CS 240 Spring 2023 Midterm Reference Sheet

Order Notation Summary

O-notation: $f(n) \in O(g(n))$ if there exist constants $c > 0$ and $n_0 \geq 0$ such that $|f(n)| \leq c\,|g(n)|$ for all $n \geq n_0$.

Ω-notation: $f(n) \in \Omega(g(n))$ if there exist constants $c > 0$ and $n_0 \geq 0$ such that $c\,|g(n)| \leq |f(n)|$ for all $n \geq n_0$.

Θ-notation: $f(n) \in \Theta(g(n))$ if there exist constants $c_1, c_2 > 0$ and $n_0 \geq 0$ such that $c_1\,|g(n)| \leq |f(n)| \leq c_2\,|g(n)|$ for all $n \geq n_0$.

o-notation: $f(n) \in o(g(n))$ if for all constants $c > 0$, there exists a constant $n_0 \geq 0$ such that $|f(n)| \leq c\,|g(n)|$ for all $n \geq n_0$.

ω-notation: $f(n) \in \omega(g(n))$ if for all constants $c > 0$, there exists a constant $n_0 \geq 0$ such that $c\,|g(n)| \leq |f(n)|$ for all $n \geq n_0$.

Techniques for Order Notation

Suppose that $f(n) > 0$ and $g(n) > 0$ for all $n \geq n_0$. Suppose that

$$L = \lim_{n \to \infty} \frac{f(n)}{g(n)} \quad \text{(in particular, the limit exists).}$$

Then

$$f(n) \in \begin{cases} o(g(n)) & \text{if } L = 0 \\ \Theta(g(n)) & \text{if } 0 < L < \infty \\ \omega(g(n)) & \text{if } L = \infty. \end{cases}$$

Note that this result gives *sufficient* (but not necessary) conditions for the stated conclusion to hold.

Useful Sums

Arithmetic sequence:

$\sum_{i=0}^{n-1} i = ???$ $\qquad \sum_{i=0}^{n-1}(a + di) = na + \frac{dn(n-1)}{2} \in \Theta(n^2)$ if $d \neq 0$.

Geometric sequence:

$$\sum_{i=0}^{n-1} 2^i = ??? \qquad \sum_{i=0}^{n-1} a\,r^i = \begin{cases} a\dfrac{r^n - 1}{r - 1} & \in \Theta(r^{n-1}) & \text{if } r > 1 \\ na & \in \Theta(n) & \text{if } r = 1 \\ a\dfrac{1 - r^n}{1 - r} & \in \Theta(1) & \text{if } 0 < r < 1. \end{cases}$$

Harmonic sequence:

$\sum_{i=1}^{n} \frac{1}{i} = ???$ $\qquad H_n := \sum_{i=1}^{n} \frac{1}{i} = \ln n + \gamma + o(1) \in \Theta(\log n)$

A few more:

$\sum_{i=1}^{n} \frac{1}{i^2} = ???$ $\qquad \sum_{i=1}^{n} \frac{1}{i^2} = \frac{\pi^2}{6} \in \Theta(1)$

$\sum_{i=1}^{n} i^k = ???$ $\qquad \sum_{i=1}^{n} i^k \in \Theta(n^{k+1})$ for $k \geq 0$

Useful Math Facts

Logarithms:

- $c = \log_b(a)$ means $b^c = a$. e.g. $n = 2^{\log n}$.
- $\log(a)$ (in this course) means $\log_2(a)$
- $\log(a \cdot c) = \log(a) + \log(c)$, $\log(a^c) = c\log(a)$, $\log x \leq x$
- $\log_b(a) = \frac{\log_c a}{\log_c b} = \frac{1}{\log_a(b)}$, $a^{\log_b c} = c^{\log_b a}$
- $\ln(x) = \text{natural log} = \log_e(x)$, $\frac{\mathrm{d}}{\mathrm{d}x}\ln x = \frac{1}{x}$

Factorial:

- $n! := n(n-1)(n-2)\cdots 2 \cdot 1 = \#$ ways to permute n elements
- $\log(n!) = \log n + \log(n-1) + \cdots + \log 2 + \log 1 \in \Theta(n \log n)$

