Heaven's Light is Our Guide

Computer Science & Engineering
Rajshahi University of Engineering & Technology

Lab Manual

Module-04

Course Title : Sessional based on CSE 2201
Course No. : CSE 2202

Experiment No. 4

Name of the Experiment: Design and Complexity analysis of selection, matrix multiplication (divide and Conquer algorithm) and Convex Hull (basic).

Date: 4th Cycle

Algorithms (Divide-and-Conquer method)**:**

- **Selection**
- **matrix multiplication (divide and Conquer algorithm)**
- **Convex Hull**

Selection:

The PARTITION algorithm may also be used to obtain an efficient solution to the selection problem. In this problem, we are given n elements A(1:n) and are required to determine the kth smallest element. If the partitioning element v is positioned at A(j), then j - 1 elements are less than or equal to A(j) and n - j elements are greater than or equal to A(j). Hence if k < j then the kth smallest element is in A(lj - 1); if k = j then A(j) is the kth smallest element; if k > j then the kth smallest element is the (k — j)th smallest element in A(J + l:n). The resulting algorithm is procedure SELECT. This procedure places the kth smallest element into position A(k) and partitions the remaining elements such that A(i) < A(k), 1 < i < k and A(i) > A(k), k < i < n.

```
Algorithm Select1(a,n,k)
//Select the kth smallest element in a[1:n] and place it in the kth position of
//a[]. The remaining elements are rearranged such that a[m]≤a[k] for
//1≤m<k, and a[m]≥a[k] for k<m≤n
{
   Low:=1;up:=n+1;
   A[n+1]:=∞;
   repeat
   {
       //Each time the loop is entered
      //1≤low≤k≤up≤n+1
     j:=partition(a,low,up);
    If (k=j) then return;
    else if (k<j) then up:=j;
         Else low:=j+1;
  } until (false)
}
```

Algorithm: Finding the kth smallest element

Let,

A: 65, 70, 75, 80, 85, 60, 55, 50 with A[10]=∞.

If k = 5 then the first call of Partition will be sufficient since 65 is placed into A(5). Instead assume that we are looking for the seventh smallest element of A, i.e. k = 7. The next invocation of Partition is call Partition(6,10).

```
A:   (5)   (6)   (7)   (8)   (9)   (10)   i    p

     65    85    80    75    70    +∞     10   9
           |-----------------------|
     65    70    80    75    85    +∞
```

This last call of Partition has uncovered the 9th smallest element of A. The next invocation is call Partition(6,9).

```
A:   (5)   (6)   (7)   (8)   (9)   (10)   i   p

     65    70    80    75    85    +∞     7   6
          |-|
     65    70    80    75    85    +∞
```

This time, the sixth element has been found. Since k ≠ j is still true in Select1, another call to Partition is made, call Partition(7,9).

```
A:   (5)   (6)   (7)   (8)   (9)   (10)   i   p

     65    70    80    75    85    +∞     9   8
                |-------|
     65    70    75    80    85    +∞
```

Now 80 is the partition value and that is correctly placed at A[8]. However, Select1 has still not found the 7th smallest element. It needs one more call to Partition, which is call Partition(7, 8). This performs only an interchange between A[7] and A[8] and then returns having found the correct value.

A WORST-CASE OPTIMAL ALGORITHM:

```
Algorithm Select2(a,k,low,up)
//return i such that a[i] is the kth smallest number in A[low:up).
//r is a global variable
{
  repeat
  {
    n:=up-low+1;//Number of element
    if(n=r) then
    {
      InsertionSort(a,low,up);
      return low+k-1;
    }
    for i:=1 to $\lfloor n/r \rfloor$ do
    {
        InsertionSort(a,low+(i-1)*r, low+i*r-1);
        //Collect Medians in the front part of a[low:up].
        Interchange(a,low+i-1,low+(i-1)*r+$\lceil r/2 \rceil$-1);
      }
      j:=Select2(a,$\lceil \lfloor n/2 \rfloor /2 \rceil$,low,low+$\lfloor n/r \rfloor$-1);
      Interchange(a,low,up+1);
      if(k=(j-low+1)) then return j'
      else if (k<(j-low+!)) then up:=j-1;
      else
      {
        k:=k-(j-low+1);
        low:=j+1;

      }
   } until(false);
 }
```

	Required Time			
	Select1		Select2	
No. of Data	**Average Case**	**Worst Case**	**Average Case**	**Worst Case**
1000				
2500				
5000				
7500				
10000				
12500				
15000				
17500				
20000 etc.				

