CLASS 82

11 May 2023 21:09

Class 82

Dynamic Programming

Subset

Coin change problem → I, II

1, 2, 5 — Indian Rupees

amount: 11

1 * 11

2 * 5 + 1 * 1

5 * 2 + 1 → 3

5 + 2 * 3

Greedy — Subset — Min lengths — rod cutting — unbounded knapsack

[1, 15, 25] — amount = 30 — 30 − 25 = 5 ✗ Greedy

5, 5, 2 ...

```
int minCoins(int idx, int amount, vector<int>& coins)
{
    if (amount == 0)
        return 0;
    if (idx == 0)
        return INT_MAX;
    int op1 = INT_MAX, op2 = INT_MAX;
    if (coins[idx-1] <= amount)
    {
        op1 = minCoins(idx, amount - coins[idx-1], coins) + 1;
    }
    op2 = minCoins(idx-1, amount, coins);
    return min(op1, op2);
}
```

I1 I2 I3 I4

amount = 0 → 1 base case

idx = 0 → INT_MAX

Invalid case: [5, 5, 2, 2, 1, 1, 1]

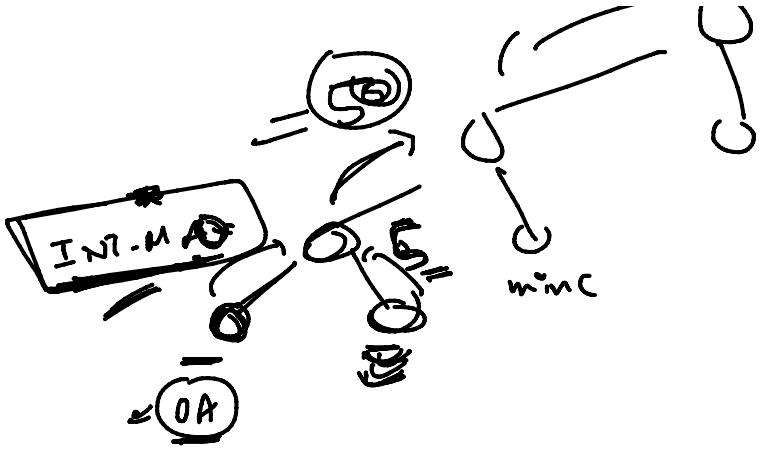
[1, 2, 5]

if (dp[idx][amount] != -1) return dp[idx][amount];

temp-fun-backtrack (coins[idx-1])

temp-fun-backtrack

dp[idx][amount] =




```
int minCoins (int amount, vector<int> &coins , vector<int>&dp,)
{
    if(amount == 0) return 0;
    if (dp[amount] != -1) return dp[amount];
    int ans = INT_MAX;
    for ( int i=0; i< coins.size(); i++)
    {
        if (coins[i] <= amount)
        {
            ans = min(ans , minCoins (amount - coins[i],
                                      coins, dp );
        }
    }
    return dp[amount] = ans;
}
```

[1,2,5] [1,2,5] [1,1,1,1,1] [2,1,1,1] [2,2,1] [5]

(1, 2, 5) (10) (9) (8) (7) Min Coins

[1, 1, 1, 1, 1, 1, 1, 1, 1, 1] → [1, 1, 1, 1, 1, 1, 1, 1, 2] → [1, 1, 1, 1, 1, 1, 1, 2, 2] → [1, 1, 1, 1, 1, 1, 2, 2, 2] → [1, 1, 1, 1, 1, 2, 2, 2, 2] → [1, 1, 1, 1, 2, 2, 2, 2, 2] → [1, 1, 1, 2, 2, 2, 2, 2, 2] → [1, 1, 2, 2, 2, 2, 2, 2, 2] → [1, 2, 2, 2, 2, 2, 2, 2, 2] → [2, 2, 2, 2, 2, 2, 2, 2, 2]

