سوال ۳۷ – شکل یکسان

صورت سوال

دو آرایه‌ی عددی A و B داده شده‌اند. می‌خواهیم همه‌ی جاهایی را پیدا کنیم که B با همان شکل کلی داخل A آمده است؛ یعنی اگر همه‌ی اعضای B به اندازه‌ی یک مقدار ثابت بالا یا پایین بروند، هنوز همان حالت حساب شود.

برای مثال، اگر $B = [3, 5, 8]$ باشد، دنباله‌ی $[10, 12, 15]$ هم همان شکل حساب می‌شود.

الگوریتمی با زمان $O(n + m)$ ارائه دهید.

با توجه به ایده های اصلی که در حل مسائل String Matching داشتیم و در الگوریتم های آن دیدیم، چالشی که در این مسئله داریم ثابت نبودن آرایه B است ← میتواند با مقدار ثابت C و جمع تفریق تغییر کند.

برای حل این مشکل باید سراغ یک ویژگی نامتغیر برویم ← در این شیفت دادن ها اختلاف بین عناصر متوالی ثابت می‌ماند.

اگر زیر آرایه‌ای از A با شروع از اندیس (i) با آرایه B هم شکل باشد بدآ هر عنصر داریم : $A[i+K] = B[K] + C$

پس تفاضل دو عنصر متوالی میشود ←

$$A[i+k+1] - A[i+K] = (B[K+1]+C) - (B[K]+C) = B[K+1] - B[K]$$

با اینکار ثابت C از میان برداشته می‌شود => با ساختن آرایه های تفاضلی از روی آرایه اولیه مسئله به همان فرمت مسائل String Matching تبدیل می‌شود. پس الگوریتم مان از این قرار است :

- اگر طول B یعنی m مساوی 1 باشد هر تک عضو از آرایه A با آن تک عضو B تطابق داشته و هم شکل است چراکه با جمع یا تفریق مقداری ثابت به هم تبدیل می‌شوند => در این حالت هم اندیس ها را خروجی میدهیم ← $O(n)$
- در حالتی که $m > n$ است هم بدیهتاً هیچ هم شکلی نخواهیم داشت.

درغیر این موارد ابتدا باید آرایه های تفاضلی جدید ΔB و ΔA را بسازیم ←

- ΔB با طول $m-1$:

$$\Delta B[K] = B[K+1] - B[K]$$

- ΔA با طول $n-1$:

$$\Delta A[K] = A[K+1] - A[K]$$

حال مسئله می‌شود مسئله String Matching ← جستجوی الگوی ΔB در ΔA: برای حل مسئله String Matching با زمان اجرای $O(n+m)$ از الگوریتم KMP استفاده می‌کنیم ← چون الفبای ما اینجا کل اعداد صحیح است ساخت DFA و محاسبات لازم برای جدول آن از اردر زمانی و حافظه‌ای $O(R \times M)$ خواهد بود ← R همان اندازه الفبا که چون به اندازه اعداد صحیح است بی‌نهایت خواهد بود. پس سراغ نسخه بهبود یافته KMP به کمک آرایه lps می‌رویم که زمان پیش‌پردازش و ساخت آرایه lps در آن $O(M)$ است پس در نهایت ساخت ΔB در $O(m)$ ساخت ΔA در $O(n)$ انجام میشود و اجرای KMP با پیش‌پردازش مورد نیاز در $O(m+n)$ انجام میشود ← زمان کل اجرای الگوریتم $O(m+n)$ خواهد بود. فضای ذخیره‌سازی مورد نیاز نیز برای KMP آرایه‌ای به طول m است + با در نظر گرفتن نگهداشتن ΔB هم مجموع فضای مورد نیاز $O(m)$ است.

longest proper prefix which is also suffix

از ابتدا تا اندیس فعلی طولانی‌ترین پیشوندی که عیناً در پسوند هم تکرار شده چه طولی دارد.

LPS پیش‌پردازش و آماده‌سازی آرایه pseudocode :

```
lps[0] = 0
len = 0
i = 1
while i<m:
    if pattern[i] == pattern[len]:
        len++
        lps[i] = len
        i++
    else:
        if len != 0
            len = lps[len-1]
        else:
          lps[i] = 0
          i++
```

وقتی pattern[len] و pattern[i] برابر نشد سراغ دومین پیشوند پسوند بزرگ مطابق هم در بخش match شده قبلی می‌رویم ←

len = lps[len-1] پیشوند کوچکتری هست که پسوند هم باشد

Pseudocode جستجو در متن با الگوریتم KMP :

```
i = 0
j = 0

while i<n:
    if text[i] == pattern[j]:
        i++
        j++
        if j==m:
            print "Match found at index: ",(i-j)
            j=lps[j-1]
    else:
        if j != 0:
            j=lps[j-1]
        else:
            i++
```

چون همه جوابها را میخواهیم

بازبه‌کشتن ادامه می‌دهیم.

چون میدانیم این مقدار از پیشوند الگو

با پسوند متنی که تاالان خواندیم برابر است

بدون مقایسه الگو را به راست شیفت می‌دهیم

که پیشوند زیر پسوند متن بیفتد

فاز پیش پردازش و بعد KMP مستقلاً و پشت هم اجرا می‌شوند ← زمان اجرا کل ← $O(m)+O(n)=O(m+n)$