Exam 1
CS-GY 6033 INET SPRING 2025
March 3rd 2025

Instructions:

Scheduling:

- The exam is runs from 6:00pm to 9:00pm on March 3rd 2025. The exam will be available to download from Gradescope at 5:50pm. The exam is to be completed by 9:00pm. Gradescope stops accepting all uploads at 9:15pm. There are absolutely no late uploads accepted by the system after that time. The time to complete the exam will depend on your preparation: a prepared student could finish it in less than two hours, an ill-prepared student might take up to three hours. It is your responsibility to allow time for scanning and uploading your exam within the deadlines.

Format:

- The written exam consists of a total of 90 points. The oral exam is worth 10 points of your Exam.
- You may write your solutions directly on the downloaded exam paper, *or* in your own format. You are responsible for providing clear and legible solutions to the problems. Your exam must be resubmitted into Gradescope electronically. Ensure that you know how to digitally scan any handwritten material. This is entirely the student's responsibility.

Questions during the exam:

- There is a ZOOM session for questions that will be open during the entire course of the exam, with MICROPHONES OFF. You may ask questions to the instructor with private chat during the exam. Any announcements made by the instructor during the exam will be made over ZOOM and also be email. **It is the student's responsibility to stay connected (either by ZOOM or email) during the exam.**

Rules:

- This exam is a **take-home exam**. You may use **only** the resources from the online class (course notes, video slides, problem solutions) and any type of calculator (although it is not needed). You may not copy or reference **any** textbook.
- Do not copy from personal notes, shared notes, or notes you have copied from any outside source.
- Your work must be entirely your own. It is **forbidden to discuss any work with *any* other person**. Furthermore, your work must be done without using internet searches (although this is completely unhelpful for this exam). Any breach of academic honesty will be handled in accordance with the *Student Code of Conduct*, (a copy of which is provided), and in this particular case, taken very seriously.
- You are asked to **read** the attached Student Code of Conduct Section III subsections A,B,C,D,E and **sign** below to acknowledge that you aware of the policy. Once signed, a copy of this page must be uploaded with your exam.

I acknowledge that my submitted Exam work is entirely my own. I have read and am in accordance with the Student Code of Conduct policy of NYU Tandon and fully accept the consequences of breaching the above instructions.

Name: Hongdao Meng

Signature: Hongdao Meng

Hongdao Meng

Q1. (a) $T(n) = \sqrt{n^3+1} + n\log n$

(a)

$\because T(n) = \sqrt{n^3+1} + n\log n \le n^{\frac{3}{2}} + n^{\frac{3}{2}} \le c \cdot n^{\frac{3}{2}}$

$\therefore T(n)$ is $O(n^{\frac{3}{2}})$ which is the upper limit $\therefore \Theta(n^{1.5})$

$\therefore$ Answer: $O(n^{1.5})$, ~~$O(n\log n)$~~ $\Omega(n\log n)$, $\Omega(n)$, $O(n^2)$, $\Omega(\log n)$ are True.

(b). $T[0, \cdots, 10]$

$h(k,i) = k^2 + 2i + 3i^2 \mod 11$

size is 11.

Answer: Yes, it's possible that we need to $i = 12$. Because Probe11 doesn't means access all 11 slots. So we need $i = 12$ or maybe bigger i.

(c). $T[0, 2, \cdots, 9]$ $h_1(k) = k \mod 10$ $h_2(k) = k \mod 10$.

$k = 1, 2, 3, 4, 5$

$k=1$	$h_2(1)=1$	$\gcd(10,1)=1$	$[0, \cdots, 9]$
$k=2$	$h_2(2)=2$	$\gcd(10,2)=2\neq 1$	$[2,4,6,8,0]$
$k=3$	$h_2(3)=3$	$\gcd(10,3)=1$	$[0, \cdots, 9]$
$k=4$	$h_2(4)=4$	$\gcd(10,4)=2$	
$k=5$	$h_2(5)=5$	$\gcd(10,5)=5$	$[5,0,5,0]$

when $k = 1, 3$ Probe can traverse allslots

when $k = 2, 4, 5$ Can't cover the entire table.

