باسمه تعالی

طراحی الگوریتم ها

دانشکده مهندسی برق و کامپیوتر

فروردین ۱۴۰۳

استاد:

دکتر فلسفین

سپهر عبادی

۹۹۳۳۲۴۳

سوال ۱ :

واضح است در حالتی که k = ۰ باشد یعنی گاو نتواند علف خود را در حین خوردن تغییر دهد حداکثر می تواند m کیلو علف بخورد.پس حالتهای خوردن را براساس وزن و k :

۰	۱	۱	...	۱	۰	...	۰
	۱	۲	...	m	M+۱	...	n

برای محاسبه تعداد حالتهایی که میتواند با یک بار تغییر بیشتر مقداری علف را بخورد نیازمند یک رابطه بازگشتی هستیم.

اگر گاو بخواهد j کیلو علف را باk بار تغییر بخورد میتوانیم بررسی کنیم که ۱ – j کیلو را با ۱ – k تغییر به چند حالت میتوانسته بخورد پس الان قطعا میتواند با یکبار تغییر نوع علف j کیلو علف را نیز بخورد. به همین صورت گاو اگر میتوانسته m - j کیلو علف را با ۱ – k بار تغییر بخورد پس الان هم میتواند j کیلو علف را باk بار تغییر بخورد.

پس رابطه برابر میشود با :

DP[i][j] = sum (DP[i-۱][j-p] for p in {۱,۲,...,m})

اکنون جدول این مسئله را میکشیم و درایه هدف ما همان DP[K][N] خواهد بود.

سوال ۲:

ابتدا یک جدول پویا به ابعاد n*n تعریف می کنیم که در آن هر خانه، جمع عناصر مستطیلی که از ابتدای جدول تا آن خانه می‌رسد را نگه دارد. سپس برای هر پرسش، مقدار جواب را با استفاده از این جدول پویا محاسبه می کنیم.

در ابتدا، جدول را به صورت زیر مقداردهی اولیه می کنیم:

dp[i][j] =

جمع عناصر مستطیل از (۱و۱) تا(i , j)

سپس برای محاسبه جواب هر پرسش با مختصات i، j، k، l، جواب به صورت زیر میشود:

```
dp[j][l] - dp[i-۱][l] - dp[j][k-۱] + dp[i-۱][k-۱]
```

سوال ۳:

ابتدا باید این فرض را در نظر بگیریم که هر کاراکتر خود یک زیر رشته متقارن به طول ۱ است.

و همینطور هر دو کاراکتر متوالی در صورت برابری یک زیر رشته متقارن به طول ۲ است.

هر بار از رشته اصلی زیر رشته ای که از **i** تا **j** است را انتخاب میکنیم. اگر کاراکتر **i**ام با کاراکتر **j**ام برابر بود بزرگترین زیر رشته متقارن از **i** تا **j** برابر است با :

```
S[i][j] = S[i+۱][j-۱]
```

در غیر اینصورت :

```
S[i][j] = max(S[i][j-۱],S[i+۱][j])
```

سوال ٤:

برای پاسخ به این سوال از مسئله longest increasing subsequence کمک میگیریم.

longest increasing subsequence :

به اولین درایه از ارایه مورد نظر عدد یک را نظیر میکنیم.و از این به بعد در صورتی که یک درایه از هرکدام از درایه های قبلش بزرگتر بود عدد نظیر شده به ان برابر عدد درایه مورد نظر به علاوه یک خواهد بود.و در غیر اینصورت به این درایه هم عدد یک میدهیم.

بزرگ ترین مجموع بلند ترین زیردنباله نزولی با شروع از یکی از عناصر دنباله را میخواهیم پس کافیست از راه حل مسئله longest increasing subsequence برای هر عنصر از ارایه خود استفاده کنیم.

سوال ۵:

ابتدا یک جدول دوبعدی dp به ابعاد n × n تعریف می‌کنیم که در آن dp[i][j] نشان‌دهنده طول کمینه وترهای موجود در چند ضلعی به ترتیب i و j است.

سپس مقادیر dp[i][j] را به ترتیب از i=۳ تا n و j=۰ تا n-۱ به‌روزرسانی می‌کنیم. برای به‌روزرسانی هر مقدار، تمام مقادیر ممکن برای تقسیم ضلعی n-ضلعی به مثلث‌ها را بررسی می‌کنیم و کمینه را انتخاب می‌کنیم.

در نهایت، مقدار dp[n][۰]مقداری خواهد بود که ما به دنبال آن هستیم، یعنی طول کمینه وترهای موجود در n-ضلعی.

سوال ٦:

ابتدا یک آرایه به نام max_suffix_sum به طول n تعریف می‌کنیم که max_suffix_sum[i] نشان‌دهنده بیشترین مقدار از A[i] تا A[n-۱] است.

سپس یک حلقه را از انتهای آرایه به سمت ابتدا اجرا می‌کنیم و برای هر عضو i، مقدار max_suffix_sum[i] را به روزرسانی می‌کنیم. به عبارت دیگر، max_suffix_sum[i] برابر با بیشترین مقداری است که می‌توانیم با شروع از A[i] و ادامه تا آخرین عنصر آرایه بدست آوریم.

در انتها، با استفاده از max_suffix_sum، می‌توانیم با دوبار پیمایش آرایه A، اندیس‌های p، q، r و s را پیدا کنیم که مقدار A[s]-A[r]+A[q]-A[p] بیشینه باشد و شرایط مسئله را همچنین برآورده کند.