$$F(n) = \frac{1}{\sqrt{5}} \left(\phi^n - \hat{\phi}^n \right)$$

$$\phi^n = \frac{1+\sqrt{5}}{2} = 1.618$$

$$\hat{\phi}^n = \frac{1-\sqrt{5}}{2} = -0.61 \quad \left[\text{between } -1 \text{ and } 0 \right]$$

F(n) grows exponentially.

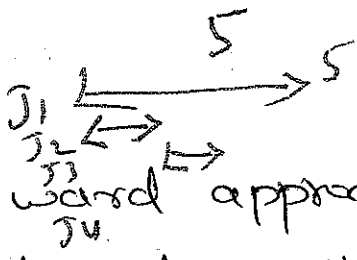
$$F(n) = 1/\sqrt{5}\, \phi^n$$

Algorithm

```
F(n)
  if n ≤ 1 return n
  else return F(n-1) + F(n-2)
```

UNIT-II

Brute force



Brute force is a straight forward approach to solving a problem, usually directly based on the problem's statement and definitions of the concepts involved.

Ex: $a^n = \underbrace{a * a \cdots * a}_{n \text{ times}}$

1. closest pair

To find the two closest points in a set of n points.

points (x, y), cartesian coordinates

The distance between two points $P_i = (x_i, y_i)$ and $P_j = (x_j, y_j)$

The standard eq. euclidean distance

$$d(P_i, P_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$

Brute force: compute the distance between each pair of distinct points and find a pair with the

Smallest distance..

BruteForceclosestPoints (P)

// Input: A list P of n ($n \geq 2$) points $P_1 = (x_1, y_1), \ldots,$

$P_n (x_n, y_n)$.

//output: index1 and index2 of the closest pair of points

dmin = ∞

For i = 1 to n-1 do

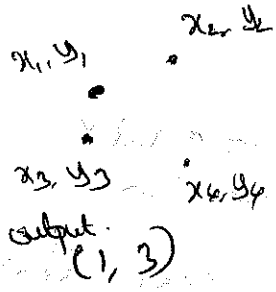
For j = i+1 to n do

$d = \text{sqrt}\left((x_i - x_j)^2 + (y_i - y_j)^2\right)$

If $d < d_{min}$

$d_{min} = d$, index1 = i, index2 = j

return index1, index2



Basic operation: Squaring a number, $(x_i - x_j)^2 +$ $(y_i - y_j)^2$, For every value of i and j, squaring two times. operation is performed two times.

Analysis

C(n) = no of times squaring operation is performed

$$C(n) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} 2$$

$$= 2 \sum_{i=1}^{n-1} (n-i+1+i) = 2 \sum_{i=1}^{n-1} (n-1+1)$$

$$= 2 \left[\sum_{i=1}^{n-1} n - \sum_{i=1}^{n-1} i \right]$$

$$\sum_{i=1}^{n} i = 1+2+3+\cdots+i = \frac{n(n+1)}{2}$$

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

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

$$= 2n^2 - 2n - \frac{n^2}{2} + n/2 \qquad = n^2 - n \pm n(n-1)$$

$$= O(n^2)$$

$$= \frac{4n^2 - n^2}{2} + n\left[\frac{1}{2}\right] - 1$$

$$\pm \frac{3n^2}{2} + n \cdot (+1/2)$$

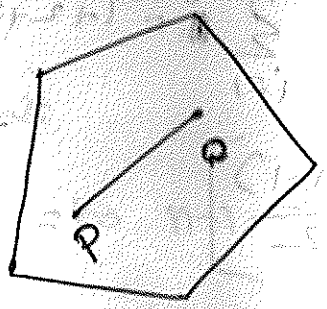
$$C(n) = \frac{3n^2}{2} - \frac{3n}{2} = O(n^2)$$

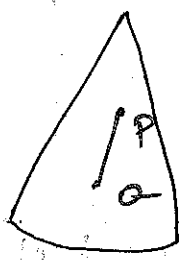
$$C(n) = O(n^2)$$

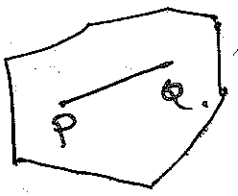
Convex-Hull problem

Convex-Set

A set of points on the plane is called convex if for any two points P and Q in the set, the entire line segment with the end points at P and Q belongs to the set.

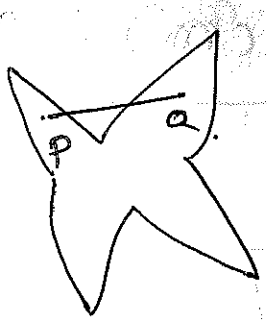


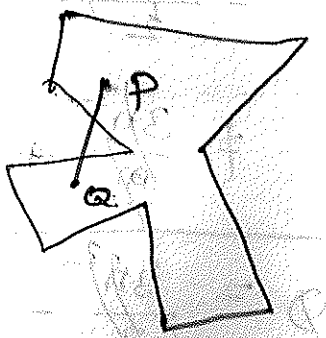




all P and Q belongs to the set.

(a) Convex Sets





PQ not belongs to the set

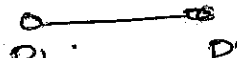
(b) Sets that are not convex.

convex hull

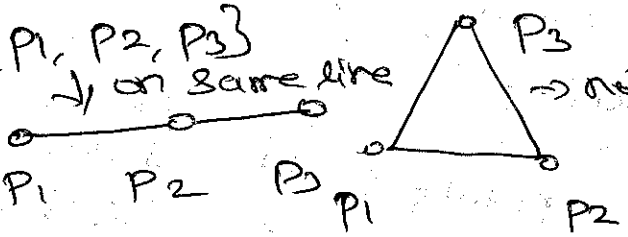
The convex hull of a set of n points in the plane is the smallest convex polygon that contains all of them (either in side or on its boundary.

Definition The convex hull of a set S of points is the smallest convex set containing S.

