Assignment 5 - Recurrences & Hash Tables

Due Wed March 19 2025

Jayden Cruz

1. (text) Recurrence [10 pts] Solve $T(n) = 3T(\frac{n}{4}) + 4n$ using repeated substitution then by master method.

Divide & Conquer recurrence relation

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

Build Solution

$T(n) = 3T(n/4) + 4n$

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

$= 3^2 T(\frac{n}{4^2}) + 12(\frac{n}{4}) + 4n$

$= 3^2[3T(\frac{n}{4^3}) + 4(\frac{n}{4^2})] + 12(\frac{n}{4}) + 4n$

$T(n) = 27T(\frac{n}{64}) + 36(\frac{n}{16}) + 12(\frac{n}{4}) + 4n$

Expand Solution

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

$T(n/4^2) = 3T(\frac{n}{4^3}) + 12(\frac{n}{4^2})$

$3^k T(\frac{n}{4^k}) + 4n + 3n(\frac{9n}{4}) + (\frac{27n}{16}) + \ldots \rightarrow 4(1-(3/4)^k) \rightarrow 4n \cdot 4(1-(\frac{3}{4})^k) = 16n(1-(\frac{3}{4})^k)$

$\frac{n}{4^k} = 1$, $k = \log_4 n$

$T(n) = O(n^{\log_4 3}) + O(n) = O(n^{\log_4 3} + n) = O(n)$

$\log_4 3 = 0.79248$ slowest growing so chop off

Master Theorem: $T(n) = aT(n/b) + f(n)$

$T(n) = 3T(n/4) + 4n$

- $T(n) = O(n^d)$ if $a < b^d$ this recurrence fits this cond. $a=3, b=4, f(n) = 4n = \theta(n^1)$

$T(n) = \theta(n)$

2. (text) Master Theorem [20 pts] apply the master method

$3 < 5^2 = 3 < 25$

a. $[T(n) = 3T(\frac{n}{5}) + n^2]$ $a=3, b=5, d=2 \rightarrow$ case 1 $a < b^d$

$T(n) = \theta(n^d)$ if $a < b^d$, plug & play $\theta(n^2)$ $\therefore \theta(n^2)$ for this recurrence

Master Thm: $T(n) = aT(n/b) + f(n)$

- If $f(n) = \theta(n^d)$, where $d \geq 0$, then

cases

let: 1 = $T(n) = \theta(n^d)$ if $a < b^d$

2 = $T(n) = \theta(n^d \log n)$ if $a = b^d$

3 = $T(n) = \theta(n^{\log_b a})$ if $a > b^d$

b. $[T(n) = 4T(\frac{n}{3}) + 7n]$ $a=4, b=3, d=1 \rightarrow$ case 3 $a > b^d \rightarrow 4 > 3^1$

$T(n) = \theta n^{\log_b a})$ if $a > b^d$, plug play $\theta(n^{\log_3 4})$ [eval $\log_3 4 \approx 1.26$] $\rightarrow \theta(n^{1.26})$

c. $[T(n) = 5T(\frac{n}{4}) + 10]$ (10 = $O(1)$ constant) $a=5, b=4, d=1 \rightarrow$ case 3 $a > b^d \rightarrow 5 > 4^1$

$(T(n) = \theta n^{\log_b a})$ if $a > b^d$, plug & play $\theta(n^{\log_4 5})$ [eval $\log_4 5 \approx 1.16$] $\rightarrow \theta(n^{1.16})$

d. $[T(n) = 9T(\frac{n}{3}) + n^4]$ $a=9, b=3, d=4 \rightarrow$ case 1 $a < b^d \rightarrow 9 < 3^4 \rightarrow 9 < 81$

$T(n) = \theta(n^d)$ if $a > b^d$, plug & play $\theta(n^4) \rightarrow O(n^4)$

e. $[T(n) = 6T(\frac{n}{8}) + n^3]$ $a=6, b=8, d=3 \rightarrow$ case 1 $a < b^d \rightarrow 6 < 8^3 \rightarrow 6 < 512$

