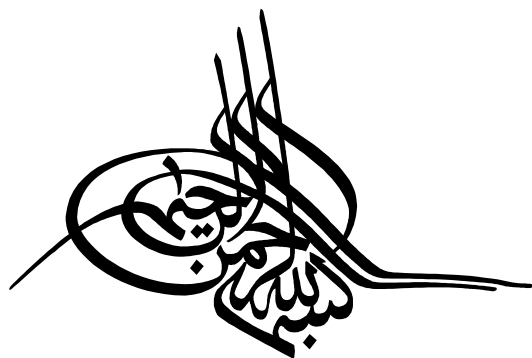




طراحی الگوریتم

سری کتاب‌های کمک آموزشی کارشناسی ارشد

مجموعه مهندسی کامپیوتر و فناوری اطلاعات
مؤلف: سمانه سرورثراد

سرشناسه	: سرور نژاد، سمانه، ۱۳۶۷
عنوان	: طراحی الگوریتم
مشخصات نشر	: تهران : مشاوران صعود ماهان، ۱۴۰۴
مشخصات ظاهری	: ۲۴۱ ص
فروست	: سری د
شابک	: ۹۷۸-۶۰۰-۴۵۸-۸۱۲-۶
وضعیت فهرست نویسی	: فیپای مختصر
یادداشت	: این مدرک در آدرس http://opac.nlai.ir قابل دسترسی است.
رده دیویی	: ۳۷۸/۱۶۶۴
شماره کتابشناسی ملی	: ۳۷۶۹۱۵۷



نام کتاب: طراحی الگوریتم

مؤلف: سمانه سرور نژاد

مدیر تولید محتوی: سمیه بیگی

ناشر: مشاوران صعود ماهان

نوبت و تاریخ چاپ: اول / ۱۴۰۴

تیراژ: ۱۰۰۰ نسخه

قیمت: ۴/۸۰۰/۰۰۰ ریال

شابک: ISBN: ۹۷۸-۶۰۰-۴۵۸-۸۱۲-۶

انتشارات مشاوران صعود ماهان: خیابان ولیعصر، بالاتر از تقاطع مطهری،

روبروی قنادی هتل بزرگ تهران، جنب بانک ملی، پلاک ۲۰۵۰

تلفن: ۴-۸۸۱۰۰۱۱۳

سخن ناشر

«ن والقلم و ما یسطرون»

کلمه نزد خدا بود و خدا آن را با قلم بر ما نازل کرد.

به پاس تشکر از چنین موهبت الهی، موسسه ماهان درصدد برآمده است تا در راستای انتقال دانش و مفاهیم با کمک اساتید مجرب و مجموعه کتب آموزشی خود برای شما داوطلبان ادامه تحصیل در مقطع کارشناسی ارشد گام موثری بردارد. امید است تلاش‌های خدمتگزاران شما در این موسسه پایه‌گذار گام‌های بلند فردای شما باشد. مجموعه کتاب‌های کمک آموزشی ماهان به‌منظور استفاده داوطلبان کنکور کارشناسی ارشد سراسری و آزاد تالیف شده‌اند. در این کتاب‌ها سعی کرده‌ایم با بهره‌گیری از تجربه اساتید بزرگ و کتب معتبر داوطلبان را از مطالعه کتاب‌های متعدد در هر درس بی‌نیاز کنیم.

دیگر تالیفات ماهان برای سایر دانشجویان به‌صورت ذیل می‌باشد.

● **مجموعه کتاب‌های ۸ آزمون:** شامل ۵ مرحله کنکور کارشناسی ارشد ۵ سال اخیر به همراه ۳ مرحله آزمون تالیفی ماهان همراه با پاسخ تشریحی می‌باشد که برای آشنایی با نمونه سوالات کنکور طراحی شده است. این مجموعه کتاب‌ها با توجه به تحلیل ۳ ساله اخیر کنکور و بودجه‌بندی مباحث در هریک از دروس، اطلاعات مناسبی جهت برنامه‌ریزی درسی در اختیار دانشجو قرار می‌دهد.

● **مجموعه کتاب‌های کوچک:** شامل کلیه نکات کاربردی در گرایش‌های مختلف کنکور کارشناسی ارشد می‌باشد که برای دانشجویان جهت جمع‌بندی مباحث در ۲ ماهه آخر قبل از کنکور مفید می‌باشد. بدین‌وسیله از مجموعه اساتید، مولفان و همکاران محترم خانواده بزرگ ماهان که در تولید و به‌روزرسانی تالیفات ماهان نقش موثری داشته‌اند، صمیمانه تقدیر و تشکر می‌نماییم. دانشجویان عزیز و اساتید محترم می‌توانند هرگونه انتقاد و پیشنهاد درخصوص تالیفات ماهان را از طریق سایت ماهان به آدرس mahan.ac.ir با ما در میان بگذارند.

موسسه آموزش عالی آزاد ماهان

سخن مولف

کتاب حاضر، پس از چندین بار ویرایش، برای ارائه به دوستان متقاضی کنکور کارشناسی ارشد رشته‌های حوزه کامپیوتر آماده گردیده است. این کتاب در ۸ فصل تنظیم و خدمت دانشجویان گرامی ارائه شده است. در برخی فصول از جمله فصل ۱، ۲، ۳ و ۶ مطالب مفید و مهمی ارائه شده که توصیه می‌شود با دقت و درایت بیشتری بررسی و خوانده شود. در پایان هر فصل نیز سعی شده سوالات مربوط به سرفصل‌های همان فصل گنجانده شود. البته در فصل یک سوالات بیشتری وجود دارد که به خاطر اهمیت آنها و ارتباطشان با فصل‌های پیش‌رو، صلاح دیده شده که در این فصل قرار گیرند. امیدوارم این کتاب بتواند شما را در راه رسیدن به اهدافتان یاری کند.

از کلیه عزیزانی که در گردآوری این کتاب زحمت کشیده‌اند سپاسگزارم.

