CLASS 88

18 May 2023 21:07

Class 88

Dynamic Programming

Matrix chain multiplication / Partition DP

Subset → Knapsack = 1D = Grid = G&P → Subproblem

Order Matters

10 20 30 40 50 ← Array

Matrix

$M_1 \to 10 \times 20$

$M_2 \to 20 \times 30$

$M_3 = 30 \times 40$

$M_4 = 40 \times 50$

Total No of Multiplication

$[M_1]\ [M_2]$ = $10 \times 20 \times 30$

$10 \times 20 \quad 20 \times 30$

(M') 10×30

$M_1 \ M_2 \ M_3 \ M_4$

$[M_1]\ [M_2 M_3 M_4] \Rightarrow [M_1]\ [M'\ \ [M_4]]$ (M'') → 74000

$\Rightarrow [[M_1 M_2][M_3 M_4]]$ → 81,000

$20 \times 30 \times 40 = 24,000$

$(20 \times 40)(40 \times 50) = 20 \times 40 \times 50 = 40,000$

$M'' = 20 \times 50$

$M_1 = 10 \times 20$

$= 10 \times 20 \times 50 = 10,000$

$24,000 + 40,000 + 10,000 = 74,000$

$M'_{10 \times 30}$; $M''\ 30 \times 50$; $= 10 \times 30 \times 50 \Rightarrow 15,000$

$10 \times 20 \times 30 \Rightarrow 6,000$

$30 \times 40 \times 50 \Rightarrow 60,000$

$\Rightarrow 6k + 15k + 60k \Rightarrow 81k$

$M_1 - M_4$

MCM

$[M_1 \ M_2 \ M_3 \ M_4]$

$[M_1][M_2 M_3 M_4]$ $[M_1 M_2][M_3 M_4]$ $[M_1 M_2 M_3][M_4]$

Partition DP

$[M_2][M_3 M_4]$ $[M_2 \ M_3][M_4]$

[M2] [M3 M4]

i j

i >= j

i == j

M1

```
int solve ( int i, int j, vector<int> arr )
{   if ( i >= j )      // Base case
        return 0;

    int ans = INT_MAX;

    for ( k = i; k <= j-1; k++ )
    {   int tempAns = solve ( i, k, arr ) + solve ( k+1, j, arr ) + arr[i-1] * arr[k] * arr[j];

        ans = min ( ans, tempAns );
    }
    return ans;
}
```

Sb1

Sb2

Merge

[M1 M2 M3] [M4] ⇒

1 - n-1

[M1 M2]

[M3 M4]

20 * 30

10 20 30 40 50

i

j

k = 1

M1' M2'

arr[i-1] * arr[i]

10 * 20

20 * 50

10 * 20 * 50

(M1) (M2 M3 M4)

M1' M2'

10 * 20

20 * 50

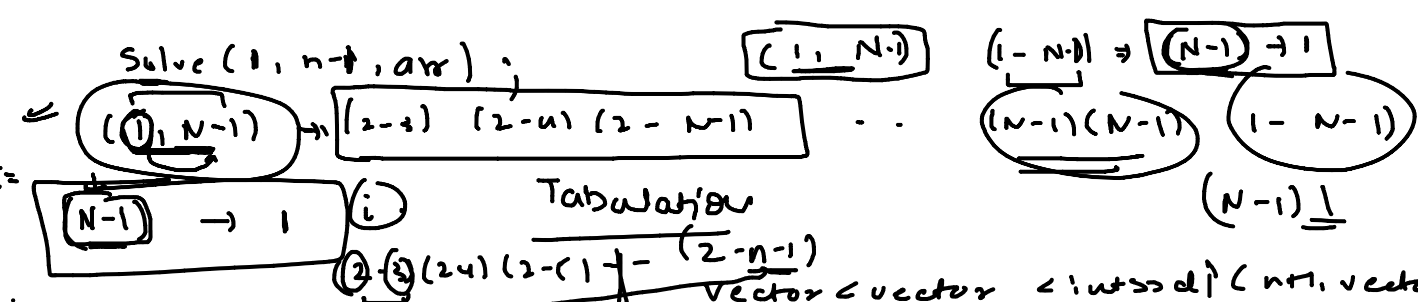
⇒

[M1] [M2 M3 M4)

