به نام خدا

دانشکده ریاضی و آمار
نام و نام خانوادگی دانشجو
پوریا مرادپور
۴۰۲۴۰۲۳۰۴۰

ساختمان داده و الگوریتم
نیم‌سال اول ۰۳-۰۴
استاد: دکتر رفیعی

تاریخ شروع: ۵ فروردین ۱۴۰۴
تاریخ تحویل: ۲ آذر ۱۴۰۳

تمرین سری اول

- مجموع نمرات: ۱۳۰
- سقف نمره: ۱۰۰
- در هر تمرین، منظور از $Big - O$ برای یک الگوریتم، پایین‌ترین کران بالا و منظور از $Big - \Omega$ بالاترین کران پایین است.

قضیه ۱ (قضیه اصلی). *اگر رابطه‌ی بازگشتی به فرم زیر داشته باشیم:*
$T(n) = aT(\frac{n}{b}) + f(n),\ where\ a \geq 1\ and\ b > 1$

آنگاه:

- *حالت اول: اگر* $f(n) = O(n^{c-\epsilon})$ *باشد، جاییکه* $\epsilon > 0$ *آنگاه* $T(n) = \theta(n^c)$.
- *حالت دوم: اگر* $f(n) = \theta(n^c \log^k n)$ *باشد، جاییکه* k *یک عدد صحیح نامنفی باشد، آنگاه* $T(n) = \theta(n^c \log^{k+1} n)$.
- *حالت سوم: اگر* $f(n) = \Omega(n^{c+\epsilon})$ *باشد، جاییکه* $\epsilon > 0$ *است و همچنین* $af(\frac{n}{b}) \leq hf(n)$ *باشد برای یک* $0 < h < 1$ *و هر* n *به اندازه‌ی کافی بزرگ، آنگاه* $T(n) = \theta(f(n))$.

در هر سه حالت ذکر شده در بالا $c = \log_b a$ *می‌باشد (با توجه به روش‌های قبلی ذکر شده برای حل روابط بازگشتی، علت این مقدار برای* c *واضح است).*

---

۱. (۲۰ نمره) در هر یک از جفت توابع زیر مشخص کنید که کدام‌یک نماد مجانبی کمتر، مساوی یا بیشتر نسبت به دیگری دارد. برای هر پاسخ توضیحی ارائه کنید.

(آ) (۲ نمره)

$$100n^2 \text{ و } 0.01n^3$$

(ب) (۲ نمره)

$$\log_2^2 n \text{ و } \log_2 n^2$$

(ج) (۲ نمره)

$$26n - 1 \text{ و } 2^n$$

(د) (۲ نمره)

$$(n-1)! \text{ و } n!$$

(ه) (۱۲ نمره) در این قسمت کارایی زمانی هر قطعه کد را با نماد مجانبی بیان کرده و آن‌ها را مقایسه کنید.

```
i = 2;
while (i < n)
    i = i * i;
```

و

```
for (i = 1; i <= n; i++)
    for (j = 1; j <= n; j = j + i)
        x++;
```

جواب:

(a)

$$\exists\, n_0,\ c_1,\ c_2\ >\ 0 \quad \text{s.t.} \quad \forall n\ >\ n_0,\ 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n),$$
$$\Rightarrow f(n) = \Theta(g(n))$$

$f(n) = 100n^2,$
$g(n) = n^2,$
$c_1 = c_2 = 100,$
$n_0 = 1,$
$\forall n\ >\ 1,\ 0 \leq 100n^2 \leq 100n^2 \leq 100n^2,$
$\Rightarrow 100n^2 = \Theta(n^2)$

$f(n) = 0.01n^3$
$g(n) = n^3$
$c_1 = c_2 = \dfrac{1}{100},$
$n_0 = 1,$
$\forall n\ >\ 1,\ 0 \leq \dfrac{1}{100}n^3 \leq \dfrac{1}{100}n^3 \leq \dfrac{1}{100}n^3$
$\Rightarrow \dfrac{1}{100}n^3 = \Theta(n^3)$

