**Problem Name : SUPW**

**Topic : DP**

**Difficulty : Medium**

**EXPLANATION**

PROBLEM IN SHORT – We have to select some elements from the array, so that the sum of selected elements is as minimum as possible with respect to constraint that there should not be 3 or more unselected consecutive elements.

KEY OBSERVATIONS – Suppose, for some i in 1,…,N, we want to calculate the answer for the array [1,i], if we select i^{th}i th element and we know the answer for arrays [1,i-1] , [1,i-2] and [1,i-3], selecting (i-1)^{th}(i−1) th , (i-2)^{th}(i−2) th and (i-3)^{th}(i−3) th element respectively. The answer for array [1,i] would be minimum of answer of the 3 arrays – minimum( [1,i-1] , [1,i-2] and [1,i-3] ) + a[i].

EXPLANATION – From key observations, we can build a recurrence. This recurrence can be easily implemented by dynamic programming. dp[i] denotes the minimum answer for array [1,i] by selecting i^{th}i th element. The recurrence code will look like this –

int r[] = {1,2,3};

for(int i = 1;i < n+1;i++)

{

for(int j = 0;j < 3;j++)

{

if(i-r[j] >= 0)

dp[i] = min(dp[i],a[i] + dp[i-r[j]]);

else

dp[i] = a[i];

}

}

The above recurrence is in accordance with the key observation. The answer to the problem would be minimum of dp[N], dp[N-1] and dp[N-2], as we have to select one of the element from the last 3 indices in order to follow the constraint that there should not be 3 or more un-selected consecutive elements.

ANOTHER INTERESTING SOLUTION – The problem IPL and this problem are very much related. So much so, that a 1-2 line change in the code of either problem can work as the solution for the other problem. This is because of a very simple relation - Sum of elements of array = answer of problem IPL + answer of problem SUPW. The proof of this relation is left on to the readers to be derived.

**Time Complexity : O(N\*3N∗3).**

#include<bits/stdc++.h>

using namespace std;

#define int long long

signed main(){

int n;

cin>>n;

vector<int> a(n);

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

cin>>a[i];

}

if(n==1){

cout<<a[0]<<endl;

return 0;

}

if(n==1){

cout<<min(a[0],a[1])<<endl;

return 0;

}

if(n==2){

cout<<min(a[0],min(a[1],a[2]))<<endl;

return 0;

}

vector<int> dp(n,INT\_MAX);

dp[0]=a[0];

for(int i=1;i<n;i++){

for(int j=0;j<3;j++){

if(i-j-1>=0){

dp[i]=min(dp[i],a[i]+dp[i-j-1]);

}

else{

dp[i]=a[i];

}

}

}

cout<<min(dp[n-1],min(dp[n-2],dp[n-3]))<<endl;

}

**Approach 2**

#include<bits/stdc++.h>

#define ll long long int

#define cin(a,n) for(i=0;i<n;i++){cin>>a[i];}

#define fast ios\_base:: sync\_with\_stdio(false); cin.tie(NULL); cout.tie(NULL);

#define forl(i,a,n) for(i=a;i<n;i++)

#define ld long double

#define cout(a,n) for(int h=0;h<n;h++)cout<<a[h]<<" ";

#define mod 1000000009

#define nl cout<<"\n";

#define de cout<<"here";

using namespace std;

int main()

{

ll q,q1,j,i,n,m,k,ans=0,x,cnt=0,l=0,r;

cin>>n;

ll a[n+1];

cin(a,n);

a[n]=0;

ll sum[n+1];

sum[0]=a[0];

sum[1]=a[1];

sum[2]=a[2];

forl(i,3,n+1){

sum[i]=a[i]+min(sum[i-1],min(sum[i-2],sum[i-3]));

}

cout<<sum[n];

}