Probability

- $E[X]$ is the expected value of X.
- $E[aX] = aE[X]$, $E[X + Y] = E[X] + E[Y]$ (linearity of expectation)

Please initial:

CS 240 Spring 2023 Midterm Reference Sheet

Some Recurrence Relations

Recursion	resolves to	example
$T(n) = T(n/2) + \Theta(1)$	$T(n) \in \Theta(\log n)$	Binary search
$T(n) = 2T(n/2) + \Theta(n)$	$T(n) \in \Theta(n \log n)$	Mergesort
$T(n) = 2T(n/2) + \Theta(\log n)$	$T(n) \in \Theta(n)$	Heapify (*)
$T(n) = T(cn) + \Theta(n)$ for some $0 < c < 1$	$T(n) \in \Theta(n)$	Selection (*)
$T(n) = 2T(n/4) + \Theta(1)$	$T(n) \in \Theta(\sqrt{n})$	Range Search (*)
$T(n) = T(\sqrt{n}) + \Theta(\sqrt{n})$	$T(n) \in \Theta(\sqrt{n})$	Interpol. Search (*)
$T(n) = T(\sqrt{n}) + \Theta(1)$	$T(n) \in \Theta(\log \log n)$	Interpol. Search (*)

- Once you know the result, it is (usually) easy to prove by induction.
- Many more recursions, and some methods to find the result, in CS341.

(*) These will be studied later in the course.

Rafiee Sevyeri, Schost (CS-UW) CS240 – Module 1 Spring 2023 42 / 44

Efficient sorting with heaps

- Idea: *PQ-sort* with heaps.
- $O(1)$ auxiliary space: Use same input-array A for storing heap.

```
HeapSort(A, n)
1.    // heapify
2.    n ← A.size()
3.    for i ← parent(last()) downto 0 do
4.        fix-down(A, i, n)
5.    // repeatedly find maximum
6.    while n > 1
7.        // 'delete' maximum by moving to end and decreasing n
8.        swap items at A[root()] and A[last()]
9.        decrease n
10.       fix-down(A, root(), n)
```

The for-loop takes $\Theta(n)$ time and the while-loop takes $O(n \log n)$ time.

Rafiee Sevyeri, Schost (CS-UW) CS240 – Module 2 Spring 2023 22 / 25

LSD-Radix-Sort

```
LSD-radix-sort(A)
A: array of size n, contains m-digit radix-R numbers
1.    for d ← least significant to most significant digit do
2.        Bucket-sort(A, d)
```

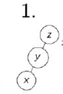
12③		2③0		①01		021
23⓪		3②0		②10		101
02①	$(d=3)$	2①0	$(d=2)$	③20	$(d=1)$	123
32⓪	$\Longrightarrow$	0②1	$\Longrightarrow$	⓪21	$\Longrightarrow$	210
21⓪		1⓪1		①23		230
23②		2③2		②30		232
10①		1②3		②32		320

- Loop-invariant: A is sorted w.r.t. digits $d, \ldots, m$ of each entry.
- **Time cost**: $\Theta(m(n+R))$ **Auxiliary space**: $\Theta(n+R)$

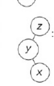
Rafiee Sevyeri, Schost (CS-UW) CS240 – Module 3 Spring 2023 40 / 41

Fixing a slightly-unbalanced AVL tree

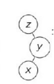
restructure(x, y, z)
node x has parent y and grandparent z
1. **case**



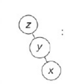
: // Right rotation
return *rotate-right*(z)



: // Double-right rotation
z.left ← *rotate-left*(y)
return *rotate-right*(z)



: // Double-left rotation
z.right ← *rotate-right*(y)
return *rotate-left*(z)



: // Left rotation
return *rotate-left*(z)

Rule: The middle key of x, y, z becomes the new root.
Remark: Break ties to prefer single rotation.

Rafiee Sevyeri, Schost (CS-UW) CS240 – Module 4 Spring 2023 21 / 27

Rafiee Sevyeri, Schost (CS-UW) CS240 – Module Midterm Reference Sheet Spring 2023 2 / 2

Please initial: