Theme: Introduction to Algorithms Date: / /

Sat Sun Mon Tue wed Thu Fri

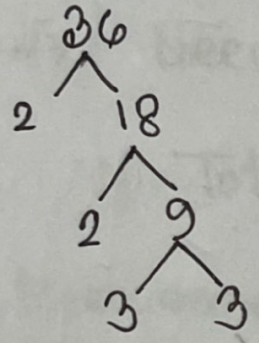
Etymology → কোনো একটা জিনিস এর উৎপত্তি কোথা থেকে এসেছে এটা বের করা।

- Algorithm well-defined হওয়া লাগবে
  - Sequential থাকা লাগবে
- Algorithm language independent.

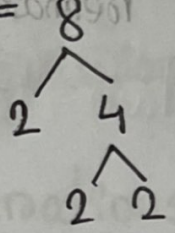
What makes an algorithm good?

Ans:- Correctness → সঠিক কিনা দেখা লাগবে
Efficiency → performance

$n_1 =$



$n_2 =$



Prime factor $n_1 = 2 \times 2 \times 3 \times 3$

$n_2 = 2 \times 2 \times 2$

$GCD = 2 \times 2$

$= 4$

Theme:

Date: / /

Sat Sun Mon Tue wed Thu Fri

Algorithm 1°–

1. Calculate Prime factors of number n1
2. Calculate Prime factors of number n2
3. Identify the common factor
4. Multiply the factors and return.

Algorithm 2°–

1. While n1 is not divided by n2
2. Calculate remainder r = n1 mod n2
3. n1 = n2
4. n2 = r
5. Return n2

Example°– $n_1 = 8$, $n_2 = 36$

1. While (36 is not divided by 8)
2. 8) 36 (4
   32

   4

   r = 4
3. $n_1 = 8$
4. $n_2 = 4$
5. R

again, $n_1 = 8$, $n_2 = 4$

1. While (8 is divided by 4)

2. return 4

Performance Measurement :- Algorithm 1

$n_1 = 8$, $n_2 = 36$

$8 = 2 \times 2 \times 2$ → Need 3 divison & 3 checks for prime

$36 = 2 \times 2 \times 3 \times 3$ → Need 4 divison & 4 checks for prime

Common factors 2, 2 → Need 3 checks for common prime

GCD $= 2 \times 2 = 4$ → Need 1 multiplication

Total = 18 operations

Performance Measurement :- Algorithm 2

$n_1 = 8$, $n_2 = 36$

Iteration 1

n1 is not divided by n2 → Need 1 check and 1 division

$n = 36 \% 8 = 4$

Iteration 2

n1 (8) is divided by n2 → Need 1 check and 1 division

Return 4 as gcd → Need 1 check for stopping

Theme:

Date: / /

Sat Sun Mon Tue wed Thu Fri

Total = 5 operations

Algorithm Analysis:-

Time Complexity:- amount বের করা

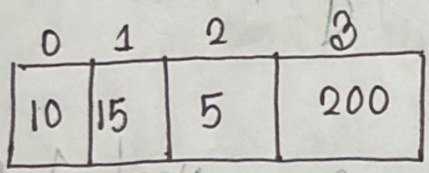
Space Complexity:- memory space বের করা

Asymptotic Notation

Big O Notation (O):- represents the upper bound

Omega notation (Ω):- represents the lower bound

Theta notation (Θ):- represents both the upper and lower bound.



item: 200
location: 3
comp = 4

comp 3 { item: 5, location: 2

comp 1 { item: 10, location: 0

Best case

{ item: 500, location: x
comp = 4

worst case

Theme:

Date: / /

Sat Sun Mon Tue wed Thu Fri

lecture-1 (Part-2)

Complexity Analysis

$f(n) \rightarrow n$ = input size

```
for (i=0; i<n; i++)
{
    printf ("Hello World");  → execute (1 unit)
}
```

$f(n) = n$

↳ এখানে বোঝা যাচ্ছে, কোড run হতে কত সময় লাগে

1 unit → 1 সংখ্যক

n " → n সংখ্যক

$f(n) = 5n^2 + 6n + 2$

যার order বেশি তার উপর depend করে time complexity.

এখানে order হলো n^2.

Single for loop হলে n

double for loop হলে n^2

Theme:

Date: / /

Sat Sun Mon Tue wed Thu Fri

type:-

$f(n)=2$
$f(n)=500$
$f(n)=1000$
} constant → $O(1)$

* f এর value যাই থাকুক না কেন n এর value না থাকলে time constant থাকবে আমরা Order $O(1)$

$f(n)=2n+3$
$f(n)=500n+700$
$f(n)=\frac{n}{5000}+6$
} → $O(n)$ [linear]

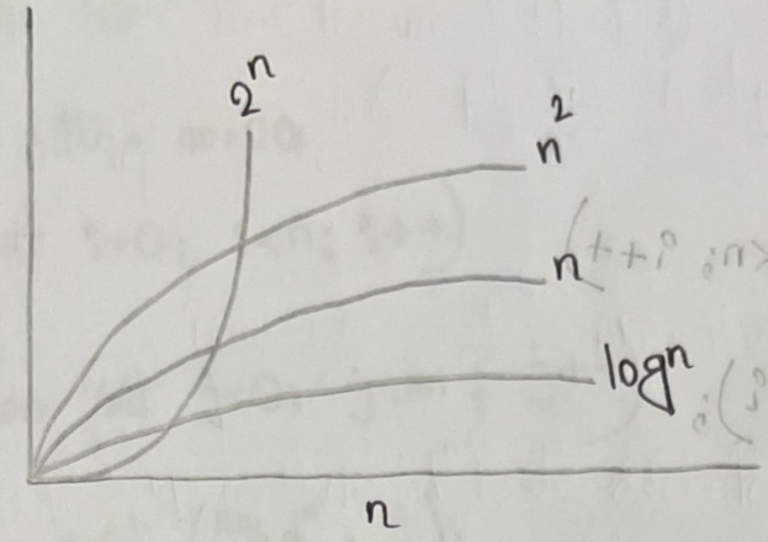
$f(n)=10n^2+3$
$f(n)=5n^2+6n+2$
} → $O(n^2)$ [Quadratic]