Table 1

Report should contain:

1. Machine configuration.
2. Complete Table 1.
3. Plot graphical output [x-axis=number of data, y-axis=Required Time][Using any Language].
4. Plot bar graph [using Microsoft Office Excel].
5. Properly analysis of **table** and **graph**.
6. Derive Time complexity of these algorithms and prepare proper analysis of those complexities.

[**N.B.** Pick randomly 1000 numbers ranging from 1 to 30,000 then write them in a file. Read them from the file and search a particular data.]

Matrix Multiplication:
Let A and B be two n x n matrices. The product matrix C = AB is also an n x n matrix whose i,jth element is formed by taking the elements in the rth row of A and the/th column of B and multiplying them to get

$$C(i,j) = \sum_{1 \le k \le n} A(i,k)\,B(k,j)$$

for all i and j between 1 and n. To compute C(i,j) using this formula, we need n multiplications. As the matrix C has n^2 elements, the time for the resulting matrix multiplication algorithm, which we shall refer to as the "conventional" method is $\Theta(n^3)$.

The divide-and-conquer strategy suggests another way to compute the product of two n x n matrices. For simplicity we will assume that n is a power of 2, i.e. that there exists a nonnegative integer k such that $n = 2^k$. In case n is not a power of two then enough rows and columns of zeros may be added to both A and B so that the resulting dimensions are a power of two. Imagine that A and B are each partitioned into four square submatrices, each submatrix having dimensions n/2 X n/2. Then the product AB can be computed by using the above formula for the product of 2 x 2 matrices, namely if AB is

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}$$

Then

$$C_{11} = A_{11}B_{11} + A_{12}B_{21}$$
$$C_{12} = A_{11}B_{12} + A_{12}B_{22}$$
$$C_{21} = A_{21}B_{11} + A_{22}B_{21}$$
$$C_{22} = A_{21}B_{12} + A_{22}B_{22}$$

If n = 2 then the above formulas are computed using a multiplication operation for the elements of A and B. These elements are typically floating point numbers. For n > 2 the elements of C can be computed using matrix multiplication and addition operations applied to matrices of size n/2 x n/2. Since n is a power of 2, these matrix products can be recursively computed by the same algorithm we are using for the n x n case. This algorithm will continue applying itself to smaller size submatrices until n becomes suitably small (n=2) so that the product is computed directly.

In order to compute AB using above equation, we need to perform eight multiplications of n/2 x n/2 matrices and four additions of n/2 x n/2 matrices. Since two n/2 x n/2 matrices may be added in time cn^2 for some constant c, the overall computing time, T(n) of the resulting divide-and-conquer algorithm is given by the recurrence

$$T(n) = \begin{cases} b, & n \le 2 \\ 8T(n/2) + cn^2, & n > 2 \end{cases}$$

Where b and c are constants.
This recurrence may be solved in the same way as earlier recurrences to obtain T (n) = O (n^3). Hence no improvement over the conventional method has been made. Since matrix multiplications are more expensive than matrix additions (O(n^3) vs. O(n^2)) one may attempt to reformulate the equations for dj so as to have fewer multiplications and possibly more additions. Volker Strassen has discovered a way to compute the C_{ij}s using only 7 multiplications and 18 additions or subtractions. His method involves first computing the seven n/2 x n/2 matrices P, Q, R, S, T, U and V. Then theC_{ij}s are computed. As can be seen, P, Q, R, S, T, U and V may be computed using 7 matrix multiplications and 10 matrix additions or subtractions. The C_{ij}s require an additional 8 additions or subtractions.

$$\begin{aligned} P &= (A_{11} + A_{22})(B_{11} + B_{22}) \\ Q &= (A_{21} + A_{22})B_{11} \\ R &= A_{11}(B_{12} - B_{22}) \\ S &= A_{22}(B_{21} - B_{11}) \\ T &= (A_{11} + A_{12})B_{22} \\ U &= (A_{21} - A_{11})(B_{11} + B_{12}) \\ V &= (A_{12} - A_{22})(B_{21} + B_{22}) \end{aligned}$$

$$\begin{aligned} C_{11} &= P + S - T + V \\ C_{12} &= R + T \\ C_{21} &= Q + S \\ C_{22} &= P + R - Q + U \end{aligned}$$

The resulting recurrence relation for T(n) is

$$T(n) = \begin{cases} b, & n \le 2 \\ 7T(n/2) + a\,n^2, & n > 2 \end{cases}$$

where a and b are constants.
Working with this formula we get

$$\begin{aligned} T(n) &= an^2(1 + 7/4 + (7/4)^2 + \dots + (7/4)^{k-1}) + 7^k T(1) \\ &\leq cn^2 (7/4)^{\log_2 n} + 7^{\log_2 n}, \quad c \text{ a constant} \\ &= cn^{\log_2 4 + \log_2 7 - \log_2 4} + n^{\log_2 7} \\ &= O(n^{\log_2 7}) \approx O(n^{2.81}) \end{aligned}$$

Dimension	Required Time	
	Divide-and-Conquer Algorithm	**Strassen Multiplication Algorithm**
2x2		
10x10		
50x50		
etc		

Table 2

Report should contain:

1. Machine configuration.
2. Complete Table 1.
3. Plot graphical output [x-axis=number of data, y-axis=Required Time][Using any Language].
4. Plot bar graph [using Microsoft Office Excel].
5. Properly analysis of **table** and **graph**.
6. Derive Time complexity of these algorithms and prepare proper analysis of those complexities.