$T(n) = \theta(n^d)$ if $a > b^d$, plug & play $\theta(n^3) \rightarrow \theta(n^3)$

Summary

a. case 1 : θn^2

b. case 3 : $\theta(n^{1.26})$

c. case 3 : $\theta(n^{1.16})$

d. case 1 : $\theta(n^4)$

e. case 1 : $\theta(n^3)$

3. (text) Radix Sort [10 pts]

CAP, COL, USD, SUN, JPY, VEE, ROW, JOB, COX, LOL, RAT, WOW, DOD, CAR, FIG, PIG, VIS, LOW, LUX, VEA, CAD, DOG, TSL

1. rightmost char

A. VEA
B. JOB
C
D. USD, DOD, CAD
E. VEE
F
G. FIG, PIG, DOG
H
I
J
K
L. COL, LOL, TSL
M
N. SUN
O.
P. CAP
Q.
R. CAR
S. VIS
T. RAT
U.
V.
W. ROW, WOW, LOW
X. COX, LUX
Y. JPY
Z

List now sorted by rightmost char: VEA, JOB, USD, DOD, CAD, VEE, FIG, PIG, DOG, COL, LOL, TSL, SUN

CAP, CAR, VIS, RAT, ROW, WOW, LOW, COX, LUX, JPY

CONT... NXT PG!

VEA, JOB, USD, DOD, CAD, VEE, FIG, PIG, DOG, COL, LOL, TSL, SUN, CAP, CAR, VIS, RAT, ROW, WOW, LOW, COX, LUX, JPY

Step 2. 2nd Char second char:

A. CAD, CAP, CAR, RAT
B.
C.
D.
E. VEA, VEE
F
G
H
I. FIG, PIG, VIS
J
K
L
M
N
O. JOB, DOD, DOG, COL, LOL, ROW, WOW, LOW, COX
P JPY
Q
R
S. USD, TSL
T
U. SUN, LUX
V.
W
X.
Y
Z

List now sorted middle char: CAD, CAP, CAR, RAT, VEA, VEE, FIG, PIG, VIS, JOB, DOD, DOG, COL

LOL, ROW, WOW, LOW, COX, JPY, USD, TSL, SUN, LUX

CONT... NXT PG!

CAD, CAP, CAR, RAT, VEA, VEE, FIG, PIG, VIS, JOB, DOD, DOG, COL, LOL, ROW, WOW, LOW, COX

JPY, USD, TSL, SUN, LUX

STEP 3: Leftmost Char.

A
B.
C. CAD, CAP, CAR, COL, COX
D. DOD, DOG
E
F. FIG
G.
H
I
J. JOB, JPY
K
L. LOL, LOW, LUX
M
N
O
P. PIG
Q
R. RAT, ROW
S. SUN
T. TSL
U. USD
V. VEA, VEE, VIS
W. WOW
X.
Y.
Z

List @ 3rd Pass: CAD, CAP, CAR, COL, COX, DOD, DOG, FIG, JOB, JPY, LOL, LOW, LUX, PIG, RAT, ROW, SUN, TSL, USD, VEA, VEE, VIS
WOW SORTED!

4. (text) Double Hashing [15 pts] 1st HashFunction h1(key) = ((key+19) x (key+11))/15 + key mod 13

M=13 slots

[25, 14, 9, 7, 5, 3, 0, 21, 6, 33, 25, 42, 24, 107] adding from left to right.

