Algorithm Design

Documentation- project2

Hananeh Esfandiar - Mohammad Afshari

ایده کلی الگوریتم :

میخواهیم مجموع بیشترین تعداد نارنگی هایی که حمید میتواند در مسیر ببیند را محاسبه کنیم. بدین منظور از ستونی دلخواه از سطر ۰ شروع کرده و به ستونی در سطر n-۱ میرسیم که آن ستون همان جواب مسئله خواهدبود.

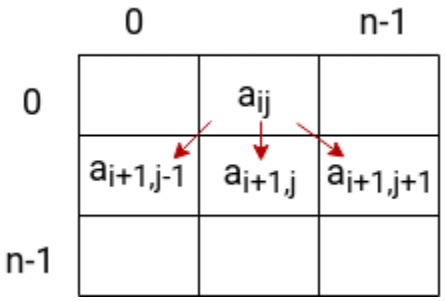
ماتریس matrix شامل اعداد صحیح مثبت a_{ij} با ابعاد n*n داده شده است . و درآن هر عنصر a_{ij} نشان دهنده تعداد نارنگی ها است که

۱۰۰۰۰۰ => a_{ij} >= ۰

n-۱ => i, j => ۰

۱۰۰ => n => ۲

حمید ۳ حرکت مجاز برای رفتن به سطر بعدی از باغ دارد.
یا میتواند به خانه‌ی (i+۱, j+۱) برود یا به خانه‌ی (i+۱,j-۱) و یا به خانه‌ی (i+۱, j) که در این حالت محدودیت دارد و فقط k بار میتواند از این حرکت استفاده کند!



الگوریتم به این صورت است که ماتریس result_matrix را به عنوان ماتریس نتیجه در نظر میگیریم و آن را با مقدار ۰ مقدار دهی اولیه میکنیم، هر عنصر a_{ij} در ماتریس result_matrix نشان دهنده مجموع تعداد نارنگی

های خانه فعلی (i, j) با بیشترین مقدار نارنگی در بین ۳ عنصر ممکن سطر قبلی است. در واقع عنصری که ذخیره میشود نشان دهنده این است که اگر از کدام یک از خانه های ممکن بالا یعنی (i-۱, j+1) یا (i-۱, j) و یا (i-۱, j-۱) به خانه (i, j) بیایم بیشترین تعداد نارنگی را دیده ایم.

همانطور که در بالا نیز اشاره کردیم حمید در انجام حرکت (i+۱, j) محدودیت دارد و فقط k بار میتواند حواسش را جمع کند و از آن استفاده کند پس حمید تنها زمانی میتواند از آن استفاده کند که k بزرگ تر از ۰ باشد و درغیر این صورت مجاز به استفاده از این حرکت نبوده و در نتیجه در این حالت عنصری که در خانه (i, j) ذخیره میشود مجموع تعداد نارنگی های خانه فعلی با بیشترین مقدار نارنگی در بین ۲ عنصر ممکن سطر قبلی ست.یعنی نشان دهنده این است که اگر از کدام یک از خانه های ممکن بالا یعنی (i+۱, j-۱) و یا (i-۱, j-۱) به خانه (i, j) بیایم بیشترین تعداد نارنگی را دیده ایم.

حال سوالی که مطرح میشود این است که چگونه تشخیص دهیم حمید حواسش را جمع کرده و در سطری حرکت (i+۱, j) را انجام داده ؟
برای این کار لازم است تا چند شرط را بررسی کنیم.
۱. آیا از بین عناصر ممکن خانه های سطر i-۱ که انتخاب میشوند تا با عنصر خانه (i, j) جمع جمع شوند عنصر (i-۱, j) انتخاب میشود ؟
اگر اینطور بود شرط بعدی را بررسی میکنیم
۲. آیا عنصر خانه (i-۱, j) با یکی از عناصر خانه های (i-۱, j+۱) و یا (i-۱, j-۱) برابر است یا خیر؟