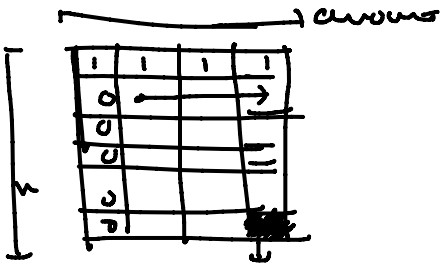
```cpp
int coinChange(vector<int>& coins, int amount) {
    int i,j,k,n,m;
    n=coins.size();
    vector<int>dp(amount+1,INT_MAX);
    dp[0]=0;
    for(i=1;i<=amount;i++){
        dp[i]=INT_MAX;
        for(j=0;j<n;j++){
            if(i-coins[j]>=0){
                if(dp[i-coins[j]]!=INT_MAX)
                    dp[i]=min(dp[i],dp[i-coins[j]]+1);
            }
        }
    }
    if(dp[amount]==INT_MAX)
    dp[amount]=-1;
    return dp[amount];
}
```

Coin Change 2

Total combinations

```
int combinations( int idx, int amount, vector<int> &coins)
{
    if (amount == 0) return 1;

    if (idx == 0) return 0;
    if (dp[idx][amount] != -1) return dp[idx][amount];
    int take = 0, discard = 0;
    if (coins[idx-1] <= amount)
    {
        take = combinations(idx , amount - coins[idx-1], coins);
    }

    discard = combinations(idx-1, amount , coins);

    dp[idx][amount] =
    return take + discard;
```

1D DP? 🙂



Driver Problem Base case

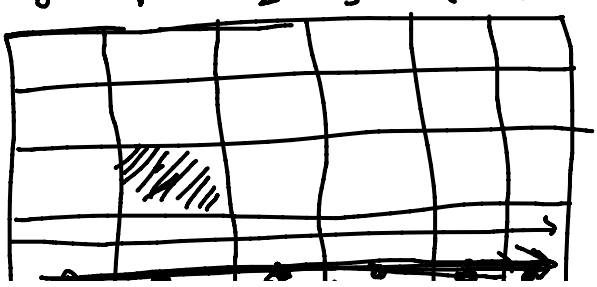
```
int change(int amount, vector<int>& coins) {
        int i,j;
        vector<vector<int> >dp(coins.size()+1,vector<int>(amount+1));
        int n=coins.size();
        for(i=0;i<=n;i++)
        {
            for(j=0;j<=amount;j++)
            {
                if(i==0 && j==0)
                    dp[i][j]=1;
                else if(j==0)
                    dp[i][j]=1;
                else if(i==0)
                    dp[i][j]=0;
            }
        }
        for(i=1;i<=n;i++)
        {
            for(j=1;j<=amount;j++)
            {
                if(coins[i-1]>j)
                {
                    dp[i][j]=dp[i-1][j];
                }
                else
                {
                    dp[i][j]=dp[i][j-coins[i-1]]+dp[i-1][j];
                }
            }
        }
        return dp[n][amount];

    }
```

Minimum subset Difference

→ Subset sum equal to K

K

tabulation → vector<vector<int>> dp(n+1, vector<int>(K+1,-1));