سمانه سرورنژاد

فصل اول – الگوریتم‌ها و مرتبه اجرایی آنها و آشنایی با الگوریتم‌های بازگشتی	۷
نمادهای مجانبی	۸
چند جمله‌ای‌ها (Polynomials)	۱۱
الگوریتم‌های بازگشتی	۱۹
برج‌های هانوی	۱۹
مراحل فراخوانی یک زیربرنامه (CALL)	۲۱
روش‌های حل روابط بازگشتی	۲۳
نکات کلیدی فصل اول	۳۹
سوالات چهارگزینه‌ای و پاسخنامه تالیفی فصل اول	۴۴
سوالات چهارگزینه‌ای و پاسخنامه سراسری فصل اول	۴۶
سوالات چهارگزینه‌ای و پاسخنامه آزاد فصل اول	۵۳
فصل دوم – برنامه‌نویسی پویا	۵۷
روش برنامه نویسی پویا	۵۸
مسئله درخت جستجوی دوتایی بهینه یا اپتیمال (Optimal Binary Search tree)	۶۴
تفاوت روش حریصانه و برنامه نویسی پویا	۶۵
مسئله فروشنده دوره‌گرد (The Traveling Salesperson Problem)	۶۸
مسئله ضرب زنجیره‌ای بهینه ماتریس‌ها	۷۳
مثلث‌بندی بهینه یک چندضلعی محدب	۷۸
نکات کلیدی فصل دوم	۸۱
سوالات چهارگزینه‌ای و پاسخنامه سراسری فصل دوم	۸۴
سوالات چهارگزینه‌ای و پاسخنامه آزاد فصل دوم	۹۰
فصل سوم – روش حریصانه	۹۳
روش حریصانه	۹۴
مسئله انتخاب فعالیت‌ها (Activity-Selection)	۹۵
متد حریصانه (Greedy method)	۹۸
مسئله کوله‌پشتی کسری (Fractional Knapsack)	۱۰۱
مسئله الگوهای بهینه ادغام (Merge Optimal Pattern)	۱۰۴
کدگذاری هافمن (Huffman Coding)	۱۰۵
نکات کلیدی فصل سوم	۱۰۸
سوالات چهارگزینه‌ای و پاسخنامه سراسری فصل سوم	۱۱۰
سوالات چهارگزینه‌ای و پاسخنامه آزاد فصل سوم	۱۱۳
فصل چهارم – گراف و درخت	۱۱۵
روش‌های مختلف نمایش گراف	۱۱۶
الگوریتم جستجوی اول – سطح	۱۱۸
دسته‌بندی یال‌ها (Classification of edges)	۱۲۲
مرتب‌سازی توپولوژیکی (Topological Sort)	۱۲۳
مؤلفه‌های همبند قوی (Strongly Connected Components)	۱۲۴
تور اویلری (Eulerian Tour)	۱۲۷
درخت پوشای مینیمم (MST: Minimum Spanning Tree)	۱۲۷
الگوریتم‌های پریم و کروسکال	۱۳۰

۱۴۱	الگوریتم دایکسترا برای یافتن کوتاه‌ترین مسیر از مبدأ واحد
۱۴۲	الگوریتم Heap-Sort
۱۴۳	نکات کلیدی فصل چهارم
۱۴۶	سوالات چهارگزینه‌ای و پاسخنامه تالیفی فصل چهارم
۱۴۷	سوالات چهارگزینه‌ای و پاسخنامه سراسری فصل چهارم
۱۶۲	سوالات چهارگزینه‌ای و پاسخنامه آزاد فصل چهارم
۱۶۷	فصل پنجم – روش بازگشت به عقب (BT)
۱۶۸	مسأله n- وزیر (n-Queens)
۱۷۴	محدودیت‌های صریح (explicit)
۱۷۴	محدودیت‌های ضمنی (implicit)
۱۷۵	زمانبندی پردازنده
۱۸۱	جنگل‌های مجموعه‌های مجزا (Disjoint Set Forests)
۱۸۴	نکات کلیدی فصل پنجم
۱۸۵	سوالات چهارگزینه‌ای و پاسخنامه سراسری فصل پنجم
۱۸۶	سوالات چهارگزینه‌ای و پاسخنامه آزاد فصل پنجم
۱۸۷	فصل ششم – روش تقسیم و غلبه
۱۸۸	روش تقسیم و غلبه (Divide & Conquer)
۱۹۰	یافتن عنصر مینیمم و ماکسیمم یک آرایه
۱۹۲	الگوریتم ضرب ماتریس استراسن (Strassen)
۱۹۴	هندسه محاسباتی
۱۹۶	الگوریتم گراهام برای تولید پوشش محدب
۱۹۹	الگوریتم Quick Hull
۲۰۰	الگوریتم Merge Hull
۲۰۵	الگوریتم اقلیدس برای محاسبه gcd (بزرگ‌ترین مقسوم علیه مشترک)
۲۰۶	الگوریتم باینری یافتن ب.م.م (Binary gcd algorithm)
۲۰۶	الگوریتم توسعه یافته اقلیدس (Extended Euclid's algorithm)
۲۰۸	نکات کلیدی فصل ششم
۲۱۲	سوالات چهارگزینه‌ای و پاسخنامه سراسری فصل ششم
۲۱۸	سوالات چهارگزینه‌ای و پاسخنامه آزاد فصل ششم
۲۲۱	فصل هفتم – روش انشعاب و تحدید
۲۲۲	روش انشعاب و تحدید (Branch & Bound)
۲۲۴	طریقه محاسبه ارزش هر گره
۲۲۶	پیچیدگی محاسبات
۲۲۷	نکات کلیدی فصل هفتم
۲۲۸	سوالات چهارگزینه‌ای و پاسخنامه سراسری فصل هفتم
۲۲۸	سوالات چهارگزینه‌ای و پاسخنامه آزاد فصل هفتم
۲۲۹	فصل هشتم – آشنایی با نظریه NP
۲۳۰	مسائل بغرنج (Intractable)
۲۳۱	تفاوت بین مسائل بهینه با مسائل تصمیم‌گیری
۲۳۳	کلاس NP-hard
۲۳۳	مرتب کردن N عدد توسط الگوریتم پوستره محدب
۲۳۳	کلاس NP-Complete
۲۳۴	مسأله تا کردن خط‌کش
۲۳۴	مسأله افراز (PARTITION)
۲۳۵	نکات کلیدی فصل هشتم
۲۳۶	سوالات چهارگزینه‌ای و پاسخنامه سراسری فصل هشتم
۲۳۹	منبع

فصل اول

الگوریتم‌ها و مرتبه اجرایی آنها و آشنایی با الگوریتم‌های بازگشتی

- ◆ نمادهای مجانبی
- ◆ چند جمله‌ای‌ها
- ◆ الگوریتم‌های بازگشتی
- ◆ برج‌های هانوی
- ◆ مراحل فراخوانی یک زیر برنامه
- ◆ روش‌های حل روابط بازگشتی

الگوریتم‌ها و مرتبه اجرایی آنها و آشنایی با الگوریتم‌های بازگشتی

نمادهای مجانبی (Asymptotic Notation) O و Ω و θ

این نمادهای مجانبی به صورت زیر تعریف می‌شوند:

$$O(g(n)) = \{f(n) \mid \exists c > 0, n_0 \in \mathbb{N} \text{ s.t. } 0 \leq f(n) \leq cg(n) \text{ as } n \geq n_0\}$$