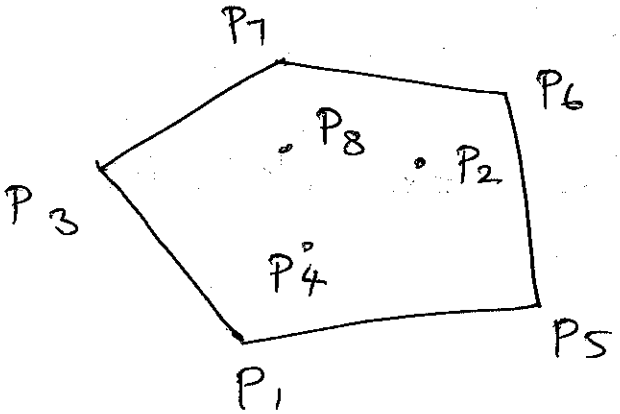
$S = \{P_1, P_2\}$



$S = \{P_1, P_2, P_3\}$



$S = \{P_1, \ldots P_8\}$



The convex hull for this set of eight points is the convex polygon with its vertices P_1, P_5, P_6, P_7 and P_3

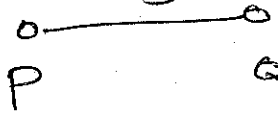
Theorem

The convex hull of any set S of $n \geq 2$ points (not all on the same line) is a convex polygon with the vertices at some of the points of S. If all the points do lie on the same line, the polygon degenerates to a line segment but still with the end points at two points of S. The convex-hull problem is the problem of constructing the convex hull for a given set S of n points. To solve it we need to find the extreme points that will serve as the vertices of the polygon.

extreme points

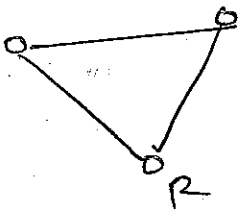
→ end points of boundary line segment

line segment



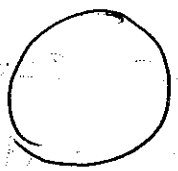
$EP = \{P, Q\}$

Triangle P



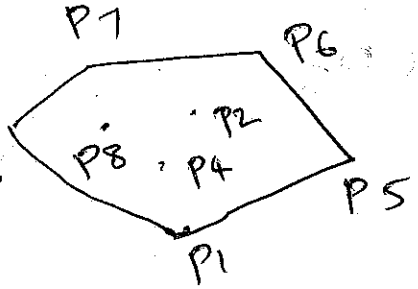
$EP = \{P, Q, R\}$

circle



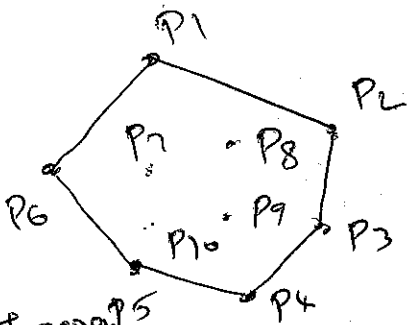
→ all points of its circumference

polygon vertices



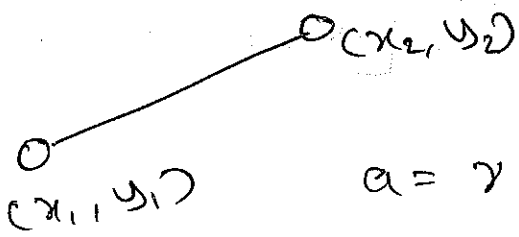
$EP = \{P1, P5, P6, P2, P3\}$

Points → line segments making up a boundary of the convex hull.



Theorem 5

A line segment connecting two points P_1 and P_2 is a part of the convex hull's boundary if and only if all the other points in the set lie on the same side of the line drawn through these points.



For all points above the line, $ax + by > c$,

for all points below the line $ax + by < c$.

The straight line through ~~these~~ two points (x_1, y_1) & (x_2, y_2) is defined by

$$ax + by = c$$

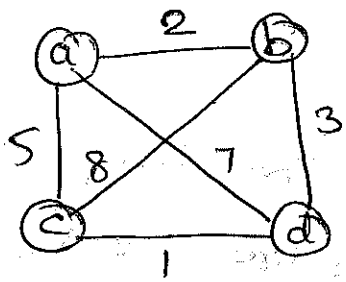
where $a = y_2 - y_1$ $b = x_1 - x_2$, $c = x_1 y_2 - y_1 x_2$

To check whether contain points lie on the same side of the line, simply check whether the expression $ax+by-c$ has the same sign at each of these points.

Efficiency of ~~the algorithm~~ is $O(n^3)$.

Exhaustive Search : It is simply a brute-force approach to combinatorial problems. (permutations, combinations (or) subsets). It suggests generating each and every element of the problem's domain, selecting those of them that satisfy the problem's constraints, and then finding a desired element.

Travelling Salesman problem



→ To find the shortest tour through a given set of n cities that visits each city exactly once.

Tour	Length
$a \to b \to c \to d \to a$	$l = 2+8+1+7 = 18$
$a \to b \to d \to c \to a$	$l = 2+3+1+5 = 11$ optimal
$a \to c \to b \to d \to a$	$l = 5+8+3+7 = 23$
$a \to c \to d \to b \to a$	$l = 5+1+3+2 = 11$ optimal
$a \to d \to b \to c \to a$	$l = 7+3+8+5 = 23$
$a \to d \to c \to b \to a$	$l = 7+1+8+2 = 18$

Knapsack problem → subsets

Given n items of known weights $w_1 \ldots w_n$ and values $v_1, \ldots v_n$ and a knapsack capacity W, find the most valuable subset of the items that fit into the knapsack.

item	weight	value
1	7	42
2	3	12
3	4	40
4	5	25

1	7	42
2	3	12
3	4	40
4	5	25

Subset	Total weight	Total value
{1}	7	42
{2}	3	12
{3}	4	40
{4}	5	25
{1,2}	10	54
{1,3}	11	82 (not feasible)
{1,4}	12	67 (nf
{2,3}	7	52
{2,4}	8	37
{3,4}	9	65
{1,2,3}	14	nf
{1,2,4}	15	nf
{1,3,4}	16	nf
{1,4,2}	19	nf
{1,2,3,4}	19	nf

Subset = {3,4}, Value = 65

3) Assignment problem [permutations] J - Job, P - person.

	J1	J2	J3	J4
P1	9	2	7	8
P2	6	4	3	7
P3	5	8	1	8
P4	7	6	9	4

Job

$$C = \begin{array}{c} \\ P1 \\ P2 \\ P3 \\ P4 \end{array} \begin{array}{c} \begin{array}{cccc} J1 & J2 & J3 & J4 \end{array} \\ \begin{bmatrix} 9 & 2 & 7 & 8 \\ 6 & 4 & 3 & 7 \\ 5 & 8 & 1 & 8 \\ 7 & 6 & 9 & 4 \end{bmatrix} \end{array}$$

persons

C[i,j] → ith person to jth Job.

