

دانش اگر در ثریا هم باشد مردمانی از سرزمین پارس بر آن دست خواهند یافت.

پیامبر اعظم (ص)

شرح جامع

دروس مشترک (ساختمان داده)

«مهندسی کامپیوتر – مهندسی فناوری اطلاعات IT»

(نرم‌افزار – هوش مصنوعی – نرم‌افزار – الگوریتم و محاسبات – تجارت الکترونیکی
امنیت اطلاعات – شبکه‌های کامپیوتری – مهندسی فناوری اطلاعات)

مولفان: شرمین موسوی – محمدرضا لحمی

انتشارات ارشد

پیشگفتار ناشر

خداوند بزرگ را شاکریم که فرصتی عنایت فرمود تا بتوانیم قدمی در جهت گسترش علم و فرهنگ کشور عزیزمان برداریم و مسیری هموار را در اختیار مخاطبان و داوطلبان علاقمند به ادامه تحصیل قرار دهیم. از آنجائی که درحوزه کتاب‌های درسی و کمک درسی کارشناسی ارشد کتاب‌های زیادی به رشته تحریر در آمده است با این حال وجود یک مجموعه جامع، دقیق، کامل و دارای رفرنس(منابع) مستند احساس می‌شود.

همیشه دغدغه داشتیم که ای کاش می‌شد تمامی منابع پراکنده و گاهاً دست نایافتنی کنکور در یک مجموعه به گونه‌ای جذاب و شیوا آماده می‌شد و همچنین سوالات دوره‌های قبل کنکور نیز به گونه‌ای تشریحی پاسخ داده می شدند، که خدشه‌ای بر آنها وارد نشود و خیال داوطلبان عزیز آنقدر آسوده باشد که اطلاعات مندرج در پاسخنامه‌ها را برای بهره‌گیری در آزمون خود بکار گیرند. این کار با یاری خداوند بزرگ و همکاری نخبه‌ترین دانشجویان و اساتید کشور در انتشارات ارشد هماهنگ گردید و در تمامی مقاطع تحصیلی از کاردانی به کارشناسی، کارشناسی ارشد و دکتری در حوزهٔ آزمون‌های سراسری – آزاد و پیام نور بسته‌های آموزشی شامل کتاب‌های بانک سوالات و کتاب‌های شرح جامع درس آماده گردید. در تهیه مجموعه‌های فوق‌الذکر از فارغ‌التحصیلان کارشناسی ارشد و دکتری که حائز رتبه‌های تک رقمی کنکور می‌باشند، استفاده کرده‌ایم، تا شما عزیزان بتوانید در راستای افزایش توان علمی خود در جهت نیل به مقصود خویش نایل آیید. بدون تردید می‌توان گفت که برای اولین بار در نظام آموزشی کشور مجموعه‌ای به این سبک عرضه می‌شود که در کیفیت و کمیت بی‌نظیر بوده و تمامی نیازهای دانشجویان و داوطلبان عزیز را پوشش می‌دهد. امید است که استفاده از این مجموعه عزیزان را در جهت موفقیت در آزمون یاری رسانده باشد. در پایان لازم به ذکر است که بی نقصی محض از آن خداوند متعال می‌باشد، بنا براین از تمامی خوانندگان و داوطلبان عزیز خواهشمندیم نظرات سازنده خود را در مورد این کتاب به سایت انتشارات ارشد اعلام نمایند تا ما را در جهت هر چه بهتر آماده کردن این محصولات یاری رسانند.

