به نام خداوند
ساختمان داده و الگوریتم
حل تمرین اول
دکتر حاجی اسماعیلی

دانشکده ریاضی و علوم کامپیوتر
فروردین ماه ۱۴۰۲

---

۱ رفتار توابع

در اینجا توابع بر اساس پیچیدگی های زمانی آنها به ترتیب صعودی مرتب شده اند:

- $f7 = (logn)$
- $f8 = log(logn)$ - این تابع کمی سریعتر از $f7$ رشد می کند اما همچنان کندتر از سایر توابع است.
- $f6 = n$
- $f5 = \binom{n}{n/2}$ به کمک تقریب استرلینگ و جایگذاری آن در فرم باز شده ی انتخاب میبینیم که این نیز یک پیچیدگی زمانی چند جمله‌ای است.
- $f2 = n^2log(n^2023)$ این تابع با نرخی متناسب با n^2 رشد می کند، اما عبارت لگاریتمی نرخ رشد را با ضریب $(log(n^(2023)) = 2023 * log(n)$ افزایش می دهد. سریعتر از توابع چند جمله ای است.
- $(logn)^2023$
- $f10 = 4^(3nlogn)$
- $f9 = 7^{n2}$ این تابع به دلیل پایه نمایی ۷ و توان n^2 سریعتر از توابع قبلی رشد می کند.

۲ پیچیدگی قطعات کد

۲-۱

- پاسخ : $O(n^2)$ راه حل: $1 + 2 + ... + n - 1 + n = (n(n-1))/2$

- پاسخ : $O(sqrt(n))$ راه حل: پیچیدگی زمانی کد ارائه شده را می توان با تجزیه و تحلیل تعداد تکرارهایی که حلقه اجرا می کند تعیین کرد.

در هر تکرار، مقدار "i" با "j" افزایش می یابد که از ۱ شروع می شود و در هر تکرار ۱ مقدار افزایش می یابد. بنابراین، حلقه بار "j" را اجرا می کند، جایی که "j" کوچکترین عدد است به طوری که:

$$1 + 2 + 3 + \cdots + j \geq n \tag{۱}$$

و طبق این خواهیم داشت که:

$$j \times (j+1)/2 \geq n \tag{۲}$$

پس خواهیم داشت که:

$$j \geq (-1 + \sqrt{1+8n})/2 \tag{۳}$$

از آنجایی که ما به پیچیدگی زمانی در بدترین حالت علاقه مندیم، می توانیم عوامل ثابت و اصطلاحات مرتبه پایین را نادیده بگیریم. بنابراین، پیچیدگی زمانی حلقه را می توان به صورت $O(sqrt(n))$ تقریب زد، زیرا عبارت غالب در فرمول "j" جذر "n" است.

- حلقه در مرحله اول n بار در مرحله دوم $n/2$ بار و به همین صورت تعداد بار تکرار میشود که خب خواهیم داشت:

$$(s) = O(n + n/2 + n/3 + ... + n/n) = n\sum_{i=0}^{n} \frac{1}{i} \tag{۴}$$

حال به حل خود سیگما میپردازیم:

$$\sum_{i=0}^{n} \frac{1}{i} \leq \int_{1}^{n} \frac{1}{x}\, dx = ln(x) \qquad (۵)$$

$$\sum_{x=1}^{n} \frac{1}{x+1} \leq \int_{1}^{n} \frac{1}{x}\, dx = ln(x) \leq \sum_{x=1}^{n} \frac{1}{x} \qquad (۶)$$

پس پاسخ نهایی خواهد بود: $O(nlogn)$

۲-۲

حلقه درونی به تعداد i بار تکرار میشود و خب i تا n بالا میرود و خب حلقه درونی n بار تکرار میشود و حلقه بیرونی نیز $log(n)$ بار تکرار میشود پس به طور کلی حلقه ها $O(nlogn)$ دفعه تکرار میشوند و صرفا لازم است درون حلقه ها $O(1)$ اجرا شود.

۳ طولانی ترین دنباله افزایشی

برای حل این مساله از $dynamicprogramming$ استفاده میکنیم و $dp[i]$ را طول بلند ترین دنباله افزایشی منتهی به نقطه i ام در نظر میگیریم و به صورت زیر کد خود $implement$ میکنیم:

```python

def LIS(nums):
    n = len(nums)
    dp = [1] * n

    for i in range(1,n):
        for j in range(i):
            if nums[j] < nums[i]:
                dp[i] = max(dp[i], dp[j]+1)
    return max(dp)
```

که خب مساله n مرتبه تکرار و در هر مرحله $O(n)$ کار صورت میگیرد پس مساله از اردر $O(n^2)$ است.

۴ حداکثر مجموع غیر مجاور

کد این سوال را میتوان به صورت زیر نوشت:

```
def MSNA(input):
    n = len(input)
    if n == 0:
        return 0
    if n == 1:
        return input[0]
    if n == 2:
        return max(input[0], input[1])
    dp = [0] * n
    dp[0] = input[0]
    dp[1] = max(input[0], input[1])
    for i in range(2, n):
        dp[i] max(dp[i-1], dp[i-2] + input[i])
    return max(dp)
```