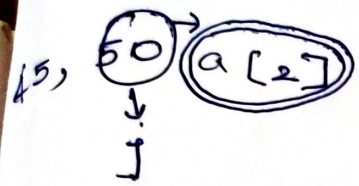
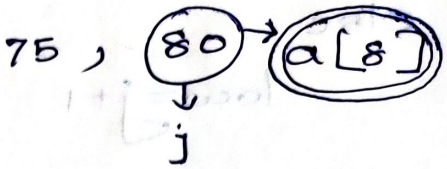




∴ The sorted array is

45,	50,	55,	60,	65,	70,	75,	80,	85
a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]

Selection Algorithm:

→ The problem is to find k^{th} smallest element in an array.

```
Algorithm Select1(a, n, k)
{
    loco = 1
    up = n+1
    a[n+1] = ∞
    repeat
    {
        j = partition(a, loco, up)
        if (k=j) then
            return
        else if (k<j) then
            up = j
```

```
else
    low = j+1
} until (false);
}
```

Ex: Given an array find k^{th} smallest element

a → (12), 8, 10, 14, 16, 19, 3 where k = 2

Solution: (12), 8, 10, 14, 16, 19, 3

↑ P

Partition (a, 1, 8)

i → 1 2 3 4

j → 8 7

12, 8, 10, 3, 16, 19, 14

i → 5

j → 4

3, 8, 10, 12, 16, 19, 14

↓ j

j = 4 > k = 2

up = 4

Partition (a, 1, 4)

a[--] 3, 8, 10, 12

i → 1 2

j → 4 3 2 1

(3) 8 10 12

j

j=1 < K=2

low = 2

partition (a, 2, 4)

(8), 10, 12

↓
P

j → 3

j → 2

(j=2) = (K=2)

MATCH FOUND

∴ a[2] is smallest i.e 8 //

Ex2: Find the 5th smallest element in the following array using selection algorithm

a → 65, 70, 75, 80, 85, 60, 55, 50, 45

Solution: loco → 1

up → 10

partition (a, 1, 10)

65, 70, 75, 80, 85, 60, 55, 50, 45

65, 45, 75, 80, 85, 60, 55, 50, 70

i → 3 65, 45, 50, 80, 85, 60, 55, 75, 70

j → 8

i → 4 65, 45, 50, 55, 85, 60, 80, 75, 70

j → 7

i → 5 65, 45, 50, 55, 60, 85, 80, 75, 70

j → 6

60, 45, 50, 55, (65), 85, 80, 75, 70

j

$(j = 5) = (k = 5)$

MATCH FOUND

∴ Smallest element at 5th position is $a[5] = 65$ //

Time Complexity

Average time Complexity ⇒ $T(n) = O(n)$

Worst case time Complexity ⇒ $T(n) = O(n^2)$

Strassen's Matrix Multiplication

Let A and B be $n \times n$ matrix, the product of both matrix C=AB requires a time complexity of $O(n^3)$

```
for (i → 0 to row)
{
    for (j → 0 to col)
    {
        mult[i][j]=0;
        for (k → 0 to col)
        {
            mult[i][j] = mult[i][j] +
            A[i][k] * B[k][j]
        }
    }
}
```

In strassen's matrix multiplication, we assume n as power of 2 [$n = 2^k$]. If there are not enough rows and columns, then rows and columns of zeroes are added to get matrix of size $n = 2^k$. If a matrix is divided into 4 part, each part is of size $\rightarrow n/4$

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

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

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

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

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

Since matrix multiplication are more expensive than matrix addition, Strassen reduced the number of computation to 7 multiplication, 18 addition or substraction

$$P = (A_{11} + A_{22})(B_{11} + B_{22})$$

$$Q = (A_{21} + A_{22})(B_{11})$$

$$R = A_{11}(B_{12} - B_{22})$$

$$S = A_{22}(B_{21} - B_{11})$$

$$T = (A_{11} + A_{12})B_{22}$$

$$U = (A_{21} - A_{11})(B_{11} + B_{12})$$

$$V = (A_{12} - A_{22})(B_{21} + B_{22})$$

$$C_{11} = P + S - T + V$$

$$C_{12} = R + T$$

$$C_{21} = Q + S$$

$$C_{22} = P + R - Q + U$$

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

by master's theorem

$$a > b^d$$

$$7 > 2^2$$

$$T(n) = \left(n^{\log_b a}\right)$$

$$= \left(n^{\log_2 7}\right)$$

$$= n^{2.81} \;//$$

Ex: Use Strassen's multiplication to multiply the matrix

$$A = \begin{bmatrix} 2 & 1 & 1 & 3 \\ 6 & 0 & 4 & 5 \\ 3 & 2 & 1 & 8 \\ 4 & 3 & 2 & 1 \end{bmatrix} \qquad B = \begin{bmatrix} 1 & 0 & 0 & 3 \\ 2 & 1 & 1 & 6 \\ 7 & 1 & 8 & 3 \\ 0 & 5 & 4 & 0 \end{bmatrix}$$