10 * 20

20 * 30 | 30 * 40 | 10 * 40


```
int solve ( int i, int j, vector<int> ar)
{   if (i >= j)
        return 0;
    if( dp[i][j] != -1) return dp[i][j];

    int ans = INT_MAX ;

    For ( int k = i;  k <= j-1 ; k++)
    {  int tempans = solve( i, k, arr) + solve (k+1, j, arr) +
                     arr[ i-1] * arr[k] * arr[j] ;
                     (mergecost)

       ans = min(ans, tempans);
    }
}   return dp[i][j] = ans ;
```



vector<vector<int>> dp(n+1, vector<int>(n+1, 0));

```
int solve(int i, int j,vector<int>arr)
  {
     if(i>=j)return 0;
     int ans=INT_MAX;
     for(int k=i;k<=j-1;k++)
     {
         int tempAns = solve(i,k,arr) + solve(k+1,j,arr)
                   + arr[i-1]*arr[k]*arr[j];

         ans=min(ans,tempAns);
     }
```

```
For ( i = N-1 ;  i >= 1 ; i-- )
{ For ( j = i+1 ; ...
```

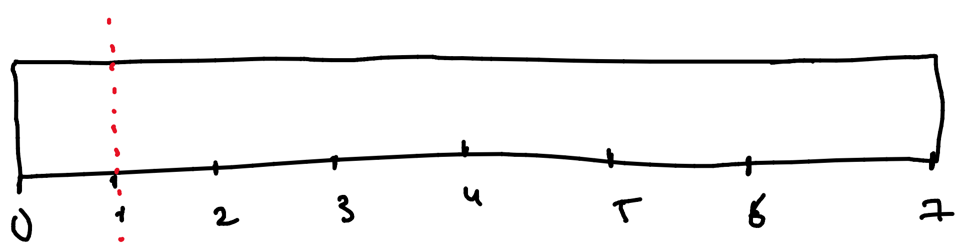
```
        int tempAns = solve(i,k,arr) + solve(k+1,j,arr)
                    + arr[i-1]*arr[k]*arr[j];

        ans=min(ans,tempAns);
    }
    return ans;
}
int matrixMultiplication(vector<int> &arr, int N)
{
    // Write your code here.
    return solve(1,N-1,arr);
}
```

