Lecture-24 Marathon Recursion Date-22/10/2023

198→LC House Robber I

Max sum of non-adjacent element

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

inclusion exclusion concept.

$\boxed{ans = 2+f(i+2, n-1)}$ $\boxed{ans = 0+f(i+1, n-1)}$

max → final Ans.

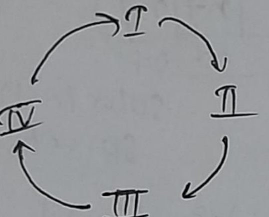
LC→213 House Robber II

I/P → nums = [2,3,2]
O/P → 3

Two adjacent house ko chori karne ke khosis ki toh alarm baj jayega.
cal → Max^m amount of money →

first ✓
Last ✗

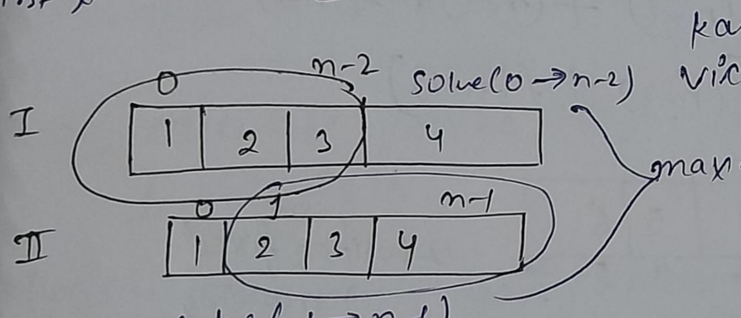
Last ✓
first ✗.



{ ✓1, 2, 3, 4✗ }

Agar first house me chori kar li toh last house me chori nhi kar payega.

solve(0→n-2) vice-versa.



solve(1→n-1)

```
int solve (vector<int> &nums, int s, int e){
    if(s>e){
        return 0;
    }
    int op1 = num[s] + solve(nums, s+2, e);
    int op2 = 0+solve(nums, s+1, e);
    int final = max(op1, op2);
    return final;

int rob(vector<int> &nums){
    int n=nums.size();
    // single elt → yha pe mai galti kar rhi hu.
    if(n==1){
        return nums[0];
    }
```

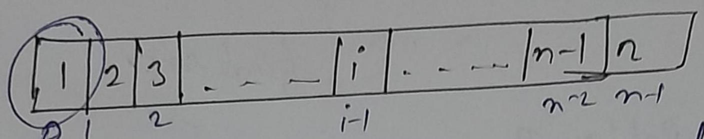
```
int op1 =
    solve(nums,0,n-2);
int op2 =
    solve(nums,1,n-1);
int ans =
    max(op1,op2);
return ans;
}
};
```

Ques2.) Count Dearangements (Permutation such that no element appears in its original position)

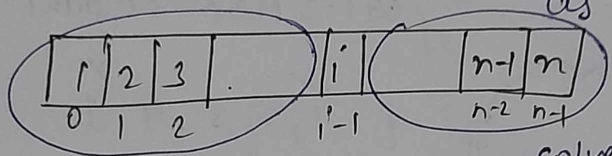
catalan Number

I/P → n=2 I/P → n=3

O/P → 1. O/P → 2.



(n-1) * f(n-2) when we swap i & 1's ~~position~~ we consider i & 1's position as fixed.



(n-1) * f(n-1)

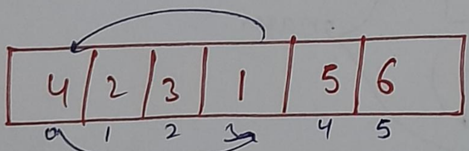
solve(n-1)

solve(n-2).