one

→ Find the assignment with the smallest total cost

permutations

⟨1, 2, 3, 4⟩ cost = 9 + 4 + 1 + 4 = 18

↑ ↑ ↑ ↑
persons 1 2 3 4

⟨1, 2, 4, 3⟩ cost = 30

⟨1, 3, 2, 4⟩ cost = 24

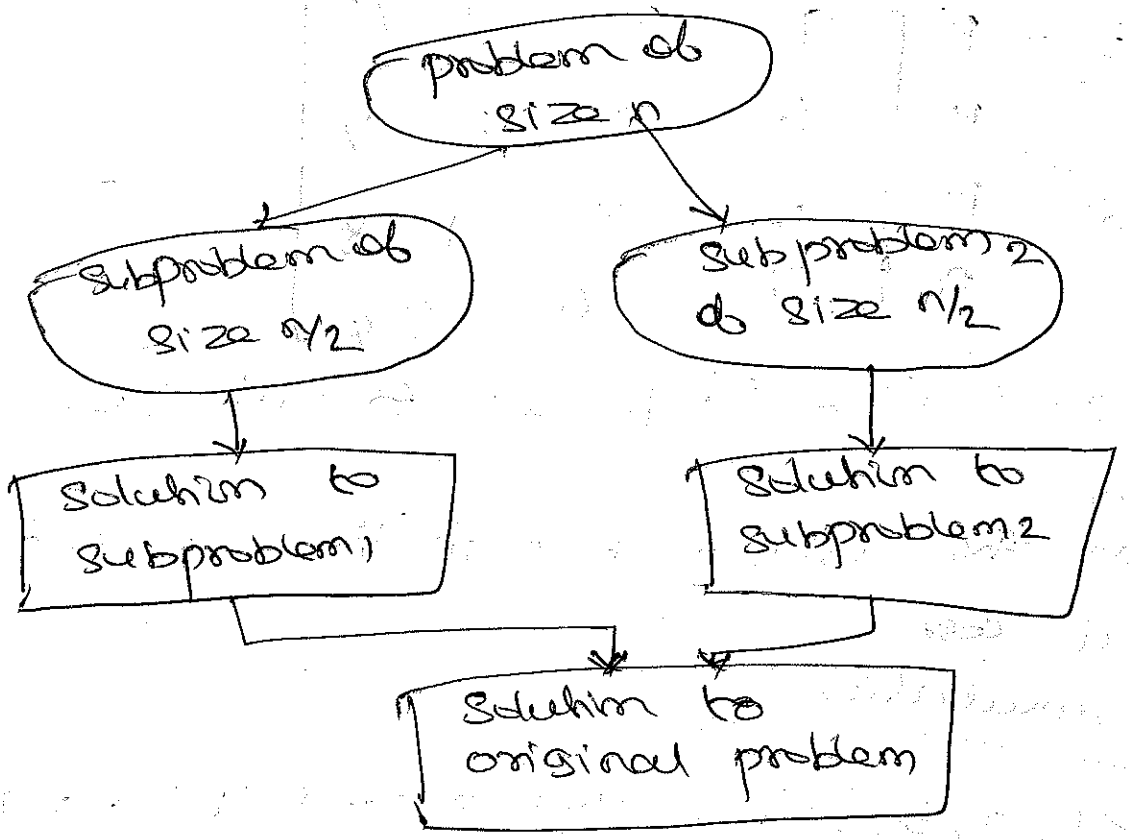
⟨1, 3, 4, 2⟩

etc

Divide-and-Conquer

3 steps

1. Divide → divide the problem into small problems
2. Solve → solve the subproblem
3. combine → combine the solution of small problem's to get the solution to the original problem.



Merge sort

It sorts a given array $A[0..n-1]$ by dividing it into two halves $A[0..\lfloor n/2 \rfloor - 1]$ and $A[\lfloor n/2 \rfloor .. n-1]$, sorting each of them recursively, and then merging the two smaller sorted arrays into a single sorted one.

```
Algorithm Mergesort (A[0..n-1])
// Sorts array A[0..n-1] by recursive
// Input: A[0..n-1] of orderable elements
//output: A[0..n-1] sorted in non-decreasing
          order

if n>1
    copy A[0.. ⌊n/2⌋-1] to B[0..⌊n/2⌋-1]
    copy A[⌊n/2⌋...n-1] to C[0...⌊n/2⌋-1]
    Mergesort (B[0..⌊n/2⌋-1)
    Mergesort (C[0..⌊n/2⌋-1)
    Merge (B, C, A)
```

Algorithm Merge (B[0..p-1], C[0..q-1], A[0..p+q-1])

// merge two sorted arrays into one sorted array.

// Input: B and C sorted.

// Output: sorted array A of elements B & C

i=0, j=0, k=0

while i<p and j<q do

if B[i] ≤ C[j]