بنابراین هرگاه داشته باشیم $f(n) = O(g(n))$ منظور این است که $f(n) \in O(g(n))$.

$$\Omega(g(n)) = \{f(n) \mid \exists c > 0, n_0 \in \mathbb{N} \text{ s.t. } 0 \leq cg(n) \leq f(n) \text{ as } n \geq n_0\}$$

به همین ترتیب $f(n) = \Omega(g(n))$ یعنی اینکه $f(n) \in \Omega(g(n))$.

$$\theta(g(n)) = \{f(n) \mid \exists c_1, c_2 > 0, n_0 \in \mathbb{N} \text{ s.t. } 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n) \text{ as } n \geq n_0\}$$

به طور شهودی $f(n) = O(g(n))$ یعنی اینکه سرعت رشد f کمتر یا مساوی g است.

به طور شهودی $f(n) = \Omega(g(n))$ یعنی اینکه سرعت رشد f بیشتر یا مساوی g است.

به طور شهودی $f(n) = \theta(g(n))$ یعنی اینکه سرعت رشد f مساوی g است.

نکته:

$$f(n) = \theta(g(n)) \Leftrightarrow f(n) = O(g(n)) \text{ And } f(n) = \Omega(g(n))$$

$$\omega(g(n)) = \{f(n) \mid \forall c > 0, \exists n_0 \in \mathbb{N} \text{ s.t. } 0 \leq cg(n) < f(n) \forall n \geq n_0\}$$

نمادهای مجانبی ω ، o

این نمادها به صورت زیر تعریف می‌شوند:

$$o(g(n)) = \{f(n) \mid \forall c > 0, \exists n_0 \in \mathbb{N} \text{ s.t. } 0 \leq f(n) < cg(n) \forall n \geq n_0\}$$

یعنی مقدار تابع f برای n ها بزرگ در برابر g ناچیز و قابل صرف نظر کردن است.

نکته بسیار مهم: استفاده از حد برای تعیین مرتبه

$$f(n) = o(g(n)) \Leftrightarrow \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = 0$$

$$f(n) = \omega(g(n)) \Leftrightarrow \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = \infty$$

$$f(n) = \theta(g(n)) \Leftrightarrow \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = a \neq 0$$

با استفاده از حد می‌توان ثابت نمود که:

$$(1) \quad \text{if } \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = 0 \Leftrightarrow f(n) = O(g(n))$$

مثال: با توجه به رابطه (۱) داریم:

$$\frac{n^2}{2} \in O(n^3)$$

$$\lim_{n \rightarrow \infty} \frac{\frac{n^2}{2}}{n^3} = \lim_{n \rightarrow \infty} \frac{1}{2n} = 0$$

چون

$$(2) \quad \text{if } \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = \infty \Leftrightarrow f(n) = \omega(g(n))$$

مثال: با توجه به رابطه (۲) داریم:

$$300n^4 + 5n^2 \in 2n^5 + n$$

$$\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = \lim_{n \rightarrow \infty} \frac{2n^5 + n}{300n^4 + 5n^2} = \infty$$

چون

$$(3) \quad \text{if } \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = a \neq 0 \Leftrightarrow f(n) = \theta(g(n))$$

مثال: با توجه به رابطه (۳) داریم:

$$5n + 3 \log n + 10n \log n + 7n^2 \in \theta(n^2)$$

$$\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = \lim_{n \rightarrow \infty} \frac{5n + 3 \log n + 10n \log n + 7n^2}{n^2} = a \neq 0$$

چون

نکته: قاعده هوییتال، اگر $f(x)$ و $g(x)$ هر دو دیفرانسیل پذیر باشند و مشتقات آنها به ترتیب $f'(x)$ و $g'(x)$ باشد و اگر:

$$\lim_{n \rightarrow \infty} f(x) = \lim_{n \rightarrow \infty} g(x) = \infty$$

در آن صورت هرگاه حد سمت راست وجود داشته باشد:

$$\lim_{n \rightarrow \infty} \frac{f(x)}{g(x)} = \lim_{n \rightarrow \infty} \frac{f'(x)}{g'(x)}$$

برای توابعی با متغیرهای حقیقی نکته مذکور برقرار است، حال آنکه توابع پیچیدگی‌ها توابعی از اعداد صحیح هستند. ولی اکثر توابع پیچیدگی‌ها (برای مثال $\log n$ و n و غیره) توابع متغیرهای حقیقی نیز هستند. به علاوه، اگر تابع $f(x)$ تابعی از متغیر حقیقی x باشد، در آن صورت:

$$\lim_{n \rightarrow \infty} f(n) = \lim_{n \rightarrow \infty} f(x)$$

$$f(n) = o(g(n)) \Leftrightarrow g(n) = \omega(f(n))$$

نکته:

$$3n^2 - 4 = \omega(2n), \quad 2n = o(n^2 + 1), \quad 4n^3 - 7n^2 + 10n^2 - 100 = \theta(n^3)$$

مثال:

نکته: نمادهای مجانبی $\omega, o, \theta, \Omega, O$ برای مقایسه دو تابع به طور مجانبی (یعنی رفتار در بی‌نهایت) استفاده می‌شوند.

نکته: به جهت شباهتی که مقایسه مجانبی دو تابع با مقایسه دو مقدار عددی حقیقی وجود دارد می‌توان این شباهت‌ها را به فرم زیر نشان داد:

$$f(n) = O(g(n)) \approx a \leq b$$

$$f(n) = \Omega(g(n)) \approx a \geq b$$

$$f(n) = \theta(g(n)) \approx a = b$$

$$f(n) = o(g(n)) \approx a < b$$

$$f(n) = \omega(g(n)) \approx a > b$$

ویژگی‌های نمادهای مجانبی

۱- تعدی (Transitivity)

$$f(n) = \theta(g(n)) \text{ and } g(n) = \theta(h(n)) \Rightarrow f(n) = \theta(h(n))$$

$$f(n) = O(g(n)) \text{ and } g(n) = O(h(n)) \Rightarrow f(n) = O(h(n))$$

$$f(n) = \Omega(g(n)) \text{ and } g(n) = \Omega(h(n)) \Rightarrow f(n) = \Omega(h(n))$$

$$f(n) = o(g(n)) \text{ and } g(n) = o(h(n)) \Rightarrow f(n) = o(h(n))$$

$$f(n) = \omega(g(n)) \text{ and } g(n) = \omega(h(n)) \Rightarrow f(n) = \omega(h(n))$$

بنابراین تمامی عملگرهای مجانبی خاصیت تعدی را دارا می‌باشند.

