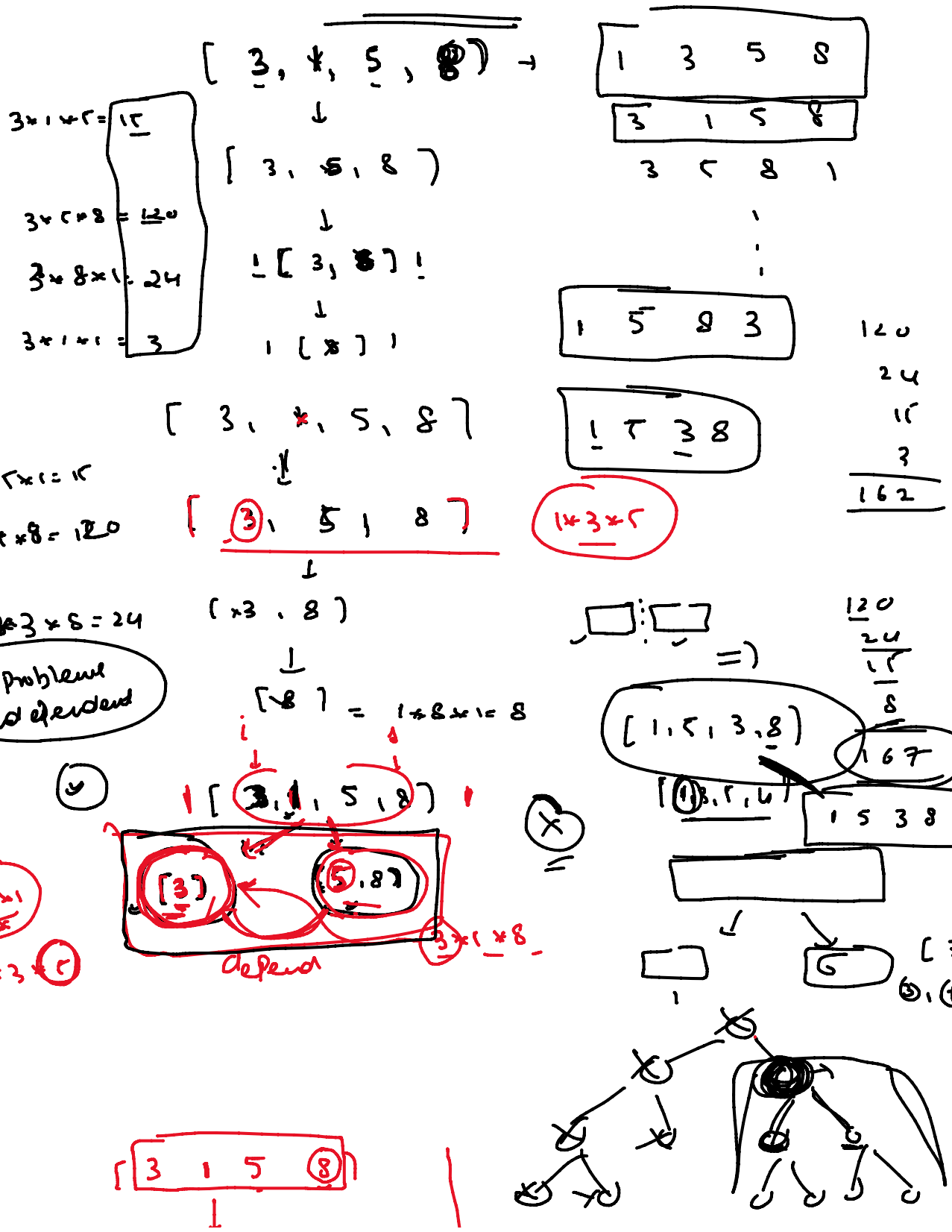
Class 89

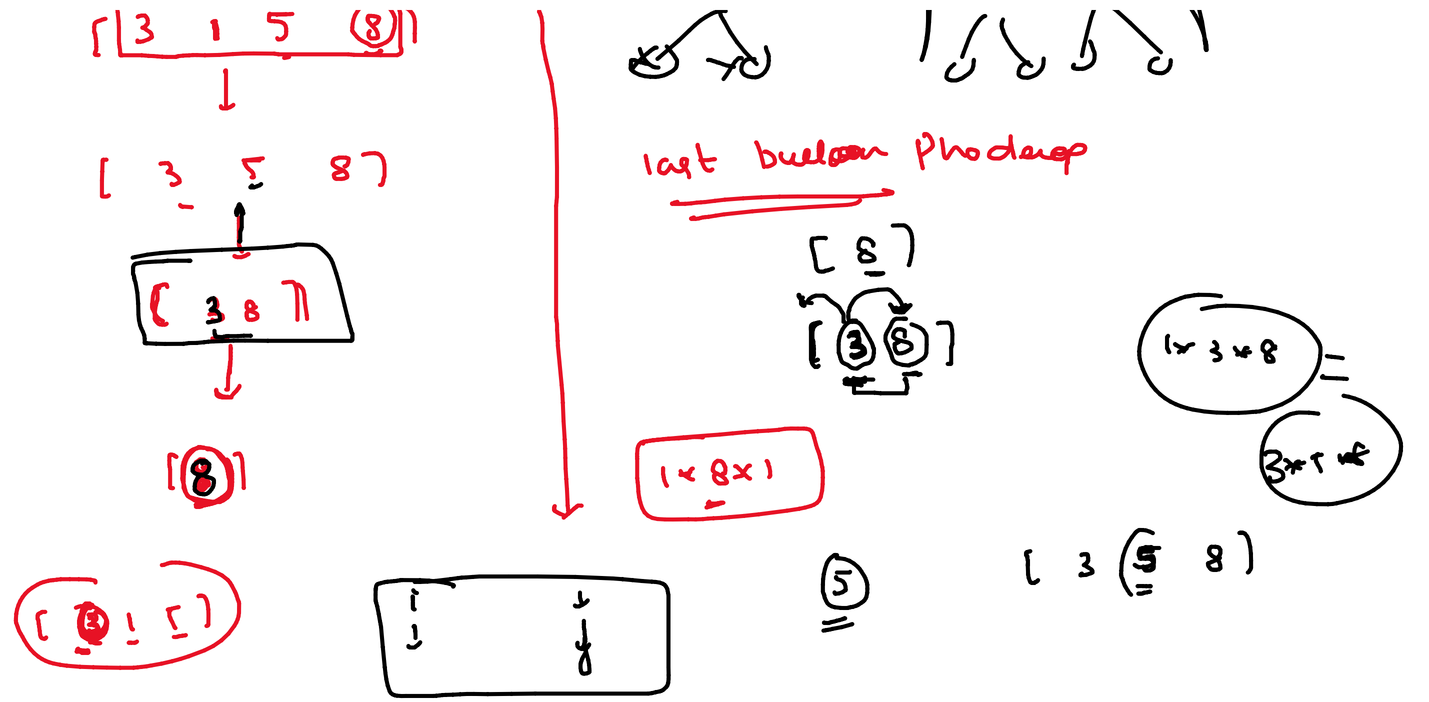
19 May 2023 21:08

Class 89

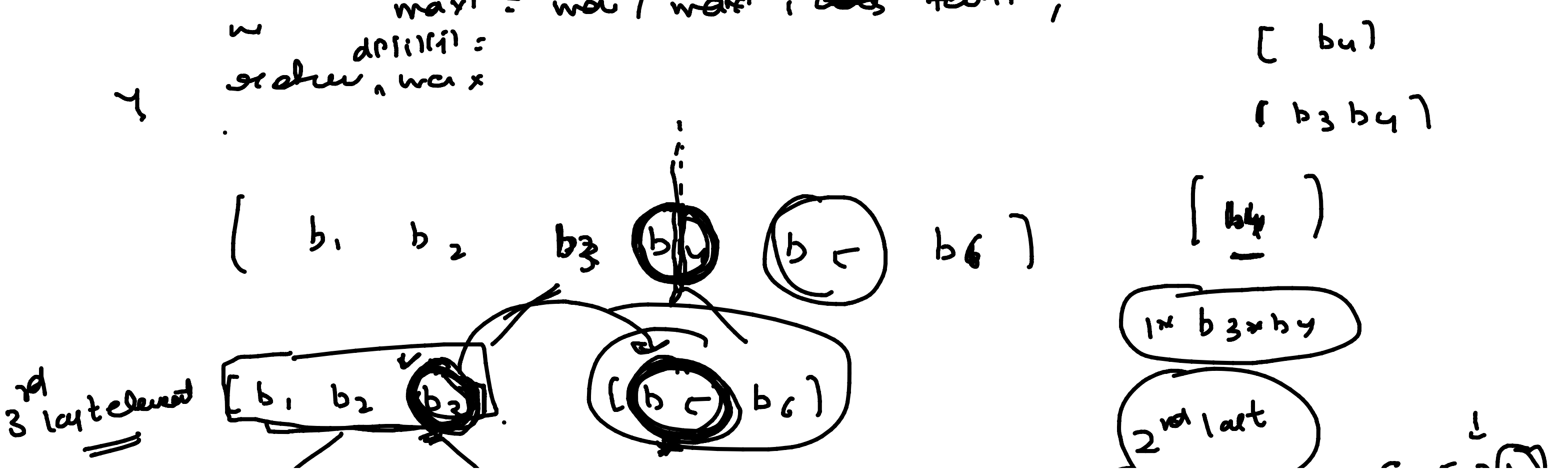
Dynamic Programming

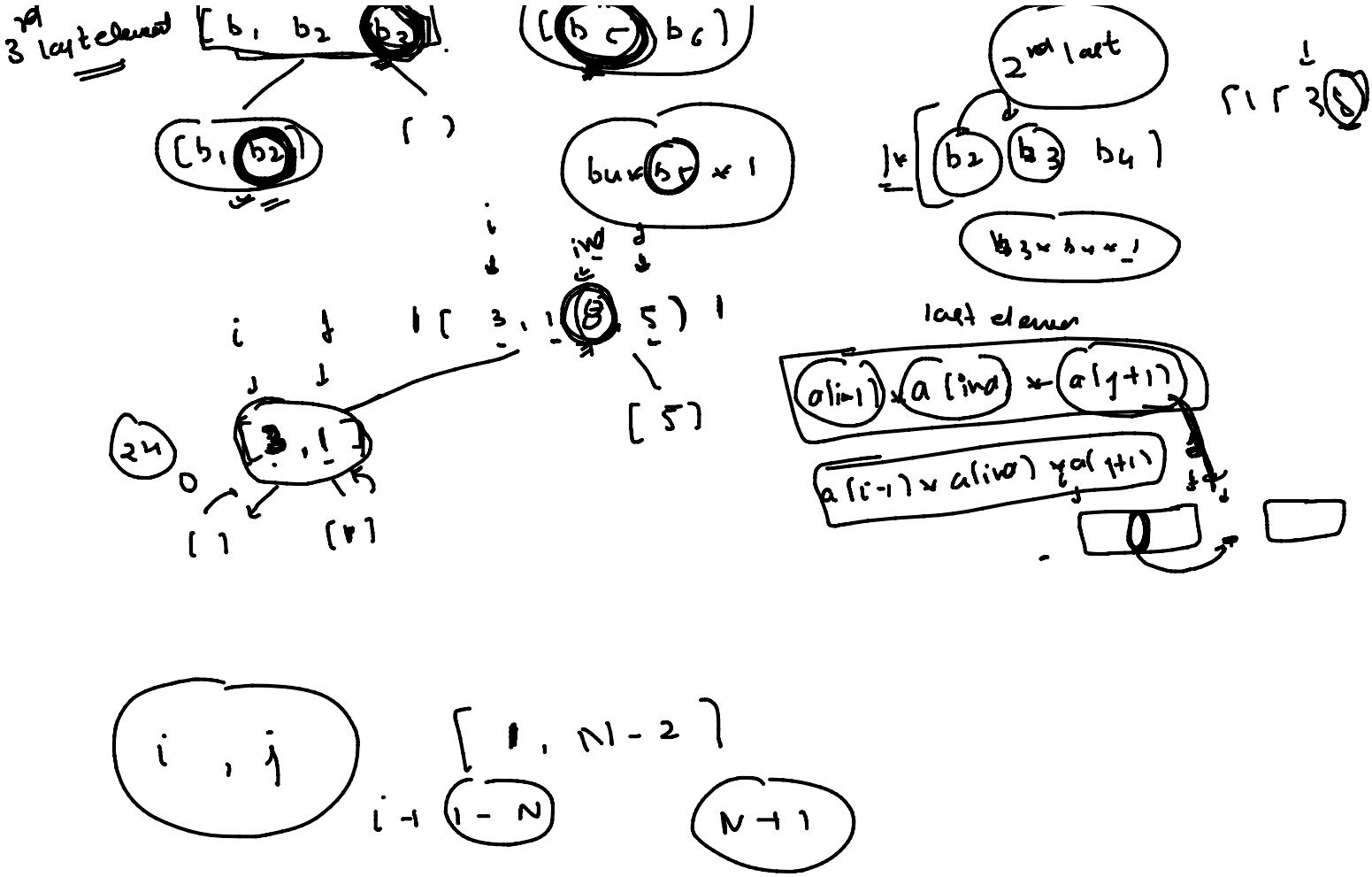
Burst Baloons





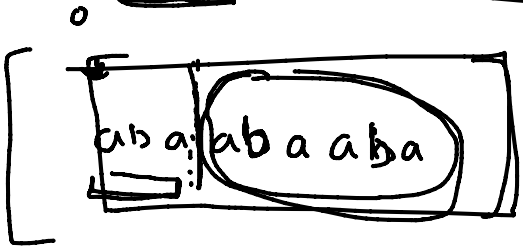

```
int f ( int i, int j, vector<int> &a )
{   if (i > j)
        return 0;