A[k] = B[i]; i=i+1

else A[k] = C[j]; j=j+1

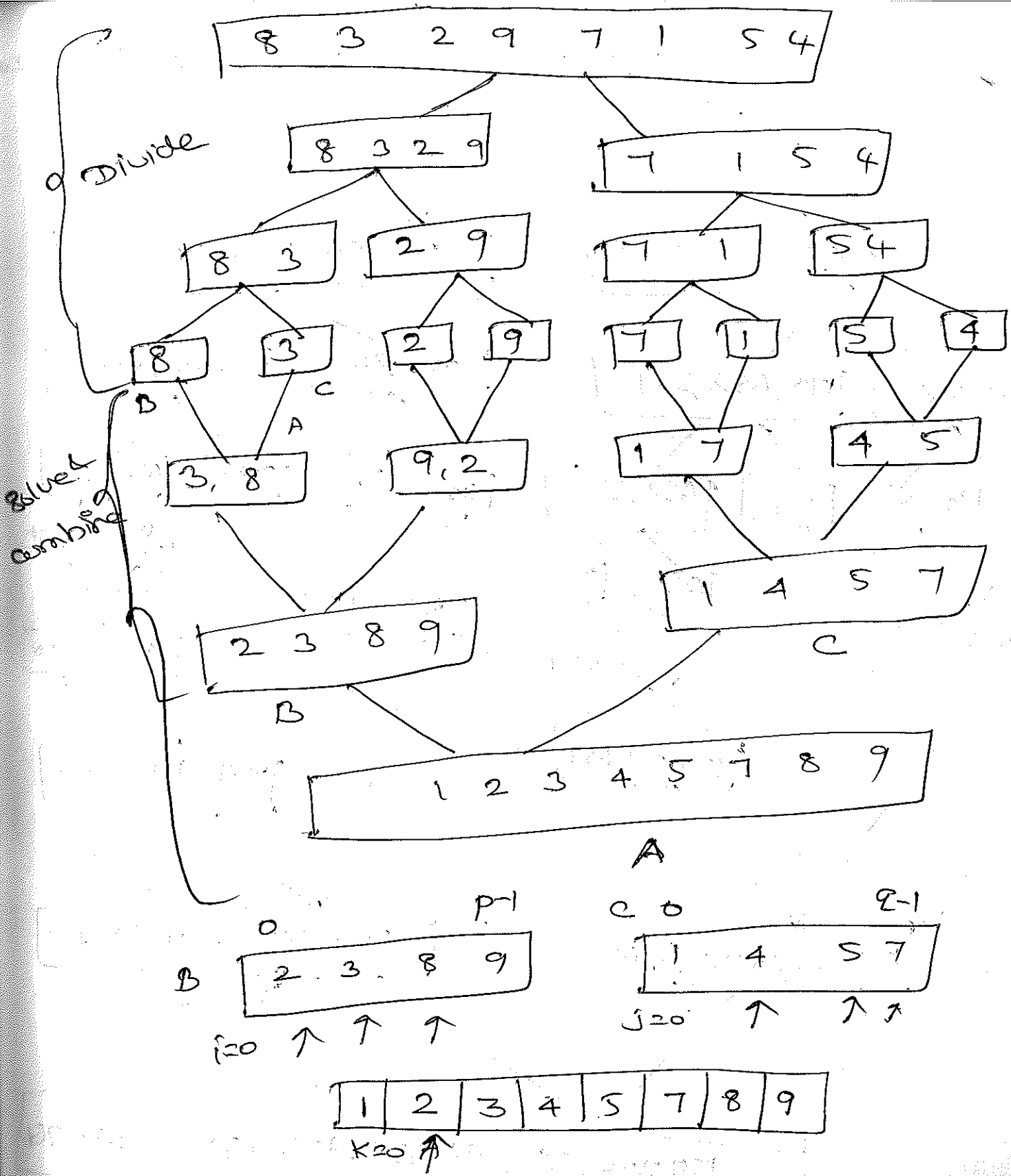
k=k+1

if i = p

copy C[j...q-1] to A[k...p+q-1]

else copy B[i...p-1] to A[k...p+q-1]

Ex: 8 3 2 9 7 1 5 4

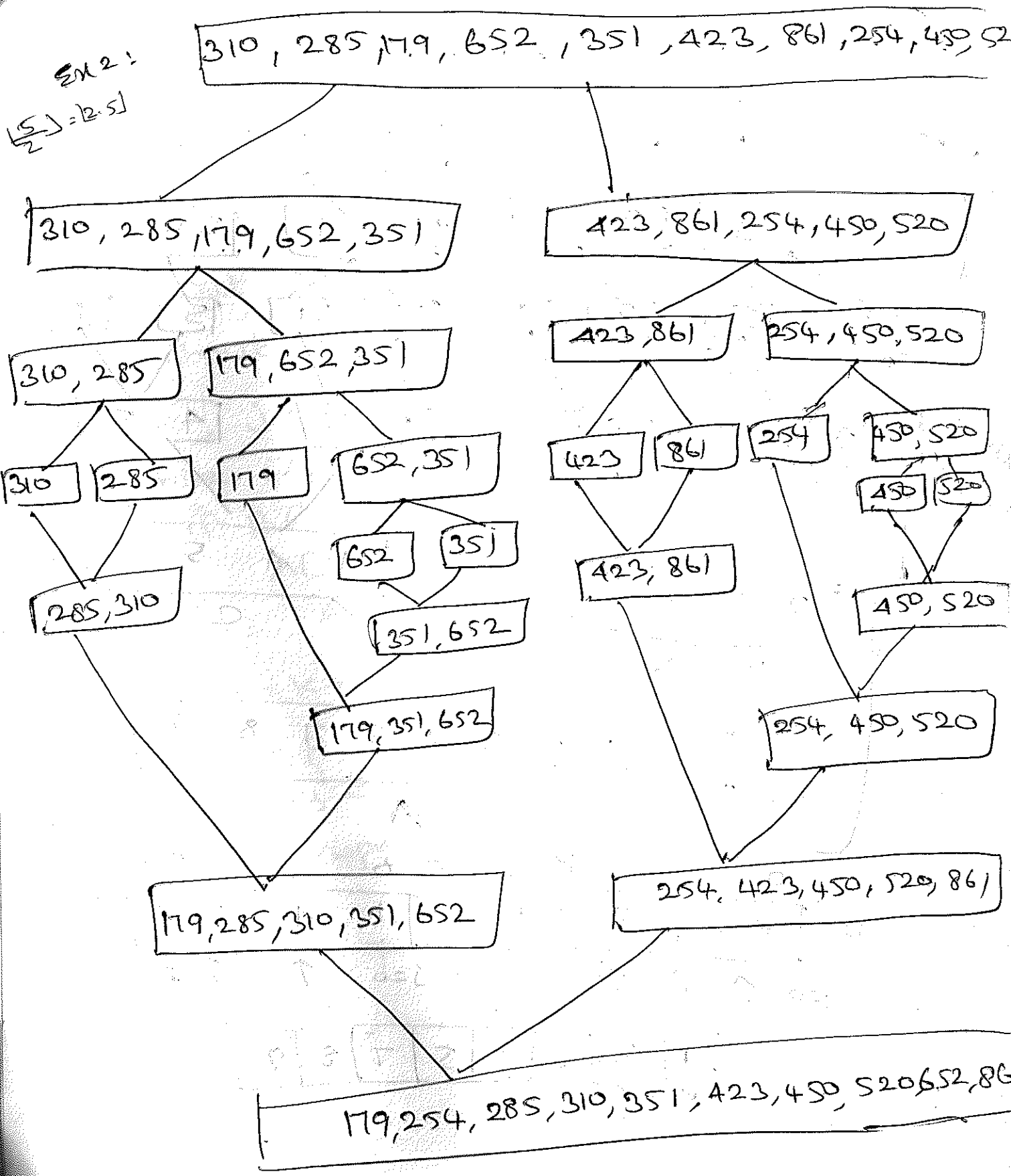

8 3 2 9 7 1 5 4
Divide
8 3 2 9
7 1 5 4
8 3
2 9
7 1
5 4
8
3
2
9
7
1
5
4
B
C
A
3, 8
9, 2
1 7
4 5
Bluet combine
2 3 8 9
1 4 5 7
B
C
1 2 3 4 5 7 8 9
A

0
p-1
B
2 3 8 9
i=0
c 0
q-1
1 4 5 7
j=0

1 2 3 4 5 7 8 9
k=0

Ex 2:

$\lfloor \frac{5}{2} \rfloor = \lfloor 2.5 \rfloor$



Analysis

The recurrence relation for the number of key comparisions $C(n)$ is

$$C(n) = 2C(n/2) + C_{merge}(n), \quad n>0, \quad C(1)=0$$

$C_{merge}(n)$ → number of key comparisions performed during the merging stage.

$$C_{merge}(n) = n-1$$

Worst case → [Smallest element may come from alternating arrays]

$$C_{worst}(n) = 2C_{worst}(n/2) + n-1, \quad n>1,$$

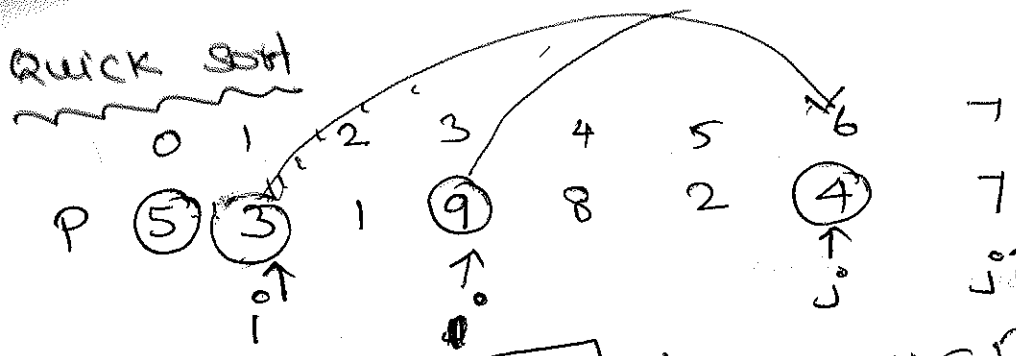
According to master theorem

$$C_{worst}(n) = n\log_2 n - n + 1$$

$$\boxed{C_{worst}(n) = O(n \cdot \log n)}$$

Apply M.S to sort the list E, X, A, M, P, L, E in alphabetical order.

