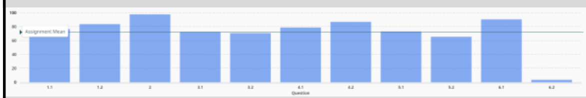
CSE 417
Algorithms and Complexity

Winter 2023
Lecture 17
Divide and Conquer

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)

- $T(n) = n + 5T(n/8)$
- $T(n) = n + 9T(n/8)$
- $T(n) = n^2 + 4T(n/2)$
- $T(n) = n^3 + 7T(n/2)$
- $T(n) = n^{1/2} + 3T(n/4)$

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

```
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;
        }
```

Recursive Matrix Multiplication

Multiply 2 x 2 Matrices:

| r s | = | a b| |e g|
| t u | | c d| | f h|

r = ae + bf
s = ag + bh
t = ce + df
u = cg + dh

A N x N matrix can be viewed as a 2 x 2 matrix with entries that are (N/2) x (N/2) matrices.

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

Recursive Matrix Multiplication

- How many recursive calls are made at each level?
- How much work in combining the results?
- What is the recurrence?

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

- Recurrence:

Strassen's Algorithm

Multiply 2 x 2 Matrices:

| r s| = |a b| |e g|
| t u| |c d| |f h|

$r = p_1 + p_2 - p_4 + p_6$

$s = p_4 + p_5$

$t = p_6 + p_7$

$u = p_2 - p_3 + p_5 - p_7$

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?

$\log_2 7 = 2.8073549221$

Strassen's Algorithms

- Treat n x n matrices as 2 x 2 matrices of n/2 x n/2 submatrices
- Use Strassen's trick to multiply 2 x 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 $O(n^{2.807})$

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, $\mathbf{S_1}$ and $\mathbf{S_2}$ such that:
   - elements in $\mathbf{S_1}$ are all < *v*
   - elements in $\mathbf{S_2}$ are all > *v*
3. Return concatenation of QuickSort($\mathbf{S_1}$), *v*, QuickSort($\mathbf{S_2}$)

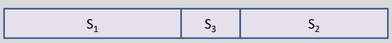
Recursion ends if Quicksort( ) receives an array of length 0 or 1.

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?

BFPRT Algorithm


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

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

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

BFPRT Recurrence

- T(n) <= T(3n/4) + T(n/5) + c n

Prove that T(n) <= 20 c n