(n-1) * f(n-2) + (n-1) * f(n-2)

(n-1) * [f(n-2) + f(n-1)]

solve(6)

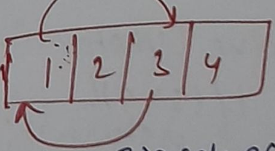


2 position are fixed. remaining

solve(6) = 5 * solve(4)

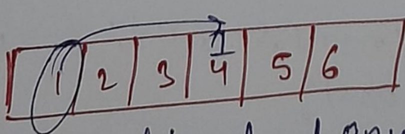
= 5 × 3

= 15



solve(4) = 3 × solve(2)

= 3 × 1 = 3



4 is placed at anywhere

solve(6) = 5 * solve(5)

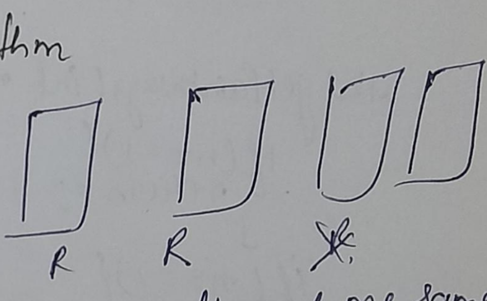
```
int solve(int n){
    // base case
    if(n==1){
        return0;
    }
  if(n==2){
        return 1;
   }
   int ans= (n-1)*(solve(n-1)+solve(n-2));
   return ans;
}
int main(){
    int n=4;
   cout<<solve(n)<<endl;
   return0;
}
```

Ques 3: Painting Fence Algorithm

Total → n post

→ n colors.

⇒ calculate possible ways to calculate.



Two adjacent me same color kar sakte hai bs.

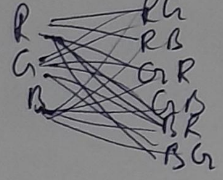
Two case →
same color
different color

same ka means last ke 2 elt.
diff. ka bhi last elt ko dekh rhe hai

	n=1	n=2	n=3	
same	R G B k=3	RR GG BB k=3	RGG RBB GRR GBB BRR BGG 3x2 k(k-1)	
diff		RB RG GB GR BR BG 3x2 k*(k-1)	RGR RGB RRB RRG RBR RBG BRB BRG BGB BGR BBR BBG GRB GRG GGB GGR GBR GBG	

X (XX) → diff

↓ 3 X 6 = 18.



f(n-2)

XX (XX)

same

f(n-2) → RR
→ GG
→ BB

diff

f(n-1) → R
→ G
→ B

(k-1)f(n-1)

(k-1) f(n-2)

$\boxed{(k-1)\times(f(n-1)+f(n-2}$

n=3

f(n-1) | X(XX)

R
G
B

RR
GG
BB

(k-1) f(n-2)

n=3
↓
diff ⇒ n=2
same × (k-1)
+
diff.

RR R
BB B
GG G

n=2

diff = same × (k-1) + diff (k-1)

```
int getPaintways(int n, int k){
    if(n==1){
      return k;
    }
   if(n==2){
     return ~~k*(k-1)~~ k+k*(k-1);
   }
   int ans =(k-1) * (solve(n-1,k)+ solve(n-2,k));
                       ↓              ↓
                   getPaintways ——→
     return ans;
  }
int main(){
   int n=3;
   int k=3;
   int ans= getPaintways(n,k);
     cout<<ans<<endl;
  return0;
 }
```

Ques 4 → LC·72

Edit Distance

Calc. the minⁿ no of operation required to convert word1 to word2.

~~otp~~ → Operation → ① insertion
remove
replace.

I/P → Word1 = h o r s e
word 2 h r o s

word1[i]
word2[j] → i & j → case → Match → aage badh jayega agale character pe.
→ Not Match → insert / replace / remove → min