۲- انعکاس (Reflexivity)

$$f(n) = \theta(f(n)), f(n) = O(f(n)), f(n) = \Omega(f(n))$$

نمادهای مجانبی Ω, θ, O خاصیت انعکاسی را دارا می‌باشند.

۳- تقارنی (Symmetry)

این خاصیت فقط برای نماد θ به صورت زیر برقرار است:

$$f(n) = \theta(g(n)) \Leftrightarrow g(n) = \theta(f(n))$$

نکته:

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

$$f(n) = o(g(n)) \Leftrightarrow g(n) = \omega(f(n))$$

نکته: اگر $f(n)$ و $g(n)$ توابع به طور مجانبی مثبتی باشند یعنی مقادیر این توابع به ازای n های به حد کافی بزرگ مثبت باشند آنگاه داریم:

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

نکته: اگر a و b عضوی از اعداد حقیقی باشند و $b > 0$ آنگاه داریم:

$$(n + a)^b = \theta(n^b)$$

$$o(g(n)) \cap \omega(g(n)) = \emptyset$$

نکته:

نکته: می‌توان نمادهای θ, Ω, O را به توابعی که دومتغیره هستند تعمیم داد. به عنوان مثال برای تابع دومتغیره $g(n, m)$ می‌توان $O(g(n, m))$ به صورت زیر تعریف می‌شود:

$$O(g(n, m)) = \{f(n, m) \mid \exists c > 0, n_0, m_0 \in \mathbb{N}, \text{ s.t. } 0 \leq f(n, m) \leq cg(n, m) \text{ as } n \geq n_0, m \geq m_0\}$$

به طور مشابه می‌توان این تعریف را برای θ و Ω نیز ارائه کرد.

تعاریف توابع نزولی صعودی به طور خلاصه به صورت زیر است:

$$\forall m, n; m \leq n \Rightarrow f(m) \leq f(n)$$

تابع یکنوا صعودی:

$$\forall m, n; m < n \Rightarrow f(m) < f(n)$$

تابع اکیدا صعودی:

$$\forall m, n; m \leq n \Rightarrow f(m) \geq f(n)$$

تابع یکنوا نزولی:

$$\forall m, n; m < n \Rightarrow f(m) > f(n)$$

تابع اکیدا نزولی:

توابع floor و ceiling به صورت زیر تعریف می‌شوند:

اگر x عددی حقیقی باشد آنگاه $\lceil x \rceil$ سقف (Ceiling) مقدار x نامیده می‌شود و برابر کوچکترین عدد صحیح بزرگتر از x می‌باشد. به طور مثال: $\lceil 2.5 \rceil = 3$

$\lfloor x \rfloor$ را کف (Floor) مقدار x می‌نامیم و برابر با بزرگترین عدد صحیح کوچکتر از x می‌باشد، مثلاً: $\lfloor 2.5 \rfloor = 2$

$$x - 1 < \lfloor x \rfloor \leq x \leq \lceil x \rceil < x + 1$$

نکته: برای هر عدد حقیقی x داریم:

$$\left\lfloor \frac{n}{2} \right\rfloor + \left\lceil \frac{n}{2} \right\rceil = n$$

نکته: برای هر عدد صحیح n داریم:

نکته: برای هر عدد حقیقی و مثبت مانند n و هر دو عدد صحیح و مثبت a و b داریم:

$$\left\lceil \left\lfloor \frac{n}{a} \right\rfloor / b \right\rceil = \lceil n / ab \rceil$$

$$\left\lfloor \left\lceil \frac{n}{a} \right\rceil / b \right\rfloor = \lfloor n / ab \rfloor$$

$$\left\lceil \frac{a}{b} \right\rceil \leq (a + (b - 1)) / b$$

$$\left\lfloor \frac{a}{b} \right\rfloor \geq ((a - (b - 1)) / b)$$

نکته: توابع floor ($\lfloor x \rfloor$) و ceiling ($\lceil x \rceil$) هر دو یکنوازی صعودی می‌باشند.

محاسبات مانده‌ای (Modular Arithmetic)

برای هر عدد صحیح a و هر عدد صحیح و مثبت n داریم:

$$a \bmod n = a - \left\lfloor \frac{a}{n} \right\rfloor \times n$$

(نماد $\equiv$) گوییم $a \equiv b \pmod{n}$ اگر و تنها اگر $a \bmod n = b \bmod n$ در این حالت گوییم a و b به پیمانه n هم‌ارز هستند.

چندجمله‌ای‌ها (Polynomials)

یک چندجمله‌ای برحسب n از درجه d عبارتست از تابعی به فرم $p(n) = \sum_{i=0}^d a_i n^i$ که در آن $a_d, \dots, a_1, a_0$ را ضرایب چندجمله‌ای نامند و $a_d \neq 0$.

ضرایب چندجمله‌ای اعدادی ثابت می‌باشند.

مثال: $p(n) = 10n^4 + 7n - 300$ یک چندجمله‌ای از درجه 4 می‌باشد.

نکته:

$$\text{if } p(n) = \sum_{i=0}^d a_i n^i \Rightarrow p(n) = \theta(n^d) \\ a_d \neq 0$$

تابع کران‌دار چندجمله‌ای: اگر تابع $f(n)$ به‌ازای برخی اعداد صحیح k به‌گونه‌ای باشد که $f(n) = O(n^k)$ گویند f کران‌دار چندجمله‌ای می‌باشد.

توان‌ها

اگر $n, m, a > 0$ هر سه عدد حقیقی باشند آنگاه داریم:

$$1) a^0 = 1$$

$$2) a^1 = a$$

$$3) a^{-1} = \frac{1}{a}$$

$$4) (a^m)^n = a^{mn}$$

$$5) (a^m)^n = (a^n)^m$$

$$6) a^m a^n = a^{m+n}$$

$$7) n^b = o(a^n) \Rightarrow \lim_{n \rightarrow \infty} \frac{n^b}{a^n} = 0 \quad ; \quad \forall n, a > 1, b \in \mathbb{R}$$

نکته: از آنجایی که به ازای هر $a > 1$ داریم $n^b = o(a^n)$ می توان نتیجه گرفت که هر تابع نمایی با یک پایه اکیداً بزرگتر از یک دارای رشدی سریع تر از هر تابع چند جمله ای می باشد.

