

DYNAMIC PROGRAMMING

Day 4

Q) <https://www.geeksforgeeks.org/maximum-sum-path-in-a-matrix-from-top-left-to-bottom-right/?ref=rp>

Given a matrix: $n \times m$

eg.

A 2×3 matrix -

1 2 3

4 5 6

$dp[2][2] \rightarrow 1, 2, 5$

$\rightarrow 1, 4, 5$

$dp[2][2] = 10$

$dp[1][2] = 1, 2 \rightarrow 3$

$dp[1][3] = 6$

$dp[2][1] = 5$

$dp[2][3] = \max(dp[1][3], dp[2][2]) + 6$

$= \max(6, 10) + 6 = 16$

movement: right or down

dest: bottom right(n, m)

start:top left(1,1)

1-2-3-6-->6+3+2+1=12

1-2-5-6-->1+2+5+6=14

1-4-5-6-->16

max val find?

(i,j)-->(i-1,j)(down)

→(i,j-1)(right)

(n,m)-->(n-1,m),(n,m-1)

dp[i][j]-->store max value we can obtain

(1,1)-->(i,j)

dp[n][m]-->ans

int dp[n+1][m+1]

i,j

dp[i][j]=max(dp[i-1][j],dp[i][j-1])+arr[i][j]

base case:(i==0||j==0)return 0;

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

```
for(int i=1;i<=n;i++)
{
    for(int j=1;j<=m;j++)
    {
        dp[i][j]=max(dp[i-1][j],dp[i][j-1])+
        arr[i][j];
    }
}
ans==dp[n][m]
```

Q)<https://codeforces.com/problemset/problem/1286/A>

N bulbs--numbering 1 to n

prev bulb odd numbered--current odd
numbered

prev even--current even

missing bulbs--0

odd->even,even->odd→++1

complexity-->min

5

0 5 0 2 3

odd=1

even=1

missing:1,4

1-->odd 5 even 2 3

complexity=2(5,4),(2,3)

2-->even 5 odd 2 3

complexity=(4,5),(1,2),(2,3)==3

min complexity achieve?

odd-->fill,even-->fill

ith index-->upto n

odds rem,even rem

missing=8

odd=3

even=5

odd--,odd=2

even=8-2=6

odd+even=missing

even-->missing

-->ith index

-->odds rem

-->even rem

ith index==odd,prev??

-->prev(odd--1/even--0)

prev==2(i==0)

ith→(i+1)th

ith→(i-1)th

(i+1)th-->ith

ith fill

ith index-->assume garland upto i

i+1th

ith-->odd→(i+1)th/even→(i+1)th

ith-->odd/even

if(arr[i]==0)//filling req

else//already filled

```
int fun(int i,int oc,int ec,int
prev)
{
    int res=INT_MAX;

    if(arr[i]!=0)
    {
        if(arr[i]%2==prev || i==0)
            res= fun(i+1,oc,ec,arr[i]%2);
        else
            res= 1+fun(i+1,oc,ec,arr[i]%2);
    }
    else
    {
        if(oc>0)
        if(i==0)
        {
            if(arr[i]==0)
            {
            }
            el
        }
    }
    res=min(res,fun(i+1,oc-1,ec,1)+(prev
```

```

==0)//slot odd fill
if(ec>0)

res=min(res,fun(i+1,oc,ec-1,0)+(prev
==1)//slot even fill
return res;
}

```

0 5 0 2 3

```

int arr[105];
int n;
int dp[105][55][55][3];

int fun(int i,int oc,int
ec,int prev)
{

    if(i==n)
        return 0;

    if(dp[i][oc][ec][prev]!=-1)

```

```
        return
dp[i][oc][ec][prev];
    int res=1e9;
    if(arr[i]!=0)
    {

if ((prev==2) || ((arr[i]%2)==p
rev))

res=fun(i+1,oc,ec,arr[i]%2);
        else

res=1+fun(i+1,oc,ec,arr[i]%2
);
    }
    else
    {
        if (prev==2)
        {
            if (oc>0)
```



```
res=min(res, fun(i+1, oc-1, ec,  
(int) 1));
```

```
    if(ec>0)
```

```
res=min(res, fun(i+1, oc, ec-1,  
(int) 0));
```

```
    }
```

```
    else
```

```
    {
```

```
        if(oc>0)
```

```
        {
```

```
res=min(res, fun(i+1, oc-1, ec,  
1) + (prev==0));
```

```
    }
```

```
    if(ec>0)
```

```
res=min(res, fun(i+1, oc, ec-1,  
0) + (prev==1));
```

```
    }
```

```
}
```

return

```
dp[i][oc][ec][prev]=res;  
}
```

int main()

