October 25, 2025

DP FORM 4 & 5 DEEP DIVE

Ques 1. Burst Balloons

https://leetcode.com/problems/burst-balloons/description/

> You are given `n` balloons, indexed from `0` to `n - 1`. Each balloon is painted with a number on it represented by an array `nums`. You are asked to burst all the balloons.
>
> If you burst the `i^th` balloon, you will get `nums[i - 1] * nums[i] * nums[i + 1]` coins. If `i - 1` or `i + 1` goes out of bounds of the array, then treat it as if there is a balloon with a `1` painted on it.
>
> Return *the maximum coins you can collect by bursting the balloons wisely*.

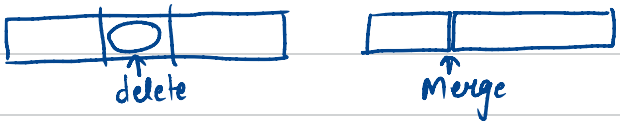
* Form 4 – LRDP

→ deals with intervals.

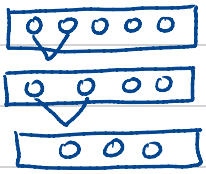
→ finding a solution from L to R and then using that to build recurrence

We see different kinds of diagrams here like :

Type 1 → delete an index & then insert it back.



Type 2 : let's say we have 5 elements in an array, if we merge 2 elements into one then the size will be 4 and we can keep merging



So, basically whenever we encounter a question where we have to delete & Merge (type 1) or just Merge (type 2) then we shall use Form 4.

Does our current question (ques 1) have any of this?

Yes, we can see that ques 1 will fall under Type 1 which means we can basically apply Form 4.

① Form : Form 4

② State :

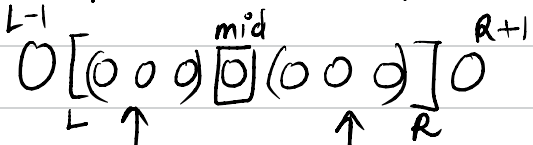
what is asked in ques

DP(l, r, ---) → ( ) for [L→R]

any restriction if any

DP(l, r) → (Max coin you can collect) for [L→R]

Assuming (L-1) and (R+1) are active indexes and we say that mid element will be deleted at the very last. In some order, we will burst L→mid-1 and mid+1→L. Its as if these bursting of elements in further 2 portions are independent.



DP(l, mid-1) + DP(mid+1, R) + DP(mid) * arr[l-1] * arr[R+1]

AlgoZenith

Now, we know that if we delete the mid at the last then we can easily calculate the score. But, for the MAX score, what will be the ideal mid? So, we can try at all elements as mid and find out the max across.

DP(l, r)

$$ans = \max_{L \le m \le R} \left(DP(l, m-1) + DP(m+1, R) + arr[m] * arr[l-1] * arr[R+1] \right)$$

Time Complexity

l → goes till N

r → N

states = $O(N^2)$

transition = N

$\Rightarrow N^2(1+N)$

$= O(N^3)$

```
int n;
int arr[505];

int dp[505][505];
int rec(int l,int r){
    if(l>r)return 0;
    if(dp[l][r]!=-1)return dp[l][r];

    int ans = 0;
    for(int mid = l;mid<=r;mid++){
        ans = max(ans, rec(l,mid-1) + rec(mid+1,r) + arr[mid]*arr
        [l-1]*arr[r+1]);

    }

    return dp[l][r]=ans;
}
void solve(){
    cin>>n;
    for(int i=1;i<=n;i++)cin>>arr[i];
    arr[0]=1; arr[n+1]=1;

    memset(dp,-1,sizeof(dp));
    cout<<rec(1,n)<<endl;
}
```

Homework Problem :

https://leetcode.com/problems/minimum-cost-to-merge-stones/

Ques 2. Merge Elements 1.

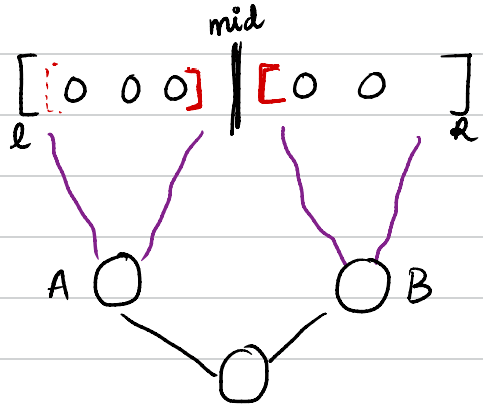
https://maang.in/problems/Merge-Elements-1-161

You are given N elements in an array A. You can take any two consecutive elements a and b and merge them. On merging you get a single element with value $(a+b)\%100$ and this process costs you $a \times b$. After merging you will place this element in place of those two elements.
If the sequence is $[A_1, A_2, A_3, A_4]$ and you merge A_2 and A_3, you incur a cost of $A_2 \times A_3$ and the array becomes $[A_1, (A_2+A_3)\%100, A_4]$.
Find the minimum cost to merge all the elements into a single element.

