Mark Cantuba MJC862 11214496

1) a) Most Quickly Growing Term → $n^4 \log_{280} n$ → $\boxed{O(n^4 \log n)}$

b) Quickly Growing Term → $0.4n^4$ → $\boxed{O(n^4)}$

c) Quickly Growing Term → $2n \log_2 n$ = $\boxed{n \log_2 n}$

2) a) i) False
ii) False
iii) True
iv) True

b) $\Theta(n^3)$

3) a) $O(n^2) + O(\log n) + O(n \log n)$
$= O(n^2 + \log n) + O(n \log n)$
$= O(2 \cdot \max(n^2 + \log n)) + O(n \log n)$
$= O(n^2) + O(n \log n)$
$= O(n^2 + n \log n)$
$= O(2 \cdot \max(n^2, n \log n))$
$\boxed{= O(n^2)}$

b) $O(2^n) * O(n^2)$
$\boxed{= O(2^n \cdot n^2)}$

c) $420(n \log n) + 180(n^3)$
$= O(n \log n) + O(n^3)$
$= O(n \log n + n^3)$
$= O(2 \cdot \max(n \log n, n^3))$
$= \boxed{O(n^3)}$

d) $O(n \log_2 n^2) + O(m)$
$= O(2n \log_2 n) + O(m)$
$= O(n \log_2 n) + O(m)$
$= O(n \log_2 n + m)$
$= O(2 \cdot \max(n \log_2 n, m))$
$\boxed{= O(\max(n \log_2 n, m))}$

1, 2, 3, 4, 5 ... n
1, 2, 3, 4, 5 ... n

4) a) Inner for loop count → # of statements = 2
# of iterations = n
n-1 1, 2, 3, 4, 5 ... n-1, n
= 2n+1 (including false condition

Outer loop → # of inner loop statements = 1 + 1 = 2 (including false)
# of iterations = n times

$= 2n(n+2) + 2$
$= 2n^2 + 4n + 2 = O(n^2)$

b) $\Theta(n^2) = \Theta(f(n))$

5) a) Inner loop statement count = 2
Inner loop iteration count → i+1, i+1+1, i+1+2 ... i+1+n-1
= i+n
↳ i+n times
Cost = 2n+i+1 (including the false condition)

# of outer loop iteration → 0, 1, 2 ... n-1
= n-1
# of statements → 2 (including false statement)
total cost = 2n+1

overall $= i + 2n(2n+1) + i + 1 = 4n^2 + 2n + i + 1$
$= O(n^2)$

b) $\Theta(f(n)) = \Theta(n^2)$

Mark Cantuba

(6) active operation = print statement inside inner for loop

number of times executed → $n-1$ times = $O(n)$

each time the loop is executed, inner loop iterates $n+i-1$ times

The outer loop is executed $n-1$ times. = $O(n$

The active operation (print statement) runs $n+i-1$ times.

∴ The time complexity of this of this operation is

$O(n \cdot n) = \boxed{O(n^2)}$

4) Worst Case: when the array is at end of list and number is end of list
data.getFirst() = 1

Active operation → Statement inside while statement
# of iterations = n+1 (including false condition)

Tcount(n) = n+1

Active Operation → found = binarySearch(data.currentItem(), target)
Cost of Binary Search = log m
Worst Case; when number is at end of the list

Total Cost of Binary Search = log m

Inside loop → Binary Search excecuted n+1 times

Total Cost n+1·(log m)

= n log m

$$= O(n \cdot \log m) \quad \text{(worst case)}$$

Mark Contuba MJC862 11/24/96

Name: PriorityQueue<G>

Sets:

G: set of items that can be stored inside the PriorityQueue
P: set of PriorityQueues that contains items from G
B: set of Boolean containing true and false
Mo: set of non negative integers. For Capacity
No: set of non-negative integers for Priority
Fo: Set of non-negative integers for Frequency

Signatures

new PriorityQueue<G>(n): N ↛ P
P.insert(g,m): G, M ↛ P
P.isEmpty: → B
P.isFull: → B
P.maxItem: ↛ G
P.minItem: ↛ G
P.deleteMax: ↛ P
P.deleteMin: ↛ P
P.deleteAllMax ↛ P
P.frequency(g): G → F

Pre-Conditions: For all g∈G. D∈P, n∈No, m∈Mo

new PriorityQueue<G>(mo): mo > 0
p.insert(g,n): p is not full
p.isEmpty: None
p.isFull: None
p.maxItem: p is not empty
p.minItem: p is not empty
p.deleteMax: p is not empty
p.deleteAllMax: p is not empty
p.deleteMin: p is not empty
p.frequency(g): None

Semantics: For $g \in G$, $p \in P$, $m \in M_0$, $n \in N_0$, $f \in F_0$

newPriorityQueue<G>(m): create PriorityQueue p, that contains items from G, with capacity m

p.insert(g, n): enqueues item g into back of the queue, with its matching priority number n

p.isEmpty: Returns true if p is empty, otherwise returns false

p.isfull: Returns true if p is full, otherwise false

p.maxItem: Get the item from p that holds the highest priority

p.minItem: Get the item from p, that holds the lowest priority.

p.deleteMax: deletes the item in the queue that holds highest priority

p.deleteAllMax: deletes all the items that holds highest priority (all items that holds max priority # will also be deleted)

p.deleteMin: deletes an item that holds lowest priority number

p.frequency: Obtains frequncy f, determining how many times g occured. Counts frequencys of item g in the queue