با تشکر

ناشـــــر

www.arshadbook. ir

فهرست مطالب

عنوان **صفحه**

پیشگفتار ناشر ۳
پیشگفتار مولف ۴

فصل اول: الگوریتم‌ها ۱۱
توابع گسترش ۱۵
مرتبه اجرایی توابع بازگشتی ۱۷

فصل دوم: ساختارهای داده‌ای ۳۹
نوع داده انتزاعی ۳۹
آرایه ۳۹
آرایه یک بعدی ۳۹
آرایه‌های دوبعدی ۴۳
آرایه‌های چند بعدی ۴۹
لیست‌های پیوندی ۴۹
لیست یک طرفه خطی ۵۱
لیست حلقوی ۵۳
لیست پیوندی دوگانه ۵۴
لیست عمومی ۵۵

فصل سوم: پشته ۸۱
پیاده سازی پشته با آرایه ۸۱
پیاده سازی پشته با لیست پیوندی ۸۳
چند مثال کاربردی از پشته ۸۴
فراخوانی توابع ۸۷

فصل چهارم: صف ۱۰۱
صف ساده ۱۰۱
صف حلقوی ۱۰۳

عنوان **صفحه**

پیاده‌سازی صف با لیست پیوندی (صف پیوندی) ۱۰۴
صف اولویت ۱۰۶

فصل پنجم: گراف ۱۲۱
پیاده‌سازی گراف ۱۲۴
انواع زیرگراف ۱۲۵
الگوریتم جستجوی عمقی (Depth first search) ۱۲۵
الگوریتم جستجوی سطحی (Breadth Fist Search) ۱۲۷
درخت‌پوشای کمینه ۱۲۸
الگوریتم کراسکال ۱۲۸
الگوریتم پریم ۱۳۰
کوتاهترین مسیر در گراف وزن‌دار ۱۳۰

فصل ششم: درخت ۱۳۹
درخت عمومی ۱۴۰
پیاده سازی درخت عمومی ۱۴۰
پیمایش درخت عمومی ۱۴۱
درخت دودویی ۱۴۲
پیاده سازی درخت های دودویی ۱۴۵
پیمایش درخت دودویی ۱۴۷
تبدیل درخت عمومی به درخت دودویی ۱۵۱
جنگل ۱۵۲

فصل هفتم: درخت‌های جستجوی دودویی ۱۶۷
درخت AVL ۱۷۰
درخت ۲-۳-۴ ..... ۱۷۳
درخت Heap ۱۷۴
الگوریتم هافمن ۱۷۶

فصل هشتم: مرتب‌سازی ۱۹۳
مرتب‌سازی تعویضی (Exhange) ۱۹۳
مرتب‌سازی حبابی (bubble sort) ۱۹۴
مرتب‌سازی انتخابی (Selection Sort) ۱۹۵

عنوان **صفحه**

مرتب‌سازی درجی (Insertion Sort) ۱۹۶
مرتب‌سازی سریع (Quick Sort) ۱۹۷
مرتب‌سازی ادغام (Merg Sort) ۲۰۰
مرتب‌سازی شمارشی (Counting Sort) ۲۰۲
۸-۷- مرتب‌سازی مبنا (Radix) ۲۰۴
مرتب‌سازی هرمی (heap sort) ۲۰۵
مرتب‌سازی درختی (Tree Sort) ۲۰۶

تست‌های طبقه‌بندی

پاسخنامه ساختمان داده‌ها (کامپیوتر) سراسری سال ۱۳۸۹ ..... ۲۲۱
پاسخنامه ساختمان داده‌ها (کامپیوتر) سراسری سال ۱۳۸۹ ..... ۲۲۳
سوالات ساختمان داده‌ها (کامپیوتر) سراسری سال ۱۳۸۹ ..... ۲۲۵
پاسخنامه ساختمان داده‌ها (کامپیوتر) سراسری سال ۱۳۸۹ ..... ۲۲۶

فصل اول

الگوریتم‌ها

فصل اول

الگوریتم‌ها

مقدمه

الگوریتم یک روش برای حل یک مسئله است که دارای ویژگی‌های زیر می‌باشد:

۱- هر الگوریتم دارای یک سری ورودی است که آنها را پردازش می‌کند.