(d). size 25 ; key 100 ; uniform hash

Excepeted length of chain: $100 \div 25 = 4$

$\because$ uniform $\therefore$ Excepted length = 4

(E) $T(n) = 4T(2n/3) + n^3$ $a = 4$ $b = \frac{3}{2}$ $f(n) = n^3$

$k = \log_b a = \log_{\frac{3}{2}}(4) \approx 3.42$ $n^k = n^{3.42} > f(n)$

$\therefore T(n) \simeq \Theta(n^{3.42})$ $T(n)$ is $\Theta(n^{\log_{\frac{3}{2}} 4})$

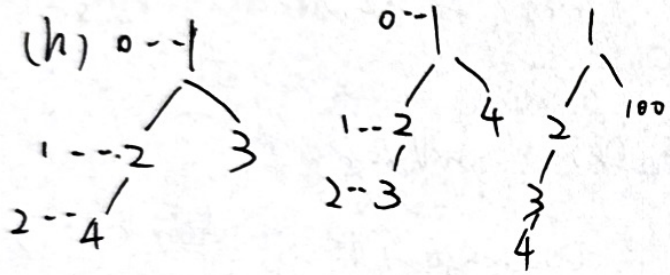
Q1.(f) q is mid.

Yes. the ~~left~~ subarray ~~use~~ is sorted by recursion, the right subarray is sorted by insertion sort. And then complete order can be obtained by merging two ordered array.

(g) ∵ uniform

$\therefore P = \frac{3}{10} = 0.3$

(h)



The fourth largest element could be in first ~~or~~ second level. or third

(j) Best-case : $\Theta(n)$ when first Pivot is ranked kth.

(k) Bubble-sort, Partition, Randomized-Sort, Select have a best-case Runtime of $O(n)$

Q2.

$3^{\log_3 n}$ is $\theta(n)$

$\log(2n^{50})$ is $\theta(\log n)$

$\sqrt{\log n + 100^{100}}$ is $\theta(\sqrt{\log n})$

$(\sqrt{n}+1)^2(\log n + 5)$ is $\theta(n \log n)$

$\frac{1+\log n + n^2}{n^2 + 6\log n}$ is $\theta(n)$

~~log~~ $\frac{1+20n^2}{n^2+6\log n}$ is $\theta(1)$

$(n\log n + 2^{10})^2$ is $\theta(n^2 \log^2 n)$

$\log(5+2^n)$ is $\theta(n)$

$\frac{3^n + n}{n+2^n}$ is $\theta(3^n)$

2^{3n+6} is $\theta(2^{3n})$

3^{2n+6} is $\theta(3^{2n})$

$n!$ is $\theta(n!)$

n^n is $\theta(n^n)$

Answer:

final sorted:

~~$\frac{1+\log n+n^2}{n^2+6\log n}$ $\theta(n)$~~

$\frac{1+20n^2}{n^2+6\log n}$	$\theta(1)$
$\sqrt{\log n + 100^{100}}$	$\theta(\sqrt{\log n})$
$\log(2n^{50})$	$\theta(\log n)$
$\log(5+2^n)$	$\theta(n)$
$3^{\log_3 n}$	$\theta(n)$
$\frac{1+\log n+n^2}{n+6\log n}$	$\theta(n)$
$(\sqrt{n}+1)^2(\log n+5)$	$\theta(n\log n)$
$(n\log n+2^{10})^2$	$\theta(n^2\log^2 n)$
$\frac{3^n+n}{n+2^n}$	$\theta(3^n)$
2^{3n+6}	$\theta(2^{3n})$
3^{2n+6}	$\theta(3^{2n})$
$n!$	$\theta(n!)$
n^n	$\theta(n^n)$

Q3. $f(n) = \frac{3n^3 + (\log n)^2}{n^2 - n} + 1000 = \Theta(n) + 1000 = \Theta(n)$

$\therefore g(n)$

$c_1 n \le f(n) \le c_2 n$ $\quad$ $c_1 n \le \frac{3n^3 + (\log n)^2}{n^2 - n} + 1000 \le c_2 n$

when $n \ge 1000$: $3n \le f(n) \le 8n + 1000 \le 9n$

$\therefore c_1 = 3 \quad c_2 = 9$

$\therefore f(n) \in \Theta(n)$ $\quad g(n) = n$ as long as $n \ge 1000$.