در نتیجه تابع دوم نماد مجانبی بزرگتری دارد

(b)

$\exists\, n_0,\ c_1,\ c_2\ >\ 0\quad \text{s.t.}\quad \forall n\ >\ n_0,\ 0 \le c_1 g(n) \le f(n) \le c_2 g(n),$
$\Rightarrow f(n) = \Theta(g(n)$

$f(n) = (\log_2(n))^2,$
$g(n) = (\log_2(n))^2,$
$c_1 = c_2 = 1,$
$n_0 = 1,$
$\forall n\ >\ 1,\ 0 \le (\log_2(n))^2 \le (\log_2(n))^2 \le (\log_2(n))^2,$
$\Rightarrow (\log_2(n))^2 = \Theta((\log_2(n))^2)$

$f(n) = \log_2(n^2) = 2\log_2(n)$
$g(n) = \log_2(n)$
$c_1 = 1, \quad c_2 = 3,$
$n_0 = 1,$
$\forall n\ >\ 1,\ 0 \le \log_2(n) \le 2\log_2(n) \le 3\log_2(n)$
$\Rightarrow \log_2(n^2) = \Theta(\log_2(n))$
در نتیجه تابع اول نماد مجانبی بزرگتری دارد

(c)

$\exists\, n_0,\ c_1,\ c_2\ >\ 0\quad \text{s.t.}\quad \forall n\ >\ n_0,\ 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n),$
$\Rightarrow f(n) = \Theta(g(n)$

$f(n) = 26n - 1,$
$g(n) = n,$
$c_1 = 1, c_2 = 26,$
$n_0 = 1,$
$\forall n\ >\ 1,\ 0 \leq n \leq 26n - 1 \leq 26n,$
$\Rightarrow 26n - 1 = \Theta(n)$

$f(n) = 2^n$
$g(n) = 2^n$
$c_1 = c_2 = 1,$
$n_0 = 1,$
$\forall n\ >\ 1,\ 0 \leq 2^n \leq 2^n \leq 2^n$
$\Rightarrow 2^n = \Theta(2^n)$
در نتیجه تابع دوم نماد مجانبی بزرگتری دارد

(c)

$\exists\, n_0,\ c_1,\ c_2\ >\ 0 \quad \text{s.t.} \quad \forall n\ >\ n_0,\ 0 \le c_1 g(n) \le f(n) \le c_2 g(n),$
$\Rightarrow f(n) = \Theta(g(n)$

$f(n) = 26n - 1,$
$g(n) = n,$
$c_1 = 1, c_2 = 26,$
$n_0 = 1,$
$\forall n\ >\ 1,\ 0 \le n \le 26n - 1 \le 26n,$
$\Rightarrow 26n - 1 = \Theta(n)$

$f(n) = 2^n$
$g(n) = 2^n$
$c_1 = c_2 = 1,$
$n_0 = 1,$
$\forall n\ >\ 1,\ 0 \le 2^n \le 2^n \le 2^n$
$\Rightarrow 2^n = \Theta(2^n)$
در نتیجه تابع دوم نماد مجانبی بزرگتری دارد

(d)

$\exists\, n_0,\ c_1,\ c_2\ >\ 0\quad \text{s.t.}\quad \forall n\ >\ n_0,\ 0 \le c_1 g(n) \le f(n) \le c_2 g(n),$
$\Rightarrow f(n) = \Theta(g(n)$

$f(n) = (n-1)!,$
$g(n) = (n-1)!,$
$c_1 = c_2 = 1,$
$n_0 = 1,$
$\forall n\ >\ 1,\ 0 \le (n-1)! \le (n-1)! \le (n-1)!,$
$\Rightarrow (n-1)! = \Theta((n-1)!)$

$f(n) = n!$
$g(n) = n!$
$c_1 = c_2 = 1,$
$n_0 = 1,$
$\forall n\ >\ 1,\ 0 \le n! \le n! \le n!$
$\Rightarrow n! = \Theta(n!)$