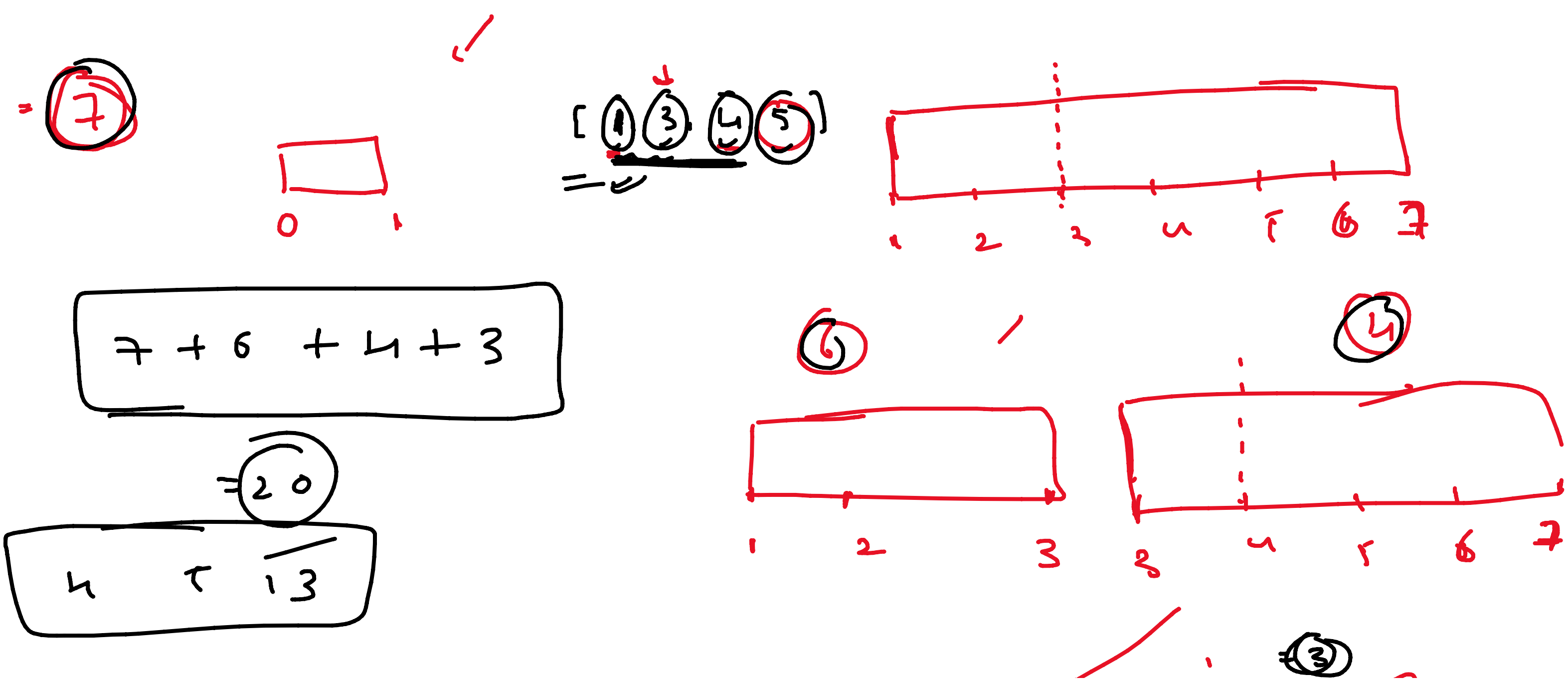
Matrix

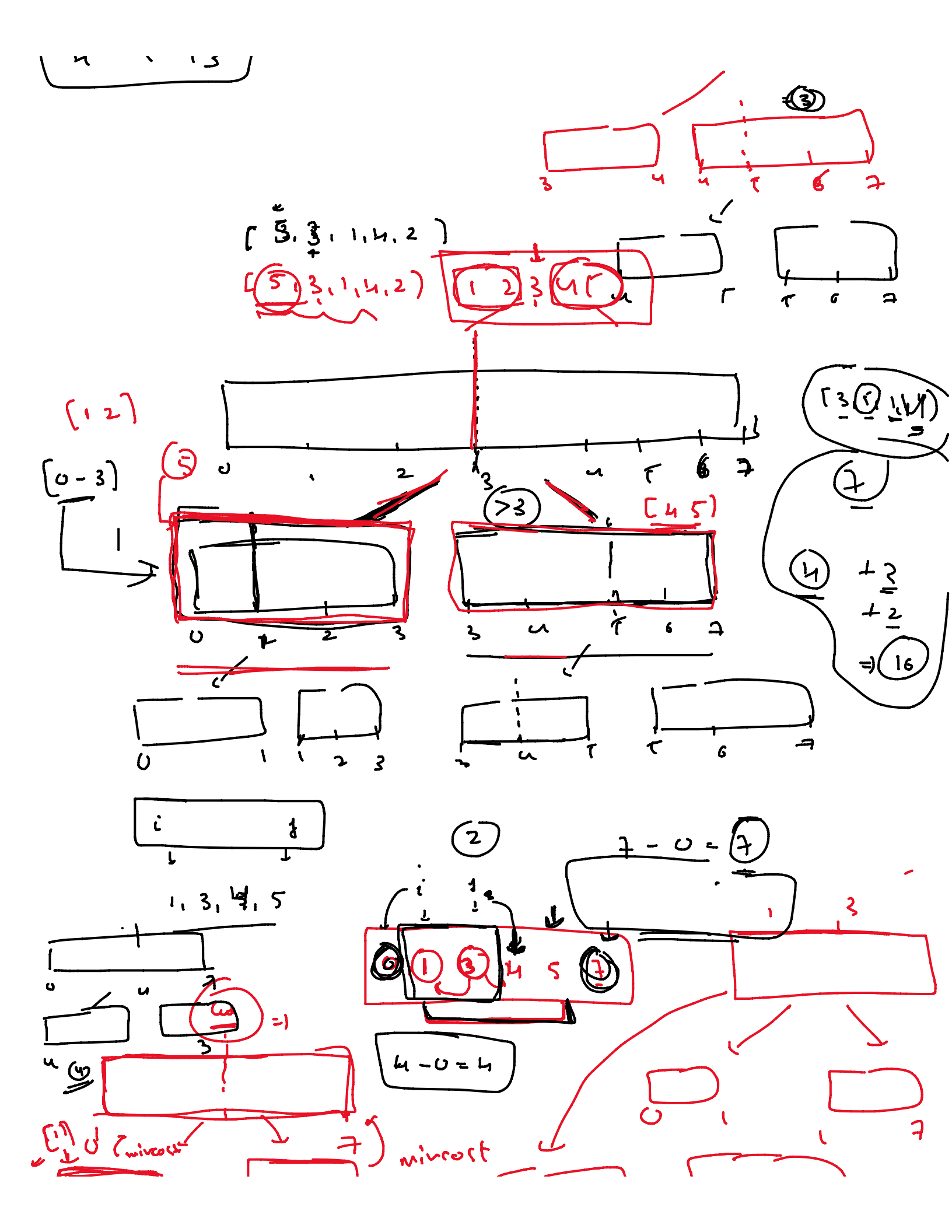
[2-2] = 0 [M1] = 0

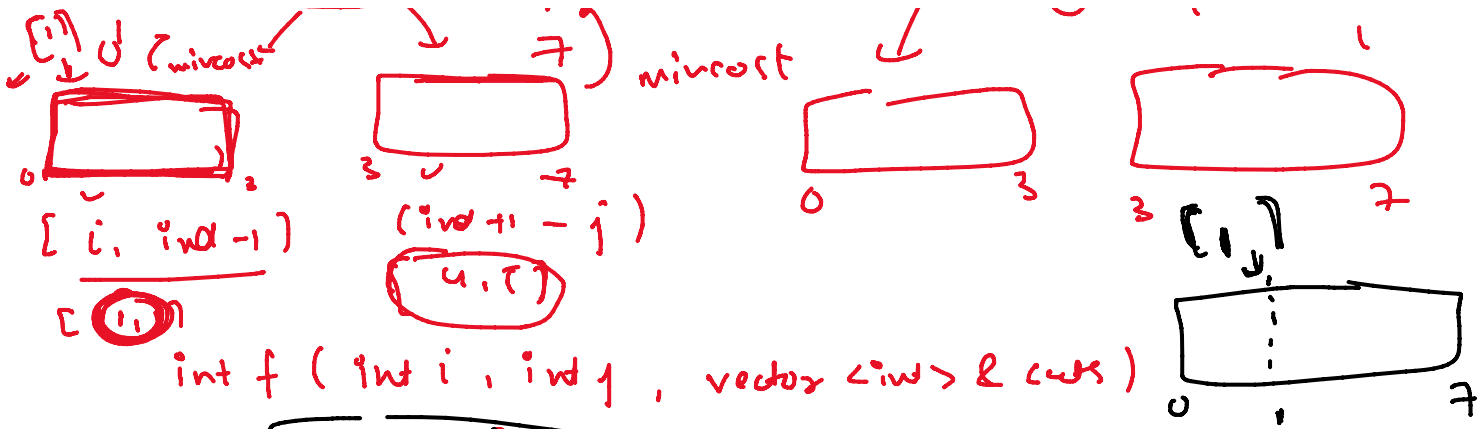
```
for (i = N-1; i >= 1; i--)
{ for (j = i+1; j < N; j++)
  {
    int ans = INT_MAX;
    for (k = i; k <= j-1; k++)
    { int tempans = dp[i][k] +
                    dp[k+1][j] +
                    arr[i-1] * arr[k] * arr[j];
      ans = min(ans, tempans);
    }
    dp[i][j] = ans;
  }
}
return dp[1][N-1];
```

Cut the Stick










```
int f ( int i , int j , vector<int> & cuts )
