(n) = \Theta(n^{\log_4 3})$
(D) $T(n) = \Theta(n^2)$

$T(n) = 3T(\frac{n}{4}) + n^{\frac{3}{2}}$ $\log_4 3 < 1$

$a=3, b=4.$ $f(n) = n^{\frac{3}{2}}$

$Q = n^{\log_4 3}$ $3(\frac{n}{4})^{\frac{3}{2}} \leq$

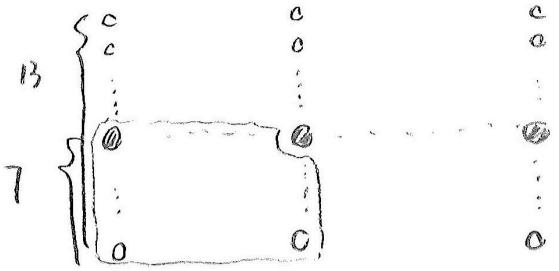
(A) $(n^{\frac{3}{2}})$

6. A system is processing real-time data logs, where a new data log arrives every second. Each data log is represented as an arbitrarily large integer, and the entire dataset must be continuously maintained in sorted order as new logs arrive. Which sorting algorithm would be the most efficient to run, in terms of asymptotics, when each new data log arrives? ..... (B)

(A) Counting Sort
(B) Insertion Sort
(C) Merge Sort
(D) Radix Sort

7. Recall that in the Deterministic Select algorithm, to compute the "approx-median" (element within the middle 40%), we break the elements into small sets of size 5, compute the median for each small set, and then output the median of these $n/5$ medians as the "approx-median". What if instead, we break into small sets of size 13? What is the smallest k such that the "approx-median" we get will be guaranteed to be in the middle k portion? ..... (A)

(A) 6/13
(B) 4/13
(C) 7/13
(D) 5/13



$7 \times \frac{n}{26}$ smaller $= \frac{7}{26}n$

$\frac{7}{26}n$ greater

$\frac{12}{26} = \frac{6}{13}$

$\frac{n}{13} \times \frac{1}{2} = \frac{n}{26}$

8. Suppose s is a string of length n, and we define a dynamic programming (DP) algorithm using the following recurrence:

$$\mathsf{DP}[i,j] = \begin{cases} 1 & \text{if } i = j, \\ \mathsf{DP}[i+1, j-1] + 2 & \text{if } i \neq j \text{ and } s[i] = s[j], \\ \max(\mathsf{DP}[i+1, j], \mathsf{DP}[i, j-1]) & \text{otherwise.} \end{cases}$$

What is the asymptotic runtime of computing $\mathsf{DP}[1, n]$ using a memoized or bottom-up version of this DP algorithm? ..... (D)

(A) $\Theta(n)$
(B) $\Theta(n \log n)$
(C) $\Theta(n^3)$
(D) $\Theta(n^2)$

# of subproblems

2 Stock Trading (30 points)

Imagine you're a stock trader looking at an **unsorted** array of stock prices for a single stock over different days. You want to maximize your profit by strategically buying on one day and selling on another, where the buying day occurs before the selling day.

Put formally, you are given an unsorted array A of n positive integers, your goal is to find a pair (i, j) with $1 \leq i < j \leq n$ that maximizes $A[j] - A[i]$.

(a) (4 points) Consider the following input $A = [8, 3, 7, 6, 2, 9, 5]$. What is the maximum profit you can make? And what are the corresponding choices of i and j?

(b) (8 points) In a few sentences, describe a brute-force algorithm that solves this problem in worst case $O(n^2)$ time. Very briefly justify the runtime. No need to justify the correctness.

(c) (18 points) Design an efficient deterministic algorithm that solves this problem in worst case $O(n \log n)$ time. Briefly justify the correctness and runtime of your proposed algorithm.
(*Hint: Consider splitting the array in two halves, and calculating the maximum profit you can get from each half. What is missing?*)

a) max profit = 7
correspondingly, I choose $i = 5$, $j = 6$, so that $A[j] - A[i] = 9 - 2 = 7$

b) first go through every i, then for every i, go through every j that $1 \leq i < j \leq n$, we denoted $A[j] - A[i]$ as m_{ij}, if we find a bigger m_{ij}, use it to replace the original one. At the end, we find the biggest $m_{ij} = A[j] - A[i]$.

runtime: we need to go through every i ($1 \leq i < n$), that is $n-1$ time, and also look at every j for every i, so we have runtime = $O(n^2)$

(c) on next page

Use this page if you need additional space for your solution to Problem 2.

c) $X = [x_1, \dots x_n]$ $O(n \log n)$

split the array in half. find the max profit from each half

X < $[x_1 \dots x_{n/2}]$ <
$[x_{n/2} \dots x_n]$ <

what is missing?

