باسمه تعالی
دانشگاه تهران - دانشکده مهندسی برق و کامپیوتر

ساختمان‌داده‌ها و الگوریتم‌ها

پاسخ کوییز اول - پیچیدگی و الگوریتم‌های بازگشتی

تاریخ: ۱۴۰۳/۰۱/۱۹

۱. ۴۰ نمره

پیچیدگی زمانی هر دو رابطه بازگشتی زیر را به روش دلخواه بدست آورید.

$$T(n) = \sqrt{n}T(\sqrt{n}) + n$$

$$T(n) = T\left(\frac{n}{۴}\right) + T\left(\frac{۳n}{۴}\right) + n^{۲}$$

پاسخ :

الف)

$$T(n) = \sqrt{n}T(\sqrt{n}) + n = \sqrt{n}T(\sqrt{n}) + (n)$$

فرض میکنیم که $n = ۲^k$ است آنگاه نتیجه می شود که $\sqrt{n} = ۲^{k/۲}$ است و همچنین میتوان گفت که $k = \log n$ است . حالا با کمک تغییر متغییر هایی که دادیم به سراغ حل مسئله میرویم :

$$T(۲^k) = ۲^{k/۲}T(۲^{k/۲}) + ۲^k$$

حالا دو طرف معادله را بر ۲^k تقسیم میکنیم .

$$\frac{T(۲^k)}{۲^k} = \frac{۲^{k/۲}T(۲^{k/۲})}{۲^k} + ۱$$

با ساده سازی به معادله زیر میرسیم :

$$\frac{T(۲^k)}{۲^k} = \frac{T(۲^{k/۲})}{۲^{k/۲}} + ۱$$

حالا تغییر متغییر $y(k) = \frac{T(۲^k)}{۲^k}$ را اعمال میکنیم و مسئله به مسئله زیر تبدیل میگردد :

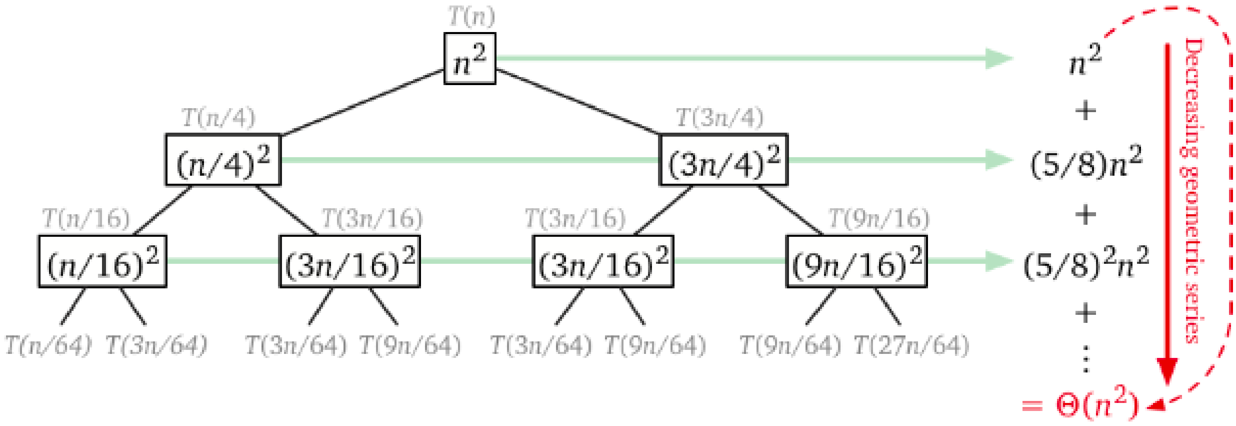
$$y(k) = y\left(\frac{k}{۲}\right) + ۱$$

حالا این سوال با قضیه اصلی به سادگی قابل حل خواهد بود . جواب نهایی مسئله به صورت زیر است :

$$T(n) = \Theta(n \log \log n)$$

ب)