Convex Hull:
Some Geometric Primitives:

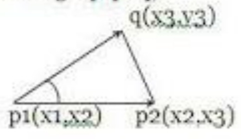
The area of ABC triangle,

$$= \frac{1}{2} \left| \det \begin{pmatrix} x1 & x2 & x3 \\ y1 & y2 & y3 \\ 1 & 1 & 1 \end{pmatrix} \right|$$

= ½*(x2y3-x3y2 – x1y3+x3y1+x1y2-x2y1)
= ½*((x2-x1)(y3-y1)-(x3-x1)(y2-y1)).

Q1. Let there are two point p1 and p2. Now draw a line through p1 and p2. If we have a point q, we will determine that q is to the left or right of p1p2 line.

Solution: Draw a triangle p1p2q,

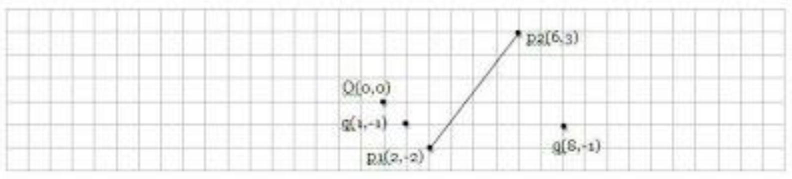


Determine,

The area of the triangle, Δ=½*((x2-x1)(y3-y1)-(x3-x1)(y2-y1)).

If Δ> 0, then q is to the left of p1p2 line.
If Δ<0, then q is to the right of p1p2 line.
If Δ=0, then q is on the p1p2 line.

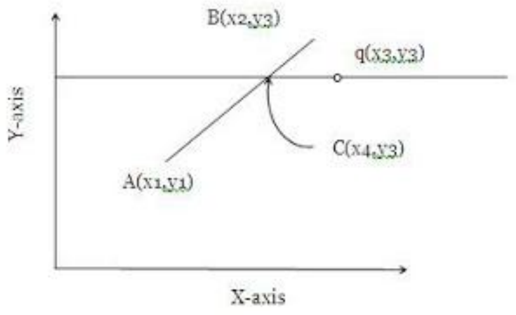
Example:



See the above figure, first determine the location of point q(8,-1) respect to p1p2 line.
Let p1(x1,y1) = p1(2,-2), p2(x2,y2)=(6,3), q(x3,y3) =(8,-1)
The area of triangle p1p2q, = ½* ((6-2)(-1+2) –(8-2)(3+2)) = ½*(4-30) = -13
So q(8,-1) is to the right of the line p1p2.
Now determine the location of point q(1,-1) respect to p1p2 line.
The area of triangle p1p2q, = ½* ((6-2)(-1+2) –(1-2)(3+2)) = ½*(4+5) = 9
So q(1,-1) is to the left of the line p1p2.

[Note: Code for above **Geometric Primitives]**

Q1. Let there are two point p1 and p2. Now draw a line through p1 and p2. If we have a point q, we will determine that q is to the left or right of p1p2 line.
A solution has been given in previous post (see the above link). Now I will discuss another simple solution.



From above figure, a line goes through A and B points. We want to determine that q to the left of AB line or not. We draw a parallel line of X-axis. This line intersect the point C(x4,y3). If x4 > x3 then, the point q is not to the left of the line AB.
We know the equation of the line passing through the points A and B is,

$$y - y_1 = \left(\frac{y_1 - y_2}{x_1 - x_2}\right)(x - x_1)$$

Let, $\boxed{m = \left(\frac{y_1 - y_2}{x_1 - x_2}\right)}$ then

$$y - y_1 = m(x - x_1)$$

Or $x = \dfrac{y - y_1 + m x_1}{m}$

Or $x = \frac{y-c}{m}$ where $\boxed{c = y_1 - mx_1}$

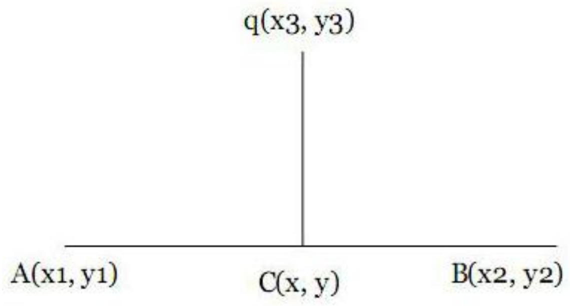
Now we find,

$x_4 = \frac{y_3 - c}{m}$, now check $x_4 > x_3$, $q(x_3, y_3)$ is to the left of AB line.