Hash Table

0:	25
1:	0
2:	21
3:	33
4:	14
5:	25
6:	42
7:	9
8:	6
9:	5
10:	3
11:	24
12:	7

1. 25 -> 1584/15 = 105.6 + 25 = 130 % 13 = 0. 25 goes in slot 0 [1 probe]
2. 14 -> 825/15 + 14 % 13 = 55 + 14 % 13 = 69 % 13 = 4. 14 into slot 4 [1 probe]
3. 9 -> 560/15 = 37.3 + 9 = 46 % 13 = 7. 9 goes into slot 7. [1 probe]
4. 7 -> 468/15 = 31.2 + 7 = 38 % 13 = 12 7 goes into slot 12 [1 probe]
5. 5 -> 384/15 = 25.6 + 5 = 30 % 13 = 4, slot 4 is taken collision reverse(5) = 5 1st probe 4+5 mod 13 = 9 has 5 now
6. 3 -> 308/15 = 20.5 + 3 = 23 % 13 = 10. 3 into slot 10 [1 probe]
7. 0 -> 209/15 = 13 + 0 = 13 % 13 = 0. [1 probe collision] 0 is take reverse(0) = 0 + 0 + 1 % 13 = 1, 0 into slot 2
8. 21 -> 1260/15 = 85 + 21 = 106 % 13 = 2 21 goes into slot 2 1 (probe)
9. 6 -> 425/15 = 28 + 6 = 34 % 13 = 8. 6 goes into slot 8 (1 probe)
10. 33 -> 2288/15 = 152 + 33 = 185 % 13 = 3. 33 goes into slot 3 (1 probe)
11. 25 -> 1584/15 = 105.6 + 25 = 130 % 130 = 0 -> slot taken [1 probe 1 collision] reverse(25) = (0 + 1.52) % 13 = 10 collision again. • each time a slot is taken add a collision

(0 + 1 x 1) mod 13 = 1 (taken) -> (0 + 2 x 1) mod 13 = 2 (taken) -> (0 + 3 x 2) mod 13 = 3 (taken) -> (0 + 4 x 1) mod 13 = 4 (taken) -> (0 + 5 x 1) % 13 = 5 25 into slot 5 7 collisions 8 probes

12. 42 -> 3233/15 = 216 + 42 = 257 mod 13 = 10 alr taken -> reverse(42) = (10 + 1 x 24) mod 13 = 8 -> (10 + 2 x 24 % 13 = 6 [2 collisions 3 probes]

13. 24 -> 1505/15 = 100 + 24 = 124 mod 13 = 7 alr taken reverse(24) (7 + 1 x 42) mod 13 = 10 alr taken

(7 + 2) x 42) mod 13 = 0 slot alr taken -> (7 + 3 x 42)) % 13 = 3, slot alr taken -> (7 + 4 · 42)) % 13 = 6 slot alr taken -> (7 + 5 x 42) mod 13 = 1 taken -> (7 + 6 x 42) mod 13 = 12 taken

cont... (7 + 7 x 42) mod 13 = 2 taken -> 7 + 8 x 42 % 13 = 5 taken -> 7 + 8 · 42 % 13 = 8 taken -> 7 + 10 · 42 % 13 = 11 [11 probes 10 collisions] 24 goes into 11

↳ now full so we have to resize 13 x 2 = 26

New Hash Table

0:	25
1:	0
2:	21
3:	33
4:	14
5:	25
6:	42
7:	9
8:	6
9:	5
10	3
11	24
12	7
:	
;	
:	
18:	107
:	
;	
:	
25:	

-> has 26 slots

13. 107 -> 14868/15 = 991 + 107 = 1098 % 27 = 18 <1 probe>

7. (text) Algorithm Analysis [5 pts]

Double Hashing:
Time Complexity: O(1) on average for each insertion given our hash function, but O(n) when resizing & rehashing happens (n is the # of slots in hash table)
Space Complexity: O(M), where M is the number of slots in the hash table

Radix Sort:
Time Complexity: each pass over the strings is O(n·k), where n is the # of strings & k is the length of the longest string in the array ... O(n·k) k = max string length
Space Complexity: O(n+256), use O(n) space to store the input array & O(256) space for 256 buckets in the counting sort. 256 is # of ASCII chr
↳ O(n+256) chop off the constant ∴ O(n)

Word Pattern:
Time Complexity: O(n), where n is the length of the pattern, as it also determines the # of words in the split string
- splitting the string has a time complexity of O(m), where m is the length of the string s. Bc this step happens before each iteration, time complexity is linear

Space Complexity: O(n), where n is the length of the pattern / # of words after splitting string s
↳ space used by maps patternToWord & wordToPattern depends on the # of unique chr's & words, which is at most n the input