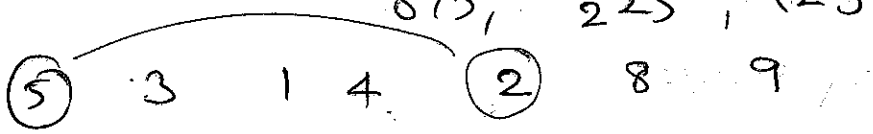
Quick sort



3<5, 1<5, 9>5 stop 7>5, 4<5 stop

i<j, swap A[i] & A[j], increment i & j

8>5, 2<5, i<j, swap A[i] & A[j]



j i

i>j

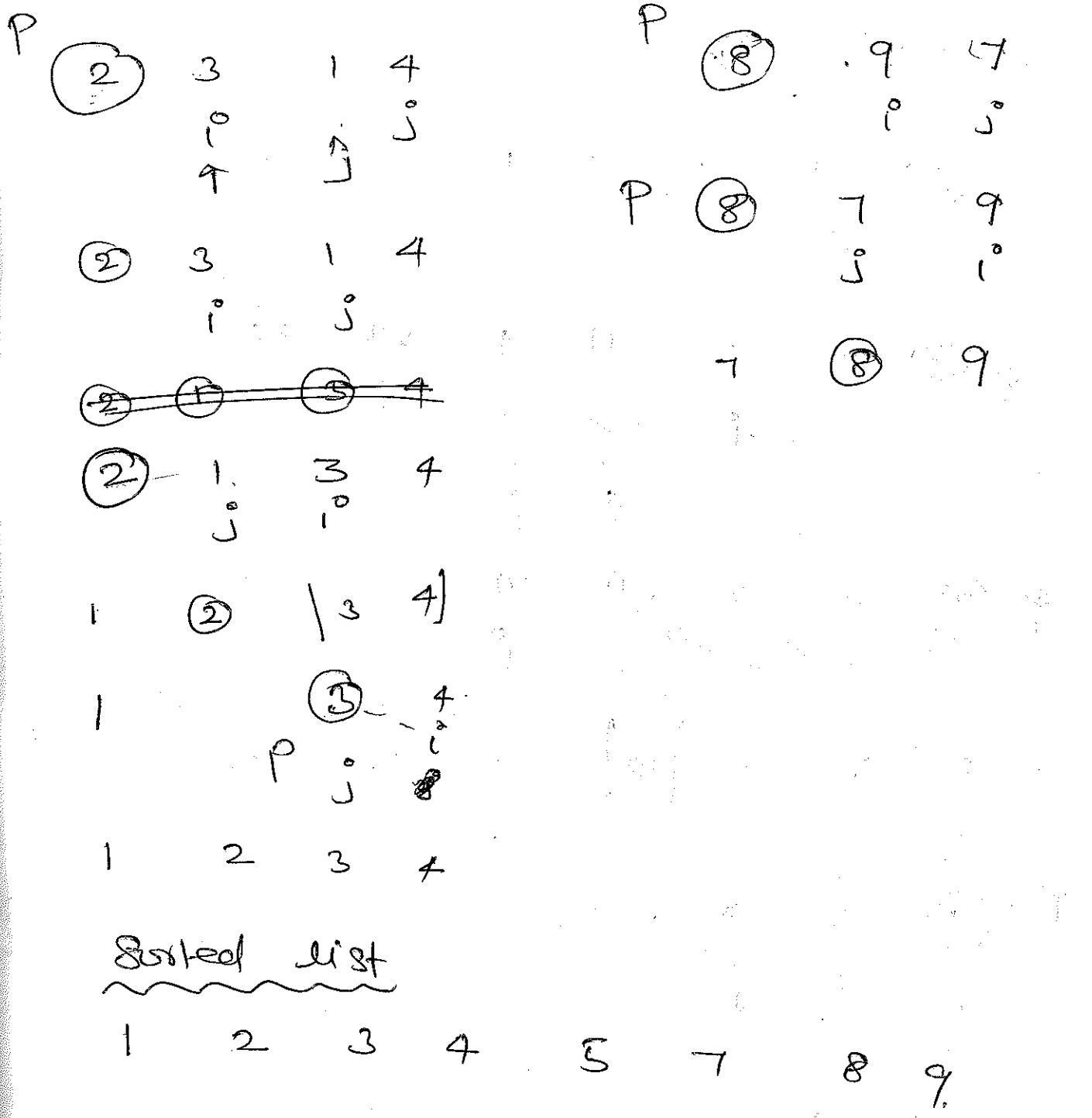
if i crossed j, swap pivot with A[j]