{

```
memset(dp,-1,sizeof(dp));
```

```
cin>>n;
```

```
int oc=0,ec=0;
```

```
for(int i=0;i<n;i++)
```

```
{
```

```
    cin>>arr[i];
```

```
    if(arr[i]%2)
```

```
        oc++;
```

```
    else if(arr[i]!=0)
```

```
        ec++;
```

```
}
```

```
oc=(n+1)/2-oc;
```

```
ec=(n/2)-ec;
```

```
//cout<<oc<<"
```

```
"<<ec<<"\n";
```

```
int res=fun(0,oc,ec,2);
```

```
    cout<<res<<"\n";  
    return 0;  
}
```

Matrix Chain Multiplication (MCM-type problems in DP)

$A(BC)$ or $(AB)C$

$A \rightarrow i*j, B \rightarrow j*k$

$A*B \rightarrow i*k * j$

ABCDEF \rightarrow minimize the number of addition operations

$A \rightarrow 10*30, B \rightarrow 30*5, C \rightarrow 5*60$

$(AB)C \rightarrow 10*30*5 + 10*5*60 = 4500$

$A(BC) \rightarrow 30*5*60 + 10*30*60 = 27000$

$27000/4500 = 6$ times

$A[0....N]$

Size of i th matrix is $A[i-1]*A[i]$

Hint : $(AB)(CDEF) \rightarrow$ placing brackets lets us solve independent problems

Dimension of (AB) = first dimension of A * second dimension of B

Dimension of $(CDEF)$ = first dimension of C * second dimension of F

DP state $\rightarrow i, j \rightarrow$ index of first matrix and index of last matrix

$M_1, M_2, M_3, \dots, M_n$

$Dp[i][j] \rightarrow$ min number of addition operations required if you multiply matrices $M_i \dots M_j$ optimally

$Dp[i][j] = \min(dp[i][k] + dp[k+1][j] + A[i-1] * A[k] * A[j])$, k from i to $j-1$

$(ABCDE) \rightarrow (A)(BCDE)$ or $(AB)(CDE)$ or $(ABC)(DE)$ or $(ABCD)(E)$

$(A)(BCDE) \rightarrow M_1 = A$ and $M_2 = BCDE$

$M_1 \rightarrow$ first dim of A * second dim of A

$M_2 \rightarrow$ first dim of B * second Dim of E

```

int dp[n+1][n+1];
memset(dp, -1, sizeof dp);

int f(int l, int r)
{
    if(l==r) return 0;
    if(dp[l][r]!=-1) return dp[l][r];
    dp[l][r]=1e9;
    for(int k=l;k<r;k++)
        dp[l][r]=min(dp[l][r],
            dp[l][k]+dp[k+1][r]+
                a[l]*a[k+1]*a[r]);
    return dp[l][r];
}

```

Now, try these problems:

1. <https://www.spoj.com/problems/MIXTURES/>
2. https://atcoder.jp/contests/dp/tasks/dp_n
3. <https://leetcode.com/problems/minimum-cost-to-cut-a-stick/>

Principle of Inclusion and Exclusion (PIE)

$$|A \cup B| = |A| + |B| - |A \cap B|$$

$$|U A_i| = \sum(|A_i|) - \sum(|A_i \cap A_j|) + \sum(|A_i \cap A_j \cap A_k|) - \dots$$

$D(n)$ -> derangement of length n

Number of permutation P where P_i not equal to i for all i .

$$N! - F$$

F -> number of permutations where there is at least one index i such that $P_i = i$.

A_i -> number of permutations where i th element is fixed, i.e. $P_i = i$

$$F = |U A_i| = nC_1 * (n-1)! - nC_2 * (n-2)! + \dots + (-1)^{n+1} * nC_n * (n-n)!$$

Homework:

<https://codeforces.com/problemset/problem/559/C>