۲- الگوریتم‌ها خاتمه‌پذیر هستند. و بعد از اجرای یک سری دستورالعمل‌ها در یک نقطه به پایان می‌رسند.

۳- ردپای الگوریتم‌ها را می‌توان دنبال کرد. بدین معنی که با اعمال ورودی‌ها و اجرای دستورالعمل‌های الگوریتم باید بتوان به خروجی معینی رسید.

برای یک مسئله می‌توان الگوریتم‌های مختلفی ارائه کرد. تفاوت این الگوریتم‌ها در مدت زمان اجرای آنهاست. بدین معنی که اگر تعداد ورودی‌های یکسانی را به الگوریتم‌ها اعمال کنیم، ممکن است که زمان به پایان رسیدن الگوریتم‌ها متفاوت باشد. طبیعی است که الگوریتمی مورد پذیرش است که سرعت بیشتری داشته باشد.

مثال:

تعداد دفعات اجرای دستور(*)Print در قطعه کد زیر چند است.

```
for(i = ۰;i < n;i++)
    print(*);
```

شمارنده حلقه i از ۰ شروع و به $n-۱$ ختم می‌گردد. به این ترتیب print(*) دقیقاً n بار اجرا می‌گردد.

🕮 نکات

- دستورات بدنه حلقه‌های زیر $n_۲ - n_۱$ بار اجرا می‌شوند.

```
for(i = n۱;i < n۲;i++)          i = n۱
{                               while (i < n۲)
                                {
بدنه حلقه                       بدنه حلقه
}                                   i++
                                }
```

- دستورات بدنه حلقه‌های زیر $n_۲ - n_۱ + ۱$ بار اجرا می‌شوند.

```
for(i = n۱; i <= n۲; i++)
{
بدنه حلقه
}
```

```
i = n۱
while (i <= n۲)
{ بدنه حلقه
i++
}
```

- در حلقه‌های تکرار اگر بدنه حلقه n بار اجرا شود، خود دستور حلقه (while, for) n+1 بار تکرار می‌شود. این مسئله به دلیل بررسی شرط حلقه جهت خروج از آن می‌باشد.

مثال:

تعداد دفعات تکرار دستور $x = x + ۱$ در قطعه کد زیر را محاسبه کنید:

```
۱)for(i = ۱;i <= n;i++)
۲) for(j = ۱; j <= m; j++)
۳) x = x + ۱;
```

حلقه داخلی باعث اجرای m بار دستور $x = x + ۱$ می‌گردد. حلقه خارجی نیز باعث اجرای n بار دستورات سطر ۲و۳ می‌گردد. به این ترتیب سطر ۳ دقیقاً mn بار تکرار می‌گردد.

مثال:

در قطعه کد زیر دفعات اجرای دستور سطر ۳ را محاسبه کنید.

```
۱) for  k := ۰   to  n − ۱  do
۲)        for  i := ۱   to  n − k  do
۳)              x = x * ۲
```

مقدار k	مقدار i	تعداد دفعات تکرار سطر ۳
$k = ۰$	$i = ۱,...,n$	n
$k = ۱$	$i = ۱,۲,...,n-۱$	$n-۱$
$k = ۲$	$i = ۱,۲,...,n-۲$	$n-۲$
⋮		⋮
$k = n-۱$	$i = ۱$	۱

تعداد دفعات تکرار سطر ۳ برابر است با: $۱+۲+...+(n-۱)+n = \frac{n(n+۱)}{۲}$

نکات

- برای شمارش تعداد دفعات اجرای یک دستور بهتر است با دو خاصیت دستور Σ آشنا شوید.

$$\sum_{i=m}^{n} a = (n-m+1)*a$$

$$\sum_{i=1}^{n} i = 1+2+...+n = \frac{n(n+1)}{2}$$

- تعداد دفعات تکرار بدنه حلقه‌های زیر ($\log_m^n$) است.

```
i = n                          i = ۱
while(i > ۱)                   while(i < n)
{ بدنه حلقه                    { بدنه حلقه
    i = i / m;                     i = i * m;
}                              }
```