0	1	2	3	4 Pivot	5	6	7
2	3	1	4	5	8	9	7

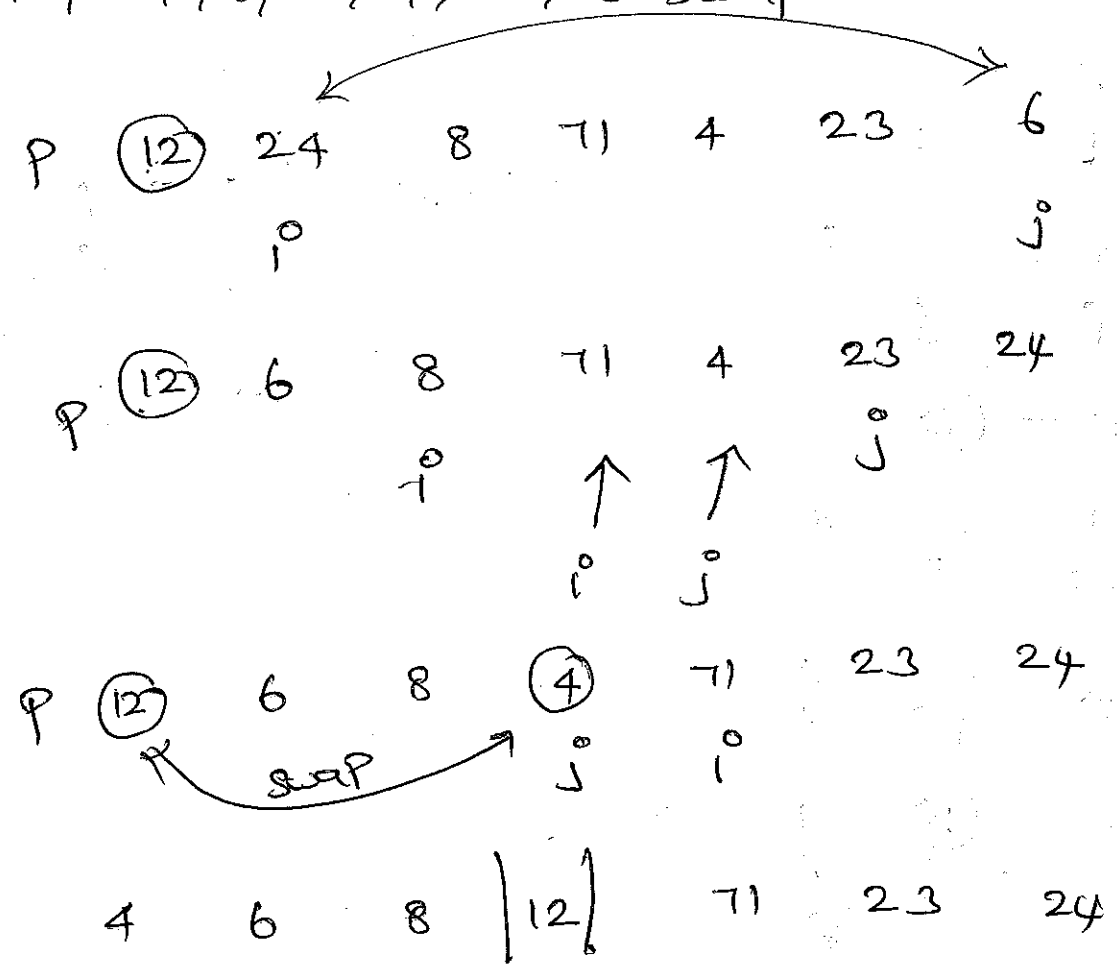
pivot element (5) final position in the sorted list is 4. All the elements in the left subarrays is less than pivot. All element in the right subarray is greater than pivot.

