حل به روش برنامه نویسی پویا :

اگرفرض کنیم تعداد ای ردیف های 3 الماسی را h است آنوقت یک ماتریس 3*h درنظر می گیریم.آن را به اینگونه پر می کنیم که سط اول آن را مقدار هر کدام از سه الماس سطر اول قرار می دهیم و بعد هر خانه از ماتریس را برابر مقدار الماس همان خانه به علاوه با مقدار max ،2الماسی که سطر قبل قرار دارند ولی در یک دیف نیستند قرارمیدهیم.

اسم هر کدام از الماسها:

A_{h1} A_{h2} A_{h3}

.

.

.

A_{31} A_{32} A_{33}

A_{21} A_{22} A_{32}

A_{11} A_{12} A_{13}

$A_{h1}+max(A_{(h-1)2},A_{(h-1)3})$	$A_{h2}+max(A_{(h-1)1},A_{(h-1)3})$	$A_{h3}+max(A_{(h-1)2},A_{(h-1)1})$
. . .	. . .	. . .
$A_{(i+1)1}+max(A_{(i+1)2},A_{(i+1)3})$	$A_{(i+1)2}+max(A_{(i+1)1},A_{(i+1)3})$	$A_{(i+1)3}+max(A_{(i+1)2},A_{(i+1)1})$
. . .	. . .	. . .
$A_{21}+max(A_{12},A_{13})$	$A_{22}+max(A_{11},A_{13})$	$A_{23}+max(A_{12},A_{11})$
A_{11}	A_{12}	A_{13}

بعد ار پر کردن همه خانه های این آرایه جواب ما بیشترین مقدار آخرین سطر این آرایه میشود.

برای آنکه مسیرمان هم داشته باشیم می توانییم یک آرایه دیگر درست کنیم به اندازه ماتریس قبلی و اینبار هر کدام از دو مقداری که در max به دست می آوریم را قرار دهیم با اینکار ما می توانیم در

آخر که مقدار max کل مسیر را به دست آوردیم با به عقب رفتن مسیر max را هم داشته باشیم.

شبه کد:

```
Give higth and named that h
Give monny of any place and put them on A[3][n]
Make array with size 3 * h and named that map
Make array with size 3 * (h-1) and named that way
Max_monny()
        {
        Map[1][1]=A[1][1]
        Map[1][2]=A[1][2]
```

```
Map[1][3]=A[1][3]
For(i=2 to h)
        {
        If(map[i-1][2] > map[i-1][3])
                {
                Way[i][1]=2
                Map[i][1]=A[i][1]+ map[i-1][2]
                }
        Else
                {
                Way[i][1]=3
                Map[i][1]=A[i][1]+ map[i-1][3]
                }
        If(map[i-1][1] > map[i-1][3])
                {
                Way[i][1]=1
                Map[i][1]=A[i][2]+ map[i-1][1]
                }
        Else
                {
                Way[i][1]=3
                Map[i][1]=A[i][2]+ map[i-1][3]
                }
        If(map[i-1][2] > map[i-1][1])
                {
                Way[i][1]=2
                Map[i][1]=A[i][3]+ map[i-1][2]
                }
        Else
```

```
                {
                Way[i][1]=1
                Map[i][1]=A[i][3]+ map[i-1][1]
                }
        }
Return(max(map[h][1],map[h][2],map[h][3]))
}
```

محاسبه پیچیدگی:

این االگوریتم با یک حلقه h تایی نوشته می شود پس آن از Θ(h) است که h ارتفاع کوه است.

ب) الگوریتم غیر پویای دیگر

این مساله به روشش تقسیم و حل نیز حل می شود به این صورت که در هر مرحله ما max راهی که به هرکدام از این الماسهای مرحله قبل را داشته باشیم واز آن سطر خودمان را به دست بیاوریم.

```
My_Max(int row,int a)
{
If(row==1)
        Return A[1][a]
Else
        {
        If(a==1)
                Return(A[row][a]+max(my_max(row-1,2), my_max(row-1,3)))
        If(a==2)
                Return(A[row][a]+max(my_max(row-1,1), my_max(row-1,3)))
        If(a==3)
                Return(A[row][a]+max(my_max(row-1,2), my_max(row-1,1)))
```

```
}

Final_max=max(my_max(h,1), my_max(h,2), my_max(h,3))
```

ج)تحلیل

در روش دوم ما max را برای بعضی خانه ها چنین بار حساب کردیم ولی در روش برنامه نویسی پویا برای هر کدام از این خانه ها تنها یک با حساب کردیم که این مساله موجب بهتر بودن حل به وسیله برنامه نویسی پویا می شود.

با تشکر از زحمات شما

محمد صادق بورونی