$O(n^3)$ [cubic]

$O(\log n)$ [logarithm]

$O(2^n)$ [Exponensia]

$1 < \log n < \sqrt{n} < n < n\log n < n^2 < n^3 \ldots\ldots\ldots < 2^n < 3^n < n^2$



Complexity analysis

Example-1:-

```
int search (int arr[], int n, int x)
{
    int i;
    for (i=0; i<n; i++)
    {
        if (arr[i] = x)
        return i;
    }
    return -1;
}
```

Complexity → O(N)

Theme:

Date: /

Sat Sun Mon Tue wed T

Example 2:-

```
int n=10;
for (int i=0; i<n; i++)
{
   printf("%d", i);
}
```

Complexity → O(n)

Example 3:-

```
int n=10, m=20
for (int i=0; i<n; i++)
{
   printf("%d", i);
}
for (int i=0; i<m; i++)
{
   printf("%d", i);
}
```

Complexity → O(n+m)

Theme:

Date: / /

Sat Sun Mon Tue wed Thu Fri

Example 4:-

```
int n = 10 , m=20;
for (int i=0; i<n; i++)
{
    for (int j=0; j<m ; j++)
    {
        printf ("%d", j);
    }
}
```

Complexity → O(n*m)

Example 5:-

```
int n= 10;
for (int i=0; i<n; i++)
{
    for (int j=0; j<i; j++)
    {
        printf ("%d", j);
    }
}
```

Complexity → O(n*n)

Theme:

Date: / /

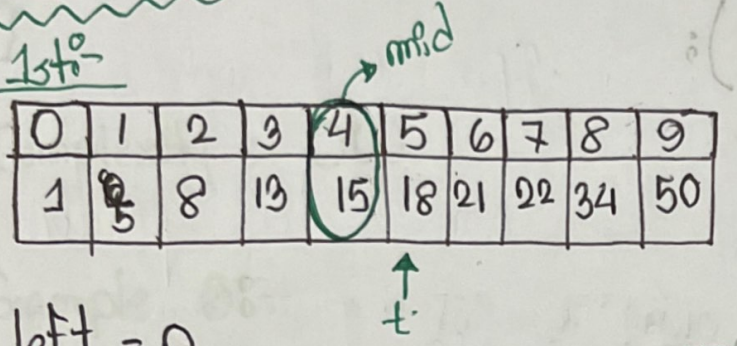
Sat Sun Mon Tue wed Thu Fri

Binary Search Algorithm:-

1. If target == arr[mid], then return mid
2. If target < arr[mid], search in the left half
3. If target > arr[mid], search in the right half

Scenario:-

1st:-



left = 0

Right = 9

mid = 4

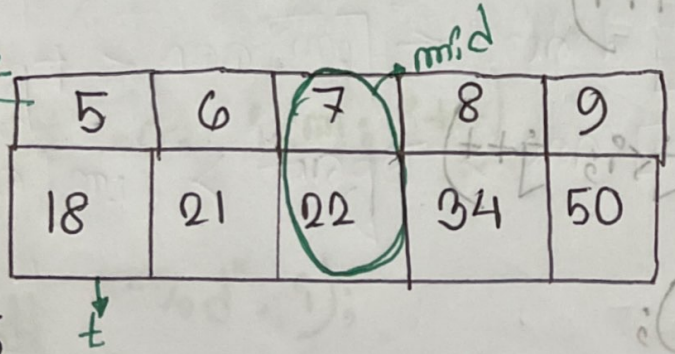
t = 18

$$m = l + \frac{(R-l)}{2}$$

$$= 0 + \frac{(9-0)}{2}$$

$$= 4.5$$

2nd:-



left = 5

Right = 9

mid = 7

t = 18

$$mid = 5 + \frac{(9-5)}{2}$$

$$= 5 + \frac{4}{2}$$

$$= 7$$

Theme:

Date: / /

Sat Sun Mon Tue Wed Thu Fri

3rd:

5	6
18	21

arr[mid] == target

target found in position 5.

Binary time Complexity:

$\Rightarrow n \to 1$

$\Rightarrow \frac{n}{2} \to 2$

$\Rightarrow \frac{\frac{n}{2}}{2} \to \frac{n}{4} \to \frac{n}{2^2} \to 3$

$\Rightarrow \frac{n}{8} \to \frac{n}{2^3} \to 4$

$\Rightarrow \frac{n}{16} \to \frac{n}{2^4} \to 5$

∴ $\boxed{\frac{n}{2^k} = 1}$

$\Rightarrow n = 1 * 2^k$

$\Rightarrow n = 2^k$

$\Rightarrow 2^k = n$

$\Rightarrow \log_2^{2k} = \log_2^n$

$\Rightarrow K \cdot \log_2^2 = \log_2^n$

$\Rightarrow K * 1 = \log_2^n$

$\Rightarrow \log_2^n = K$

Theme:

Date: / /

Sat Sun Mon Tue wed Thu Fri

f(n) কে simplify করে asimptotic notation

Best Case:- $O(1)$

$\Theta(1)$

$\Omega(1)$

worst case:- $O(\log_2^n)$

$\Theta(\log_2^n)$

$\Omega(\log_2^n)$

Average case:- Same

Ternary Search Algorithm:-

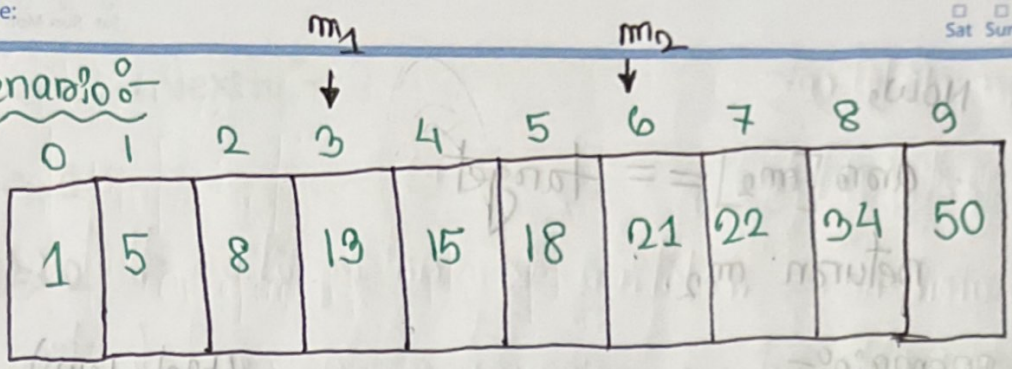
1. arr[m_1] == target → return m_1
2. arr[m_2] == target → return m_2
3. target < arr[m_1] → search [left, m_1-1]
4. target > arr[m_2] → search [right, m_2+1]
5. arr[m_1] < target < arr[m_2] → search [m_1+1, m_2-1]

Theme:

Date: / /

Sat Sun Mon Tue wed Thu Fri

Scenario:-



1st step:-

low = 0

high = 9

target = 34

$$m_1 = low + \frac{(high - low)}{3}$$

$$= 0 + \frac{(9-0)}{3}$$

$$= 3$$

$$m_2 = high - \frac{(high - low)}{3}$$

$$= 9 - \frac{(9-0)}{3}$$

$$= 6$$

Now,

target > arr[m_2]

return ternarysearch(arr, m2+1, high, target)

(arr 7, 9, 34)

2nd step:-

low = 7

high = 9

target = 34

$$m_1 = low + \frac{(high - low)}{3}$$

$$= 7 + \frac{(9-7)}{3}$$

$$= 7$$

$$m_2 = 8$$

Theme:

Date: / /

Sat Sun Mon Tue wed Thu Fri

Now,

arr[m_2] == target

return m_2.

another scenario:-

low = 0

high = 9

target = 15

$m_1 = low + \frac{(high - low)}{3}$

$= 0 + \frac{(9-0)}{3}$

$= 0 + 3$

$= 3$

$m_2 = high - \frac{(high - low)}{3}$

$= 9 - \frac{(9-0)}{3}$

$= 9 - 3$

$= 6$

Now,

arr[m_1] < target < arr[m_2]

return ternarysearch (arr, m_1+1, m_2-1, target)

(arr, 4, 5, 34)

low = 4

high = 5

target = 34

Theme:

Date: / /

Sat Sun Mon Tue Wed Thu Fri

Time Complexity:-

$n \to 1$

$\frac{n}{3} \to 2$

$\frac{n}{3^2} \to 3$

$\frac{n}{3^3} \to 4$

$\therefore \boxed{\frac{n}{3^k} = 1}$

$n = 3^k$

$\Rightarrow \log_3^n = \log_3^{3^k}$

$\Rightarrow \log_3^n = k * 1$

$\Rightarrow k = \log_3^n$

Theme:

Date: / /

Sat Sun Mon Tue wed Thu Fri

Recursive Relation:-

~~void Test (int n)~~

A recurrence relation is an equation that express each element of a sequence as a function of the preceding ones.

Substitution Method:-

Example-01:-

```
void test (int n) ⟶ T(n)
{
  if (n>0)
  {
    print ("%d", n); →1 unit
    test (n-1); → T(n-1)
  }
}
```

$\Rightarrow T(n) = T(n-1) + 1$

$T(n) \begin{cases} 1 \\ T(n-1)+1, & n>0 \end{cases}$

Theme:

Date: / /

Sat Sun Mon Tue Wed Thu Fri

$$T(n) = T(n-1) + 1$$

$$= [T(n-2) + 1] + 1$$

$$= [[T(n-3) + 1] + 1] + 1$$

$$= [[[T(n-4) + 1] + 1] + 1] + 1$$

$$= T(n-4) + 4$$

$$= T(n-k) + k$$

When

$k = n$

$$= T(n-n) + n$$

$$= T(0) + n$$

$$= 1 + n$$

$$f(n) = 1 + n$$

$$= O(n)$$

$T(n-1) + 1$

$[T(n-1) + 1] + 1$

$[T(n-1) + 1] + 1$

Theme:

Date: / /

Sat Sun Mon Tue wed Thu Fri

Example-02:

```
Void test (int n) ⟶ T(n)
{
 if (n>1)
{ printf ("%d", n); ⟶ 1 unit
  Test (n/2); ⟶ T(n/2)
}
```

$\Rightarrow T(n) = 1 + T(n/2)$

$$T(n) \begin{cases} 1, & n \le 1 \\ T(n/2)+1, & n > 1 \end{cases}$$

$$T(n) = T\left(\frac{n}{2}\right) + 1$$

$$= \left[T\left(\frac{n}{4}+1\right)\right] + 1$$

$$= \left[\left[T\left(\frac{n}{8}+1\right)+1\right]+1\right]+1$$

$$= T\left(\frac{n}{2^k}\right) + k$$

Theme:

Date: / /

Sat Sun Mon Tue Wed Thu Fri

After, K steps, $\frac{n}{2^k} = 1$

$$\frac{n}{2^k} = 1$$

$$\Rightarrow 2^k = n$$

$$\Rightarrow \log_2^n = \log_2^{2^k}$$

$$\Rightarrow K = \log_2^n$$

$$\therefore O\left(\log_2^n\right)$$

Master Method:-

The Master method depends on the Master Theorem:-

$$T(n) = a * T\left(\frac{n}{b}\right) + O\left(n^d\right)$$

$a \geq 1;$
$b > 1$

Where,

$T(n) \Rightarrow$ Time complexity of the algorithm on an input of size n.

$a \Rightarrow$ Number of recursive subproblems created in each division step.