    if (dp[i][j] != -1) return dp[i][j]
    int maxi = INT_MIN ;
    for ( int ind = i ; ind <= j ; ind++ )
    {   int temp = a[i-1] * a[ind] * a[j+1] +
                   f ( i, ind - 1, a ) + f ( ind+1, j, a );
        maxi = max ( maxi , temp ) ;
    }
    return dp[i][j] = maxi
}
```




```
For ( i= n-2 ; i >= 1; i-- )
{ For( j= i ; j <= n-2 ; j++ )
{
    int maxi = INT_MIN ;
  For ( int ind= i; ind<= j ; ind++ )
  { int cost = arr[i-1] * a[ind] * a[j+1] + dp[i][ind-1] +
                                            dp[ind+1][j];
    maxi = max (maxi, cost);
  }
  dp[i][j]=maxi;
}
```

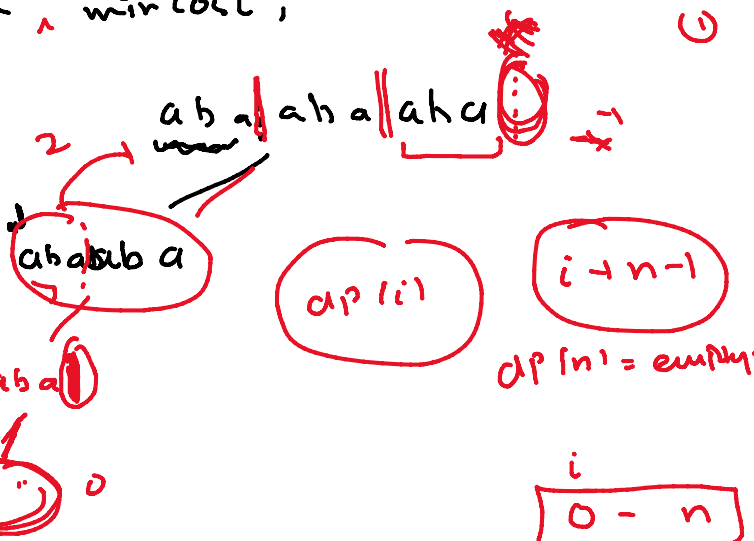
Palindrome Partitioning II



Front Partition ||

f(0) 0 - n

```
int f ( int i, int n, string str )
{
    if (i == n) return 0;          → if ( dp[i] != -1) return dp[i]
    int minCost = INT_MAX;
    for ( j = i ; j < n ; j++ )
    {
        if ( is_Palindrome ( i, j, str ))
        {
            int cost = 1 + f (j+1, n, str);
            minCost = min( minCost, cost);
        }
    }
    return dp[i] = minCost;
}
```



