ساختمان داده‌ها و الگوریتم‌ها

نیم‌سال دوم ۹۷ــ۹۸

دانشکده‌ی مهندسی کامپیوتر
گردآورندگان: صادق مهدوی ، پدرام خورسندی

پاسخنامه تمرین عملی پنجم

مسئله‌ی ۱. خوب، بد، زشت

در هر زیر رشته باید در $O(1)$ بفهمیم که این زیر رشته زیباست یا نه. برای این کار ابتدا یک پیش پرداز از $O(n)$ باید روی تمامی زیر رشته‌ها انجام بدیهم و با باستفاده از partial sum تعداد حروف بد را از ابتدا تا مکان x حساب کنیم. سپس برای هر رشته کافی است عدد انتهای رشته را از عدد ابتدای رشته کم کنیم و جواب را به‌دست آوریم حال که می‌توانیم برای هر زیر رشته چک کنیم که زیباست یا نه. در انتها کافی است روی تمام زیر رشته‌ها با $O(n^2)$ حرکت کنیم و اگر خوب بود هش آن را به مجموعه زیر رشته‌های خوب اضافه کنیم (چون خود آن می‌تواند بسیار بزرگ باشد و در حافظه‌ی ما جای نگیرد) همچنین هش هر زیر رشته را می‌توان در $O(1)$ حساب کرد. کافی است هر کاراکتر که اضافه می‌شود، عدد مربوط به آن را اضافه کنیم و هر کاراکتر که کم می‌شود عدد مربوط به آن را از مقدار هش فعلی کم کنیم و در مواقعی که کاراکتر از سمت چپ کم می‌شود علاوه بر کم کردن هش آن باید مقدار کل هش را تقسیم بر یک عدد نیز بکنیم.

مسئله‌ی ۲. چراغ‌های شهر

برای پیدا کردن xــامین چراغ، ابتدا باید خیابانی که چراغ داخل آن قرار دارد را پیدا کنیم. برای این کار با $O(N)$ باید داخل یک آرایه برای هر خیابان تعداد چراغ‌های کم ارتفاع‌تر از چراغ‌هایی که داخل آن خیابان وجود دارد را به‌دست بیاوریم. بدیهی است که اگر این مقدار را داشته باشیم، تعداد چراغ‌های با طول بزرگتر از چراغ‌های این خیابان را هم خواهیم داشت. بنابر این هر شهر را که انتخاب کنیم، می‌توانیم بفهمیم که xــامین چراغ داخل آن خیابان قرار دارد، داخل خیابان‌های با چراغ‌های کوتاه‌تر است، یا داخل خیابان‌های با چراغ‌های بلند تر. بنابر این می‌توان با باینری سرچ این خیابان را پیدا کرد و در انتها عدد چراغ را به‌دست آورد. اگر تعداد چراغ‌ها در شهرهای کوتاه‌تر را a بنامیم. عدد چراغ برابر $x-a+l-1$ خواهد بود. که l نقطه‌ی شروع بازه‌ی خیابان مورد نظر است.

مسئله‌ی ۳. بازم مقسوم‌علیه مشترک!!

برای حل این مساله برای هر زیر رشته از ابتدا تا انتهای رشته‌ی اول،چک می‌کنیم که آیا اگر این زیر رشته را به تعداد $|s1|/|x|$ بار برای رشته‌ی اول و $|s2|/|x|$ بار برای رشته‌ی دوم تکرار کنیم، رشته‌های $s1$ و $s2$ ساخته خواهند شد یا خیر. برای این کار از هش چند جمله‌ی می‌توانیم استفاده کنیم. (باید به صورت داینامیک موقع زدن for روی پیش‌وندهای رشته‌ی $s1$ پس از اضافه شدن

هر کاراکتر، هش آن‌را با استفاده از هش زیررشته‌ی قبلی به‌دست بیاوریم)
حال که می‌توانیم هش هر زیر رشته را در $O(1)$ به‌دست بیاوریم، باید تشخیص دهیم که اگر این زیر رشته را چند بار تکرار کنیم، رشته‌ی اصلی به‌دست می‌آید یا خیر. برای این کار هم باید به این نکته دقت کنید که اگر رشته‌ی x را m بار تکرار کنیم. هش چند جمله‌ای آن به صورت زیر خواهد شد (که پس از آن می‌توان هش به‌دست آمده را با هش دو رشته‌ی $s1$ و $s2$ مقایسه کرد):

$$hash(s) = 1 \times hash(x) + p^{|x|} \times hash(x) + p^{2\times|x|} \times hash(x) + ... + p^{m\times|x|} \times hash(x)$$

که اگر از اتحاد چاق و لاغر استفاده کنیم:

$$hash(s) = hash(x) \times \frac{p^{(m+1)\times|x|}}{p^{|x|} - 1}$$

مسئله‌ی ۴. تقارن‌یابی

ابتدا ارایه را از ابتدا به انتها به شکل زیر hash میکنیم.

$hash1 = \Sigma p^i * a_i mod 10^7$

این عمل در پیچیدگی زمانی خطی قابل اجراست. در مرحله بعد مشابه روش بالا ارایه را از انتها به ابتدا hash میکنیم.

$hash2 = \Sigma p^(n - i) * a_i$

با به کارگیری هش بالا در تحلیل زمانی ثابت با بررسی تساوی زیر تقارن رشته را در زمان ثابت بررسی میکنیم. به این طریق الگوریتمی در زمان ثابت برای بررسی تقارن یک رشته خواهیم داشت.

$hash1[l] * p^(r - l) - hash1[r] == hash2[r] * p^(r - l) - hash2[l]$

در نظر بگیرید مرکز رشته متقارن در خانه امi رشته مورد بررسی باشد. در این صورت اگر برای $i < J$ رشته $array[2 * i - J : J]$ متقارن باشد برای $j < J$ نیز متقارن خواهد بود.

ال با به کارگیری الگوریتم search binary طول بزرگترین رشته متقارن با فرض محوریت خانه امi در $O(logn)$ یافت میشود.

الگوریتم بالا را برای تمام خانه های آرایه اجرا کرده و طول بزرگتریت رشته متقارن را به این طریق پیدا میکنیم.

```
str = input()
str = str[:-1]
str2 = ''.join(''.format(str))
n = len(str2)
P = [0] * n
C = R = 0
for i in range (1, n-1):
P[i] = (R > i) and min(R - i, P[C - (C - i)])
while str2[i + 1 + P[i]] == str2[i - 1 - P[i]]:
P[i] += 1
if i + P[i] > R:
C, R = i, i + P[i]
```

```
maxLen = max(P)
print (maxLen)
```