راه حل های زیادی برای این مسئله می توان در نظر گرفت اما یکی از راه حل های خلاقیتی آن استفاده از درخت بازگشت است که به صورت زیر خواهد بود :



۲. ۴۰ نمره

رابطه بازگشتی قطعه کد زیر را به دست آورید و پیچیدگی زمانی آن را محاسبه کنید. استفاده از قضیه اصلی مجاز است

```
Function func(array Arr) :
    n = Length(Arr)
    If (n == 1):
        return A[0]
    array Part1[n/2], Part2[n/2], Part3[n/2], Part4[n/2]
    for (i = 0; i < (n/2) -1; i++):
        for (j = 0; j < (n/2) -1; j++):
            Part1[j] = Arr[i];
            Part2[j] = Arr[j+i];
            Part3[j] = Arr[n/2];
            Part4[j] = Arr[j];
    return func(Part1) * func(Part2) * func(Part3) * func(Part4);
```

پاسخ:

اجرای هر حلقه $O(n)$ زمان می‌برد. بنابراین در مجموع برای اجرای حلقه‌ها $O(n^۲)$ طول می‌کشد. سپس چهار زیرمسئله به اندازه $\frac{n}{۲}$ حل می‌شود. بنابراین رابطه بازگشتی به صورت زیر خواهد بود:

$$T(n) = ۴T(\frac{n}{۲}) + O(n^۲)$$

حال برای محاسبه پیچیدگی رابطه بازگشتی بالا، از قضیه اصلی استفاده می‌کنیم:

$$a = ۴, b = ۲, f(n) = O(n^۲)$$

$$n^{\log_۲ ۴} = n^۲$$

بنابراین طبق بند دوم قضیه اصلی داریم:

$$T(n) = O(n^۲ \log n)$$

۳.　　　　۲۰ نمره

پیچیدگی زمانی قطعه کد زیر را محاسبه کنید .

```
Function func(n) :
    s = 0
    for i = n  to n*n do :
        for j = n  to i  do :
            s = s + j - i
    return s
```

پاسخ :

لوپ بیرونی به اندازه ی $(n^۲ - n + ۱)$ بار اجرا میشود .
به ازای هر iteration از لوپ بیرونی ، لوپ داخلی $(i - n + ۱)$بار اجرا میگردد .
حالا به سراغ محاسبه ی تعداد کل اجراها در هر iteration در لوپ داخلی میرویم :

- وقتی که $i = n$ باشد ، لوپ داخلی به اندازه$۱ = (n - n + ۱)$ بار اجرا میشود .
- وقتی که $i = n + ۱$ باشد ، لوپ داخلی به اندازه $۲ = (n + ۱ - n + ۱)$بار اجرا میشود .
- ...
- وقتی که $i = n^۲$ باشد ، لوپ داخلی به اندازه $(n^۲ - n + ۱)$ بار اجرا میشود .

پس در کل $۱ + ۲ + ۳ + ... + (n^۲ - n + ۱)$ بار اجرا میشود که این مقدار برابر $\frac{(n^۲-n+۱)(n^۲-n+۲)}{۲}$ خواهد بود
پس order کلی مسئله از $O(n^۴)$ خواهد بود .

راه حل ساده تر :
برای هر iteration از لوپ خارجی ، لوپ داخلی به اندازه ی n تا i اجرا میگردد . و این مقدار تا زمانی که i به $n^۲$ برسد ادامه دارد. پس در حالت کلی ، لوپ داخلی به طور میانگین ، به ازای هر iteration از لوپ خارجی به اندازه ی $\frac{n^۲+n}{۲}$ اجرا میگردد. پس در مجموع ، پیچیدگی زمانی مسئله به اندازه ی

$$T(n) = (n^۲ - n + ۱)\frac{(n^۲ + n)}{۲}C$$

خواهد بود .
که order زمانی مسئله از $O(n^۴)$ خواهد بود.