Minimum Squares Sum

Problem

Given a number n, your task is to find the minimum number of numbers which sums to n.

Example

$$n = 26 = 4^2 + 3^2 + 1^2 \ \{3 \text{ numbers}\}$$

$$\text{or } 26 = 5^2 + 1^2 \ \{2 \text{ numbers}\}$$

So the minimum number of numbers required are 2.

Recurrence Relation

Base Case:

$$f(0) = 0$$

$$f(1) = 1 \quad \{1^2\}$$

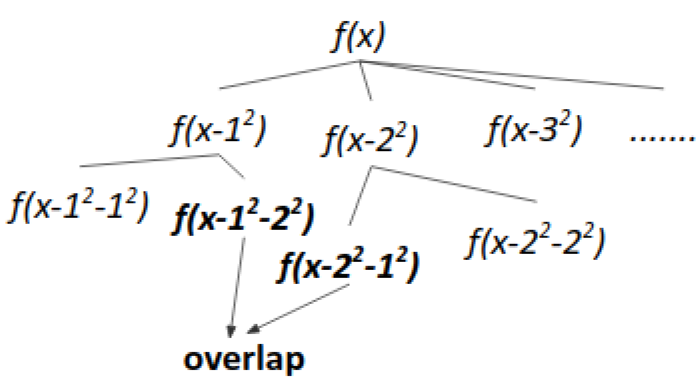
$$f(2) = 2 \quad \{1^2 + 1^2\}$$

$$f(3) = 3 \quad \{1^2 + 1^2 + 1^2\}$$

$$\boldsymbol{f(x) = \min\{f(x-i*i)+1\}} \quad \{ 1 <= i <= \sqrt{x} \}$$

Since it can be represented as recursive function, hence it has optimal substructure property.

Overlapping Subproblem property



Since it follows both overlapping subproblem problem property and optimal substructure property, hence we can apply dynamic programming here.

Algorithm

1. Write the recursive solution.
2. Memoize it by making an extra dp table.

Code (Recursive)

```
int dp[N];

int MinSquare(int n)
{
    if(n == 1 || n == 2 || n == 3 || n == 0)
        return n;

    if(dp[n] != MOD)
        return dp[n];

    for(int i=1; i*i <= n; i++)
    {
        dp[n] = min(dp[n],1 + MinSquare(n-i*i));
    }

    return dp[n];
}
```

```
void solve()
{
    rep(i,0,N)
        dp[i] = MOD;

    int n;
    cin >> n;

    cout << MinSquare(n) << endl;

}
```

Code (iterative)

```
void solve()
{
    int n;

    cin >> n;

    vi dp(n+1, MOD);

    dp[0] = 0;
    dp[1] = 1;
    dp[2] = 2;
    dp[3] = 3;

    for(int i=1; i*i <= n; i++)
    {
        for(int j=0; i*i+j <= n; j++)
        {
            dp[i*i + j] = min(dp[i*i+j], 1+dp[j]);

            // if(i*i )
        }
    }

    cout << dp[n] << endl;
}
```