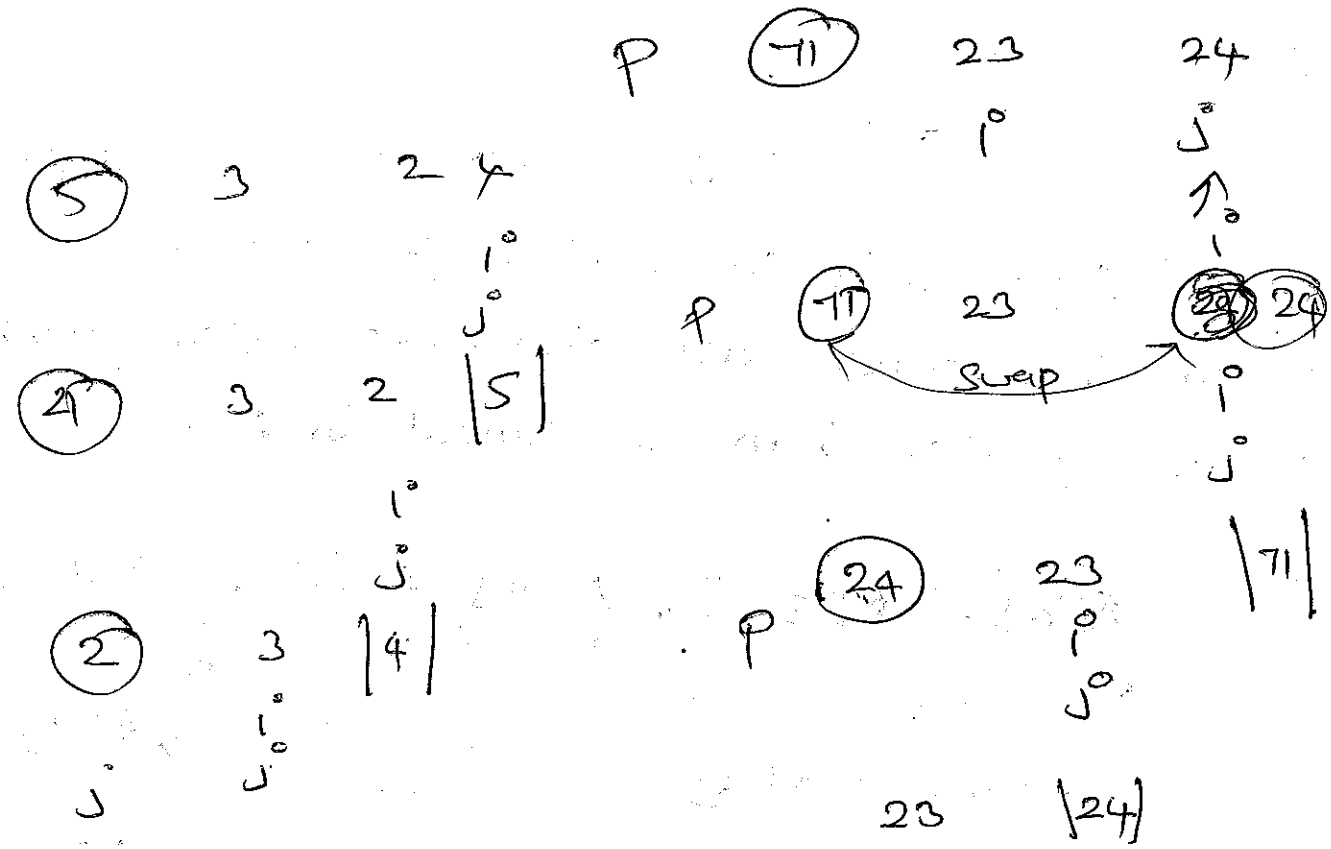
2, 3, 1, 4 < 5,

8, 9, 7 > 5

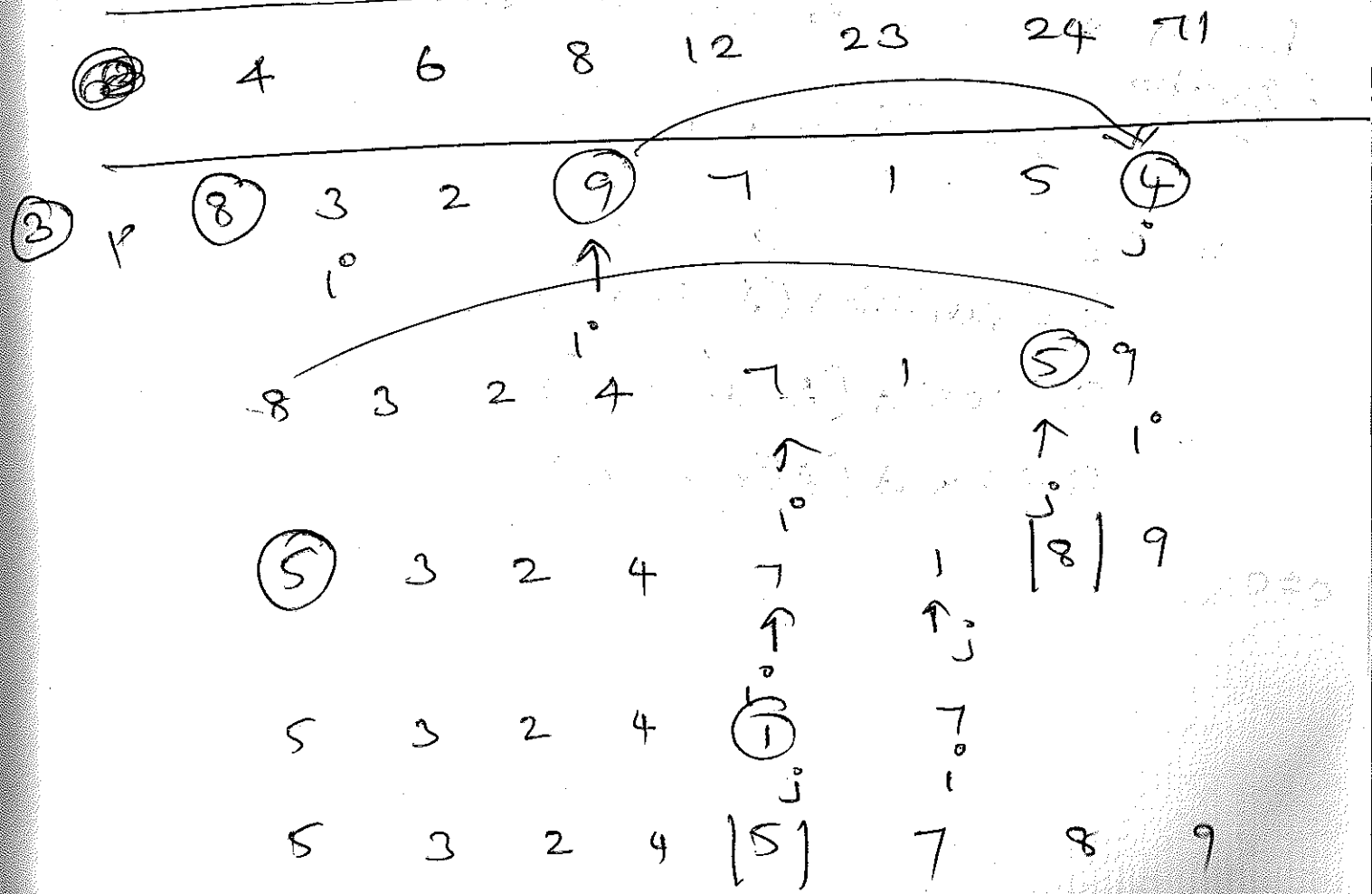

P
2 3 1 4
i j
j
2 3 1 4
i j
2 1 3 4
j i
1 2 3 4
P
i
j
1 2 3 4
Sorted list
1 2 3 4 5 7 8 9
P
8 9 7
i j
P 8 7 9
j i
7 8 9

(2) 12, 24, 8, 71, 4, 23, 6 swap



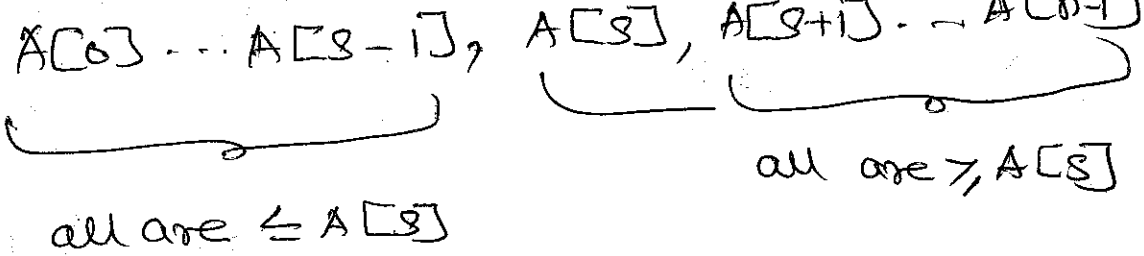

P
71
23
24
i
j
5
3
2
4
4
3
2
5
2
3
4
swap
24
23
71

Sorted list


2
4
6
8
12
23
24
71
3
P
8
3
2
9
7
1
5
4
8
3
2
4
7
1
5
9
5
3
2
4
7
1
8
9
5
3
2
4
1
7
5
3
2
4
5
7
8
9

Quick sort

It rearranges elements of a given array $A[0..n-1]$ to achieve its partition, a situation where all elements after partition s are greater than or equal to $A[s]$



After a partition has been achieved, A[s] will be in its final position in the sorted array.

Algorithm

Quick Sort (A[l..r])

```
If l < r
    s = partition (A[l..r])
    Quicksort (A[l..s-1])
    Quicksort (A[s+1....r])
```

partition

```
partition (A[l..r])
  p = A[l]
  i = l ; j = r+1
  repeat
      repeat i = i+1 until A[i] ≥ p
      repeat j = j-1 until A[j] ≤ p
      swap (A[i], A[j])
  until i ≥ j
  swap (A[i], A[j])
  swap (A[l], A[j])
  return j.
```

Analysis

Best case

all the splits happen in the middle of corresponding sub arrays, called best case

$$C_{best}(n) = 2C_{best}(n/2) + n, \quad n > 1, \quad C_{best}(1) = 0$$

According to Master theorem

$$C_{best}(n/2) = n \log_2 n$$

$C_{best}(n) = n \cdot \log n + n$

$$C_{best}(n) = n \cdot \log n$$

Worst case

All the splits will be skewed to the extreme; one of the two subarrays will be empty while the size of the other will be just one less than the size of a subarray being partitioned.

$$C_{worst}(n) = (n+1) + n + \cdots + 3 = \frac{(n+1)(n+2)}{2} - 3$$

$$C_{worst}(n) = n^2$$

Average case

Split can happen in each position s, $(0 \le s \le n-1)$ with the same probability $1/n$.

$$C_{avg}(n) = \frac{1}{n} \sum_{s=0}^{n-1} [(n+1) + C_{avg}(s) + C_{avg}(n-1-s)] \quad n \ge$$

$$C_{avg}(n) = 1.38 n \cdot \log_2 n$$

Time complexity of quick sort

Best case	Average case	Worst case
$n \cdot \log n$	$1.38 \cdot n \cdot \log_2 n$	$n2$

Binary Search

It works by comparing a search key K with the array's middle element A[m]. If they match, the algorithm stops; the same operation is repeated recursively for the first half of the array if K < A[m] and for the second half if K > A[m].

precondition: i/p in Ascending order.