$$A_{11} = \begin{bmatrix} 2 & 1 \\ 6 & 0 \end{bmatrix} \quad A_{12} = \begin{bmatrix} 1 & 3 \\ 4 & 5 \end{bmatrix} \quad B_{11} = \begin{bmatrix} 1 & 0 \\ 2 & 1 \end{bmatrix} \quad B_{12} = \begin{bmatrix} 0 & 3 \\ 1 & 6 \end{bmatrix}$$

$$A_{21} = \begin{bmatrix} 3 & 2 \\ 4 & 3 \end{bmatrix} \quad A_{22} = \begin{bmatrix} 1 & 8 \\ 2 & 1 \end{bmatrix} \quad B_{21} = \begin{bmatrix} 7 & 1 \\ 0 & 5 \end{bmatrix} \quad B_{22} = \begin{bmatrix} 8 & 3 \\ 4 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 2 & 1 \\ 6 & 0 \end{bmatrix} + \begin{bmatrix} 1 & 8 \\ 2 & 1 \end{bmatrix} = \begin{bmatrix} 3 & 9 \\ 8 & 1 \end{bmatrix}$$

A_{11} A_{22}

$$\begin{bmatrix} 1 & 0 \\ 2 & 1 \end{bmatrix} + \begin{bmatrix} 8 & 3 \\ 4 & 0 \end{bmatrix} = \begin{bmatrix} 9 & 3 \\ 6 & 1 \end{bmatrix}$$

$$P = \begin{bmatrix} 3 & 9 \\ 8 & 1 \end{bmatrix} \begin{bmatrix} 9 & 3 \\ 6 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 81 & 18 \\ 78 & 25 \end{bmatrix}$$

$$\begin{bmatrix} 3 & 2 \\ 4 & 3 \end{bmatrix} + \begin{bmatrix} 1 & 8 \\ 2 & 1 \end{bmatrix} = \begin{bmatrix} 4 & 10 \\ 6 & 4 \end{bmatrix}$$

A_{21} A_{22}

$$Q = \begin{bmatrix} 4 & 10 \\ 6 & 4 \end{bmatrix} * \begin{bmatrix} 10 \\ 21 \end{bmatrix} = \begin{bmatrix} 24 & 10 \\ 14 & 4 \end{bmatrix}$$

B_{11}

$$\begin{bmatrix} 2 & 1 \\ 6 & 0 \end{bmatrix} * \begin{bmatrix} 0 & 3 \\ 1 & 6 \end{bmatrix} - \begin{bmatrix} 8 & 3 \\ 4 & 0 \end{bmatrix}$$

$$R = \begin{bmatrix} 2 & 1 \\ 6 & 0 \end{bmatrix} * \begin{bmatrix} -8 & 0 \\ -3 & 6 \end{bmatrix}$$

$$= \begin{bmatrix} -19 & 6 \\ -48 & 0 \end{bmatrix}$$

$$S = \begin{bmatrix} 1 & 8 \\ 2 & 1 \end{bmatrix} * \begin{bmatrix} 7 & 1 \\ 0 & 5 \end{bmatrix} - \begin{bmatrix} 1 & 0 \\ 2 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 8 \\ 2 & 1 \end{bmatrix} * \begin{bmatrix} 6 & 1 \\ -2 & 4 \end{bmatrix}$$

$$= \begin{bmatrix} -10 & 33 \\ 10 & 6 \end{bmatrix}$$

$$T = \begin{bmatrix} 2 & 1 \\ 6 & 0 \end{bmatrix} + \begin{bmatrix} 1 & 3 \\ 4 & 5 \end{bmatrix} * \begin{bmatrix} 8 & 3 \\ 4 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 3 & 4 \\ 10 & 5 \end{bmatrix} * \begin{bmatrix} 8 & 3 \\ 4 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 40 & 9 \\ 100 & 30 \end{bmatrix}$$