اگر برابر نباشد در اینصورت این احتمال وجود دارد که حمید به صورت ستونی حرکت کرده باشد پس این عنصر را در لیست possible_k نگه میداریم تا بعدا چک کنیم ببینیم آیا واقعا حمید به صورت ستونی حرکت کرده ؟

اگر برابر باشد حمید حرکتی ستونی خود را هدر نمیدهد و خانه ای را انتخاب میکند که بتواند به صورت مورب حرکت کند

با توجه به اینکه هدف این است که حمید بیشترین تعداد نارنگی ها را در مسیر ببیند پس در انتهای هر سطر بیشترین عنصر آن سطر را پیدا میکنیم و چک میکنیم که آیا عدد انتخاب شده در لیست possible_k هست یا خیر اگر بود پس حمید به صورت ستونی حرکت کرده و یکی از مقدار k کم میشود.

نکته: تعداد ستون های ماتریس نتیجه (result_matrix) دو ستون بیشتر از ماتریس ورودی است. بدلیل اینکه هنگام مقایسه‌ی ۳ انتخاب برای مسیر، ممکن است مسیر مورب چپ یا راست موجود نباشد که باعث ارور میشود.

برای مثال ماتریس زیر را در نظر بگیرید

	0	1	2
0	0	10	0
1	0	10	0
2	10	11	0

با توجه به اینکه قبل از سطر اول ماتریس matrix مسیر دیگری وجود ندارد سطر اول matrix را عینا در result_matrix کپی میکنیم و خواهیم داشت:

0	0	10	0	0
0	0	0	0	0
0	0	0	0	0

حال گام به گام عناصر سطر دوم را تولید میکنیم بدین منظور بررسی میکنیم که اگر از کدام یک از مسیر ها قبلی در سطر ۰ به خانه مورد نظر برویم بیشینه خواهد بود.
فرض کنید میخواهیم خانه ۱ و ۱ را پر کنیم بدین منظور ۳ انتخاب داریم یا میتوانیم از خانه (۲, ۰) به آن برویم یا (۱, ۰) و یا (۰, ۰) همانطور که میبنیم بین عناصر این خانه ها عنصر (۲, ۰) یعنی ۱۰ از بقیه بزرگ تر است پس اگر حمید از آن به خانه (۱, ۱) بیاید بیشترین تعداد نارنگی را خواهد دید یعنی ۰ + ۱۰ در نتیجه داریم:

0	0	10	0	0
0	10	0	0	0
0	0	0	0	0

حال فرض کنید که میخواهیم عنصر ستون بعدی را محاسبه کنیم. حمید از خودش میپرسد اگر از کدام یک از سه خانه ممکن سطر ۰ به خانه (۲, ۱) بروم بیشترین تعداد نارنگی را میبینم؟

یا میتواند از (۳, ۰) به (۲, ۱) برود.

یا میتواند از (۲, ۰) به (۲, ۱) برود.

و یا میتواند از (1, 0) به (۲, ۱) برود.

و با توجه به اینکه اگر از (۲, 0) به (۲, ۱) برود میتواند ۲۰ پر تقال ببیند پس آن را انتخاب میکند.

و همینطور برای ستون بعدی این سطر از بین عناصر خانه های (۲, 0) و (۳, ۰) و (۴, ۰) بازهم (۲, ۰) را انتخاب میکند و در نتیجه داریم:

0	0	10	0	0
0	10	20	10	0
0	0	0	0	0

برای سطر ۲ (دقت کنید که شماره سطر ها از ۰ شروع شده است)نیز به ازای هر خانه حمید سوال مشابه بالا را از خودش میپرسد اگر از کدام یک از سه خانه ممکن سطر ۱ به خانه های (j و۲) بروم که 1 <= j و 3 => j بیشترین تعداد نارنگی را میبینم؟ که مانند دو سطر قبلی عمل میکند و در نتیجه:

0	0	10	0	0
0	10	20	10	0
0	30	21	20	0

بررسی کد :

تابع get_input ورودی های برنامه را از کاربر می‌گیرد. (خط ۳-۱۴)

تابع find_maximum_path تابع اصلی است. (خط ۲۱-۲۶) ماتریس نتیجه را تعریف می‌کند و آن را با صفر، مقداردهی اولیه می‌کنیم.