For Example, A(5,4) and B(16, 15) and q (10,10)

[Note: Code for above **Geometric Primitives]**

Now let A(x1,y1) and B(x2,y2) are two point. We draw a line passing through A and B. we have a point q(x3, y3). We have to determine the perpendicular distance from the point q to AB line.



Here Cq is the perpendicular on the line AB. The equation of line AB is

$$y - y_1 = \left(\frac{y_1 - y_2}{x_1 - x_2}\right)(x - x_1)$$

Let, $\boxed{m = \left(\frac{y_1 - y_2}{x_1 - x_2}\right)}$ then

$$y - y_1 = m(x - x_1)$$

Or $mx - y + (y_1 - mx_1) = 0$

Let, $\boxed{c = y_1 - mx_1}$

So $mx - y + c = 0$

Since Cq is the perpendicular of AB line, so the equation of line Cq is,

$$-x - my + k = 0$$
$$\text{or} \quad x + my - k = 0$$

Cq line passes through the point (x3,y3),
So

AB line and Cq line intersect at point C(x,y).
By solving below equation, we find,

The perpendicular distance from C(x,y) to q(x3,y3) is,

$$= \sqrt{(x3 - x)^2 + (y3 - y)^2}$$

[Note: Code for above **Geometric Primitives]**

Determining Whether a Point Is Inside a Complex Convex Polygon:

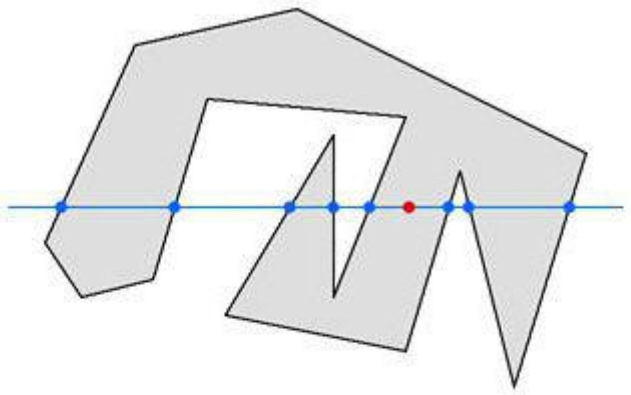


Figure 1

Figure 1 demonstrates a typical case of a severely concave polygon with 14 sides. The red dot is a point which needs to be tested, to determine if it lies inside the polygon. The solution is to compare each side of the polygon to the Y (vertical) coordinate of the test point, and compile a list of nodes, where each node is a point where one side crosses the Y threshold of the test point. In this example, eight sides of the polygon cross the Y threshold, while the other six sides do not. Then, if there are an odd number of nodes on each side of the test point, then it is inside the polygon; if there are even numbers of nodes on each side of the test point, then it is outside the polygon. In our example, there are five nodes to the left of the test point, and three nodes to the right. Since five and three are odd numbers, our test point is inside the polygon.

Straight forward Rules:
(1)

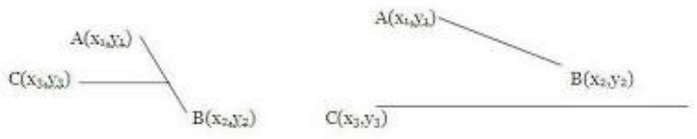


Figure 1 Figure 2

In figure 1, we see that if we draw a line from +infinite to -infinite through point C, which intersect the line AB. You check just

```
min(y1,y2) <= y3<max(y1,y2).
```

In figure 2, we see that if we draw a line from +infinite to -infinite through point C, which does not intersect the line AB

(2) Now check, the position of point C with respect to line AB. (discuss it previous above paragraph).

(3) If there are an odd number of intersect points on each side of the test point C, then it is inside the polygon; if there are even numbers of intersect points on each side of the test point C, then it is outside the polygon.

[Note: Code for above **Geometric Primitives]**

Recommended Exercise:
Programming Exercises of Chapter 3: "Divide-And-Conquer" of "Fundamentals of Computer Algorithm", Ellis Horowitz, Sartaj Sahni, Sanguthevar Rajasekaran.