در نتیجه تابع دوم نماد تتای بزرگتری دارد اما اگر نماد او را بررسی کنیم داریم:

$\exists\, n_0,\ c\ >\ 0\quad \text{s.t.}\quad \forall n\ >\ n_0,\ 0 \le f(n) \le cg(n),$
$\Rightarrow f(n) = O(g(n)$

$f(n) = (n-1)!,$
$g(n) = n!,$
$c = 1,$
$n_0 = 1,$
$\forall n\ >\ 1,\ 0 \le (n-1)! \le n!,$
$\Rightarrow (n-1)! = O(n!)$

$f(n) = n!,$
$g(n) = n!,$
$c = 1,$
$n_0 = 1,$
$\forall n\ >\ 1,\ 0 \le n! \le n!$
$\Rightarrow n! = O(n!)$

در نتیجه نماد اوی هردو تابع میتواند برابر باشد

(e)

با توجه به این که در هر دو قطعه کد عمل اصلی درخط سوم انجام میشوند تعداد تکرار آن را میسنجیم

قطعه کد سمت راست

$n-2(O(1) \;\Rightarrow\; O(n)$

قطعه کد سمت چپ

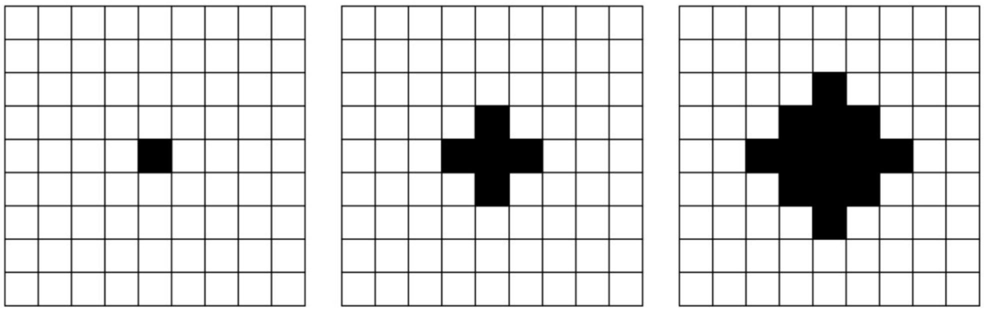
$(n)(n)(O(1)) \;\Rightarrow\; O(n^2)$

$\Rightarrow$ **قطعه کد سمت چپ دارای نماد مجانبی بزرگتری است**

منابع:

لکچر نوت ها

۲. (۲۰ نمره) مسئله محله‌ی فون نیومان: یک صفحه شطرنجی را تصور کنید که یک مربع سیاه در مرکز آن قرار دارد. در هر مرحله، کنار تمام اضلاع بیرونی مربع مربع‌های جدیدی تولید می‌شود. برای نمونه، ۳ مرحله از این الگوریتم در تصویر زیر نمایش داده شده‌اند:



$n = 0$ $\quad$ $n = 1$ $\quad$ $n = 2$

(آ) (۱۰ نمره) ابتدا یک رابطه بازگشتی برای تعداد مربع‌های صفحه در مرحله‌ی n ام پیدا کنید.

(ب) (۱۰ نمره) به کمک روش جای‌گذاری بازگشتی آن را حل کنید. تابع به دست آمده در چه مرتبه‌ای قرار می‌گیرد؟

جواب:

(a)
$S(n) = S(n-1) + 4n$

(b)
Substituting
$S(n) = S(n-1) + 4n,\ S(n-1) = S(n-2) + 4(n-1) = S(n-2) + 4n - 4$
$\Rightarrow S(n) = S(n-2) + 8n - 4$
$S(n) = S(n-2) + 8n - 4,\ S(n-2) = S(n-3) + 4(n-2) = S(n-2) + 4n - 8$
$\Rightarrow S(n) = S(n-3) + 12n - 12$
$S(n) = S(n-3) + 12n - 12,\ S(n-3) = S(n-4) + 4(n-3) = S(n-3) + 4n - 12$
$\Rightarrow S(n) = S(n-4) + 16n - 24$
.
.
.
$\Rightarrow S(n) = S(n-i) + 4in - f(i)$
پیدا کردن رابطه ای برای تابع اف بر حسب آی
(مجبورم فارسی بنویسم چیکار کنم)
$f(1) = 0, \quad f(2) = 4, \quad f(3) = 12, \quad f(4) = 24,$
حدس و آزمایش $\Rightarrow f(i) = 2i(i-1), \quad \Rightarrow \quad S(n) = S(n-i) + 4in - 2i(i-1)$
$S(0) = 1$ جواب اولیه $\quad \Rightarrow\ n - i = 0\ \Rightarrow\ n = i$
$\Rightarrow S(n) = S(0) + 4n^2 - 2n(n-1)\ \Rightarrow\ S(n) = 2n^2 - 2n + 1$
با توجه به قضیه ای که در کلاس اثبات شد چند جمله ای ها از مرتبه کا دارای نماد مجانبی تتای ان به توان کا هستند
$\Rightarrow\ S(n) = \Theta(n^2)$

منابع:

لکچر نوت ها

۳. (۲۰ نمره) فرمول زیر را که برای به دست آوردن جمله n ام دنباله‌ی فیبوناچی به کار می‌رود در نظر بگیرید.

(آ) (۱۰ نمره) شبه کد آن را با استفاده از رویکرد بازگشتی بنویسید.

(ب) (۱۰ نمره) درستی الگوریتم را ثابت کنید

$$\begin{bmatrix} F(n-1) & F(n) \\ F(n) & F(n+1) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^n \quad \text{برای } n \geq 1$$

جواب:

(a)

Algorithm 1 Fibonacci Using Matrix Exponentiation

```
function Fibonacci(n)
    if n = 0 then
        return 0
    end if
    A ← [0 1; 1 1]
    Result ← MatrixPower(A, n)
    return Result[1][2]                ▷ Return the top-right value of the matrix
end function
function MatrixPower(A, n)
    if n = 1 then
        return A
    end if
    Half ← MatrixPower(A, ⌊n/2⌋)
    Half ← MatrixMultiply(Half, Half)
    if n is odd then
        Half ← MatrixMultiply(Half, A)
    end if
    return Half
end function
function MatrixMultiply(A, B)
    return [A[1][1]·B[1][1] + A[1][2]·B[2][1]   A[1][1]·B[1][2] + A[1][2]·B[2][2];
            A[2][1]·B[1][1] + A[2][2]·B[2][1]   A[2][1]·B[1][2] + A[2][2]·B[2][2]]
end function
```

ب - اثبات با استقرا

شرط اولیه ($n = 1$)

For $n = 1$، سمت چپ:

$$\begin{bmatrix} F(0) & F(1) \\ F(1) & F(2) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}.$$

سمت راست:

$$\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^1 = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}.$$

شرط اولیه اثبات شد.

استقرا

با فرض درستی $n = k$،

$$\begin{bmatrix} F(k-1) & F(k) \\ F(k) & F(k+1) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^k.$$

ضرب میکنیم $\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$:

$$\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^{k+1} = \begin{bmatrix} F(k-1) & F(k) \\ F(k) & F(k+1) \end{bmatrix} \cdot \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}.$$

ماتریس ها را ضرب میکنیم

$$\begin{bmatrix} F(k-1) & F(k) \\ F(k) & F(k+1) \end{bmatrix} \cdot \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} F(k) & F(k+1) \\ F(k+1) & F(k+2) \end{bmatrix}.$$

درایه های ماتریس طبق سری فیبوناچی جواب میدهند:

$$F(k+2) = F(k) + F(k+1),$$

منابع:

GeeksForGeeks (for the idea)
Lecture Notes

۴. (۲۰ نمره) فرض کنید برای یک الگوریتم، رابطه بازگشتی پیچیدگی زمانی زیر داده شده است:

$$T(n) = \sqrt{n}\dot{\log} n + \left(\frac{n}{4}\right) 3T(n-1)$$

(آ) (۱۰ نمره) با توجه به شرایط و حالات قضیه اصلی (Master Theorem)، بررسی کنید که آیا می‌توان مستقیماً از این قضیه برای تعیین درجه پیچیدگی زمانی این رابطه استفاده کرد؟ علت پاسخ خود را توضیح دهید.