مثال:

مقدار محاسبه شده برای Total را تعیین کنید. **(علوم کامپیوتر ۷۹)**

```
Total = ۰
for i=1 to n do
begin
    k : N
    while(k <> ۱)do
    begin
        k := k  div  ۲  ;
        Total  : = Total + ۱ ;
    end
end;
```

۱) $N(\log_2^N - 1)$ ۲) $N\log_2^N$ ۳) $N(\log_2^N + 1)$ ۴) $\log_2^N + 1$

حل:

پاسخ گزینه: «۲»: حلقه for، N بار و حلقه while، $\log_2^N$ تکرار می‌گردد.

مثال:

پس از اجرای قطعه کد زیر، مقدار نهایی x چه مقداری خواهد بود. (۸۵ – IT)

```
for (i = ۱;i <= n;i++){
    for(j = ۱; j <= n; j++)
        x++;
    j = ۱;
    while(j < n){
        x++;
        j = j*۲;
    }
}
```

۱) $n\left\lceil \log_{۲}^{n} \right\rceil$ ۲) $n^{۲} + \left\lceil \log_{۲}^{n} \right\rceil$ ۳) $n^{۲} + n\left\lceil \log_{۲}^{n} \right\rceil$ ۴) $n + \left\lfloor \log_{۲}^{n} \right\rfloor$

پاسخ گزینه: «۳»: حلقه for داخلی n بار اجرا می‌گردد و حلقه while، $\log_{۲}^{n}$ بار حلقه خارجی for نیز n بار اجرا می‌شود. به این ترتیب دستور $x++$ به تعداد $n^{۲} + n\left\lceil \log_{۲}^{n} \right\rceil$ بار تکرار می‌شود.

نکته: دستورات مربوط به تعریف تابع، متغیر و ابتدا و انتهای بلاک از پردازنده زمانی را جهت اجرا اخذ نمی‌کنند. بنابراین در شمارش کلی تعداد دفعات اجرای یک برنامه از آنها صرف‌نظر می‌گردد.

مثال:

تعداد کل مراحل اجرای برنامه زیر را بدست آورید:

```
                                    → ۰
void  test(n[ ], m)
{                                   → ۰
    int     sum;                    → ۰
    sum  = ۰;                       → ۱
    for(i = ۰;i < m;i++)            → m + ۱
        sum+ = n[i];                → m
    write(sum / m);                 → ۱
}                                   ۲m + ۳
```

توابع گسترش

در اکثر اوقات محاسبه تعداد دقیق تکرار و یا اجرای یک برنامه، کار بسیار دشواری خواهد بود. به همین دلیل غالباً در تمام برنامه یک عمل اصلی را مشخص کرده و تعداد دفعات اجرای آن برنامه براساس آن عمل اصلی تخمین زده می‌شود.

به عنوان مثال در برنامه زیر بررسی این که عناصر آرایه x از ۱۰ بیشتر باشد عمل اصلی است (دستور if) و به این ترتیب می‌توان گفت این قطعه برنامه حداکثر n بار اجرا می‌گردد.

```
for i = ۱ to n do
    if x[i] > ۱۰ then
        Break;
```

در تخمین زمان اجرای الگوریتم‌ها از نمادهای Ω, θ, O استفاده می‌گردد.

مرتبه اجرایی O

گوئیم تابع $f(n)$ از مرتبه O تابع $g(n)$ است و می‌نویسیم $f(n) = O(g(n))$ یا $f(n) \in O(g(n))$، هرگاه اعداد صحیح n_0, C یافت شوند به طوری که $\forall n \geq n_0 \rightarrow f(n) \leq Cg(n)$ باشد.