Theme:

Date: / /

Sat Sun Mon Tue wed Thu Fri

$O(n^d)$ => Time complexity of combining the results of the subproblems and any additional work done outside the recursive calls.

Time complexity $T(n)$ of

Merge Sort

Date: / / Sat Sun Mon Tue wed Thu Fri

Theme:

mergeSort (arr, l, r)

{ if (l < r)

m = l + (h-l)/2

ms = (arr, l, m)

ms = (arr, m+1, r)

merge (arr, l, m, r)

}

ms (0,3)

{ m = 1

ms (0,1)

ms (2,3)

m (0,1,3)

}

ms (2,3)

{ m = 2

ms (2,2) ⇌ ms (2,2) ××

ms (3,3) ⇌ ms (3,3) ××

m (

ms (0,1)

{ m = 0

ms (0,0) ⇌ ms (0,0) ××

ms (1,1) ⇌ ms (1,1) ××

m (0,0,1) (1,9)

ms (4,7)

{ m = 5

ms (4,5)

ms (6,7)

m (4,5,7)

}

ms (4,5)

{ m = 4

ms = (4,4) ××

ms = (5,5) ××

m (4,4,5)

}

ms (6,7)

{ m = 6

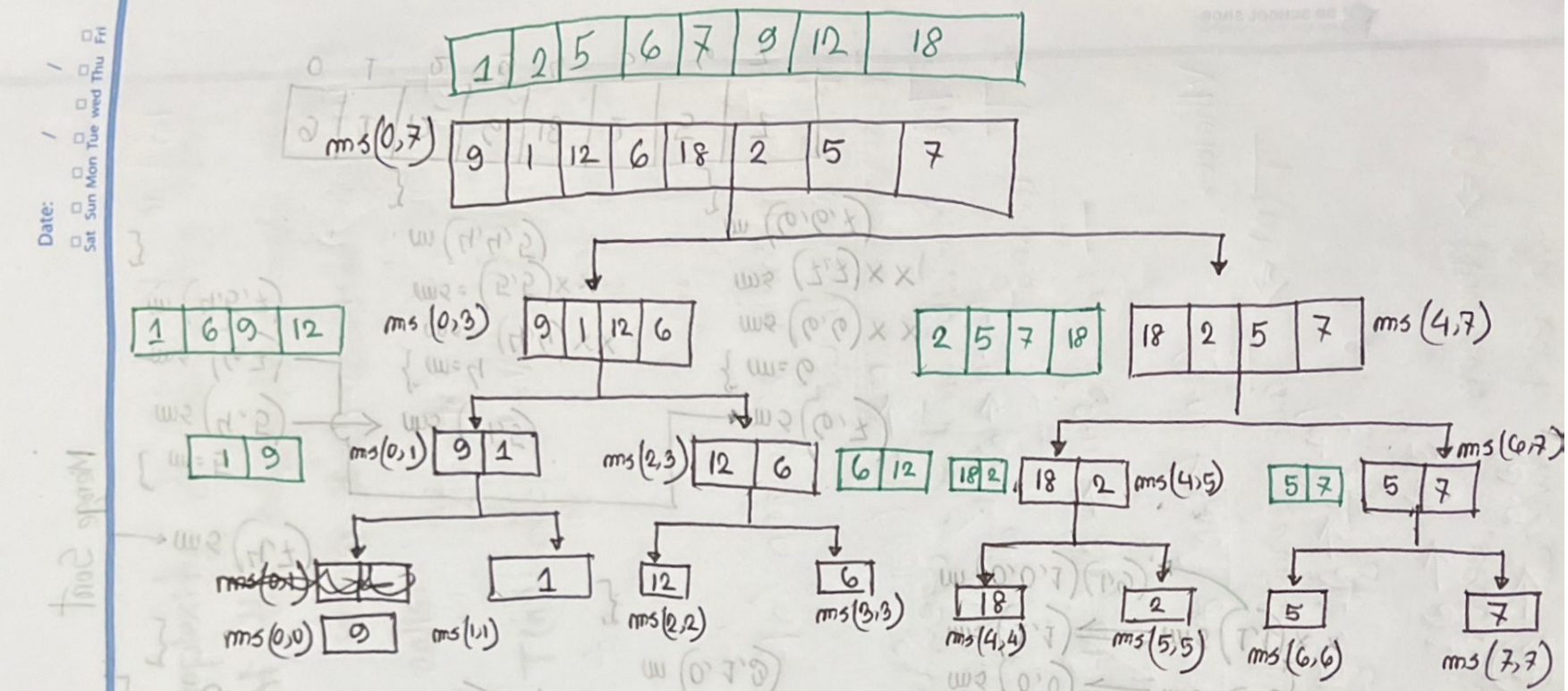
ms (6,6) ××

ms (7,7) ××

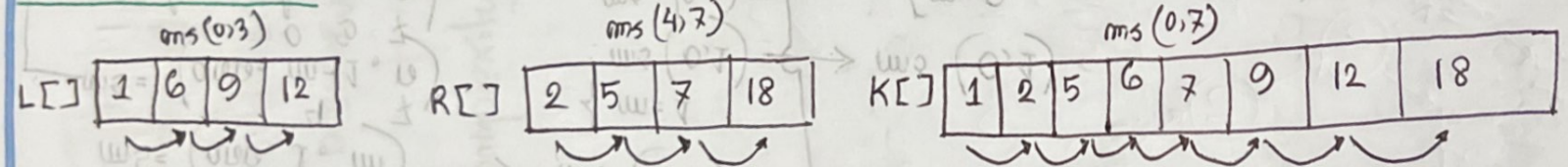
m (6,6,7)

}

9	1	12	6	18	2	5	7
0	1	2	3	4	5	6	7



For 2nd Level:-



1st step

L[0] < R[0], so K[0] = L[0]

L++; K++

2nd Step:

R[0] < L[1], so, K[1] = R[0]

R++; K++

3rd step:

K[2] = R[1]