(ب) (۱۰ نمره) اگر استفاده از قضیه اصلی مستقیماً ممکن نیست، از روش درخت بازگشت استفاده کنید تا رابطه بازگشتی را حل کرده و مرتبه زمانی آن را به دست آورید.

جواب:

ب - استفاده از روش درخت بازگشت برای حل مسئله

باز کردن درحت بازگشت

$$T(n) = \sqrt{n}\cdot\log n + \frac{n}{4}\cdot 3\left(\sqrt{n-1}\cdot\log(n-1) + \frac{n-1}{4}\cdot 3T(n-2)\right).$$

$$T(n) = \sqrt{n}\cdot\log n + \frac{n}{4}\cdot 3\cdot\sqrt{n-1}\cdot\log(n-1) + \frac{n}{4^2}\cdot 3^2 T(n-2).$$

تا مرحله کی ام:

$$T(n) = \sum_{j=0}^{k}\frac{n}{4^j}\cdot 3^j\cdot\sqrt{n-j}\cdot\log(n-j) + \frac{n}{4^k}\cdot 3^k T(n-k).$$

شرط اولیه

رابطه بازگشتی را تا شرط اولیه ادامه میدهیم:

$$\frac{n}{4^k}\cdot 3^k T(1),$$

که این جمله با توجه به این که کسری از ان است قابل چشم پوشی است.

هزینه کلی

$$T(n) = \sum_{j=0}^{h}\frac{n}{4^j}\cdot 3^j\cdot\sqrt{n-j}\cdot\log(n-j),$$

برای n های بزرگ، $\sqrt{n-j}\approx\sqrt{n}$ و $\log(n-j)\approx\log n$.

پیچیدگی زمانی

$$T(n) = O(n \cdot \sqrt{n} \cdot \log n).$$

منابع:

Lecture Notes

۵. (۳۰ نمره) این الگوریتم بازگشتی برای مسأله یکتایی عناصر را در نظر بگیرید.

Algorithm 2 UniqueElements(A[0...$n-1$]): Determines whether all the elements in a given array are distinct

Input: An array $A[0 \ldots n-1]$
Output: Output: Returns `true` if all the elements in A are distinct and `false` otherwise

```
if n == 1 then
    return true
else if not UniqueElements(A[0...n-2]) then
    return false
else if not UniqueElements(A[1...n-1]) then
    return false
else
    return A[0] ≠ A[n-1]
end if
```

(آ) (۵ نمره) کارایی زمانی الگوریتم بالا چقدر است و چرا ناکاراست؟

(ب) (۱۰ نمره) الگوریتم بالا را به یک الگوریتم iterative تبدیل کنید و درستی آن رو اثبات کنید؟

(ج) (۱۵ نمره) الگوریتم خود را در صورت امکان بهینه کنید و شبه کد و کد الگوریتم خود را ارائه دهید؟

جواب:

آـ

به علت دو شاخه شدن و روی هم رفتن آرایه های ایجاد شده در هر مرحله هر بار پیچیدگی زمانی:

$$O(2^n)$$

که به شدت ناکارا و غیر بهینه است جدای از ناممکن بودن به کارگیری این الگوریتم برای نمونه های بزرگ این الگوریتم تسک خواسته شده را نیز به درستی انجام نمیدهد برای مثال با پیش بردن الگوریتم روی آرایه زیر متوجه این ناکامدی میشویم:

$$A = [1, 2, 1]$$

ب-

Algorithm 3 UniqueElements(A): Determines if all elements in the array are distinct