مثال:

$g(n) = n, f(n) = ۲n + ۱$ را در نظر بگیرید. می‌توان گفت $۲n + ۱ \in O(n)$ است چرا که برای $n \geq ۲$ داریم $۲n + ۱ \leq ۳n$

مرتبه اجرایی θ

گوییم تابع $f(n)$ از مرتبه $\Omega(g(n))$ است و می‌نویسیم $f(n) \in \Omega(g(n))$ یا $f(n) = \theta(g(n))$ هرگاه اعداد صحیح و مثبت n_0, c_2, c_1 وجود داشته باشند به طوری که برای هر $n \geq n_0$ بتوان رابطه زیر را نوشت:

$$c_1 g(n) \leq f(n) \leq c_2 g(n)$$

مثال:

تابع $f(n) = n^3 + n$ از مرتبه θ تابع $g(n) = n^3$ است زیرا داریم: $n \geq ۱, c_2 = ۱, c_1 = ۱$

$$n^3 \leq n^3 + n \leq ۲n^3$$

مرتبه اجرایی Ω

گوئیم تابع $f(n)$ از مرتبه $\Omega(g(n))$ است و می‌نویسیم $f(n) \in \Omega(g(n))$ یا $f(n) = \Omega(g(n))$ هرگاه اعداد صحیح n_0, c وجود داشته باشند که به ازاء هر $n \geq n_0$ بتوان نوشت $f(n) \geq cg(n)$

نکات

$f(n) = O(g(n)) \Leftrightarrow g(n) = \Omega(f(n))$

$\theta(f(n) + g(n)) = \theta(\max(f(n), g(n))$

$n! = O(۲^n)$

$\log n! = \theta(n \log n)$

$\sum_{i=۱}^{k} i^k = \theta(z^{k+۱})$

- در چند جمله‌ای $an^m + bn^{m-۱} + \ldots$ مرتبه اجرایی $\theta(n^m)$ خواهد بود.

مثال:

درجه الگوریتم زیر کدام است؟

```
for i ← ۰ to n do
{       j ← i
        while j ≠ ۰ do
          j ← j div ۲
}
```

۱) $\theta(n^۲)$ ۲) $\theta(n \log n)$ ۳) $\theta(n + \log n)$ ۴) $\theta(n)$

حل:

حلقه for، n بار تکرار می‌شود. حلقه while نیز $\log_۲^n$ بار. به این ترتیب می‌توان گفت مرتبه اجرایی الگوریتم $\theta(n \log_۲^n)$ خواهد بود.

نکته:

$O(a) < O(\log_۲^n) < O(n) < O(n \log_۲^n) < O(n^۲) < \ldots < O(n^a) < \ldots < O(۲^n) < \ldots < O(a^n) < O(n!) < O(n^n)$

مثال:

مرتبه بزرگی قطعه کد زیر چیست؟

```
k = ۰
for i := ۱ to n do begin
  for j = ۱ to m do
      k := k + ۱
  j = ۱
  while (j < n)do begin
        k := k + ۱
        j := ۲ * j
   end
end
```

۱) $O(n^۲)$ ۲) $O(nm)$ ۳) $O(mn + n^۲)$ ۴) $O(mn + n \log n)$

حل)

پاسخ گزینه: «۴»: در بدنه حلقه بیرونی، دو حلقه وجود دارد. یکی با مرتبه $O(m)$ و دیگری با مرتبه اجرایی $O(\log_2^n)$. حلقه خارجی نیز از مرتبه $O(n)$ است به این ترتیب داریم $O(mn + n\log_2^n)$.

- **تذکر:** اگر در مثال فوق حلقه for داخلی از $O(n)$ بود در آن صورت مرتبه بزرگی نهایی از $O(n^2)$ است.