Q4. q is random

The Algo isn't correct. It may cause infinite recursion.

For example. $A[3, 1]$ $\xrightarrow{\text{SlowSort}(A,0,1)}$ $q = 1$ $\longrightarrow$ SlowSort(A, 0, 1)

If q always 1, the Algo will infinite recursion.

Best-case: q is mid. $T(n) = 2T(n/2) + O(n)$

$a = 2 \quad b = 2 \quad k = \log_2 2 = 1 \quad n^k = n$

$\therefore T(n)$ is $\Theta(n \log n)$

Worst-Case: $q = s$ or $q = f$ $\quad T(n) = T(n-1) + O(n)$

$T(n-1) = T(n-2) + O(n)$

$T(n)$ is $\Theta(n^2)$

Q5 A = [3, 2, 4, 6, 1, 9, 8, 5, 7]

1. [1, 2, 4, 6, 3, 9, 8, 5, 7]

(1, 2, 4, 6) is sorted ∴ Selection Sort

2. [2, 3, 4, 6, 1, 9, 8, 5, 7]

∵ 2 swap with 3, 4 and 6 is original place

∴ Insertion Sort

3. [3, 2, 4, 6, 1, 5, 7, 9, 8]

∵ [9, 8] bigger than 7, left subarray smaller than 7

∴ Quick Sort

4. [2, 3, 4, 1, 6, 8, 5, 7, 9]

∵ 9 The biggest is in the end

∴ Bubble Sort

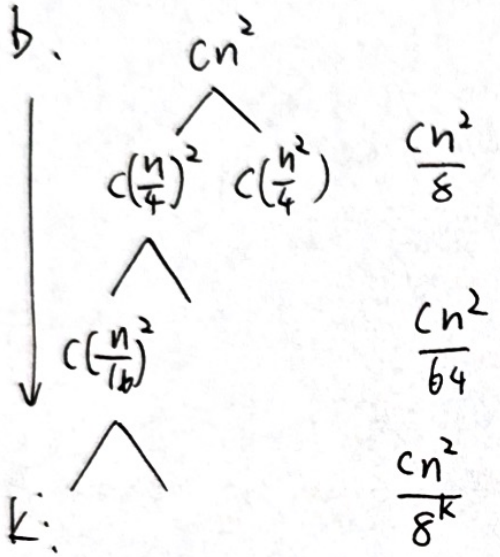
Q6. a.

~~$T(n) = T(n/2) + T(n/4) + O(n^2)$~~

$T(n) = 2T(n/4) + O(n^2)$ as long as $n = 7 - 5 + 1 \geq 5$

$T(n) = O(1)$ as long as $n < 5$

b.



$T(n) = \sum_{k=1}^{\log_4 n} \frac{cn^2}{8^k} = cn^2 \cdot \frac{1}{1 - \frac{1}{8}} = \frac{8}{7} c n^2$

$\therefore T(n)$ is ~~$\Theta(n^2)$~~ $O(n^2)$

c. $T(n) = 2T(\frac{n}{4}) + O(n^2)$ is $O(n^2)$ $\quad T(n) \leq cn^2$

$T(\frac{n}{4}) = 2T(\frac{n}{16}) + O(\frac{n^2}{16}) \leq c(\frac{n^2}{16})$

$T(n) \leq c \cdot n^2$

$T(\frac{n}{4}) \leq c \cdot (\frac{n}{4})^2$ $\quad \therefore T(n) \leq 2 \cdot c \cdot (\frac{n^2}{4}) + dn^2$