sum K = sum

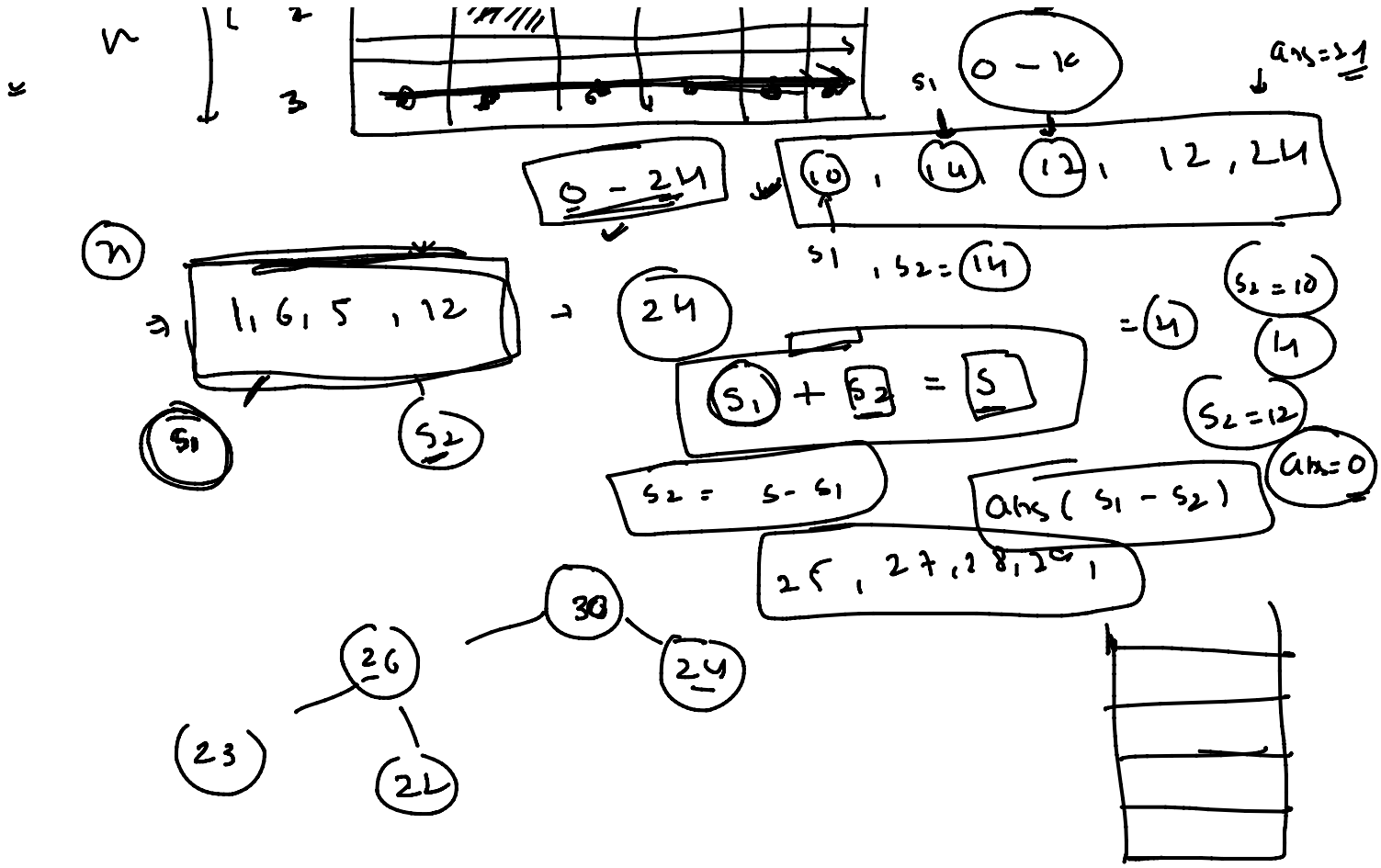
0 - 24

0 - 24

n

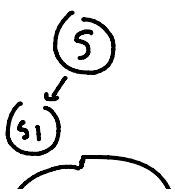
0 - K

ans = 1




```
int i,j,k,n,m;
    n=nums.size();
    int s=0;
    for(auto i:nums){
        s+=i;
    }
    vector<vector<int>>dp(n+1,vector<int>(s+1,0));
    for(i=0;i<=n;i++){
        dp[i][0]=1;
    }
    for(i=0;i<=n;i++){
        for(j=1;j<=s;j++){
            if(i==0)continue;
            dp[i][j]=dp[i-1][j];
            if(j>=nums[i-1])
            dp[i][j]|=dp[i-1][j-nums[i-1]];

        }
    }
```

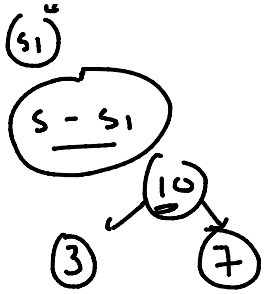
subset sum equals to



int ans= INT_MAX

For (i= 0; i<= s; i++)