R++; K++

4th step:

K[3] = L[1]

L++; K++

5th step

K[4] = R[2]

R++; K++

6th step

K[5] = L[2]

L++; K++

7th step

K[6] = L[3]

L++; K++

[Point to remember,
K এর index always left থেকে start হবে]

Theme: Dynamic Programming

Date: / /

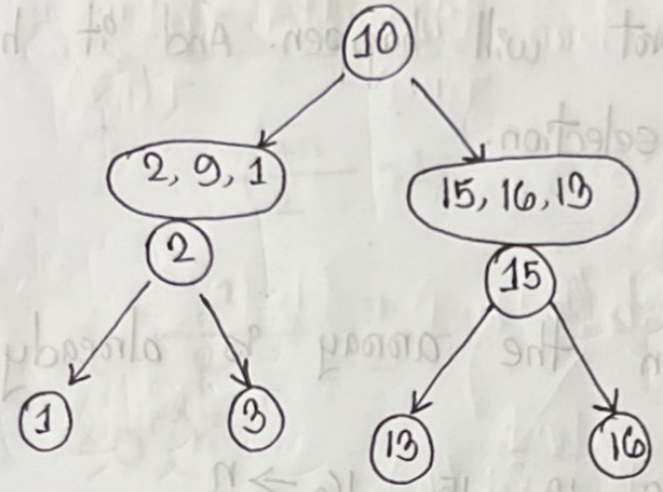
Sat Sun Mon Tue wed Thu Fri

Time Complexity of Quick Sort :-

Best Case :-

10, 15, 1, 2, 9, 16, 13

↳ 2, 9, 1, 10, 15, 16, 13 (After pivot selection)



Time Complexity :-

no. of levels × no. of comparison at each level

As we divide it into two balanced parts (n divides by 2)

↳ $\log_2^n$

Divided operation visits each element in the array once (n)

Now,

Time complexity = $\log_2^n \times 2$

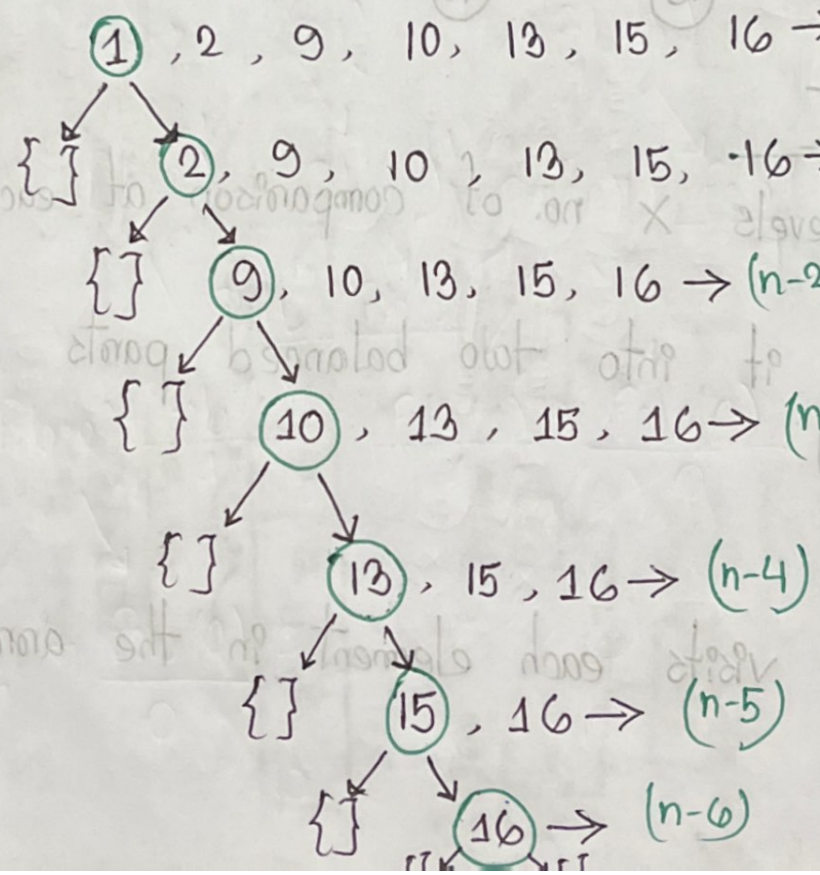
$= n\log_2^n$

→ Best Case for QuickSort

If the array is evenly distributed then the best case of quick sort will happen. And it happens because of pivot selection.

Worst Case:-

It happens when the array is already sorted.



Theme:

Date: / /

Sat Sun Mon Tue Wed Thu Fri

$2 + 3 + 4 + 5 + 6 + \cdots\cdots\cdots + n$

Similar ←

↓

We know,

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

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

$$= \frac{n^2+n}{2} - 1$$

$$= \frac{n^2}{2} + \frac{n}{2} - 1$$

$$= O(n^2)$$

Dynamic Programming

↳ Combining the solutions of subproblems.

↳ storing intermediate results.

Divide & Conquer:- → Disjointed subproblems

Dynamic Programming:- → Overlapping subproblems

Theme:

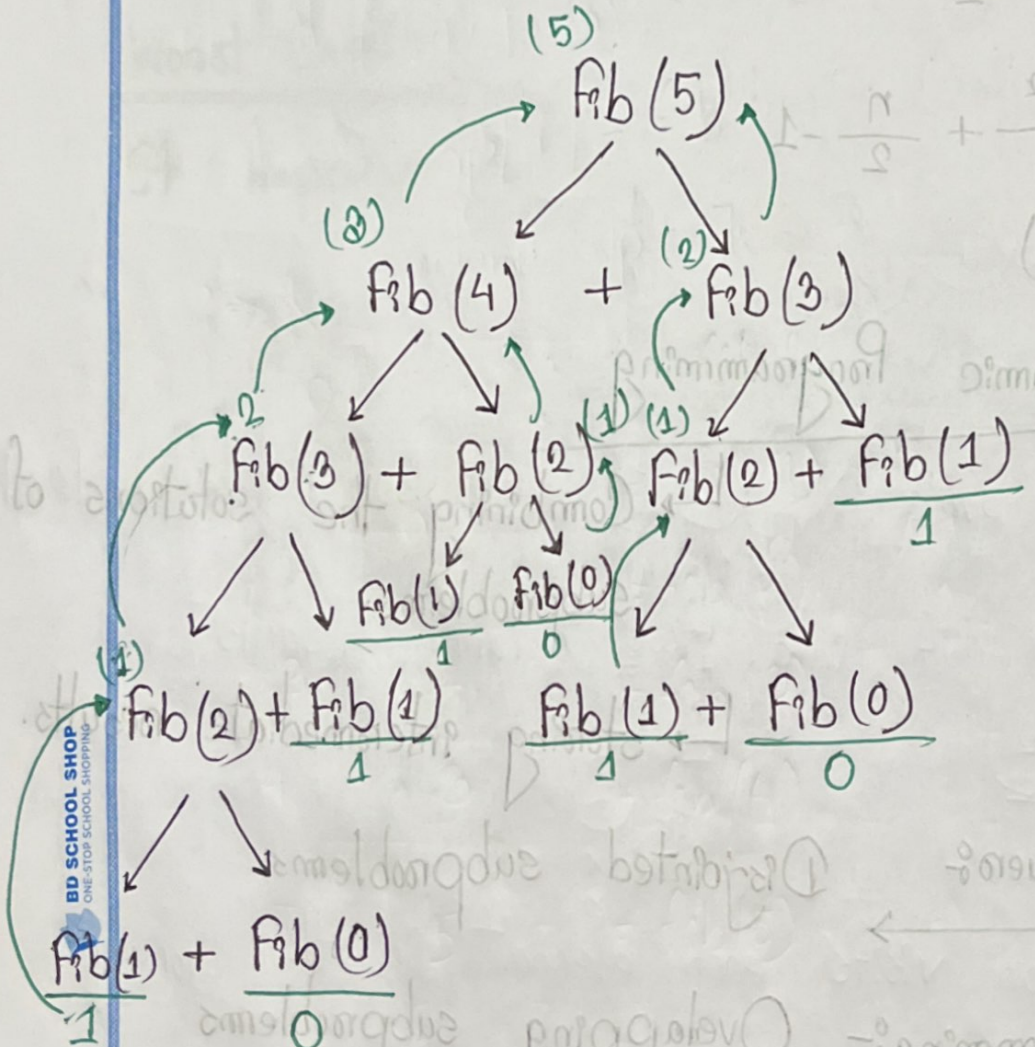
Date: / /

Sat Sun Mon Tue wed Thu Fri

Fibonacci Series

```
int fib (int n) { ⟶ T(n)
    if (n<=1)
        return n;
    else
        return fib(n-1) + fib(n-2);
}
```

T(n-1) + T(n-2) + 1



Theme:

Date: / /

Sat Sun Mon Tue wed Thu Fri

Time complexity for this algo:-

$$T(n) = T(n-1) + T(n-2) + 1$$

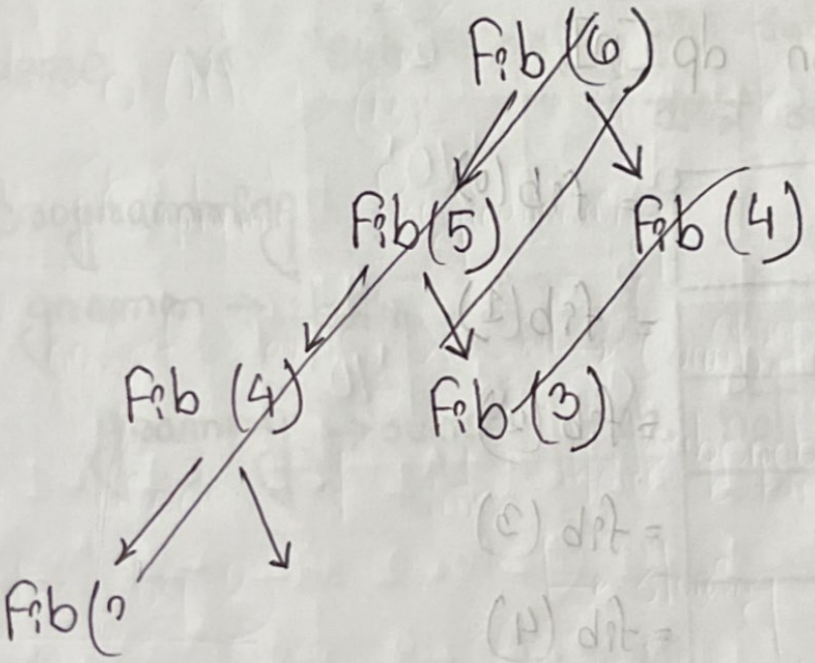
$$\approx T(n-1) + T(n-1) + 1$$

$$T(n) \approx 2T(n-1) + 1$$

$$\therefore O(2^n)$$

Here each recursive call makes 2 more recursive call.