$$U = \begin{bmatrix} 3 & 2 \\ 4 & 3 \end{bmatrix} - \begin{bmatrix} 2 & 1 \\ 6 & 0 \end{bmatrix} * \begin{bmatrix} 1 & 0 \\ 2 & 1 \end{bmatrix} + \begin{bmatrix} 0 & 3 \\ 1 & 6 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 1 \\ -2 & 3 \end{bmatrix} * \begin{bmatrix} 1 & 3 \\ 3 & 7 \end{bmatrix}$$

$$= \begin{bmatrix} 4 & 10 \\ 7 & 15 \end{bmatrix}$$

$$V = \begin{bmatrix} 1 & 3 \\ 4 & 5 \end{bmatrix} - \begin{bmatrix} 1 & 8 \\ 2 & 1 \end{bmatrix} * \begin{bmatrix} 7 & 1 \\ 0 & 5 \end{bmatrix} + \begin{bmatrix} 8 & 3 \\ 4 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & -5 \\ 2 & 4 \end{bmatrix} * \begin{bmatrix} 15 & 4 \\ 4 & 5 \end{bmatrix}$$

$$= \begin{bmatrix} -20 & -25 \\ 46 & 28 \end{bmatrix}$$

$$C_{11} = P + S - T + V$$

$$= \begin{bmatrix} 81 & 18 \\ 78 & 25 \end{bmatrix} + \begin{bmatrix} -10 & 33 \\ 10 & 6 \end{bmatrix} - \begin{bmatrix} 40 & 9 \\ 100 & 30 \end{bmatrix} + \begin{bmatrix} -20 & -25 \\ 46 & 28 \end{bmatrix}$$

$$= \begin{bmatrix} 11 & 17 \\ 34 & 29 \end{bmatrix}$$

$$C_{12} = R + T$$

$$= \begin{bmatrix} -19 & 6 \\ -48 & 0 \end{bmatrix} + \begin{bmatrix} 40 & 9 \\ 100 & 30 \end{bmatrix}$$

$$= \begin{bmatrix} 21 & 15 \\ 52 & 30 \end{bmatrix}$$

$$C_{21} = Q + S$$

$$= \begin{bmatrix} 24 & 10 \\ 14 & 4 \end{bmatrix} + \begin{bmatrix} -10 & 33 \\ 10 & 6 \end{bmatrix}$$

$$= \begin{bmatrix} 14 & 43 \\ 24 & 10 \end{bmatrix}$$

$$C_{22} = P + R - Q + U$$

$$= \begin{bmatrix} 81 & 18 \\ 78 & 25 \end{bmatrix} + \begin{bmatrix} -19 & 6 \\ -48 & 0 \end{bmatrix} - \begin{bmatrix} 24 & 10 \\ 14 & 4 \end{bmatrix} + \begin{bmatrix} 4 & 10 \\ 7 & 15 \end{bmatrix}$$

$$= \begin{bmatrix} 42 & 24 \\ 23 & 36 \end{bmatrix}$$

$$C = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} 11 & 17 & 21 & 15 \\ 34 & 29 & 52 & 30 \\ 14 & 43 & 42 & 24 \\ 24 & 10 & 23 & 36 \end{bmatrix}$$

Greedy Method

Given a problem with n inputs we need to find a subset that satisfies some constraint. Any subset that satisfies the constraint is called feasible solution. A feasible solution that maximises or minimizes an objective function is called Optimal solution.

```
Greedy Algorithm (a)
{
    Solution = ∅
    for i = 1 to n do
    {
        if (feasible (solution a)) then
            Solution = union (solution a)
    }
    return solution
}
```

Knapsack Problem: (Fractional Knapsack)

Consider object i has weight w_i and the knapsack has capacity m. If fraction x_i, $0 \leq x_i \leq 1$ of object i is placed into the knapsack then the profit $\sum p_i x_i$ should be maximum. Such that $\sum w_i x_i \leq m$, $0 \leq x_i \leq 1$, $1 \leq i \leq n$

Algo

```
GreedyKnapsack (m, n)
// n objects ordered such that
// p[i]/w[i] >= p[i+1]/w[i+1]
{
for i=1 to n do
    x[i]=0

U=m
for i=1 to n do
{
    if (w[i]> U) then
        break
    x[i]= 1
    U= U- w[i]
}
if (i<n) then
    x[i]= U/w[i]
}
```

Ex1: Consider the following instance of the Knapsack problem $n=3$, $m=20$, $(p_1, p_2, p_3) = (25, 24, 15)$ and $(w_1, w_2, w_3) = (18, 15, 10)$

Solution:

$p_1/w_1 = 25/18 = 1.388$

$p_2/w_2 = 24/15 = 1.6$

$p_3/w_3 = 15/10 = 1.5$

Arrange object in decreasing order

$\rightarrow x_2, x_3, x_1$

$n = 3$

$m = 20 \qquad U = 20$

(i) $\omega_2(15) < 20$

$U = 20 - 15 = 5 \Rightarrow x[2] = 1$

(ii) $\omega_3(10) > 5$

break

$\Rightarrow x[3] = 5/10 = 1/2$

∴ The solution or selected objects are $\{0, 1, 1/2\}$

$x_1 \; x_2 \; x_3$

Ex 2: Find optimal solution to knapsack instance $n=7$, $m=15$, $(p_1, p_2, p_3 \cdots p_7) = (10, 5, 15, 7, 6, 18, 3)$ and $(\omega_1, \omega_2 \cdots \omega_7) = (2, 3, 5, 7, 1, 4, 1)$

Solution: $\frac{p_1}{\omega_1} = \frac{10}{2} = 5$

$\frac{p_2}{\omega_2} = \frac{5}{3} = 1.6$

$\frac{p_3}{\omega_3} = \frac{15}{5} = 3$ $\quad \frac{p_4}{\omega_4} = \frac{7}{7} = 1$ $\quad \frac{p_5}{\omega_5} = \frac{6}{1} = 6$

$\frac{p_6}{\omega_6} = \frac{18}{4} = 4.5$, $\frac{p_7}{\omega_7} = \frac{3}{1} = 3$

$m = 15$, $n = 7$, $U = 15$

Arrange objects in decreasing order

$\rightarrow x_5, x_1, x_6, x_3, x_7, x_4, x_2$

(1) $\omega_5(1) \leq 15$

$U = 15 - 1 = 14 \Longrightarrow x[5] = 1$

(2) $\omega_1(2) < 14$

$U = 14 - 2 = 12 \Longrightarrow x[1] = 1$

(3) $\omega_6(4) < 12$

$U = 12 - 4 = 8 \Longrightarrow x[6] = 1$

(4) $\omega_3(5) < 8$

$U = 8 - 5 = 3 \Longrightarrow x[3] = 1$

(5) $\omega_7(3) == 3$

$U = 3 - 3 = 0 \Longrightarrow x[7] = 1$

$\therefore$ The solution is $\{1, 0, 1, 0, 1, 1, 1\}$

Job Sequencing with Deadlines:-

- There are n jobs and each job i has a integer deadline $d_i \geq 0$ and a profit $p_i > 0$.
- job i earns a profit p_i if and only if the job completes by its deadline.
- Only one machine available to process job.
- The job has to processed per unit time.

// A Greedy algorithm for sequencing unit job with deadline

Algorithm JS(d, j, n)

//$d[i] \geq 1$, $1 \leq i \leq n$ are deadline, $n \geq 1$. The jobs
//are ordered such that $p[1] \geq p[2] \geq \cdots \geq p[n]$.
//$J[i]$ is the ith job in the optimal solution,
//$1 \leq i \leq K$. At termination $d[J[i]] \leq d[J[i+1]]$, $1 \leq i < K$

```
{
    d[0] := J[0] = 0
    J[1] = 1
    K = 1
    for i = 2 to n do
    {
        r = K
        while ((d[J[r]] > d[i]) and
               (d[J[r]] ≠ r)) do {
            r = r - 1
        }
        if ((d[J[r]] ≤ d[i]) and (d[i] > r)) then
        {
            for q = K to (r+1) step -1 do {
                J[q+1] = J[q];
```

```
        J[r+1] = i;
        K = K+1;
    }
}
return K;
}
```

(OR)

```
Algorithm GreedyJob(J, n)
{
        = {1}
    for i=2 to n do
    {
        if Call jobs in  U{i}
        can be completed by their
        deadline) then
            = U{i}
    }
}
```

```
        J[r+1] = i;
        K = K+1;
      }
    }
    return K;
}
```

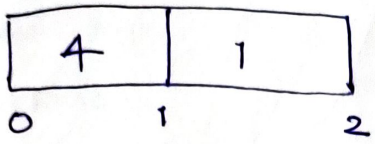
(OR)

```
Algorithm GreedyJob(J, n)
{
        = {1}
    for q=2 to n do
    {
        if Call jobs in  U{i}
        can be completed by their
        deadline) then
                = U{i}
    }
}
```

Ex 1: Let $n=4$, $P = \{100, 10, 15, 27\}$

$d = \{2, 1, 2, 1\}$. Using greedy algorithm Schedule the job for above sets.

