CSE 3500 Algorithms and Complexity – Fall 2016
Lecture 12: October 6, 2016

Worst case linear time selection

- In the last lecture we introduced the quick select algorithm and showed that it's worst case run time is $\Omega(n^2)$. It has a best case and average case run time of $O(n)$.
- We also started our discussion on the worst case linear time algorithm of BFPRT.
- The BFPRT algorithm is nothing but the quick select algorithm wherein the pivot is picked using a special algorithm.
- If $X = k_1, k_2, \ldots, k_n$ is the input sequence, the algorithm groups the elements of X into groups of size 5 each. Let these groups be $G_1, G_2, \ldots, G_{n/5}$. The median of each of these groups is found. Let these medians be $M_1, M_2, \ldots, M_{n/5}$, respectively. The median M of these medians is found recursively and used as the pivot.
- The entire algorithm can be summarized as follows:

BFPRT(X, i)
  0) **if** $|X| = 1$ **then** output k_1 and quit;
  1) Group X into groups $G_1, G_2, \ldots, G_{n/5}$ each of size 5;
    Find the median M_i of G_i for $i = 1, 2, \ldots, \frac{n}{5}$;
    Let $S = \{M_1, M_2, \ldots, M_{n/5}\}$;
  2) $M =$BFPRT$(S, \frac{n}{10})$; (M is the median of group medians);
  3) Partition X into $X_1 = \{q \in X : q < M\}$ and $X_2 = \{q \in X : q > M\}$;
  4) **if** $|X_1| = i - 1$ **then** output M and quit;
    if $|X_1| \geq i$ **then** BFPRT(X_1, i);
    else BFPRT$(X_2, i - |X_1| - 1)$;

Analysis of the BFPRT algorithm

- The element M we pick as the pivot is expected to be an approximate median of X. In this case we can expect X_1 and X_2 to be nearly of the same size and subsequently, our divide and conquer algorithm can be expected to yield a good performance.
- We will prove that X_1 and X_2 will be nearly of the same size.

- Note that out of the $n/5$ groups we have created in step 1, half of the groups will have a median $\leq M$ and the other half of the groups will have a median $\geq M$. Let G_i be one group whose median is $\leq M$. If the median of G_i is M_i, then $M_i \leq M$. In this group G_i there are three elements that are less than or equal to M_i (by definition of the median) and all of these elements will also be $\leq M$ (since $M_i \leq M$). This means that at least $3 \times \frac{n}{10}$ elements of X will be $\leq M$. This in turn means that $|X_2| \leq \frac{7}{10}n$.

- Along the same lines, we can also show that $|X_1| \leq \frac{7}{10}n$.

- Now we are ready to write a recurrence relation for the run time of the BFPRT algorithm.

- Let $T(n)$ be the run time of the BFPRT algorithm on any input of size n and for any i.

- Step 1 takes $\Theta(n)$ time since we can find the median of each group in $\Theta(1)$ time. Step 2 takes $T(n/5)$ time. Partitioning in step 3 can be done in $\Theta(n)$ time. In step 4, in the worst case, we recurse on X_1 or X_2. We know that the size of neither is more than $\frac{7}{10}n$. As a result, we get:

$$T(n) \leq T\left(\frac{n}{5}\right) + T\left(\frac{7}{10}n\right) + \Theta(n) \leq T\left(\frac{n}{5}\right) + T\left(\frac{7}{10}n\right) + dn$$

  for some constant d.

- We can prove by induction that the above recurrence relation solves to: $T(n) = O(n)$.

- Induction Hypothesis: $T(n) \leq cn$ for some constant c. The base case can be proven easily.

- Induction step: Assume that the hypothesis holds for all the inputs of size up to $n-1$. We'll prove it for inputs of size n.

- $T(n) \leq T\left(\frac{n}{5}\right) + T\left(\frac{7}{10}n\right) + dn \leq c\frac{n}{5} + c\frac{7}{10}n + dn = 0.9cn + dn$. The RHS will be $\leq cn$ if $c \geq 10d$.