(خط ۳۰-۳۱) سطر اولیه ماتریس ورودی را در ماتریس نتیجه ذخیره می‌کنیم چون قبل از آن مسیر دیگری وجود ندارد که بخواهیم بیشترین مقدار نارنگی هارا پیدا کنیم و با مقدار نارنگی های سطر بعد جمع کنیم.

در خط ۳۵، آرایه‌ی possible_k را تعریف می‌کنم که در آن مقدار مجموع نارنگی های سطر فعلی با بیشترین مقدار(بهترین انتخاب بین ۳ مسیر) نارنگی های سطر قبل را ذخیره می‌کنیم که مسیر مستقیم (i+1,j) را انتخاب کردیم.(یعنی از بین سه مسیر مستقیم و دو مسیر مورب، مسیر مستقیم بیشترین تعداد را به ما می‌داد.)
این آرایه به ما کمک می‌کند تا در نهایت بعد از بررسی تمام درایه‌ها در یک سطر، اگر بهترین مسیر، مسیر مستقیم بوده و مسیر مستقیم را انتخاب کردیم، یکی از تعداد k کم کنیم.
(خط ۳۴-۵۲) از سطر دوم ماتریس نتیجه شروع می‌کنیم و برای هر درایه، سه مسیر مستقیم و مورب چپ و مورب راست را بررسی می‌کنیم و می‌بینیم که کدام بیشترین مقدار را دارد و درواقع بهترین مسیر برای رسیدن به درایه فعلی از درایه سطر قبل در همین ستون است.
شرط k>0 در حلقه for دوم، بررسی می‌کند که اگر تعداد k بیشتر از صفر بود یعنی مجاز به انتخاب مسیر مستقیم به پایین(یک سطر به پایین)، بین سه مسیر، بیشترین را انتخاب کنیم و در غیر این صورت از بین دو مسیر مورب بیشترین را انتخاب می‌کنیم.
شرط if در خط ۴۳، بررسی می‌کند که اگر تعداد نارنگی ها در مسیر مستقیم، با تعداد نارنگی با دو مسیر مورب(یا یکی از مسیر های مورب) برابر باشد، میسر مورب را انتخاب می‌کنیم زیرا برای انتخاب مسیر مستقیم محدودیت داریم.
در متغیر max_of_row، بیشترین تعداد نارنگی ها را در هر سطر(در هر مرحله از اجرای حلقه for داخلی) در آن ذخیره می‌کنیم. (خط ۵۳)
(در خط ۵۴-۵۵) بررسی می‌کنیم که آیا بیشترین تعداد نارنگی ها در سطر فعلی، از انتخاب مسیر مستقیم بدست آمده یا خیر!اگر از مسیر مستقیم بدست آمده بود، باید از تعداد k ها یکی کم کنیم.
و در نهایت در خط ۵۹-۶۰ جاب نهایی که در سطر اخر است را، که برابر با بیشترین مقدار در سطر آخر است پیدا می‌کنیم.

پیچیدگی زمانی :

در کد، دو حلقه‌ی فور در هم داریم(nested loop) و هرکدام کل سطر را طی می‌کنند و در نتیجه پیچیدگی زمانی کد برابر($O(n^2)$) است.

محاسبه پیچیدگی زمانی:

فرض می‌کنیم(T(n تعداد دفعات اجرای دستور زیر (در خط ۴۸) باشد:

```
possible_k.append(result_matrix[i][j])
```

پیچیدگی زمانی خط ۳۴-۵۲:

$$T(n) = \sum_{i=1}^{n-1} \sum_{j=1}^{n} 1 = \sum_{i=1}^{n-1} n = (n-1)n = n^2 - n$$

حلقه های for در خط های ۱۰، ۲۲ ، ۳۰ و ۵۹، n بار اجرا میشوند در نتیجه پیچیدگی زمانی هر کدام O(n) است.
درنهایت داریم :

$$T(n) = 4n + n^2 - n = n^2 + 3n$$

$$T(n) \in O(n^2)$$