⇒ we get the max profit when one in the left array and one in the right array.

- we use the idea of merge sort, before doing the merge sort. we ~~need~~ first need to memorize all the ~~place of~~ index of the original array.

after using merge sort to sort the array, we have

we take $X[1]$ and $X[n]$

if the index of $X[1]$ is smaller than $X[n]$ in the original array, then we output it

if the index of $X[1]$ is greater than $X[n]$ in the original array, we move forward to check $(X[2], X[n])$ and $(X[1], X[n-1])$

runtime: merge sort takes $O(n \log n)$
and checking the max/min takes n ⇒ runtime = $O(n \log n)$

Use this page if you need additional space for your solution to Problem 2.

Use this page if you need additional space for your solution to Problem 2.

3 Strategic Fishing (30 points)

Your trading strategy in problem 2 turned out to be a huge success, so you retire early to enjoy your life by the sea. You grow a passion for fishing, but given your expertise, you want to also do it strategically. You start by producing estimates of the amount of fishing you can harvest each day for the following n days, and then you want to decide which of these days you should go fishing. However, to keep things relaxed, you impose a simple constraint: **you never fish on two consecutive days**, i.e., if you go fishing on day i, you must rest on both day $i-1$ and day $i+1$.

Given an array of n integers $X = [x_1, x_2, \ldots, x_n]$, where x_i represents the amount of fish you can catch on day i, determine the maximum amount of fish you can harvest over these n days while respecting the no-consecutive-fishing constraint. Notice that you only need to find the max possible amount of fish, not the detailed fishing plan though.

For example, for $X = [3, 9, 7, 5, 1]$, by fishing on day 2 and 4, you can get the max amount of fish of $x_2 + x_4 = 9+5 = 14$. Although fishing on day 2 and 3 gives $x_2 + x_3 = 9+7 = 16 > 14$, this is not allowed because it involves consecutive fishing days.

(a) (4 points) Find the maximum amount of fish you can catch for $X = [5, 2, 7, 8, 1, 4, 3]$ while following the no-consecutive-fishing rule.

(b) (20 points) Design an efficient deterministic algorithm that solves this problem in worst case $O(n)$ time. Briefly justify the correctness and runtime of your proposed algorithm. (*Hint: Use a 1-dimensional DP.*)

(c) (6 points) Suppose you become even more passionate about fishing and decide to relax the constraint. Instead of avoiding **two** consecutive fishing days, you now only need to avoid **three** consecutive fishing days. In other words, in any three-day window, you can fish at most twice.

Modify the subproblem formulation and the recurrence in part (b) to accommodate this new constraint while still achieving an $O(n)$ runtime. Very briefly justify the runtime. No need for justification of correctness.

a) maximum amount 17 (we fish on first, fourth, and sixth day getting 5+8+4=17)

(b) on next page.

Use this page if you need additional space for your solution to Problem 3.

b) ① Subproblems:

for $i = 0, 1, \dots, n$, let $M[i]$ be the max sum of fish we can get from no-consecutive days $X = [x_1, \dots x_i]$

The solution is given by $M[n]$.

② Guess: if we are fishing on the ith day, we want to guess whether fishing on the $(i-1)$th day or $(i-2)$th day (we can only choose one)

③ Recurrence:

$$M[i] = \begin{cases} 0 & \text{if } i = 0 \\ \max(M[i-1], x_i + M[i-2]) & \text{if } i \geq 1 \end{cases}$$

④ we use memoization / buttom up ☺

Correctness:

a) Optimal structure:

consider the $X = [x_1, \dots x_i]$

first condition → 1° if we want to include x_i in the sum $M[i]$ we cannot take x_{i-1}, the we look at the max we can get from x_1 to x_{i-2}, i.e $M[i-2]$ in this case $M[i] = x_i + M[i-2]$

2° however, if we donot include x_i in the sum $M[i]$ we can take x_{i-1}, thus we have $M[i] = M[i-1]$

this optimal structure include all the possible cases

b) Correct Base Case

$M[i] = 0$ if $i = 0$ ⇒ means there is no fish we can get when the array is empty. ☑

Use this page if you need additional space for your solution to Problem 3.

b) continued

runtime:

subproblems: n
time per subproblem: $O(1)$ $\Rightarrow$ $n \times O(1) = O(n)$

c) 

Use this page if you need additional space for your solution to Problem 3.