vector <int> dp(n+1);

dp[...] = 0;

```
vector <int> dp ....... ;

dp[n] = 0;

for ( i = n-1 ; i >= 0 ; i-- )
{   → int mincost = INT_MAX;
    for ( j = i ; j <= n-1 ; j++ )
    {   if ( isPalindrome( str, i, j ) )
        {   int cost = 1 + dp[j+1];
            mincost = min( mincost, cost );
        }
    }
    dp[i] = mincost;
}

return dp[0] - 1;
```

Square submatrices with all ones

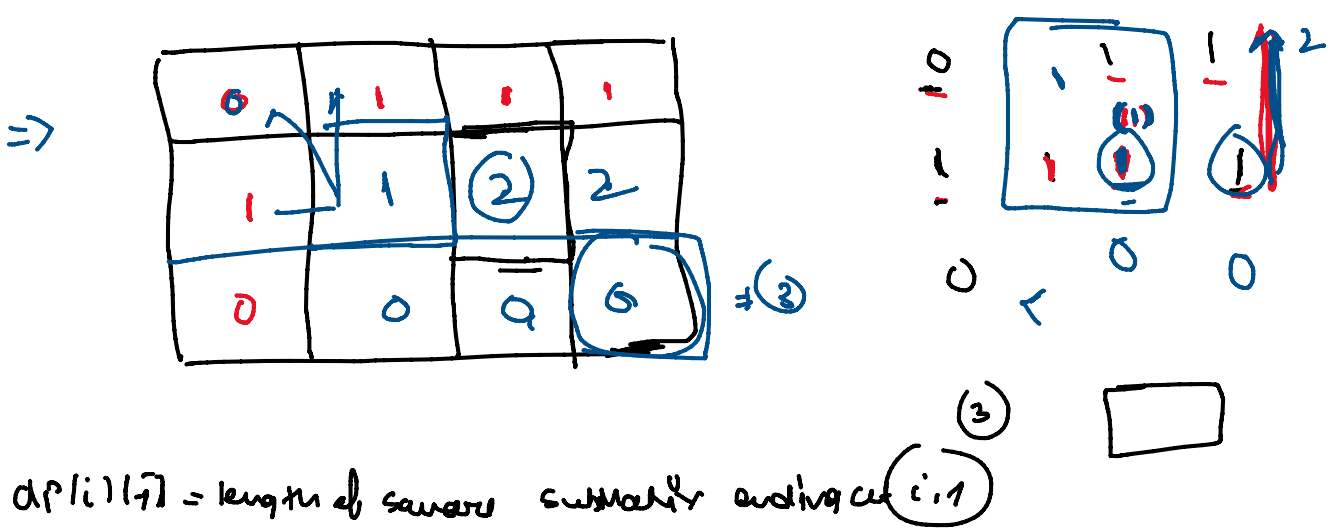
0	1	1	1
1	1	1	1
0	1	1	1

```
vector<vector<int>> dp(r, vector<int>(n, 0));

for (j = 0; j < m; j++)  dp[0][j] = arr[0][j];
for (i = 0; i < n; i++)  dp[i][0] = arr[i][0];

for (i = 1; i < n; i++)
{
  for (j = 1; j < m; j++)
```

```
for (j = 1; j<m; j++)
  if (arr[i][j]==0) dp[i][j]= 0;
  else
    dp[i][j] = 1+min( dp[i-1][j], min(dp[i-1][j-1], dp[i][j-1]);
```



dp[i][j] = length of square submatrix ending at (i,j)

Find max of dp vector