نکته: عدد نپر e پایه ای برای لگاریتم نپر: قضایای مربوط به عدد نپر e به صورت زیر می باشند:

$$1) e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots = \sum_{i=0}^{\infty} \frac{x^i}{i!}$$

$$2) \forall x \in \mathbb{R} : e^x \geq 1 + x$$

$$3) \text{ if } |x| \leq 1 \Rightarrow 1 + x \leq e^x \leq 1 + x + x^2$$

$$4) \text{ if } |x| > 1 \Rightarrow e^x = 1 + x + \theta(x^2)$$

$$5) \forall x \in \mathbb{R} : \lim_{n \rightarrow \infty} (1 + \frac{x}{n})^n = e^x$$

نکته: اگر a و b و c اعداد حقیقی و مثبت باشند، آنگاه خواص زیر برای لگاریتم ها صدق می کند:

$$1) a = b^{\log_b a}$$

$$2) \log_c(ab) = \log_c a + \log_c b$$

$$3) \log_b a = \frac{\log_c a}{\log_c b}$$

$$4) \log_b \left(\frac{1}{a}\right) = -\log_b a$$

$$5) a^{\log_b c} = c^{\log_b a}$$

$$6) \log_b a = \frac{1}{\log_a b}$$

نکته: برای هر ثابت $a > 0$ و هر b دلخواه داریم: $\log^b n = o(n^a)$

✓ تست: اگر $T(n) = \log(n!)$ آنگاه:

$$T(n) = \Omega(n \log n) \quad (۲)$$

$$T(n) = O(n \log n) \quad (۱)$$

$$T(n) = \theta(n \log^2 n) \quad (۴)$$

$$T(n) = \theta(n \log n) \quad (۳)$$

✓ حل: گزینه «۳»

روش اول: تقریب استرلینگ یک حد قوی برای فاکتوریل به صورت زیر می باشد:

$$n! = \sqrt{2\pi n} \left(\frac{n}{e}\right)^n e^{a_n} \quad \frac{1}{12n+1} < a_n < \frac{1}{12n}$$

$$\log(n!) \approx \log\left(\frac{n^n}{e^n}\right) = \log n^n - \log e^n = n \log n - n \log e = \theta(n \log n)$$

روش دوم: راه دیگر، یافتن کران های پایین و بالا می باشد.

$$\log(n!) = \log(n * (n-1) * (n-2) * \dots * 2 * 1) = \log n + \log(n-1) + \log(n-2) + \dots + \log(2) + \log(1)$$

$$= \sum_{i=1}^n \log i \leq \sum_{i=1}^n \log n = n \log n = O(n \log n)$$

$$\sum_{i=1}^n \log i \geq \sum_{i=\frac{n}{2}+1}^n \log\left(\frac{n}{2} + 1\right) = \frac{n}{2} \log\left(\frac{n}{2} + 1\right) \geq \frac{n}{2} \log\left(\frac{n}{2}\right) \geq \frac{n}{2} \log(\sqrt{n}) \quad (\text{for } 4 \leq n) = \frac{n}{2} \log(n^{\frac{1}{2}}) = \frac{n}{4} \log(n)$$

$$\Rightarrow \log(n!) = \Omega(n \log n)$$

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

✓ تست: اگر $T(n) = \sum_{i=1}^{\log n} i 2^i$ ؛ آنگاه:

$$T(n) = \Omega(n \log n) \quad (۲)$$

$$T(n) = q(n \log^2 n) \quad (۴)$$

$$T(n) = O(n \log n) \quad (۱)$$

$$T(n) = \theta(n \log n) \quad (۳)$$

✓ حل: گزینه «۳»

روش اول: می‌دانیم که:

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

بنابراین:

$$\sum_{i=1}^{\log n} i 2^i = (\log n - 1)2^{\log n + 1} + 2 = (\log n - 1)2^{\log n} * 2^1 + 2 = 2(\log n - 1)n + 2 = 2(n \log n - n + 1)$$

$$\Rightarrow \sum_{i=1}^{\log n} i 2^i = \theta(n \log n)$$

روش دوم: راه دیگر، یافتن کران‌های پایین و بالا می‌باشد.

$$\sum_{i=1}^{\log n} i 2^i \leq \sum_{i=1}^{\log n} \log n 2^i \leq \log n \sum_{i=1}^{\log n} 2^i = \log n (2^{\log n + 1} - 2) = \log n (2 * 2^{\log n} - 2) = \log n (2n - 2) \Rightarrow \sum_{i=1}^{\log n} i 2^i = O(n \log n)$$

$$\sum_{i=1}^{\log n} i 2^i \geq \sum_{i=\frac{\log n}{2}+1}^{\log n} (\frac{\log n}{2} + 1) 2^i \geq \sum_{i=\frac{\log n}{2}+1}^{\log n} (\frac{\log n}{2}) 2^i = (\frac{\log n}{2}) \sum_{i=\frac{\log n}{2}+1}^{\log n} 2^i = (\frac{\log n}{2}) (\sum_{i=0}^{\log n} 2^i - \sum_{i=0}^{\frac{\log n}{2}} 2^i)$$

$$= (\frac{\log n}{2}) (2^{\log n + 1} - 1 - [2^{\frac{\log n}{2} + 1} - 1])$$

$$= (\frac{\log n}{2}) (2^{\log n + 1} - 2^{\frac{\log n}{2} + 1}) = (\frac{\log n}{2}) (2 * 2^{\log n} - 2 * 2^{\frac{\log n}{2}}) = \log n (2^{\log n} - 2^{\frac{\log n}{2}}) = \log n (n - \sqrt{n}) = \Omega(n \log n)$$

$$\Rightarrow \sum_{i=1}^{\log n} i 2^i = \theta(n \log n)$$

✓ تست: اگر $T(n) = \sum_{i=1}^n \frac{1}{i}$ ؛ آنگاه:

$$T(n) = \Omega(\log n) \quad (۲)$$

$$T(n) = \theta(\log^2 n) \quad (۴)$$

$$T(n) = O(\log n) \quad (۱)$$

$$T(n) = \theta(\log n) \quad (۳)$$

✓ حل: گزینه «۳»

روش اول: می‌دانیم که:

$$1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n} \approx \ln n$$

بنابراین داریم:

$$T(n) = \sum_{i=1}^n \frac{1}{i} \approx \ln n \Rightarrow T(n) = \theta(\ln n) = \theta(\log n)$$

