**,2022, ,26** int count(int n)

{

int a[10][n+1];

for (int r=1; r<=9; r++)

a[r][1]=1;

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

a[9][c]=1;

for (int c=2; c<=n; c++)

for (int r=8; r>0; r--)

a[r][c]=a[r+1][c]+a[r][c-1];

int s=0;

for (int i=1; i<=9; i++)

s=s+a[i][n];

return 2\*s;

}

a[r][c] מכיל את מספר ה"שלמים המסודרים" החיוביים בעלי c ספרות שספרת האחדות שלהם r.

הערה:מכיוון שלא נאמר בשאלה זו במפורש שצריכים לכתוב את הקוד בשפת C, אז מי שבחרה לפתור את השאלה בפסאודו קוד, זה גם פתרון שצריך להתקבל.

## int palLength(chai s[], int n)

## {

## int b[n][n];

## foi (int i=n-1; i>-1; i--)

## foi (int c=i; c<n; c++)

## if (c==i || c-1==i && s[c]==s[i])

## b[i][c]=c-i+1;

## else if (s[i]==s[c])

## b[i][c]=b[i+1][c-1]+2;

## else if (b[i+1][c]>b[i][c-1])

## b[i][c]=b[i+1][c];

## else b[i][c]=b[i][c-1];

## ietuin b[0][n-1];

## }

ב b[i][c] מאוחסן אורך הפלינדרום הארוך ביותר שהוא תת-מחרוזת של s[i..c].

אם במחרוזת תו יחיד או שני תווים שונים זה מזה, אורך הפלינדרום המבוקש 1.

אם במחרוזת שני תווים זהים אז אורך הפלינדרום המבוקש 2.

אם התו הראשון והאחרון זהים אז אורך הפלינדרום המבוקש 2+ אורך הפלינדרום הארוך ביותר שהוא תת-מחרוזת של s[i+1..c-1].

בכל מקרה אחר אורך הפלינדרום המבוקש הוא המקסימום בין: אורך הפלינדרום הארוך ביותר שהוא תת-מחרוזת של s[i+1..c], ואורך הפלינדרום הארוך ביותר שהוא תת-מחרוזת של s[i..c-1].

namespace כמה\_אפשרויות\_לקבלת\_סכום\_בקוביה

{

class Program

{

static int Kubia(int s, int n, int k)

{

int[,] mat = new int[n + 1, s + 1];

int sum = 0;

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

{

if(i<s+1)

mat[1, i] = 1;

}

for (int j = 2; j < s + 1; j++)

{

sum = 0;

if (j <= k)

{

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

{

sum += mat[1, i];

}

}

else

for (int i = k; i >=j-k; i--)

{

sum += mat[1, i];

}

mat[2, j] = sum;

}

for (int i = 3; i < n + 1; i++)

{

for (int j = 2; j <= s; j++)

{

mat[i, j] = mat[i - 1, j - 1] + mat[i, j - 1];

}

}

return mat[n, s];

}

static void Main(string[] args)

{

Console.WriteLine(Kubia(2,2, 6));

}

}

}

namespace מינוס\_ופלוס

{

class Program

{

static int MinusPlus(int []a,int s)

{

int sum = 0;

foreach (var item in a)

{

sum += item;

}

int[,] mat = new int[a.Length, sum \* 2 + 1];

mat[0, a[0] + sum]++;

mat[0, -a[0] + sum]++;

for (int i =1; i <a.Length; i++)

{

for (int j = 0; j < sum\*2+1 ; j++)

{

if(mat[i-1,j]!=0)

{

mat[i, j + a[i]]++;

mat[i, j - a[i]]++;

}

}

}

return mat[a.Length - 1, s + sum];

}

static void Main(string[] args)

{

int[] a = { 5,3, 4, 1 };

Console.WriteLine(MinusPlus(a,5));

}

}

}

**02, 03, 04, 05, 06, 09, 10, 15, 16, 18 , 25**

הנחש הארוך ביותר

namespace Snack

{

class Program

{

static int Snack(int[,] mat, int n)

{

int[,] mm = new int[n + 1, n + 1];

