#HW#5

Toufiq Mahmud

$T(1) = 0$

$T(n) = 3T(n/3) + 2n - 3$

Assume $n = 3^k$, $k = \log_3 n$

Domain Transformation:

$S(k) = T(n) = 3S(3^{k-1}) + 2(3^k) - 3$

$S(0) = T(\cancel{0}) = 0$

Range transform:

$R(k) = \frac{S(k)}{3^k} = \frac{\cancel{3}S(3^{k-1})}{3^{\cancel{k}}} + \frac{2(3^k)}{3^k} - \frac{3}{3^k}$

$= R(k-1) + 2 - \frac{1}{3^{k-1}}$

Set up telescoping:

$R(k) - \cancel{R(k-1)} = 2 - \frac{1}{3^{k-1}}$

$\vdots$

$\cancel{R(1)} - \underbrace{R(0)}_{0} = 2k - \sum_{i=0}^{k-1} \frac{1}{3^i}$

$\Rightarrow R(k) = 2k - \sum_{i=0}^{k-1} \frac{1}{3^i}$

$= 2k - \frac{\left(\frac{1}{3}\right)^k - 1}{\frac{1}{3} - 1} = 2k - \frac{1 - \left(\frac{1}{3}\right)^k}{2/3} = 2k - \frac{3}{2}\left(\frac{1}{3^k}\right)$

$S(k) = 3^k R(k) = 2k3^k - \frac{3}{2} 3^k \left[\frac{1}{1-3^k}\right]$

$S(k) = 2k3^k - \frac{3^{k+1}}{2} + \frac{3}{2}$

$T(n) = 2\log_3 \cdot 3^{\log_3 n} - \frac{3^{\log_3 n} \cdot 3}{2} + \frac{3}{2}$

$= 2n\log_3 n - \frac{3n}{2} + \frac{3}{2}$

# AVERAGE-CASE ANALYSIS OF QUICKSORT

$T(0) = T(1) = c$ ← could be 0

$$T(n) = cn + \frac{1}{n} \sum_{n=0}^{n-1} \big( \underbrace{\underset{\text{left of pivot}}{T(k)} + \underset{\text{right of pivot}}{T(n-k-1)}}_{\text{Summations are same backwards}} \big)$$

$$\boxed{cn + \frac{2}{n} \sum_{k=0}^{n-1} T(k) = T(n)} \leftarrow ⓐ$$

$$c(n+1) + \frac{2}{n+1} \sum_{k=0}^{n} T(k) = T(n+1) \quad \begin{bmatrix} \text{plug in} \\ n+1 \end{bmatrix} \leftarrow ⓑ$$

Recurrence with memory aka has to remember full history of what came before

$$ⓐ \cdot n = cn^2 + 2\sum_{k=0}^{n-1} T(k) \leftarrow ⓒ$$

$$ⓑ \cdot (n+1) = c(n+1)^2 + 2\sum_{k=0}^{n} T(k) \leftarrow ⓓ$$

Subtract ⓒ from ⓓ

$$\Rightarrow T(n+1)(n+1) - T(n) \cdot n = c(n+1)^2 + cn^2 + 2T(n)$$

$$= c[(n+1)^2 n^2] + 2T(n)$$

$$= c(2n+1) + 2T(n)$$

$$\frac{\cancel{(n+1)} T(n+1)}{\cancel{(n+1)}(n+2)} - \frac{\cancel{(n+2)} T(n)}{(n+1)\cancel{(n+2)}} = \frac{c(2n+1)}{(n+1)(n+2)}$$

