

Lecture17

CSE 417

Algorithms and Complexity

Winter 2023
Lecture 17
Divide and Conquer

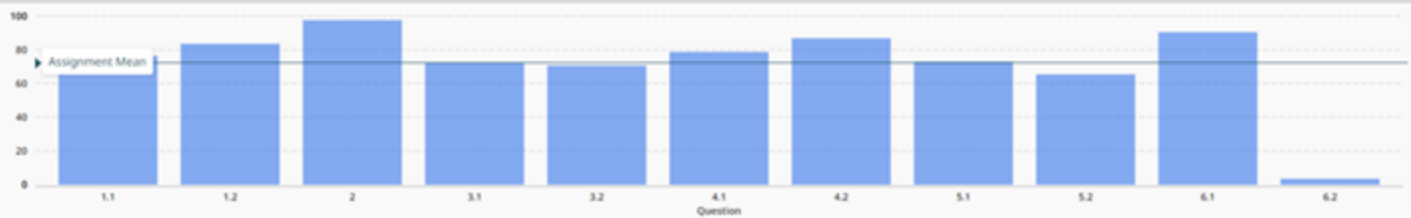
11/6/2023

CSE 417

1

Announcements

- Midterm stats (out of 60)
 - Mean: 43.2, Median: 46.25, Std Dev: 9.63



- Today and Wednesday: Divide and Conquer
- Friday: Armistice Day / Veterans Day (almost)

What you really need to know about recurrences

- Work per level changes geometrically with the level
- Geometrically increasing ($x > 1$)
 - The bottom level wins
- Geometrically decreasing ($x < 1$)
 - The top level wins
- Balanced ($x = 1$)
 - Equal contribution



Classify the following recurrences (Increasing, Decreasing, Balanced)

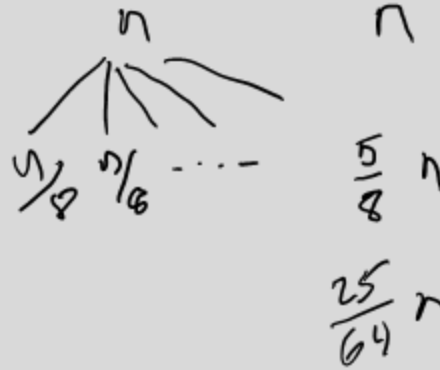
- $T(n) = n + 5T(n/8)$

- $T(n) = n + 9T(n/8)$

- $T(n) = n^2 + 4T(n/2)$

- $T(n) = n^3 + 7T(n/2)$

- $T(n) = n^{1/2} + 3T(n/4)$



$n \log \frac{9}{8}$

Divide and Conquer

- Algorithm paradigm
 - Break problems into subproblems until easy to solve
 - Work is split between “divide”, “combine”, and “base” components
- Standard examples
 - MergeSort and QuickSort
- Analysis tool: Recurrences

Matrix Multiplication

- N X N Matrix, $A B = C$

$$[A][B] = [C]$$

row each row result

```
for (int i = 0; i < n; i++)
  for (int j = 0; j < n; j++) {
    int t = 0;
    for (int k = 0; k < n; k++)
      t = t + A[i][k] * B[k][j];
    C[i][j] = t;
  }
```

Dot Product

$$O(n^3)$$

Recursive Matrix Multiplication



Multiply 2 x 2 Matrices:

$$\begin{pmatrix} r & s \\ t & u \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} e & g \\ f & h \end{pmatrix}$$

$$\begin{aligned} r &= ae + bf \\ s &= ag + bh \\ t &= ce + df \\ u &= cg + dh \end{aligned}$$

8 mult
4 add

A $N \times N$ matrix can be viewed as a 2×2 matrix with entries that are $(N/2) \times (N/2)$ matrices.

The recursive matrix multiplication algorithm recursively multiplies the $(N/2) \times (N/2)$ matrices and combines them using the equations for multiplying 2×2 matrices



$$= \begin{pmatrix} AE + BF \\ \dots \end{pmatrix}$$

Recursive Matrix Multiplication

- How many recursive calls are made at each level?

8

- How much work in combining the results?

$O(n^2)$

- What is the recurrence?

$$T(n) = 8T\left(\frac{n}{2}\right) + n^2$$

What is the run time for the recursive Matrix Multiplication Algorithm?