A[0 ... m-1] A[m], A[m+1] ... A[n-1]

Search here if K < A[m] ⇕ K. Search here if K > A[m]

	0	1	2	3	4	5	6	7	8	9	10	11	12
(1)	3	14	27	31	39	42	55	70	74	81	85	93	98

K = 70

$$m = \left\lfloor \frac{l+r}{2} \right\rfloor = \frac{0+12}{2} = 6$$

K ≠ A[6], K > A[m]

	7	8	9	10	11	12
	70	74	81	85	93	98

$$m = \left\lfloor \frac{7+12}{2} \right\rfloor = \frac{19}{2} = 9.5$$

m = 9

A[m] ≠ K.

K < A[m]

	7	8
	70	74

m = 15/2 = 7.

K = A[m], K found in A[m]. o/p : 7

Algorithm

Binary Search (A[0...n-1], K)

```
l=0, r=n-1
while l<r do
    m=(l+r)/2
    if K=A[m] return m
    else if K<A[m] r=m-1
    else l=m+1
return -1
```

Analysis

worst case

(i) element not found

(ii) element is last location.

$C_w(n) = C_w(\lfloor n/2 \rfloor) + 1, \quad n > 1 \quad C_w(1) = 1$

$\frac{n}{2^k} = 1 \Rightarrow n = 2^k$ $\boxed{k = \log n}$

$C_w(n) = \log n + 1$

$\boxed{C_w(n) = \Theta(\log n)}$

$C_{avg}(n) = \log n$

Method 2

no. of mul = 4

2 3 x 1 4

92

23

322

Multiplication of large integers

→ t

Method I

$23 = 2 \cdot 10^1 + 3 \times 10^0$ $\quad 14 = 1 \times 10^1 + 4 \times 10^0$

$23 \times 14 = (2 \cdot 10^1 + 3 \times 10^0) * (1 \times 10^1 + 4 \times 10^0)$

$= (2*1)\,10^2 + (3*1 + 2*4)\,10^1 + (3*4)\,10^0$

$23 * 14 = 322$

no. of multiplication = 4. $[n^2]$ $2^2 = 4$

20
240
120
400

25
14
18

200
110
12
322

Method III

$a = a_1 a_0$ $\quad b = b_1 b_0$

→ two n-digit number
→ n is even

$c = a * b = c_2 10^2 + c_1 * 10^1 + c_0$

$c_2 = a_1 * b_1$

$c_0 = a_0 * b_0$

$c_1 = (a_1 + a_0) * (b_1 + b_0) - (c_2 + c_0)$

$5 * 5 - 14$

$c_2 = 2$ $\quad c_0 = 12$

$c_1 = 11$

$c = 2 \cdot 10^2 + 11 \cdot 10^1 + 12$

$= 200 + 110 + 12$

$c = 322$

25-17

[c = c1 c0

23 14

a1 a0 b1 b0

$c_2 = 2$

$c_0 = 12$

$c_1 = 5 * 5 - (14$

200
110
12
322

$$c = a*b = (a_1 10^{n/2} + a_0) * (b_1 10^{n/2} + b_0)$$

$$= (a_1 * b_1) 10^{n} + (a_1 * b_0 + a_0 * b_1) 10^{n/2}$$

$$+ (a_0 * b_0)$$

$$\boxed{c = c_2 10^{n} + c_1 10^{n/2} + c_0}$$

where

$c_2 = a_1 * b_1$ $\quad c_0 = a_0 * b_0$

$c_1 = (a_1 + a_0) * (b_1 + b_0) - (c_2 + c_0)$

→ n is even number, a & b are
→ of n digits

Recurrence equation

$$M(n) = 3M(n/2), \quad n > 1, \quad M(1) = 1$$

$$M(n) = 3^{\log_2 n} = n^{\log_2 3} = n^{1.585}$$ property

$$\boxed{M(n) = n^{1.585}} \qquad \left[a^{\log_b c} = c^{\log_b a} \right]$$

Ex: $\frac{12}{a_1}\ \frac{14}{a_0}$ × $\frac{32}{b_1}\ \frac{41}{b_0}$

$a_1 \rightarrow$ first half ($n/2$ digit)

$a_0 \rightarrow$ second half

$n = 4$

$c_2 = 12 \times 32 = 384$

$c_0 = 14 \times 41 = 574$

$c_1 = (26 \times 73) - (958)$

$= 1898 - 958$

$= 940$

$c = 3840000 + 940000 + 574$

$= 3934574$

3934574

Strassen's matrix multiplication

$$C = AB$$

A, B, C and 2 dimensional matrices

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \quad B = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$$

$$C = \begin{bmatrix} 1*5+2*7 & 1*6+2*8 \\ 3*5+4*7 & 3*6+4*8 \end{bmatrix} = \begin{bmatrix} 5+14 & 6+16 \\ 15+28 & 18+32 \end{bmatrix}$$

$$C = \begin{bmatrix} 19 & 22 \\ 43 & 50 \end{bmatrix}$$

No. of multiplications = 8.

$$\begin{bmatrix} c_{00} & c_{01} \\ c_{10} & c_{11} \end{bmatrix} = \begin{bmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{bmatrix} * \begin{bmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \end{bmatrix}$$

$$= \begin{bmatrix} m_1+m_4-m_5+m_7 & m_3+m_5 \\ m_2+m_4 & m_1+m_3-m_2+m_6 \end{bmatrix}$$

$$m_1 = (a_{00} + a_{11}) * (b_{00} + b_{11})$$

$$m_2 = (a_{10} + a_{11}) * b_{00}$$

$$m_3 = a_{00} * (b_{01} - b_{11})$$

$$m_4 = a_{11} * (b_{10} - b_{00})$$

$$m_5 = (a_{00} + a_{01}) * b_{11}$$

$$m_6 = (a_{10} - a_{00}) * (b_{00} + b_{01})$$

$$m_7 = (a_{01} - a_{11}) * (b_{10} + b_{11})$$

$$m_1 = 5 * 13 = 65$$

$$m_2 = 12 * 5 = 60$$

$$m_3 = 1 * (-2) = -2$$

$$m_4 = 4 * (7 - 5) = 8$$

$$m_5 = (3) * 8 = 24$$

$$m_6 = (2) * (11) = 22$$

$$m_7 = (-2) * (15) = -30$$

$$= \begin{bmatrix} 65+8-24-30 \end{bmatrix}$$

$$= \begin{bmatrix} \end{bmatrix}$$