Solution:



I	Assigned slots	Job considered	Action	profit
$\emptyset$	none	1	feasible, assigned at slot [1,2]	100
{1}	[1, 2]	4	feasible, assigned at slot [0,1]	127
{1,4}	[0,1] [1,2]	3	Not feasible	127
{1,4}	[0,1] [1,2]	2	Not feasible	127

→ Place the job in non-increasing order

$P = \{100, 27, 15, 10\} \rightarrow \{P_1, P_4, P_3, P_2\}$

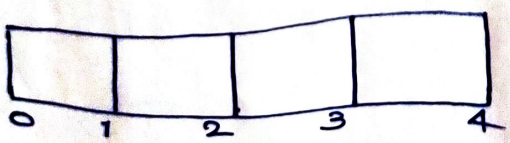
$d = \{2, 1, 2, 1\} \rightarrow \{d_1, d_4, d_3, d_2\}$

∴ The feasible solution is $\{1, 4\}$

∴ The total profit is 127 //

Ex 2: Solve the job sequencing algorithm using greedy algorithm where profit $P = \{3, 5, 20, 18, 1, 6, 30\}$ and deadline $d = \{1, 3, 4, 3, 2, 1, 2\}$

Solution:



J	Assigned Slots	Job Considered	Actions	Profit
ϕ	none	7	feasible assigned at slot [1, 2]	30
{7}	[1,2]	3	feasible assigned at [3,4]	30+20=50
{7,3}	[1,2] [3,4]	4	feasible assigned at [2,3]	50+18=68
{7,3,4}	[1,2] [3,4], [2,3]	6	feasible assigned at [0,1]	68+6=74
{7,3,4,6}	[1,2] [3,4] [2,3] [0,1]	2	not feasible	74
{7,3,4,6}	[1,2] [3,4] [2,3] [0,1]	1	not feasible	74
{7,3,4,6}	[1,2] [3,4] [2,3] [0,1]	5	not feasible	74

place job in decreasing order

$$p = \{ p_7, p_3, p_4, p_6, p_2, p_1, p_5 \}$$

$$d = \{ 2, 4, 3, 1, 3, 1, 2 \}$$

∴ The feasible solution is { 7, 3, 4, 6 }

∴ The total profit is 74 //

May/June 2019

Questions:

1. Define asymptotic notation: Big oh, Big-Omega, Theta. [3]

2. Let S= {a,b,c,d,e,f,g} be a collection of objects with profit weight values as follows:

a(10,2), b(5,3), c(15,5), d(7,7), e(6,1), f(18,4), g(3,1). What is the optimal solution to the fractional Knapsack for S assuming that the capacity of Knapsack is 15? [7]

3. Describe the divide and conquer technique with a Control abstraction. [5]

4. State master's theorem for solving recurrence

5. Write an iterative algorithm to find maximum and minimum items in a set of n elements. Transfo your iterative algorithm into an equivalent algorithm that uses recursion. Compare the effeciency of both algorithm & show which one is powerful. [10]

6. Given a set of n jobs. Associated with each job k is a deadline d_k and profit P_K. For any job k the profit P_K is earned iff the job is completed by its deadline d_k. Formulate a greedy algorithm obtain an optimal solution for this problem. S its time complexity. [6]

7. Write an Algorithm that takes positive integer

as input & returns the exponent 2^n as outp

Analyze the time complexity of your algorithm using program step table method. [8]

Nov/Dec 2019

1) What are criteria for designing an effecient algorithm. [5]

2) Let $a_i, 1 \leq i \leq n$ be a list of elements that are sorted in non-decreasing order. Devise a suitable algorithm to determine whether a given element 'x' is present in the list. by using divide and Conquer strategy. [5]

3) Discuss how divide and conquer strategy is used to multiply 2 square matrices as suggested by strassen's. Illustrate by using following matrices.

$$A = \begin{bmatrix} 2 & 3 \\ 3 & 2 \end{bmatrix} \quad B = \begin{bmatrix} 4 & 3 \\ 3 & 4 \end{bmatrix}$$

4) Let $n=5$, $(p_1, p_2 \cdots p_5) = (20, 10, 15, 1, 5)$ and $(d_1, d_2 \cdots d_5) = (2, 2, 1, 3, 3)$. Obtain the optimal schedule that finishes the job by its deadline & given maximum profit. [5]

5) Can quick sort be modified so that it performs well on every input. justify with an algorithm. [5]

) Determine time and space complexity of following algorithm.

```
Algorithm A(x, n)
{
    for i := 1 to n do
        for j := 1 to i do
            for k := 1 to j do
                x := x + 1
    end
}
```