① Form : Form 4 (Type 2)

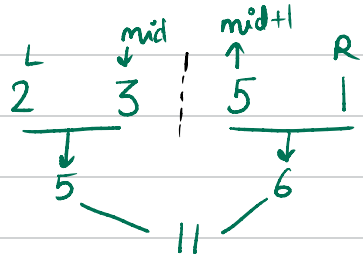
② State :

DP(l, r) → min cost to merge ele From [L → R]



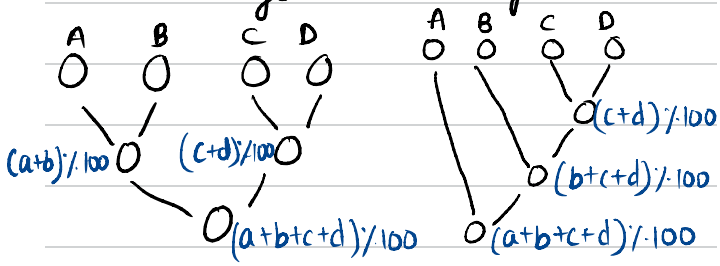
$$[DP(l, mid) + DP(mid+1, R) + A*B]$$

Example :



When two elements gets merged, we get $(A+B)\%100$

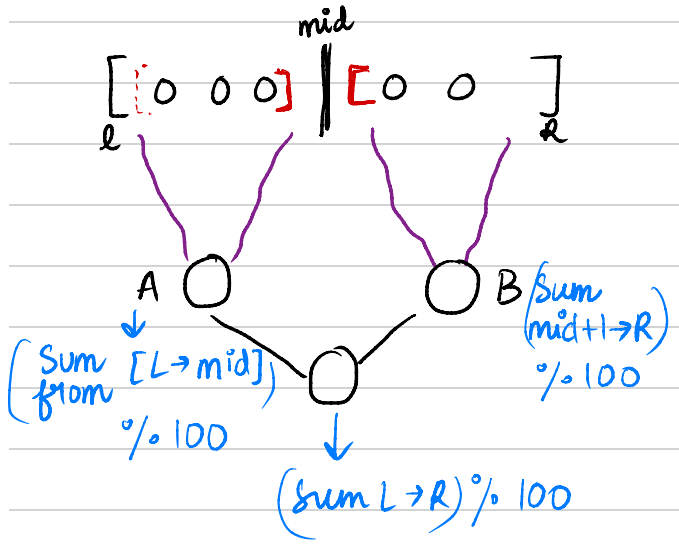
Suppose we have 4 elements, it can be merged differently like :



But, our answer will be optimal amongst it.

$\Rightarrow (a+b+c+d) \% 100$

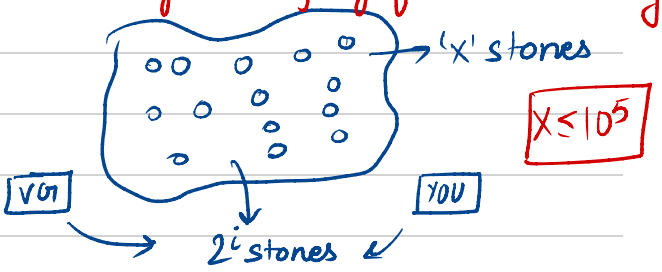
So, we observe that the A in the following diagram will be nothing but sum of the range L → mid



$$DP(l,r) = \min_{(l \le mid < R)} \left(\begin{array}{l} DP(l,mid) + DP(mid+1,R) \\ + (sum(l,mid) \% 100) \\ * (sum(mid+1,R) \% 100) \end{array} \right)$$

* Form 5 deals with game theory

Ques 2. Given a pool with 'x' stones. There's 2 players to this game, You & VG. Each one of you gets to pick 2^i stones from the pool each day. Whoever has no stones to pick from the pool, LOSES. Can it be ensured that 'YOU' will win this game. Can you win given that you are going from the first day.



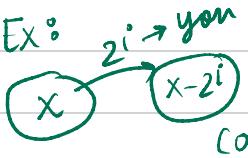
① Form : Form 5

② State :

Here, we keep configuration of the board as states. So, in this case, it will be no. of stones DP(x) → tells us that currently we have 'x' stones and whoever is playing now, can they win? (0/1)

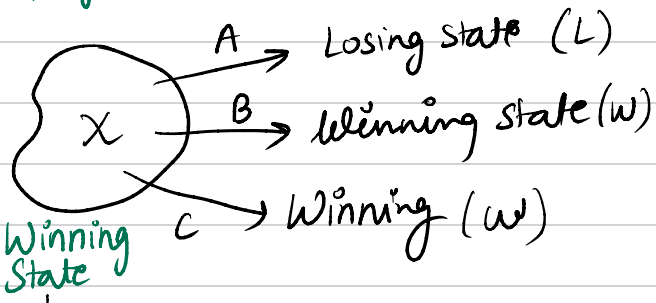
③ Transition

Ex :



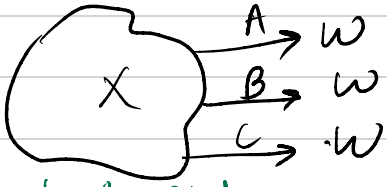
Can VG win or any player at this configuration with $x-2^i$ stones?

If $DP(x-2^i)=1$, that means player can win it.



Winning State

↳ If there is any transition to the losing state then you are at the winning state overall.



Losing State

↳ All my transitions (YOU) is going to winning state which puts me in a losing state.

T.C

$(x-2^i) \geq 0$

$i \approx \log_2(x)$

```
int dp[100100];
int rec(int x){
    if(x==0){
        return 0;
    }
    if(dp[x]!=-1)return dp[x];
    int ans = 0;
    for(int i=0;(1<<i)<=x;i++){
        if(rec(x-(1<<i))==0){
            ans = 1;
            break;
        }
    }

    return dp[x] = ans;
}
```

```
void solve(){
    int x;
    cin>>x;
    memset(dp,-1,sizeof(dp));
    cout<<rec(x)<<endl;
}
```

```
void solve(){
    int x;
    cin>>x;
    if(x%3)cout<<1;
    else cout<<y;
    // memset(dp,-1,sizeof(dp));
    // for(int i=0;i<=100;i++){
    //     cout<<i<<": "<<rec(i)<<" "<<((i%3)==0?0:1)<<endl;

    // }
}
```