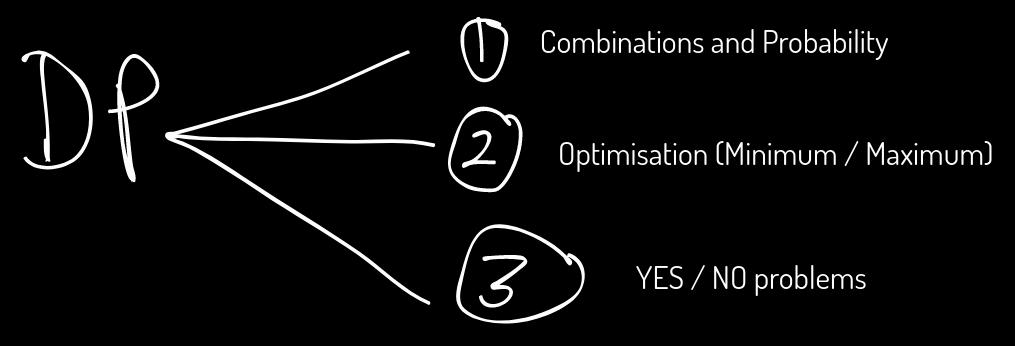
# Solving easy DP Problems

## Recap

For DP, we need:-

1. Recursive relation (with base case)

2. Overlapping subproblems



1.

<https://atcoder.jp/contests/dp/tasks/dp_c>

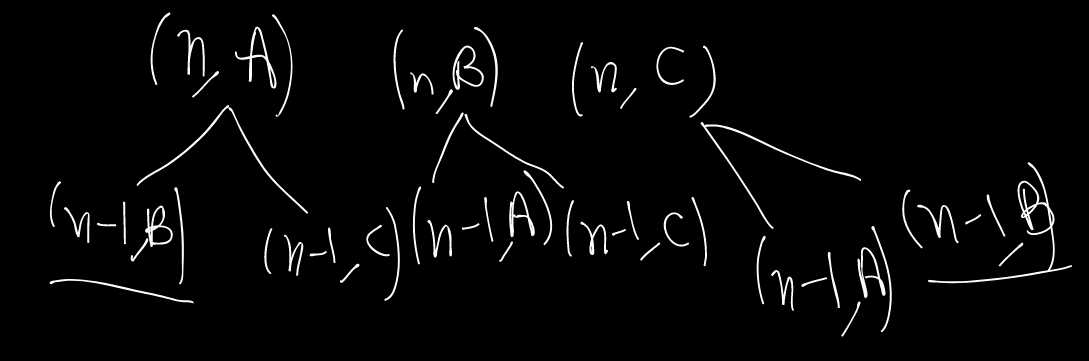
dp[i][j] = The maximum total happiness points till ith day if he performed j^th activity on the current day.

dp[i][j] = max ( dp[i-1][k] + pj, k ≠ j )

dp[0][0] = p0

dp[0][1] = p1

dp[0][2] = p2



In the above recursion tree, you can observe the overlapping subproblems.

| // Problem : C - Vacation // Contest : AtCoder - Educational DP Contest // URL : https://atcoder.jp/contests/dp/tasks/dp\_c  #include <bits/stdc++.h> using namespace std;  int points[100005][3]; int dp[100005][3];  int solve(int i, int j) {  // i: current day  // j: activity performed on i-th day  if (i == 1) {  return points[i][j];  }   if (dp[i][j] != -1) return dp[i][j];   int ans = 0;  for (int k = 0; k < 3; k++) {  // k: activity on (i-1)th day  if (k != j) {  ans = max(ans, solve(i - 1, k) + points[i][j]);  }  }  return dp[i][j] = ans; }  int32\_t main() {  int n;  cin >> n;   for (int i = 1; i <= n; i++) {  for (int j = 0; j < 3; j++) {  dp[i][j] = -1;  }  }   for (int i = 1; i <= n; i++) {  for (int j = 0; j < 3; j++) {  cin >> points[i][j];  }  }   cout << max(max(solve(n, 0), solve(n, 1)), solve(n, 2));   return 0; } |
| --- |

2.

<https://codeforces.com/problemset/problem/455/A>

9

1 2 1 3 2 2 2 2 3

Score =2 , 2 2 2 2

1 1 2 2 2 2 2 3 3

dp[1]=2;

dp[2]=2\*5=10

dp[3]=3\*2=6;

1 2 3

dp[1]=1,dp[2]=2,dp[3]=3;

I , freq[i];

i\*freq[i]

**dp[i] : max score you can get by considering numbers till i**

dp[1] = 2

dp[2] = dp[1], dp[0]+2\*freq[2]

dp[3] = dp[2],dp[1]+3\*freq[3];

dp[i] = max(dp[i-1],dp[i-2]+i\*freq[i])

ans = dp[100000]

**Sol:**

| int n;  cin>>n;  int a[n];  int freq[100001]={0};  for(int i=0;i<n;i++){  cin>>a[i];  freq[a[i]]++;  }  int dp[100001]={0};  dp[1] = freq[1];  for(int i=2;i<=1e5;i++){  dp[i] = max(dp[i-1],dp[i-2]+i\*freq[i]);  }  cout<<dp[100000]; |
| --- |

3) <https://codeforces.com/contest/1475/problem/G>

**Approach:- I’ve discussed using a whiteboard in video.**

**Solution:**

| #include<bits/stdc++.h> using namespace std; int main(){  int t;  cin>>t;  while(t--){  int n;  cin>>n;  vector<int> a(n);  for(int i=0;i<n;i++) cin>>a[i];  const int MAX=2e5+1;  vector<int> dp(MAX);  //dp[i]-> The max reside if we consider i element at the end of the beautiful array  sort(a.begin(),a.end());  dp[a[0]]=1;   for(int i=1;i<n;i++){  dp[a[i]]=dp[a[i]]+1;  // Now I’m going through all the factors of a[i] below   if(a[i]!=1) dp[a[i]]=max(dp[a[i]],dp[1]+1);  // Here factor is 1  for(int j=2;j\*j<=a[i];j++){  if(a[i]%j==0){  dp[a[i]]=max(dp[a[i]],dp[j]+1);  // Here factor is j  dp[a[i]]=max(dp[a[i]],dp[a[i]/j]+1);  // Here factor is a[i]/j  }  }   }  int ans=0;  for(int i=0;i<=MAX;i++) ans=max(ans,dp[i]);  cout<<n-ans<<endl;   } } |
| --- |
|  |

**H.W:- Solve the problem using a recursive approach.**