روش دوم: برای تابع نزولی یکنواخت $F(i)$ به صورت زیر محدود می‌شود:

$$\int_m^{n+1} f(x) dx \leq \sum_{i=m}^n f(i) \leq \int_{m-1}^n f(x) dx$$

از طرفی:

$$\sum_{i=1}^n \frac{1}{i} \geq \int_1^{n+1} \frac{dx}{x} = \ln(n+1) - \ln 1 = \ln(n+1) \quad , \quad \sum_{i=1}^n \frac{1}{i} = 1 + \sum_{i=2}^n \frac{1}{i} \leq 1 + \int_1^n \frac{dx}{x} = 1 + \ln(n)$$

$$\sum_{i=1}^n \frac{1}{i} = \theta(\ln n) = \theta(\log n)$$

✓ تست: اگر $T(n) = \sum_{i=1}^n \log^k i$ آنگاه:

$$T(n) = \theta(n \log^k (k+n)) \quad (۲)$$

$$T(n) = \theta(n \log^k n) \quad (۱)$$

$$T(n) = \theta(n^k) \quad (۴)$$

$$T(n) = \theta(\log^k n) \quad (۳)$$

که حل: گزینه «۱»

می دانیم که:

$$\int_1^m \log^k(x) dx = \sum_{i=0}^k m \log^{k-i}(m) \frac{k!}{(k-i)!} (-1)^i$$

از آنجا که $\log^k(i)$ تابع صعودی و یکنواخت است؛ پس بنابر عکس مطلب تست قبل داریم:

$$\sum_{i=1}^n \log^k i \leq \int_1^{n+1} \log^k(x) dx = \sum_{i=0}^k (n+1) \log^{k-i}(n+1) \frac{k!}{(k-i)!} (-1)^i = O(n \log^k(n))$$

$$\sum_{i=1}^n \log^k i = \log^k 1 + \sum_{i=2}^n \log^k i \geq \int_1^n \log^k i = \sum_{i=0}^k (n) \log^{k-i}(n) \frac{k!}{(k-i)!} (-1)^i = \Omega(n \log^k(n))$$

بنابراین داریم:

$$T(n) = \sum_{i=1}^n \log^k i = \theta(n \log^k n)$$

● مثال: اگر $f_1(n) = O(g_1(n))$ و $f_2(n) = O(g_2(n))$ آنگاه درستی یا نادرستی رابطه زیر را بررسی نمایید.

$$f_1(n)^{f_2(n)} = O(g_1(n)^{g_2(n)})$$

که حل: رابطه فوق نادرست است. به عنوان مثال نقض داریم:

$$g_1(n) = 0.5, \quad g_2(n) = n, \quad f_1(n) = 2, \quad f_2(n) = n$$

در این صورت داریم:

$$2^n \neq O((0.5)^n)$$

● مثال: درستی یا نادرستی گزاره زیر را بررسی نمایید.

"بهازای هر دو تابع دلخواه f و g یکی از روابط $f(n) = \Omega(g(n))$ or $f(n) = O(g(n))$ or $f(n) = \theta(g(n))$ برقرار است."

که حل: گزاره فوق نادرست است. به عنوان مثال نقض داریم:

$$f(n) = n \quad \text{و} \quad g(n) = n^{1+\sin n}$$

● مثال: توابع زیر را بر حسب رشد مرتب نمایید.

$(\sqrt{2})^{\log n}$	n^2	$n!$	e^n	2^n
n^3	$\log(n!)$	$n^{\frac{1}{\log n}}$	n	$n^{\log \log n}$
$\ln n$	$\ln \ln n$	$3^{\log n}$	$(\log n)^{\log n}$	$n \log n$
$(\log n)!$	$(2^2)^n$	$n 2^n$	$2^{\log n}$	$2^{\sqrt{2 \log n}}$
$(2^2)^{n+1}$	$\sqrt{\log n}$	$(n+1)!$	$4^{\log n}$	$(\frac{3}{2})^n$

که حل:

نکته: $a^{\log_b c} = c^{\log_b a}$ در نتیجه:

$$(\log n)^{\log n} = n^{\log \log n}$$

$$4^{\log n} = n^2, \quad 2^{\log n} = n$$

$$n^{\frac{1}{\log n}} = n^{\frac{\log 2}{\log n}} = (n^{\log 2})^{\frac{1}{\log n}} = (2^{\log n})^{\frac{1}{\log n}} = 2$$

$$(\sqrt{2})^{\log n} = n^{\log \sqrt{2}} = n^{\log 2^{\frac{1}{2}}} = n^{\frac{1}{2} \log 2} = n^{\frac{1}{2}} = \sqrt{n}$$

$$e^n = (2.71)^n$$

$$\log(\sqrt{2})^{\log n} = \log(2^{\frac{1}{2} \log n}) = \frac{1}{2} \log n * \log 2 = \frac{1}{2} \log n$$

$$\log 2^{\sqrt{2} \log n} = \sqrt{2} \log n$$

$$\frac{1}{2} \log n = \omega \sqrt{2} \log n$$

$$\Rightarrow (\sqrt{2})^{\log n} = \omega(2^{\sqrt{2} \log n})$$

$$\log \log^2 n = \log(\log n)^2 = 2 \log(\log n)$$

$$\log 2^{\sqrt{2} \log n} = \sqrt{2} \log n$$

$$2^{\sqrt{2} \log n} \geq \log^2 n \Rightarrow \log 2^{\sqrt{2} \log n} \geq \log \log^2 n$$

$$\sqrt{2} \log n = \omega(2 \log(\log n))$$

$$\Rightarrow 2^{\sqrt{2} \log n} = \omega(2^{2 \log(\log n)}) = \omega((\log n)^{2 \log 2}) = \omega(\log^2 n)$$

$$n! = \theta\left(\frac{n^n}{e^n}\right)$$

$$(\ln n)! = \theta\left(\frac{(\ln n)^{\ln n}}{e^{\ln n}}\right) = \theta\left(\frac{\ln^{\ln n} n}{n}\right)$$

با توجه به توضیحات فوق ترتیب توابع داده شده به صورت زیر می باشد:

$$(2^2)^{n+1} > (2^2)^n > (n+1)! > n! > e^n > n2^n > 2^n > \left(\frac{3}{2}\right)^n > n^{\log \log n} = (\log n)^{\log n} > \log(n!) > n^3 > 4^{\log n} \\ > \log(n!) = n \log n > 3^{\log n} > 2^{\log n} = n > (\sqrt{2})^{\log n} = \sqrt{n} > 2^{\sqrt{2} \log n} > \ln n > \sqrt{\log n} > \ln \ln n > n^{\frac{1}{\log n}}$$

نکته: نمادهای مجانبی تابع $n! = 1 \times 2 \times \dots \times n$ به صورت زیر می باشد:

$$1) n! = o(n^n)$$

$$2) n! = \omega(2^n)$$

$$3) \lg n! = \theta(n \lg n)$$

$$4) n! = \sqrt{2\pi n} \left(\frac{n}{e}\right)^n \left(1 + \theta\left(\frac{1}{n}\right)\right)$$

$$5) n! = \sqrt{2\pi n} \left(\frac{n}{e}\right)^n e^{\alpha_n}, \quad \frac{1}{12n+1} < \alpha_n < \frac{1}{12n}$$

فرمول استرلینگ:

نکته: اگر $f(n)$ و $g(n)$ توابع یکنوای صعودی باشند؛ آنگاه توابع $f(n) + g(n)$ و $f(g(n))$ نیز چنین هستند.
نکته: اگر $f(n)$ و $g(n)$ توابع یکنوای صعودی و غیرمنفی باشند؛ آنگاه تابع $f(n).g(n)$ نیز یکنوای صعودی می باشد.