W1 → (l) B a b b a r
W2 → l a b

B != l → No match → insertion / Remove / replace

insert → i, j+1

W1 = [B] a b b a r (i+1)
W2 = [l] a b

remove → i+1, j

W1 = [B] a b b a r (i+1)
W2 = [l] a b (j+1)

replace → i+1, j+1

w1 = [h] [o] [r] [s] e
w2 = [l] a [b]
replace
① +1+1+1+1
⑤ →

for base case → w1.length() - 1
5-3=2.
w1 → h o r s e (i, i+1, i+2)
w2 → . h o r (j, j+1, j+2, j → ruk jao.)

```
~~class~~ int solve(string& a, string& b, int i, int j){
    //base case
    if(i>=a.length()){
    //word1 wali string end hogyi
    // yani word1 ki length maybe word2 se choti hai
        return b.length()-j;
    }
    if( j>=b.length()){
    // word2 end hogya
    // yani word1 ki length may be greater than w2 hai
    return a.length()-i;
  }
  int ans=0;
  if(a[i]==b[j]){
     ans=0+solve(a,b,i+1,j+1);
  }
  else{
  // not match
  // operation perform karo   (1+) -> bhul rhe hai.
                                      keep in mind.
  // (insert)                          1 operation
   int op1 = 1+solve(a,b,i,j+1);       add kar
   // remove                           rhe hai
   int op2 = 1+solve(a,b,i+1,j);       isliye 1+kiye
                                       hai
   // replace
     int op3 = 1+solve(a,b,i+1,j+1);
   ans=min(op1,min(op2,op3));

  }
  return ans;
 }
int minDistance(string w1, string w2){
        int i=0;
        int j=0;
        int ans= solve(~~word~~w1,w2,i,j);
        return ans;
     }
  };
```

LC-221

Ques→ Maximal Square.

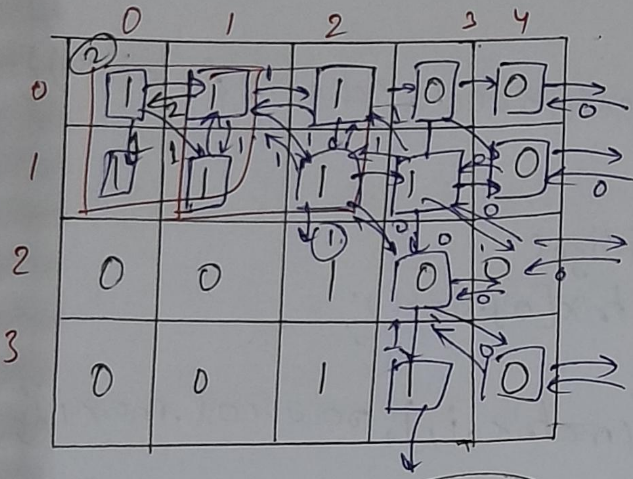
find largest square containing 1's.

H.W.

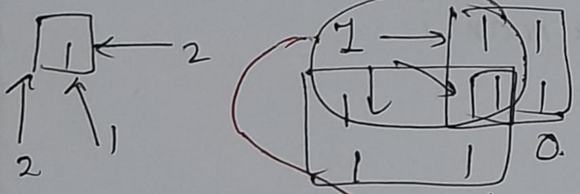
① 0/1 knapsack Problem

② (1039) Min^m score Triangular of Polygon

③ LC-1155 No of dice rolls with Target sum



	0	1	2	3	4
0	1	1	1	0	0
1	1	1	1	1	0
2	0	0	1	0	
3	0	0	1		0



Min^m isliye le rhe hai taki ye return kar de

```
int solve(vector<vector<char>> &matrix, int i, int j, int row,
          int col, int &maxi){
    // base case
    if(i>=row || j>=col){
        return 0;
    }
    //explore all 3 directions
    int right = solve(matrix, i, j+1, row, col, maxi);
    int diagonal = solve(matrix, i+1, j+1, row, col, maxi);
    int left = solve(matrix, i+1, j, row, col, maxi);
    // check can we build square for current position
    if(matrix[i][j] == '1'){
        int ans = 1 + min(right, min(down, diagonal))
        maxi = max(maxi, ans);
        return ans;
    }
else{
    // agar 0 pe hi khade hai toh ans 0 hoga
    return 0;
    }
}
int maximalSquare(vector<vector<char>> &matrix){
    int i=0,j=0;
    int row= matrix.size();
    int col = matrix[0].size();
    int maxi= 0;
    int ans= solve(matrix,i,j,row,col,maxi);
    return maxi*maxi;
  }
};
```