$\leq \frac{2c}{16} n^2 + dn^2$

$\leq (\frac{c}{8} + d) n^2$

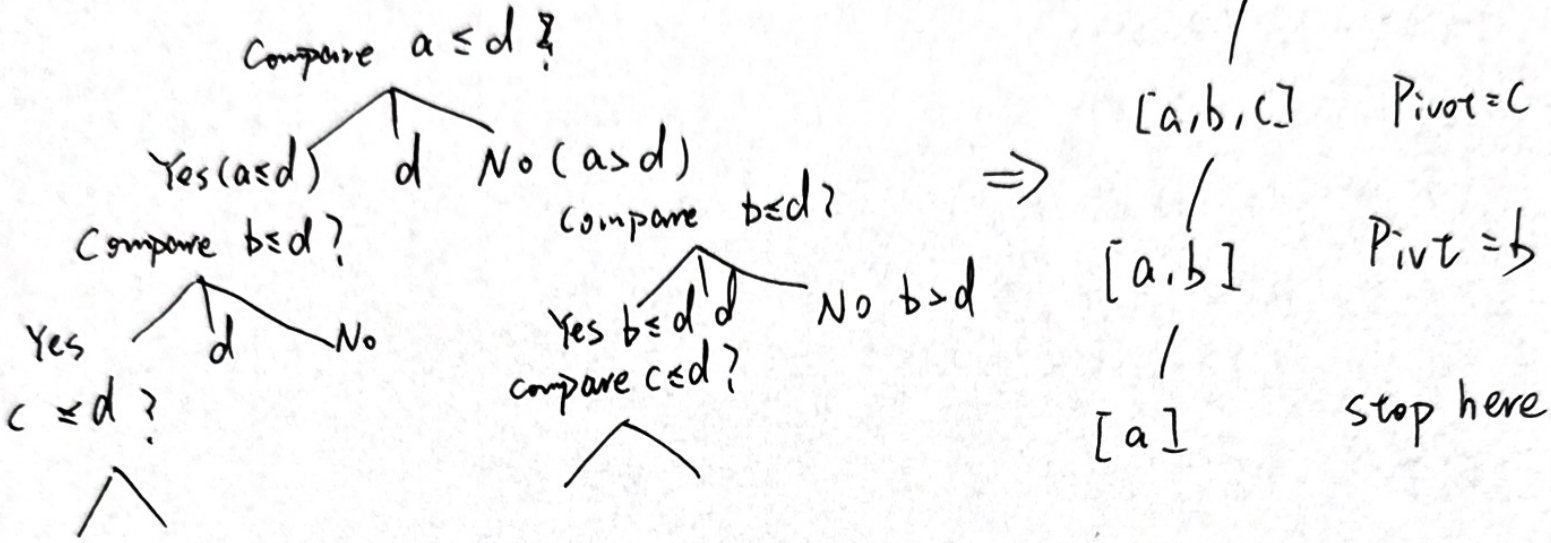
when: $\frac{c}{8} + d \leq c$

$c \geq \frac{8}{7} d$ and $n \geq 5$

$T(n) \leq c n^2$ $\quad T(n)$ is $O(n^2)$

Q7. Input = [a, b, c, d]

Use d Pivot = d



Height = $O(n)$

Worst-case : $O(n^2)$

Average-case : $O(n \log n)$

Q8 $T[0, \cdots, 12]$

$h(k,i) = k + i + 4i \mod 13$ key: 3, 23, 17, 27, 9, 19, 6, 16, 26

3: $h(3,0) = 3$

23: $h(23,0) = 10$

17: $h(17,0) = 4$

27: $h(27,0) = 1$

9: $h(9,0) = 9$

19: $h(19,0) = 6$

6: $h(6,0) = 6$ Collision $h(6,1) = 11$

16: $h(16,0) = 3$ Collision $h(16,1) = 8$

26: $h(26,0) = 0$

$\therefore T[0] = 26$, $T[1] = 27$, $T[3] = 3$, $T[4] = 17$, $T[6] = 19$, $T[8] = 16$

$T[9] = 9$, $T[10] = 23$, $T[11] = 6$,

Q9.

Best case: q is the position of k. And we need only 1 time comparision.