- Thus we conclude that $T(n) \leq 10dn$, i.e., $T(n) = O(n)$. We get the following:

  Theorem. We can select the ith smallest element from any given sequence of n elements in $O(n)$ time. $\square$

A randomized algorithm

- Floyd and Rivest (1975) have given a randomized algorithm that employs random sampling. The steps in this algorithm are:

  1) Pick a random sample S from the input sequence X;
  2) Find two elements l_1 and l_2 from S such that:

the ith smallest element of X (call it q) has a value in the interval $[l_1, l_2]$ and $|\{x \in X : l_1 \leq x \leq l_2\}|$ is 'small' with a high probability.

3) Identify the set $Y = \{x \in X : l_1 \leq x \leq l_2\}$;
4) Make sure that q is in Y and perform an appropriate selection in Y;

- We can show that the number of comparisons made by the above algorithm is $n + \min\{i, n - i\} + \widetilde{o}(n)$. This is one of the best-known algorithms for selection.

Matrix Multiplication

- Matrix multiplication is a very important problem in science and engineering with numerous applications.

- Input for this problems are two matrices A and B of size $n \times n$ each. The goal is to compute the product C of A and B.

- We can come up with a simple cubic time algorithm for this problem as follows:

for $i = 1$ **to** n **do**
  for $j = 1$ **to** n **do**
    $C[i, j] = 0.0;$
    for $k = 1$ **to** n **do**
      $C[i, j] = C[i, j] + A[i, k] * B[k, j];$

- The above algorithm takes n multiplications and $n - 1$ additions for each output element and there are a total of n^2 elements to be output. Thus the total time is $n^2(2n-1) = \Theta(n^3)$.

Strassen's algorithm

- Strassen has given an elegant divide-and-conquer algorithm for matrix multiplication that takes subcubic time.

- Consider the problem of multiplying two 2×2 matrices. A straight forward algorithm for this problem will take 8 multiplications. Strassen has come up with a way of mutiplying them with only 7 multiplications.

- Let the matrices of interest be $A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$ and $B = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}$.

- Strassen computed the product $C = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix}$ of A and B as follows:

$d_1 = (a_{11} + a_{22})(b_{11} + b_{22});$
$d_2 = (a_{21} + a_{22})b_{11};$
$d_3 = a_{11}(b_{12} - b_{22});$
$d_4 = a_{22}(b_{21} - b_{11});$
$d_5 = (a_{11} + a_{12})b_{22};$
$d_6 = (a_{21} - a_{11})(b_{11} + b_{12});$
$d_7 = (a_{12} - a_{22})(b_{21} + b_{22});$
$c_{11} = d_1 + d_4 - d_5 + d_7;$
$c_{12} = d_3 + d_5;$
$c_{21} = d_2 + d_4;$
$c_{22} = d_1 - d_2 + d_3 + d_6.$

- If A and B are generic $n \times n$ matrices, we could use the above algorithm to derive a divide-and-conquer recursive algorithm to multiply them. W.l.o.g. assume that $n = 2^k$ for some integer k.

- Partition A and B as: $A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$ and $B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$, where A_{ij} and B_{ij} are $\frac{n}{2} \times \frac{n}{2}$ submatrices, for $1 \leq i, j \leq 2$.

- After partitioning A and B as above, use Strassen's formulas to multiply A and B. What is a scalar multiplication (or addition) in the above formulas will now become a submatrix multiplication (or addition). Submatrix addition is easy. Submatrix multiplication is done recursively.

- Let $T(n)$ be the run time of Strassen's algorithm to multiply two $n \times n$ matrices. Then, we have:
$$T(n) = 7T\left(\frac{n}{2}\right) + \Theta(n^2).$$

- Note that we have to do seven submatrix multiplications (each taking $T\left(\frac{n}{2}\right)$ time) and 18 submatrix additions (each taking $\frac{n^2}{4}$ time).

- Using the Master theorem, we can solve for $T(n)$ to get: $T(n) = \Theta(n^{\log_2 7})$.

- Since the publications of Strassen's algorithm in 1969, a number of improvements have been made: V. Pan (1978): $O(n^{2.796})$; Coppersmith and Winograd (1983): $O(n^{2.376})$; V. Williams (2014): $O(n^{2.373})$.