~~DP-FIB Series~~



Theme:

Date: / /

Sat Sun Mon Tue wed Thu Fri

```
int dp[n+1], dp[0] = 0, dp[1] = 1;

    fib (int n)
{
  if (n==0)
      return 0;
else if (n==1)
      return 1;
else if (dp[n]==0)
      dp[n] = fib (n-1) + fib (n-2);
      return dp[n];
}
```

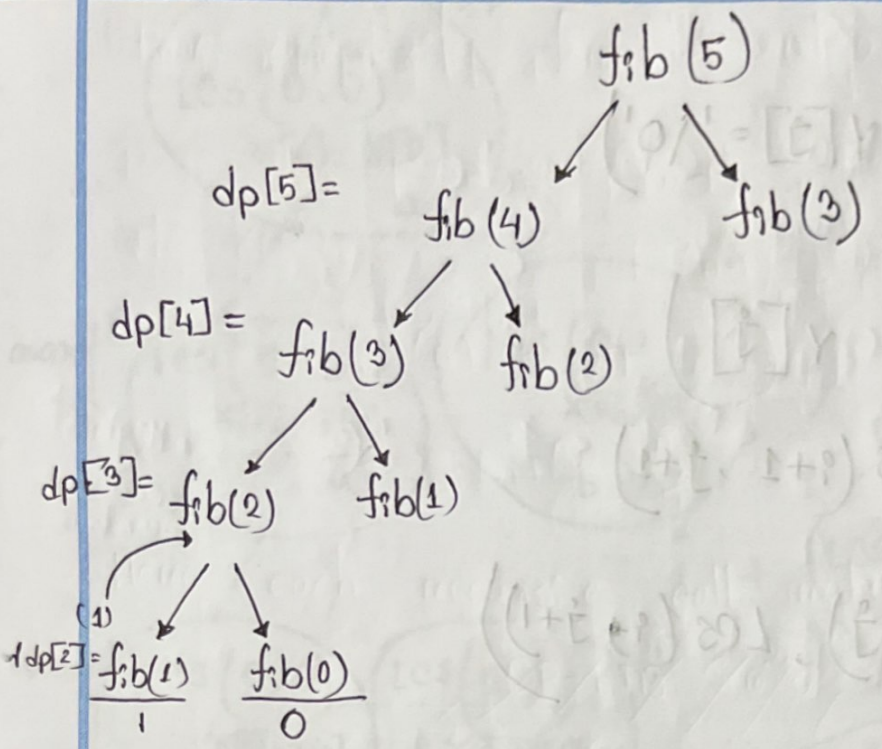
0	0	= fib(0)
1	1	= fib(1)
2	1	= fib(2)
3	2	= fib(3)
4	3	= fib(4)
5	5	= fib(5)

dp

Theme:

Date: / /

Sat Sun Mon Tue wed Thu Fri



Subsequence vs substring [সকল subsequence substring কিন্তু সকল substring subsequence না]

programming

gramm → both

gramng → subsequence, not substring [কারন এখানে কিছু word বাদ চলে গেছে]

Theme:

Date: / /

Sat Sun Mon Tue wed Thu Fri

```
int LCS (i, j)
if (X[i] = '\0' || Y[j] = '\0')
    return 0;
else if (X[i] == Y[j])
    return 1 + LCS (i+1, j+1)
else
    max (LCS (i+1, j), LCS (i, j+1))
```

Example:-

i -> X

0	1	2
b	d	'\0'

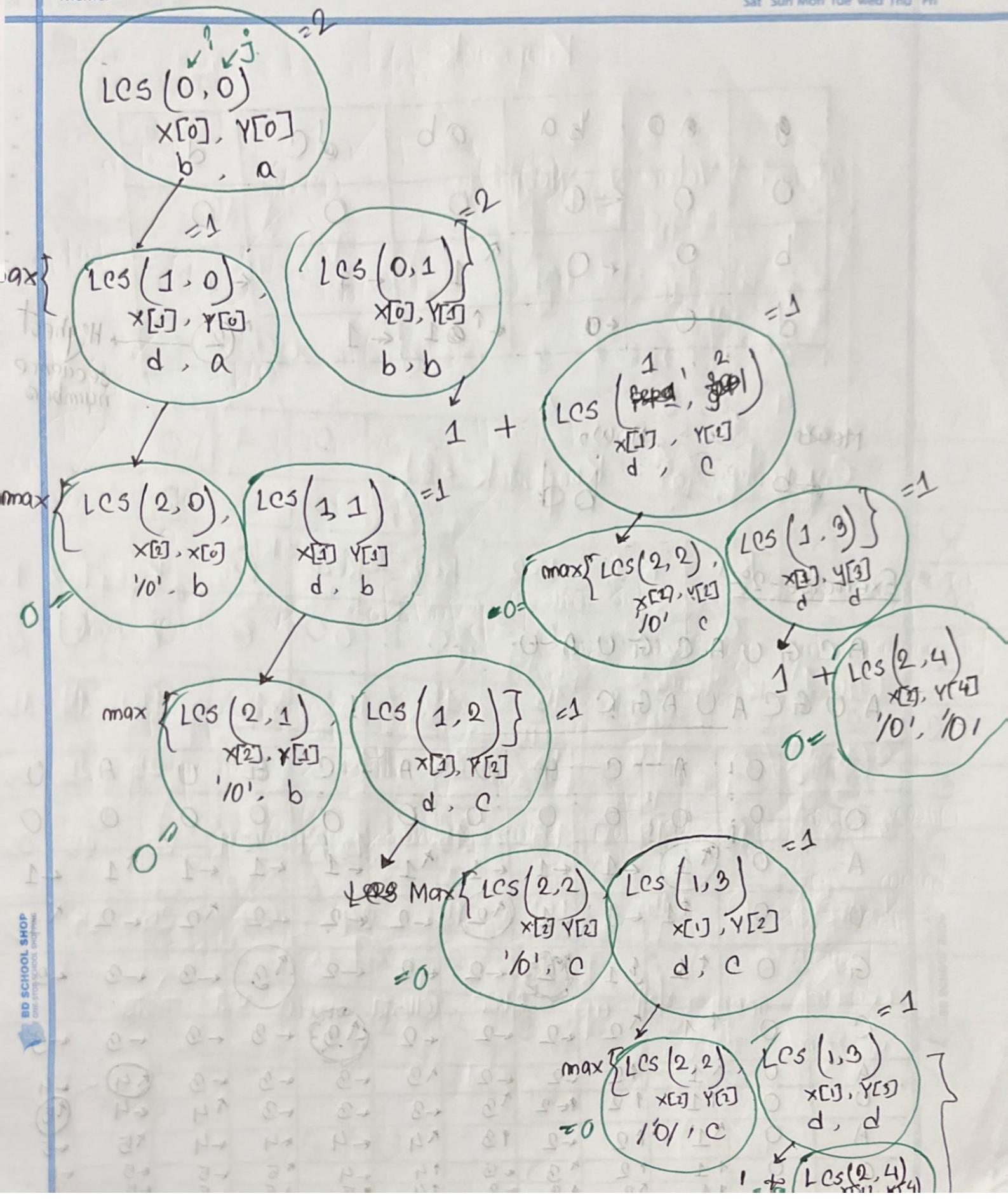
j -> Y

0	1	2	3	4
a	b	c	d	'\0'