- Recurrence: $T(n) = 8T\left(\frac{n}{2}\right) + n^2$



$$8\left(\frac{n}{2}\right)^2 = \frac{8n^2}{4} = 2n^2$$

$$\text{leaves} = 8^{\log n} = \frac{2^{3 \log n}}{8^3}$$

4 Add, 8 Mult

Strassen's Algorithm

18 Add, 7 mult

Multiply 2 x 2 Matrices:

$$\begin{vmatrix} r & s \\ t & u \end{vmatrix} = \begin{vmatrix} a & b \\ c & d \end{vmatrix} \begin{vmatrix} e & g \\ f & h \end{vmatrix}$$

$$r = p_1 + p_2 - p_4 + p_6$$

$$s = p_4 + p_5$$

$$t = p_6 + p_7$$

$$u = p_2 - p_3 + p_5 - p_7$$

Where:

$$p_1 = (b - d)(f + h)$$

$$p_2 = (a + d)(e + h)$$

$$p_3 = (a - c)(e + g)$$

$$p_4 = (a + b)h$$

$$p_5 = a(g - h)$$

$$p_6 = d(f - e)$$

$$p_7 = (c + d)e$$

From Aho, Hopcroft, Ullman 1974

Recurrence for Strassen's Algorithms

- $T(n) = 7 T(n/2) + cn^2$
- What is the runtime?

$$7^{\log_2 n} = n^{\log_2 7}$$
$$n^{2.807\dots}$$

$$\log_2 7 = 2.8073549221$$

CSE 417

11

$n^{2.807}$

Strassen's Algorithms

- Treat $n \times n$ matrices as 2×2 matrices of $n/2 \times n/2$ submatrices
- Use Strassen's trick to multiply 2×2 matrices with 7 multiplies
- Base case standard multiplication for single entries
- Recurrence: $T(n) = 7 T(n/2) + cn^2$
- Solution is $O(7^{\log n}) = O(n^{\log 7})$ which is about $O(n^{2.807})$

$$n = 32, \quad n = 64$$

Quicksort [Tony Hoare, 1959]

QuickSort(S):

1. Pick an element v in S . This is the *pivot* value.
2. Partition $S - \{v\}$ into two disjoint subsets, S_1 and S_2 such that:
 - elements in S_1 are all $< v$
 - elements in S_2 are all $> v$
3. Return concatenation of QuickSort(S_1), v , QuickSort(S_2)

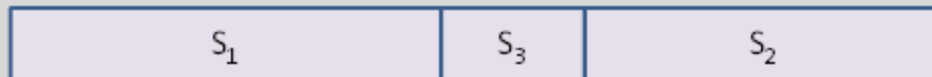
Recursion ends if QuickSort() receives an array of length 0 or 1.

Computing the Median

- Given n numbers, find the number of rank $n/2$
- One approach is sorting
 - Sort the elements, and choose the middle one
 - Can you do better?
- *Selection*, given n numbers and an integer k , find the k -th largest

Select(A, k)

```
Select(A, k){  
  Choose element x from A  
  S1 = {y in A | y < x}  
  S2 = {y in A | y > x}  
  S3 = {y in A | y = x}  
  if (|S2| >= k)  
    return Select(S2, k)  
  else if (|S2| + |S3| >= k)  
    return x  
  else  
    return Select(S1, k - |S2| - |S3|)  
}
```



Deterministic Selection

- Random pivot gives an expected $O(n)$ run time. The question of a deterministic algorithm was more challenging.
- What is the run time of select if we can guarantee that *ChoosePivot* finds an x such that $|S_1| < 3n/4$ and $|S_2| < 3n/4$ in $O(n)$ time?

$$T(n) \leq T\left(\frac{3}{4}n\right) + cn$$

BFPRT Algorithm

- A very clever choose algorithm . . .
- Deterministic algorithm that guarantees that $|S_1| < 3n/4$ and $|S_2| < 3n/4$
- Actual recurrence is:

$$T(n) \leq T(3n/4) + T(n/5) + c n$$



BFPRT Algorithm

$$|S_1| < 3n/4, |S_2| < 3n/4$$

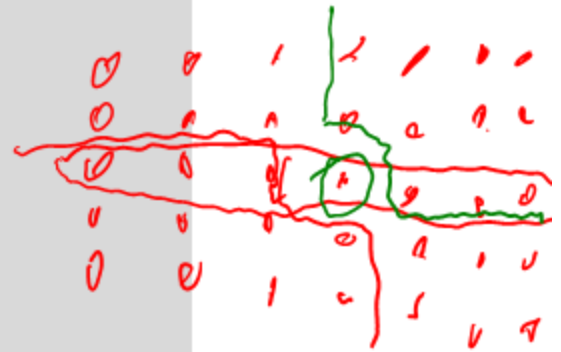
Split into $n/5$ sets of size 5

M be the set of medians of these sets

x be the median of M

Construct S_1 and S_2 using pivot x

Recursive call in S_1 or S_2



$$T(n) = T\left(\frac{3}{4}n\right) + T\left(\frac{n}{5}\right) + cn$$

BFPRT Recurrence

- $T(n) \leq T(3n/4) + T(n/5) + c n$

$$T(n) \leq 20cn$$

Prove that $T(n) \leq 20cn$

11/6/2023

CSE 417

19