Simplify RHS using Partial Fractions

$$c\left(\frac{2n+1}{(n+1)(n+2)}\right) = \frac{a}{n+1} + \frac{b}{n+2}$$

$$\Rightarrow a(n+2) + b(n+1) = 2n+1$$

$$(a+b)n = 2n \quad \Rightarrow \quad a = -1$$

$$b = 3$$

$$2a + b = 1$$

$$= \frac{-1}{n+1} + \frac{3}{n+2}$$

Range Transformation

$$R(n) = \frac{T(n)}{n+1}$$

$$R(n+1) - R(n) = c\left[\frac{-1}{n+1} + \frac{3}{n+2}\right]$$

$$\vdots$$

$$\underbrace{R(1) - R(0)} = c\left[-\frac{1}{1} + \frac{3}{2}\right]$$

$$R(n+1) = -c\sum_{i=1}^{n+1}\frac{1}{i} + c\sum_{i=1}^{n+1}\frac{3}{i+1}$$

$\rightarrow$ Harmonic Series

$$1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} \cdots \infty$$

$$= \cancel{o(\log n)})$$

$$R(0) = \frac{T(0)}{0+1} = c$$

$$R(n+1) + R(0) = R(n+1) + c \approx$$

$$= o(\log cn)$$

RHS:

$$c \cdot 2\log n = R(n+1)$$

$$\downarrow$$

Transform back

$$\Rightarrow T(n+1) = (n+1)\, c \cdot 2 \cdot \log n$$

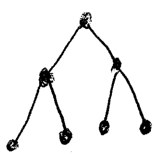
$$= o(n \log n)$$

**Lower bound for finding the minimum value**

Create a binary tree where each leaf is a possible value for the min.

There will be at least n-1 leaves because each value needs to be represented in the list of possible answers.

Comparison tree



| Depth | # leaves |
|---|---|
| 1 | 2 |
| 2 | $2^2$ |

If the tree is not complete it will have less leaves for the same depth. If there are n leaves, the depth will be log (n)

How many possible leaves are there for sorting?

- At least n! because the same permutation could appear multiple ways
- The number of leaves >= n!
- Base case height = log (n!) = nlogn – occurs when the tree is balanced and complete
    - Sterling's approximation formula

Can you do better than nlogn?

Yes. (But if the algorithm is comparison-based then the best is nlogn)

**Counting sort**

- Create an array with indexes 1 to m and initialize to some value that won't be in the input (such as –infinity)
- Read an input element and increment a[k] where k is the value of the input element
- Once all the input has been read print out the elements

Input: 25, 45, 18, 25

Array:

| Index | 1 | 2 | … | 18 | … | 25 | … | 45 |
|---|---|---|---|---|---|---|---|---|
| Value | | | | 1 | | [1+1]=2 | | 1 |

Output: 18 once, 25 twice, 45 once = 18, 25, 25, 45

**Time complexity of counting sort:**

Initializing: m

Counting: n

To write out: n+m

So the time complexity **is O(m+n)**

Problem with counting sort: Inefficient space allocation because the array has to be allocated for the entire range of the input while only a few values in the range could appear

**Bucket Sort**

Sorting exam papers analogy – assume you need to sort student grades from highest to lowest. Put papers into buckets of certain ranges (example: 0-70, 71-80, 81-90, 91-100). Once the papers have been put into piles sort each pile and put everything in order.

M = maximum value in list ( upper bound)

K = # of buckets

On average there are m/k items in each bucket

For each bucket, using insertion sort = (n/k)^2 /4 * k buckets = $n^2$ /4k

K should be proportional to n

N^2/4k*(c/d)

If k is proportional to n then the algorithm is O(n)

Worst case: All the data is in 1 bucket = O($n^{2)}$

**Radix Sort**

Assume the is 50, 6, 150, 300

| | | | | | | |
|---|---|---|---|---|---|---|
| 050 | sort based on last digit → | 050 | sort based on 2nd digit → | 300 | sort 1st digit → | 006 |
| 006 | | 150 | | 006 | | 050 |
| 150 | | 300 | | 050 | | 150 |
| 300 | | 006 | | 150 | | 300 |

Time Complexity: Time to sort * number of digits

= O(n) * d

How to pick d? Pick d so that it's a constant & not proportional to n