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

for (int j = 1; j < n - 1; j++)

{

mm[i, j] = mat[i - 1, j - 1];

}

int[,] m = new int[n, n];

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

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

{

if(((mm[i,j]==mm[i-1,j]-1)||(mm[i,j]== mm[i - 1, j] + 1))||

(mm[i,j]==mm[i,j-1]-1)||(mm[i,j]== mm[i, j - 1] + 1))

{

if(mm[i, j - 1] > mm[i - 1, j])

{

m[i - 1, j - 1] = mm[i, j - 1] + 1;

}

if (mm[i, j - 1] < mm[i - 1, j])

m[i - 1, j - 1] = mm[i - 1, j] + 1;

}

}

int max = 0;

for(int i=0;i<n-1;i++)

for(int j=0;j<n-1;j++)

{

if (mat[i, j] > max)

max = mat[i, j];

}

return max;

}

static void Main(string[] args)

{

int[,] mat = { { 7,5,2,3,1}, { 3, 4, 1, 4, 4 }, { 1, 5, 6, 7, 8 }, { 3, 4, 5, 8, 9 }, { 3, 2, 2, 7, 6 } };

int s = Snack(mat, 5);

Console.WriteLine(s);

}

}

}

public static int IfK(int []arr,int low,int high,int k)

{

bool flag = false;

int middle = (low + high) / 2;

int f = middle;

while (f % k != 0)

f--;

int l = middle+1;

while (l % k != 0)

l++;

l--;

if (high - low < k)

flag = true;

if (arr[f] == arr[l])

high =middle;

else

{

if (arr[f - k] == arr[f - 1])

low = f;

else

{

high = f - 1;

if (high - middle <=k)

return arr[low];

}

}

return IfK(arr,low,high,k);

}

יעד למספר מסוים ע"פ חבור כל המספרים/חיסור גם ניתן תוך שמוש בכל איברי המערך.

static int MinusPlus(int[] a, int s)

{

int sum = 0;

foreach (var item in a)

{

sum += item;

}

int[,] mat = new int[a.Length, sum \* 2 + 1];

mat[0, a[0] + sum]++;

mat[0, -a[0] + sum]++;

for (int i = 1; i < a.Length; i++)

{

for (int j = 0; j < sum \* 2 + 1; j++)

{

if (mat[i - 1, j] != 0)

{

mat[i, j + a[i]]++;

mat[i, j - a[i]]++;

}

}

}

return mat[a.Length - 1, s + sum];

}

static void Main(string[] args)

{

int[] a = { 5, 3, 4, 1 };

Console.WriteLine(MinusPlus(a,1));

}

static int MaxPath(int[,] mat)

{

int[,] m = new int[mat.GetLength(0) + 1, mat.GetLength(1) + 1];

for (int i = 1; i < (mat.GetLength(0) + 1); i++)

{

for(int j=1;j<(mat.GetLength(1)+1);j++)

{

m[i,j] = mat[i - 1, j - 1];

}

}

int [,]d=new int[mat.GetLength(0) + 1, mat.GetLength(1) + 1];

for (int i = 0; i < (mat.GetLength(0)); i++)

{

for (int j= 0; j < (mat.GetLength(1) ); j++)

{

if ((m[i+1, j+1] + 1 == m[i - 1+1, j+1]) || (m[i+1, j+1] - 1 == m[i - 1+1, j+1]))

{

if (i == 0)

d[i, j] = 1;

else

d[i, j] = d[i - 1, j] + 1;

}

else

{

if ((m[i+1, j+1] + 1 == m[i+1, j - 1+1]) || (m[i+1, j+1] - 1 == m[i+1, j - 1+1]))

{

if (j == 0)

d[i, j] = 1;

else

d[i, j] = d[i , j- 1] + 1;

}

else

d[i, j] = 1;

}

}

}

int max = 0;

for (int i = 0; i < (d.GetLength(0)); i++)

{

for (int j = 0; j < (d.GetLength(1)); j++)

{

if (d[i, j] > max)

max = d[i, j];

}

}

return max;

}

ככ