Theme:

Date: / /

Sat Sun Mon Tue wed Thu Fri



Theme:

Date: / /

Sat Sun Mon Tue wed Thu Fri

	a	b	c	d
0	0	←0	0	0
b	0	←0	↖1	←1
d	0	←0	←↑1	←1
				↖(2) → Highest secuence number

Now,

bd

Exercise - 2:

A C G U A C G U A U

A U G C A U A G C

	0	A	C	G	U	A	C	G	U	A	U
0	0	0	0	0	0	0	0	0	0	0	0
A	0	↖1	←1	←1	←1	↖1	←1	←1	←1	↖1	←1
U	0	↑1	←1	←1	↖2	←2	←2	←2	↖2	←2	↖1
G	0	↑1	↑1	↖2	←2	←2	←2	↖3	←3	←3	←3
C	0	↑1	↖2	←2	←2	←2	↖3	←3	←3	←3	←3
A	0	↖1	↑2	←2	←2	↖3	←3	←3	←3	↖4	←4
U	0	↑1	↑2	←2	↖3	←3	←3	←3	↖4	←4	↖5
A	0	↖1	↑2	←2	↑3	↖4	←4	←4	←4	↖5	←5
G	0	↑1	↑2	↖3	←3	↑4	←4	↖5	←5	←5	←5

Theme:

Date: / /

Sat Sun Mon Tue wed Thu Fri

Ans:-

~~AGCAU~~

A UGA U

Exercise-3:-

A C G U A C G U A U

U G C A U A G C

	0	A	C	G	U	A	C	G	U	A	U
0	0	0	0	0	0	0	0	0	0	0	0
U	0	0	0	0	↖1	1	1	1	↖1	1	↖1
G	0	0	0	↖1	1	1	1	↖2	2	2	2
C	0	0	↖1	1	1	1	↖2	2	2	2	2
A	0	↖1	1	1	1	↖2	2	2	2	↖3	3
U	0	1	1	1	↖2	2	2	2	↖3	3	↖4
A	0	↖1	1	1	2	↖3	3	3	3	↖4	4
G	0	1	1	↖2	2	3	3	↖4	4	4	4
C	0	1	↖2	2	2	3	↖4	4	4	4	4

U G A U

Theme:

Date: / /

Sat Sun Mon Tue wed Thu Fri

Knapsack

Item i	Value v_i	Weight w_i
1	15	1
2	10	5
3	9	3
4	5	4

Capacity of Knapsack = 8

left = k = 0

max (prev, present)

max (0, 15+0)

(0, 15)

	K=0	K=1	K=2	K=3	K=4	K=5	K=6	K=7	K=8
	0	0	0	0	0	0	0	0	0
1 (15, 1)	0	15	15	15	15	15	15	15	15
2 (10, 5)	0	15	15	15	15	15	25	25	25
3 (9, 3)	0	15	15	15	~~15~~ 24	24	25	25	25
4 (5, 4)	0	15	15	15	24	24	25	25	29

*একটা row fill up করতে আগের row এর উপর depend করবে.

Theme:

Date: / /

Sat Sun Mon Tue wed Thu Fri

1st hand:-

K

↓

1=1

1-1=0 = K(0) pick item 1, profit 15

$$\text{Max}\left(\frac{0}{\text{prev}}, \frac{15+0}{\text{present}}\right)$$

(0, 15)

K

↓

2 > 1

2-1=1= K(1)

pick item 1, profit 15

$$\text{Max}\left(\frac{0}{\text{prev}}, \frac{15+0}{\text{present}}\right)$$

(0, 15)

K

↓

4 > 1

4-1 = 3 = K(3)

$$\text{Max}\left(\frac{0}{\text{prev}}, \frac{15+0}{\text{present}}\right)$$

(0, 15)

K

↓

3 > 1

3-1 = 2 = K(2)

pick item 1, profit 15

$$\text{Max}\left(\frac{0}{\text{prev}}, \frac{15+0}{\text{present}}\right)$$

Max (0, 15)

K

↓

5 > 1

5-1 = K(4)

$$\text{Max}\left(\frac{0}{\text{prev}}, \frac{15+0}{\text{present}}\right)$$

(0, 15)

Theme:

Date: / /

Sat Sun Mon Tue Wed Thu Fri

2nd :-

K

↓

5=5

5-5 = 0 = K(0) pick point 5, profit 10

$$\text{Max}\left(\frac{15}{\text{prev}}, \frac{10+0}{\text{present}}\right)$$

Max (15, 10)

K

↓

6>5

6-5 = 1

pick item 5, profit 10

$$\text{Max}\left(\frac{15}{\text{prev}}, \frac{10+15}{\text{present}}\right)$$

(15, 25)

K

↓

7>5

7-5 = 2 = K(2)

pick item 5, profit 10

$$\text{Max}\left(\frac{15}{\text{prev}}, \frac{10+15}{\text{present}}\right)$$

(15, 25)

3rd :-

K

↓

3=3

3-3 = 0 = K(0) pick item 3, profit 9

$$\text{Max}\left(\frac{15}{\text{prev}}, \frac{9+0}{\text{present}}\right)$$

(15, 9)

Theme:

Backtrack

4 → 29 - 5 = 24

3 → 24 - 9 = 15

1 → 15 - 15 = 0

1, 3, 4