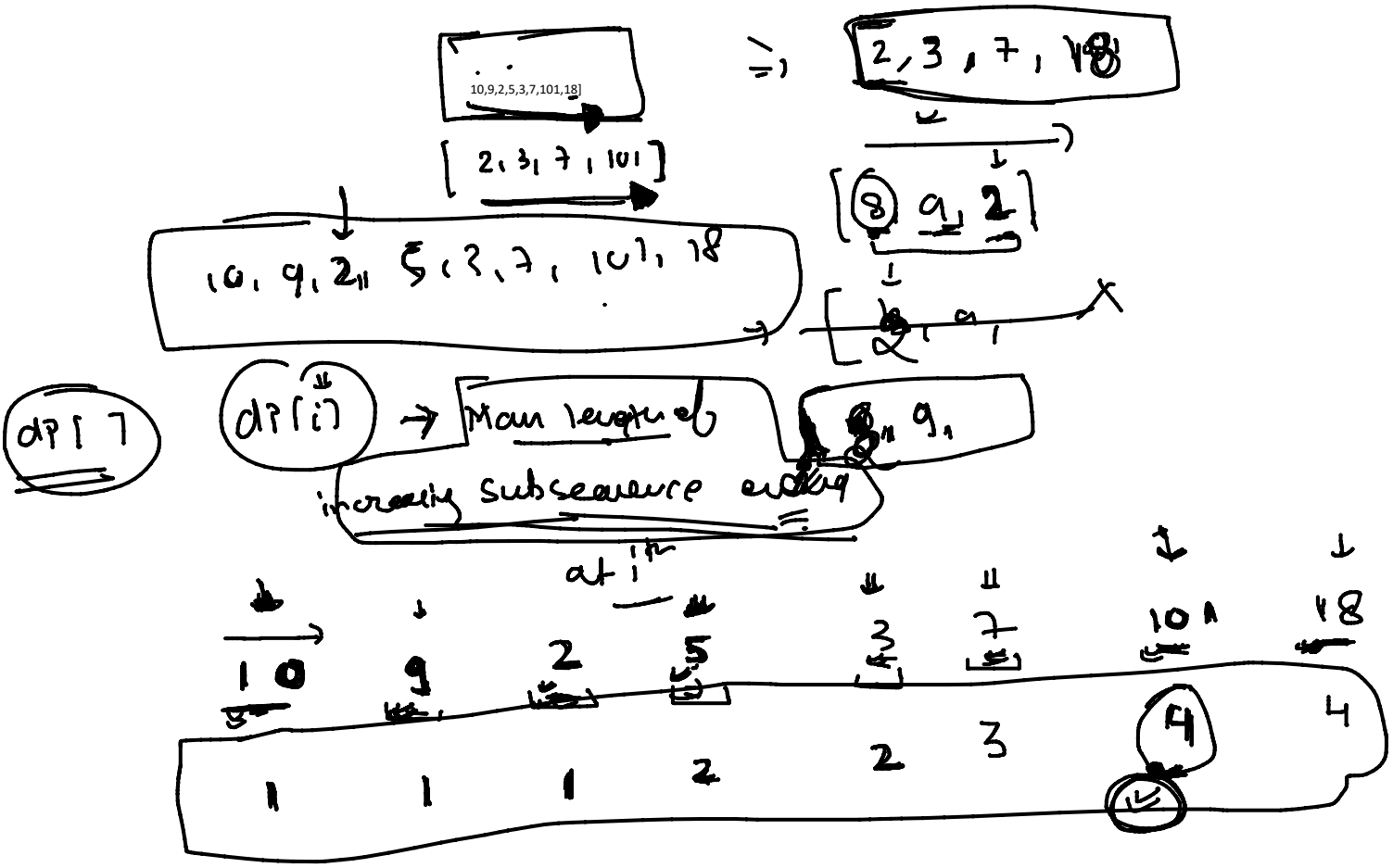
{ if (dp[n-1][i] == 1)




```
if ( dp[n-1][i] == 1)
{ int s1 = i;
  int s2 = s - s1;
  if (dp[n-1][s2]==1)
  ans = min ( ans, abs ( s1 - s2 ) );
}
```

return ans;

Longest Increasing subsequence




```
vector<int> dp(n);
dp[0] = 1;
```

```
For ( i=1; i<n; i++)
{ int ans = 0;
  For(j=0; j< i-1; j++)
  { if (nums[i] > nums[j])
    { ans = max(ans, dp[j]+1);
    }
  }
  dp[i] = ans;
}
return *max_element( dp.begin(), dp.end() );
```

n^2