Input: An array $A[0 \dots n-1]$
Output: Returns `true` if all elements are distinct, otherwise `false`

for $i = 0$ **to** $n-1$ **do**
  for $j = i+1$ **to** $n-1$ **do**
    if $A[i] = A[j]$ **then return** `false`
    end if
  end for
end for
  return `true`

خصیصه ناوردایی : در پایان هر مرحله عضو آِی در آرایه زیر یکتاست

$$A = [0, .., j]$$

گام آغازین: با توجه به اینکه فقط عضو بعدی آی در آرایه بررسی شده و برابر نبوده پس آِ یکتاست

گام نگهداری: اگر در مرحله کا ام یکتا باشد با توجه به بررسی شرط و صورت شکسته نشدن حلقه، آی یکتاست.

گام پایانی: با توجه به نبود آی در کل آرایه عضو مد نظر در آرایه یکتاست

با تکرارا این پروسه برای تمام اعضای آرایه میتوان از درستی الگوریتم اطمینان حاصل کرد.

ج-

Algorithm 4 UniqueElements(A): Determines if all elements in the array are distinct using a hash set

Input: An array $A[0 \dots n-1]$
Output: Returns `true` if all elements are distinct, otherwise `false`

Create an empty hash set S
for each element x in A **do**
  if $x \in S$ **then return** `false`
  end if
  Add x to S
end for
  return `true`

با استفاده از هش ست ها میتوان در ازای افزاش پیچیدگی فضایی به پیچیدگی خطی، پیچیدگی زمانی را هم به پیچیدگی خطی کاهش داد

کد پایتون برای پیاده سازی الگوریتم ذکر شده

```
def unique_elements(arr):
    # Create an empty hash set
    seen = set()
    for x in arr:
        if x in seen:
            return False
        seen.add(x)
    return True
```

منابع:

Lecture Notes

۶. (۲۰ نمره) با استفاده از قضیه اصلی، در صورت امکان مرتبه رشد هر یک از موارد زیر را بدست آورید.

(آ) (۸ نمره)

$$T(n) = 3T(\frac{n}{3}) + n^2 \log n$$

(ب) (۸ نمره)

$$T(n) = \sqrt{2}T(\frac{n}{2}) + \log n$$

(ج) (۴ نمره)

$$T(n) = 16T(\frac{n}{4}) + n$$

جواب:

آ - $T(n) = 3T\left(\frac{n}{3}\right) + n^2 \log n$

$$a = 3, \quad b = 3, \quad f(n) = n^2 \log n, \quad p = \log_b(a) = \log_3(3) = 1.$$

بخش سوم قضیه

$$\varepsilon = 1, f(n) = n^{p+\varepsilon} \cdot \log n = \Omega(n^{p+\varepsilon})$$

$$T(n) = \Theta(f(n)) = \Theta(n^2 \log n).$$

ب- $T(n) = \sqrt{2}T\left(\frac{n}{2}\right) + \log n$

$$a = \sqrt{2}, \quad b = 2, \quad f(n) = \log n, \quad p = \log_b(a) = \log_2(\sqrt{2}) = \frac{1}{2}.$$

$f(n) = \log n$ و $n^p = n^{1/2} = \sqrt{n}$: - $\log n = O(n^{1/2-\varepsilon})$ برای $\varepsilon > 0$.
پس از بخش اول قضیه استفاده میکنیم

$$T(n) = \Theta(n^{1/2}).$$

ج - $T(n) = 16T\left(\frac{n}{4}\right) + n$

$$a = 16, \quad b = 4, \quad f(n) = n, \quad p = \log_b(a) = \log_4(16) = 2.$$

۱. $f(n) = n$ و $n^p = n^2$: - $n = O(n^{2-\varepsilon})$ و $\varepsilon > 0$.
پس از بخش اول قضیه استفاده میکنیم

$$T(n) = \Theta(n^p) = \Theta(n^2).$$

منابع:

Lecture Notes