Algorithmic Problem Solving 2019
Dynamic Programming Exercises
VI Sem 9 April 2019

Given below are two interesting problems from dynamic programming. Trace them and build your intuition on how the logic was built.

Task 01
Coin Change Problem

There are infinite number of coins of x different values. These values are given. Using these coins, you have to make change for Rs. N. In how many ways, you can make this change? Note that there are infinite number of coins of each given value.

Example 1:
N=10
x=4
coin[]={2,5,3,6}

Expected result=5

Possible combinations are -
2+2+2+2+2
5+5
2+3+5
2+2+3+3
2+2+6

Example 2:
N=4
x=3
coin[]={1,2,3}

Expected result=4

Possible combinations are -
1+1+1+1
1+1+2
1+3
2+2

Code:

```
#include<stdio.h>
#include<stdlib.h>

int coinChange(int coin[], int x, int N)
{
   int i,j;
   int dp[N+1][x];
   int including,excluding;

   for(j=0;j<x;j++)
      dp[0][j]=1;

   for(i=1;i<=N;i++)
   {
      for(j=0;j<x;j++)
      {
         if(i>=coin[j])
            including=dp[i-coin[j]][j];
         else
            including=0;

         if(j>=1)
            excluding=dp[i][j-1];
         else
            excluding=0;

         dp[i][j]=including+excluding;
      }
   }
   return dp[N][x-1];
}

int main()
{
   int i;
   int x;
   int N;

   printf("Enter the amount whose change is required\n");
   scanf("%d", &N);

   printf("Enter the number of distinct values of coins\n");
   scanf("%d", &x);
```

```
  int coin[x];

  printf("Enter the values of coins");
  for(i=0;i<x;i++)
    scanf("%d", &coin[i]);

  printf("The number of ways is\n");
  int result = coinChange(coin,x,N);
  printf("%d", result);

  return 0;
}
```

Task 2
Make Palindrome Problem

You are given a string str. Find the minimum number of characters to be inserted to string str to convert it to a palindrome.

Example:
str= ABCDE
result = 4 (ABCDEDCBA)

Code:

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>

int max(int a , int b)
{
  if(a>b)
    return a;
  else
    return b;
}

int longestCommonSubsequece(char str1[], char str2[])
{
  printf("%s %s\n", str1, str2);
  int len1=strlen(str1), len2=strlen(str2);
  int i, j;
```

```
    int LCS[len1+1][len2+1];

    for(i=0;i<=len1;i++)
        LCS[i][0]=0;

    for(j=0;j<=len2;j++)
        LCS[0][j]=0;

    for(i=1;i<=len1;i++)
    {
        for(j=1;j<=len2;j++)
        {
            if(str1[i-1]==str2[j-1])
                LCS[i][j]=1+LCS[i-1][j-1];
            else
                LCS[i][j]=max(LCS[i-1][j],LCS[i][j-1]);
        }
    }
    return LCS[len1][len2];
}

int main()
{
    char str1[20], str2[20];

    printf("Enter the string\n");
    scanf("%s", str1);

    strcpy(str2, str1);
    int lcs = longestCommonSubsequece(str1,strrev(str2));
    int result = strlen(str1) - lcs;
    printf("The required minimal insertions are %d\n", result);
    return 0;
}
```