ویژگی‌های نماد جمع‌بندی سیگما

- 1) $\sum (ca_k + b_k) = c \sum a_k + \sum b_k$
- 2) $\sum \theta(f(k)) = \theta(\sum f(k))$
- 3) $\sum_{k=1}^n k = \frac{n}{2}(1+n) = \theta(n^2)$
- 4) $\sum_{k=0}^n k^2 = \frac{n(n+1)(2n+1)}{6} = \theta(n^3)$
- 5) $\sum_{k=0}^n k^3 = \frac{n^2(n+1)^2}{4} = \theta(n^4)$
- 6) $\sum_{k=0}^n x^k = \frac{x^{n+1} - 1}{x - 1}$ (Geometric Series)
- 7) if $|x| < 1 \Rightarrow \sum_{k=0}^{\infty} x^k = \frac{1}{1-x}$
- 8) $H_n = 1 + \frac{1}{2} + \dots + \frac{1}{n} = \sum_{k=1}^n \frac{1}{k} = \ln n + o(1)$ (Harmonic Series)
- 9) if $|x| < 1 \Rightarrow \sum_{k=0}^{\infty} kx^k = \frac{x}{(1-x)^2}$ (Integration & Differentiating Series)
- 10) $\sum_{k=1}^n (a_k - a_{k-1}) = a_n - a_0$ (Telescoping Series)
- 11) $\sum_{k=0}^{n-1} (a_k - a_{k+1}) = a_0 - a_n$ (Telescoping Series)
- 12) $\sum_{k=1}^{n-1} \frac{1}{k(k+1)} = \sum_{k=1}^{n-1} \left(\frac{1}{k} - \frac{1}{k+1} \right) = 1 - \frac{1}{n}$

✓ تست: کدام یک از گزینه‌های زیر صحیح است؟

$$f(n) = O((f(n))^2) \quad (۲)$$

(۴) هیچ‌کدام

$$f(n) + g(n) = O(\min(f(n), g(n))) \quad (۱)$$

$$f(n) + O(f(n)) = O((f(n))) \quad (۳)$$

✓ حل: گزینه «۳»

گزینه ۱ نادرست است. به عنوان مثال نقض $f(n) = n$ و $g(n) = n^2$ را در نظر بگیرید.

گزینه ۲ نادرست است. به عنوان مثال نقض $f(n) = 1/n$ را در نظر بگیرید.

گزینه ۳ صحیح است. زیرا:

$$\begin{aligned} g(n) = O(f(n)) &\Rightarrow 0 \leq g(n) < cf(n) \\ &\Rightarrow f(n) \leq f(n) + g(n) < (c+1)f(n) \\ &\Rightarrow f(n) \leq f(n) + O(f(n)) < (c+1)f(n) \\ &\Rightarrow f(n) + O(f(n)) \in f(n) \end{aligned}$$

● مثال: فرض کنید $f_i(n)$ دنباله‌ای از توابع باشد به طوری که برای هر i داشته باشیم: $f_i(n) = O(n)$. اگر $g(n) = \sum_{i=1}^n f_i(n)$

آیا $g(n) = O(n^2)$ است یا خیر؟

✓ حل: خیر. به عنوان مثال نقض فرض کنید $f_i(n) = i * n$ ، در این صورت بدیهی است که برای هر i داریم: $f_i(n) = O(n)$ اما:

$$g(n) = \sum_{i=1}^n i * n = n \sum_{i=1}^n i = n \left(\frac{n(n+1)}{2} \right) = O(n^3)$$

نکته: معادلات زمانی الگوریتم‌های بازگشتی، توابعی بازگشتی می‌باشند. به عنوان مثال تابع بازگشتی زیر را در نظر بگیرید:

```
int P(int *A, int n){
    if (n<0)
        return 0
    else
        return (A[n]+P(A,n-1));
}
```

مرتبه زمانی $T(n)$ این رویه بر حسب n به صورت زیر می‌باشد:

$$T(n) = \begin{cases} o(1) & n < 0 \\ T(n-1) + o(1) \end{cases}$$

بنابراین $T(n) = \theta(n)$.

نکته بسیار مهم:

$$1) \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = a \in \mathbb{R}^+ \Rightarrow f(n) = \theta(g(n))$$

$$2) \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = 0 \Rightarrow f(n) = O(g(n))$$

$$3) \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = \infty \Rightarrow f(n) = \Omega(g(n))$$

اما توجه داشته باشید که عکس این قضیه صحیح نمی‌باشد، یعنی ممکن است $f(n) = O(g(n))$ باشد بدون اینکه حد موردنظر $(\lim_{n \rightarrow \infty} f(n)/g(n))$ موجود باشد.

برای به دست آوردن حد توابع می‌توان از قاعده هسپیتال استفاده کرد.

مثال: $\log n \in O(\sqrt{n})$ ، زیرا داریم:

$$\lim_{n \rightarrow \infty} \frac{\log_2 n}{\sqrt{n}} \stackrel{\text{hop}}{=} \lim_{n \rightarrow \infty} \frac{\frac{1}{n}}{\frac{1}{2}n^{\frac{1}{2}}} = \frac{1}{2} \lim_{n \rightarrow \infty} \frac{1}{n^{\frac{3}{2}}} = 0$$

اما $\sqrt{n} \notin O(\log n)$ ، زیرا عکس این حد وجود ندارد.

تحلیل الگوریتم مرتب‌سازی حبابی

ورودی: آرایه نامرتب $A[1..n]$.

خروجی: این آرایه مرتب‌شده به صورت صعودی.