مرتبه اجرایی توابع بازگشتی

تابع بازگشتی به تابعی گفته می‌شود که برای محاسبه مقدار $f(n)$. مجبور به فراخوانی $f(n-1)$ و $f(n-2)$ و... باشد. یک تابع بازگشتی علاوه بر یک رابطه بین $f(n)$ و $f(n-1)$ و... دارای یک نقطه بازگشت یا شرط مرزی نیز است.

مثال:

رابطه بازگشتی دنباله فیبوناچی

$$f(n)=\begin{cases} f(0)=1 \quad f(1)=1 \\ f(n)=f(n-1)+f(n-2) \quad n\geq 2 \end{cases}$$

رابطه بازگشتی فاکتوریل عدد n

$$f(n)=\begin{cases} f(1)=1 \\ f(n)=f(n-1)*n \end{cases}$$

رابطه بازگشتی برج‌های هانوی

$$f(n)=\begin{cases} f(0)=0 \\ f(n)=2f(n-1)+1 n>1 \end{cases}$$

با استفاده از روش‌های جایگذاری و تکراری می‌توان توابع بازگشتی را محاسبه کرد و مرتبه اجرایی آن‌ها را به دست آورد.

$$f(n)=\begin{cases} f_0 \\ f(n)=cf(n-1)+g(n) \end{cases} \Rightarrow f(n)=cf_0+\sum_{k=1}^{n} c^{n-k}g(k)$$

مثال:

مقدار تابع بازگشتی برج‌های هانوی را محاسبه کنید.

$$f(n)=2*0+\sum_{k=1}^{n} 2^{n-k}=\frac{2^n-1}{2-1}=2^n-1$$

به این ترتیب بدست آوردن مقدار nام دنباله فیبوناچی از مرتبه $\theta(2^n)$ خواهد بود.

نکته. اگر $F(n) = aF[\frac{n}{b}] + g(n)$ باشد. به طوری‌که $g(n)$ تابعی دلخواه برحسب n، $\frac{n}{b}$ می‌تواند به صورت کف $\left\lfloor \frac{n}{b} \right\rfloor$ و یا سقف $\left\lceil \frac{n}{b} \right\rceil$ باشد. آن‌گاه ۳ حالت زیر را داریم:

۱- اگر $g(n)$ از درجه $n^{\log_b^a}$ بیشتر باشد. $T(n) = O(g(n))$

۲- اگر $g(n)$ از درجه $n^{\log_b^a}$ کمتر باشد. $T(n) = O(n^{\log_b^a})$

۳- اگر درجه $g(n)$ با درجه $n^{\log_b^a}$ برابر باشد آن‌گاه $T(n) = O(g(n)\log_2^n)$

مثال:

با توجه به رابطه $T(n) = 4T(\frac{n}{2}) + n$ مرتبه اجرایی $T(N)$ را محاسبه کنید.

$$\left.\begin{array}{l} a = 4 \\ b = 2 \\ g(n) = n \end{array}\right\} \rightarrow \log_2^4 = 2$$

درجه $g(n)$ از n^2 کمتر است پس $T(n) = O(n^2)$ خواهد بود.

مثال:

اگر $T(n) = 2T(\frac{n}{3}) + n\log n$ باشد مرتبه اجرایی $T(n)$ را بدست آورید.

$$\left.\begin{array}{l} a = 2 \\ b = 3 \\ g(n) = n\log n \end{array}\right\}$$

$$\log_3^2 < \log_3^3 \Rightarrow n^{\log_3^2} < n^{\log_3^3} \leq n < n\log n$$

پس $T(n) = O(n\log n)$

مثال:

تابع بازگشتی زیر را در نظر بگیرید.

```
int     test(int n)
{       if (n ≤ ۲)
            return ۱;
        else
            return      test(n − ۲) * (n − ۲);
}
```

زمان اجرای تابع فوق را بدست آورید. **(دولتی۷۵)**

۱) $O(n^2)$ ۲) $O(n\log n)$ ۳) $O(2^{\frac{n}{2}})$ ۴) $O(2^n)$