{
    if ( i > j )
        return 0;
    int mini = INT_MAX;
    if ( dp[i][j] != -1 )
        return dp[i][j];
    for ( int ind = i ; ind <= j ; ind++ )
    {
        int ans = f ( i, ind-1, cuts ) + f ( ind+1, j, cuts )
                  + cuts[j+1] - cuts[i-1];
        mini = min ( mini, ans );
    }
    return dp[i][j] = mini;
}
```

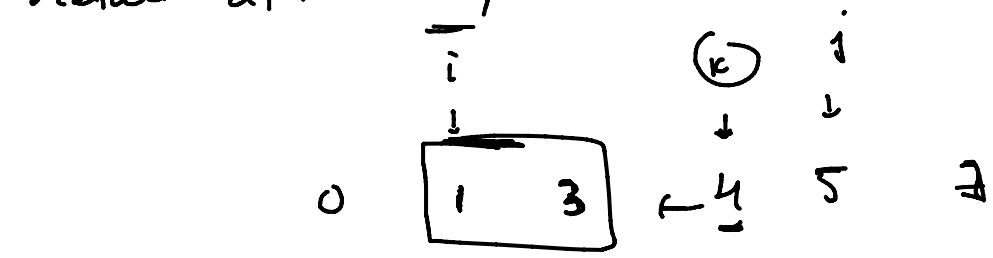
```
cuts.push_back(node-length);
cuts.insert (cuts.begin(), 0);
sort (cuts.begin(), cuts.end());
n = cuts.size();
f ( 1, n-2, cuts );
```