Worst case: 2 comparisons for each recursive call

Each partition only can exclude one element.

Last recursion need one time comperation.

∴ Time = $2n-1$

Q10. Make-Heap(A, i) is Correct

Because this Algo calls recursively from the leaf node to the root node. The completed heap structure is guaranteed every time the Bubble-up.

Make-Heap2(A, i) is Fault.

Because the Recursive calls from root to leaf may destory the parent node structure due to subsequent adjustments, and the overall heap property can't be guaranteed.

Q11. FixHeap(A, i):

```
If i > A.heapsize:
    return
left = 2·i
right = 2·i+1
largest = i
If left <= A.heapsize and A[left] > A[largest]:
    largest = left
If right <= A.heapsize and A[right] > A[largest]:
    largest = right
If largest != i:
    swap(A[i], A[largest])
    FixHeap(A, largest)
FixHeap(A, left)
FixHeap(A, right)
```

For each node is only visited one time.

∵ we have n nodes

∴ T(n) is O(n)

Q12 A: binary B: $\frac{1}{k}$ n

① Radix Sort

A: $d = \log_2 n$ $r = 2$; Convert each number to binary represent.

step 1. $d = \log_2 n$ $r = 2$

step 2. using counting sort to sort by digit → $O(n)$

Step 3. repeat step 2 by d times → ~~$O(d \log n)$~~ $d = \log n$

∴ Runtime is $d \cdot O(n) = O(n \log n)$

B. pre-processing: 1. Convert each fraction to three decimal(place.

eg. ~~999~~ $\frac{1}{1000} = 0.001$

2. Each decimal time 1000 convert to integer eg. $0.001 \times 1000 = 1$

step 1. $d = 3$ $r = 1/10$

step 2. Using counting sort to sort by digit → $O(n)$

step 3. Repeat step 2 by $d = 3$ times

∴ Runtime is $d \cdot O(n) = O(n)$

② Bucket Sort (uniform)

Pre-processing: Creat n buckets, each bucket for one natural number.

Step 1: Traverse array and put elements into its corresponding Bucket (element k for bucket k)

Step 2. Collects all elements in non-empty buckets in order

Runtime is $n \cdot O(1) = O(n)$

Q13.

```
Find_mid (A):
    lower = QuickSort(A, 0, n-1, n//4)
    upper = Quicksort(A, 0, n-1, 3*n//4)
    Return [x for x in A if lower ≤ x ≤ upper]

QuickSort (A, left, right, k)
    if left = right
        return A[left]
    Pivot = (left + right)//2
    Pivot = partition(A, left, right, Pivot)

    if k == pivot:
        return A[k]
    elif k < Pivot:
        return QuickSort(A, left, Pivot-1, k)
    else:
        return QuickSort(A, Pivot+1, right, k)

Partition (A, left, right, Pivot)
    Pivot_val = A[Pivot]
    Swap (A[Pivot, A[right])      Previous = left
    For i from left to right:
        If A[i] < Pivot_val:
            swap (A[i], A[Previous])
            Previous += 1
    swap(A[right], A[Previous])

    Return Previous
```

Runtime:
$O(n) + O(n) + O(n)$
$= O(n)$

Q14. A=[8,1,4,6,3,2] Pivot is last

Step 1. QuickSort(A, 0, 5) Pivot = 2 left right

result: [1,2,4,6,3,8] Recursive call (A,0,0) and (A,2,5)

Step 2. QuickSort(A,0,0) Pivot=1

result [1,2,4,6,3,8] No Call

Step 3. Quick ~~Sort~~ (A,2,5) Pivot=8 [4,6,3,8] invalid

result: [4,6,3,8] left (A,2,4) right (A,6,5)

Step 4. QuickSort (A, 2,4) Pivot=3 [4,6,3] invalid

result: ~~[3,4,6]~~ [3,6,4] left (A,3,1) right (A,3,4)

→ [1,2,3,6,4,8]

Step 5. QuickSort(A,3,4) Pivot=4 [6,4]

result: [4,6] → [1,2,3,4,6,8]