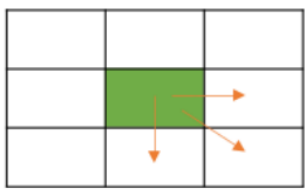
امیرعلی فرازمند

99522329

سوال و الگوریتم کامل تر(با فرق جزئی)

ویدیو توضیح الگوریتم(با فرق جزئی)

لاک پشتی می خواهد از خانه ی آبی به خانه ی قرمز برود او در هر حرکت می تواند به خانه ی راست یا خانه ی پایین یا خانه ی راست و پایین برود (او از خانه ی (i,j) می تواند به خانه ی (i,j+1) یا (i+1,j) یا (i+1,j+1) برود). هزینه ی رفتن به هر خانه در آن نوشته شده است با استفاده از DP به او کمک کنید کم هزینه ترین مسیر را پیدا کند . پیچیدگی زمانی و حافظه الگوریتم خود را بدست آورده و نحوه محاسبه آنها را توضیح دهید.



مسیر حرکت لاکپشت

1	3	3	5	4
5	7	4	4	3
2	3	2	6	2
8	5	6	4	1
3	1	2	5	3

هر خانه را میتوان بدین صورت بطور بازگشتی حساب کرد:

Cost(I , j)= Value(i, j)+ min{cost(i-1, j), cost(i, j-1), cost(i-1, j-1)}

مشکلی که با تابع بازگشتی بالا پیش میاید این است که ما چندبار cost مربوط به بعضی خانه هارا حساب میکنیم.مثلادر مثال زیر خانه ی (1,1) در محاسبه ی کاست 3 خانه ی (1,2), (2,2), (2,1) دخیل است و ما 3بار یک چیز را حساب میکنیم و این خوب نیست:

0,0	0,1	0,2
1,0	1,1	1,2
2,0	2,1	2,2

- راه حلی که میتوان داد یکی این است که تابع بازگشتی را نگه داریم و با متد memoizatation کاست ها را در آرایه 2بعدی ای نگه داریم و هر دفعه چک کنیم اگر برای خانه ای کاست حساب شده دوباره آنرا حساب نکنیم.

- راه حل دیگری که وجود دارد این است که کاست ردیف بالا و ستون سمت چپ را جدا حساب کنیم اول. چون که در ردیف اول هر خانه تنها به کاست خانه سمت چپش وابسطه است، در ستون اول هم تنها به بالایی اش وابسطه است.

1	3	3	5	4
5	7	4	4	3
2	3	2	6	2
8	5	6	4	1
3	1	2	5	3

سپس در دوحلقه ی تو در تو برای طول و عرض کاست خانه های باقی مانده را حساب کنیم و در آرایه 2بعدی ای ذخیره کنیم. که به ترتیب اینگونه کاست خانه ها حساب میشوند و به مشکلی بر نمیخوریم چون در هر خانه کاست 3 خانه مربوط بهش را داریم (tabulation).

1	3	3	5	4
5	7 1	4 5	4 9	3 13
2	3 2	2 6	6 10	2 14
8	5 3	6 7	4 11	1 15
3	1 4	2 8	5 12	3 16

پیچیدگی ها در روش دوم(جدول را M*N در نظر بگیریم):

:Time complexity

کل جدول را یکبار بررسی میکنیم »»»»O(M*N)

:Space complexity

2 جدولی M*N باید داشته باشیم »»»» O(M*N)