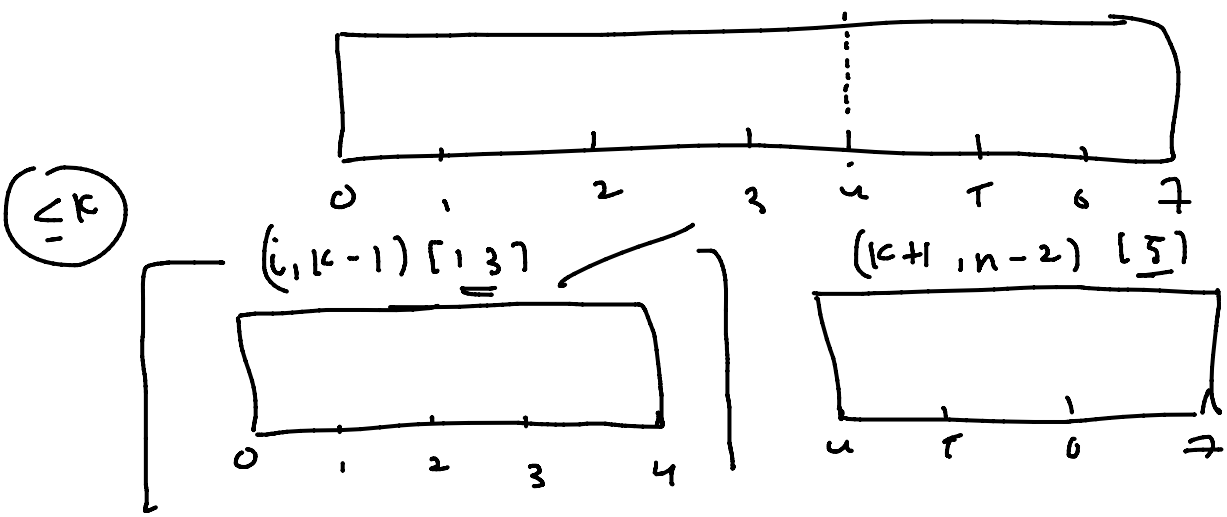
(Tabulation)

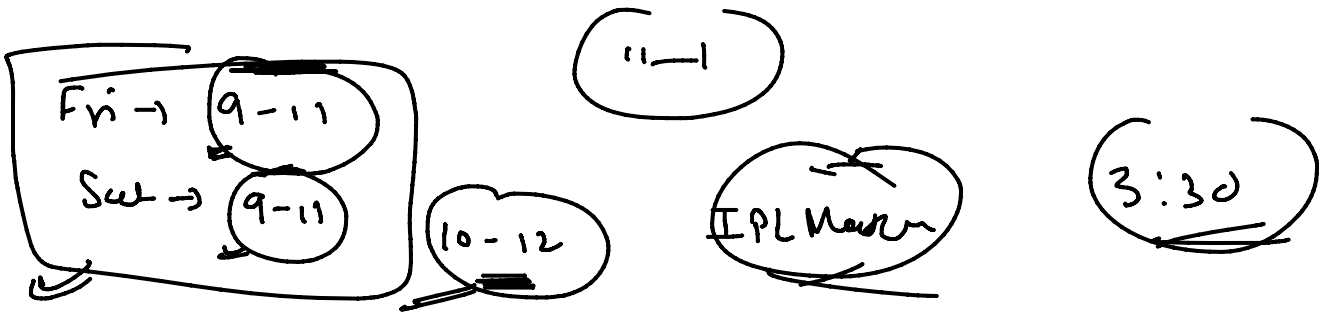
1 - n-2 — O(n-2) — (i == j)

1 - n-2 — O(n-2) — i == j

```
For ( i = n-2 ;  i >= 1 ; i-- )
{  For ( j = i ;  j <= n-2 ; j++ )
   {  int mini = INT_MAX ;
      For ( int ind = i ;  ind <= j ; ind++ )
      {  ans = cuts[i-1] + dp[i][ind-1]
                + dp[ind+1][j] ;
         mini = min(mini, ans) ;
      }
      dp[i][j] = mini ;
   }
}
return dp[1][n-2] ;
```






Fri → 9-11
Sat → 9-11
10-12
11—1
IPL Match
3:30