Algorithm Bubble Sort (A, n)

```
1 for i=1 to n-1 do
2   for j=n down to i+1 do
3     if A[j-1] > A[j] then
4       {
5         temp = A[j-1]
6         A[j-1] = A[j]
7         A[j] = temp
8       }
9   }
```

بررسی بدترین حالت: بدترین حالت زمانی اتفاق می افتد که فرمان خط ۳ همواره درست باشد. این مساله عملاً زمانی اتفاق می افتد که خود آرایه $A[1..n]$ به صورت نزولی مرتب شده باشد، در این صورت داریم:

شماره دستورالعمل	تعداد دفعات اجرا
1	$n - 1$
2	$(n - 1) + (n - 2) + \dots + 2 + 1$
3	$(n - 1) + (n - 2) + \dots + 2 + 1$
4	$(n - 1) + (n - 2) + \dots + 2 + 1$
5	$(n - 1) + (n - 2) + \dots + 2 + 1$
6	$(n - 1) + (n - 2) + \dots + 2 + 1$

$$t(n) = n - 1 + 5 \frac{n(n - 1)}{2} \quad \leftarrow \text{جمع}$$

بنابراین $t(n) = O(n^2)$

بررسی بهترین حالت: بهترین حالت زمانی اتفاق می افتد که دستور خط ۳ هیچ وقت اجرا نشود و این زمانی اتفاق می افتد که خود آرایه به صورت صعودی مرتب شده باشد، در این حالت داریم:

شماره دستورالعمل	تعداد دفعات اجرا
1	$n - 1$
2	$(n - 1) + (n - 2) + \dots + 2 + 1$
3	$(n - 1) + (n - 2) + \dots + 2 + 1$
4	0
5	0
6	0

$$t(n) = n - 1 + 2 \frac{n(n - 1)}{2} = \Omega(n^2) \quad \leftarrow \text{جمع}$$

لذا نتیجه می شود که $t(n) = \Omega(n^2)$

تحلیل الگوریتم مرتب سازی درجی

ورودی: آرایه نامرتب $A[1..n]$

خروجی: این آرایه مرتب شده به صورت صعودی.

Algorithm Insertion sort (A,n)

$A[0] = -\infty$ / برای سهولت /

for $j = 2$ to n do

{

Item = $A[j]$, $i = j - 1$

While $\boxed{\text{item} < A[i]}$ do

{

$A[i+1] = A[i]$, $i = i - 1$

}

$A[i+1] = \text{item}$

}

برای سهولت در تحلیل این الگوریتم، فقط تعداد دفعات اجرای دستورالعمل مقایسه ای را حساب می کنیم.

بررسی بدترین حالت: بدترین حالت زمانی اتفاق می‌افتد که دستور شرطی همواره درست باشد و این مساله عملاً زمانی اتفاق می‌افتد که آرایه $A[1..n]$ به صورت نزولی مرتب شده باشد.

تعداد دفعات اجرای دستورالعمل	مقدار پارامتر j
2	1
3	2
⋮	⋮
⋮	⋮
n	n

$$t(n) = 2 + 3 + \dots + n = \frac{n(n+1)}{2} - 1 = O(n^2)$$

بررسی بهترین حالت: بهترین حالت زمانی اتفاق می‌افتد که حلقه داخلی هیچ وقت اجرا نشود (آرایه به صورت صعودی مرتب شده باشد)

تعداد دفعات اجرای دستورالعمل	مقدار پارامتر j
1	2
1	3
1	⋮
⋮	⋮
n	n

$$t(n) = 1 + 1 + \dots + 1 = n - 1 = \Omega(n)$$

با توجه به مطالب فوق نتیجه می‌شود که میانگین زمان اجرای الگوریتم مرتب‌سازی درجی $O(n)$ ، $O(n \log n)$ یا $O(n^2)$ است که اثبات می‌شود برابر $\theta(n^2)$ است.

الگوریتم‌های بازگشتی

به طور کلی اگر یک الگوریتم در بدنه خود، خود را فراخوانی نماید، الگوریتم را بازگشتی می‌نامند.

برج‌های هانوی

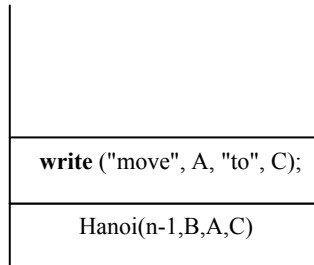
برنامه بازگشتی برج هانوی به صورت زیر می‌باشد:

```

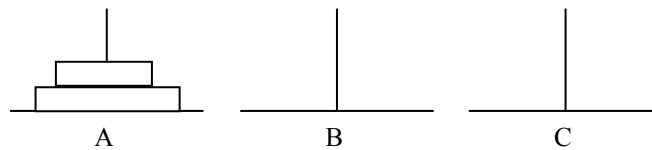
Procedure Hanoi (n,A,B,C)
Begin
    If n = 1 then write ("move", A, "to", C);
    Else Begin
        Hanoi(n-1,A,C,B);
        write ("move", A, "to", C);
        Hanoi(n-1,B,A,C)
    End;
End;

```

در مسأله برج هانوی هدف این است که n دیسک را از میله A به کمک میله B به میله C منتقل کنیم. با این شرط که در هر بار تنها می‌توان یک دیسک را منتقل نمود و دیسک بزرگتر نمی‌تواند روی دیسک کوچک‌تر قرار گیرد. به‌عنوان مثال با فراخوانی $\text{hanoi}(2, A, B, C)$ می‌توان دو مهره را از میله A به میله C به‌صورت زیر منتقل کرد. با توجه به آنکه مقدار n برابر ۲ است اجرای برنامه وارد قسمت Else می‌شود و بایستی سه دستور آن قسمت را اجرا کنیم. از آنجایی که دستور $\text{Hanoi}(1, A, C, B)$ به‌صورت بازگشتی رویه را فراخوانی می‌کند، بنابراین دو دستور بعد از آن به‌صورت زیر وارد پشته سیستم می‌شود.

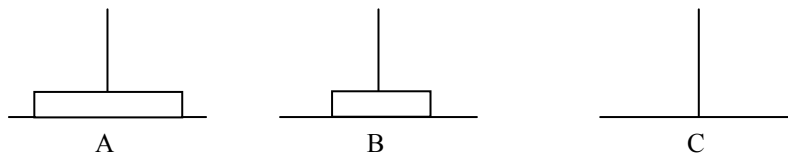


با اجرای دستور $\text{Hanoi}(1, A, C, B)$ از آنجایی که n برابر یک می‌باشد عبارت `move A to B` چاپ می‌شود. حال به سراغ پشته سیستم می‌رویم و دستور `write ("move", A, "to", C);` اجرا می‌شود که حاصل آن نوشته شدن عبارت `move A to C` می‌باشد. حال مجدداً به سراغ پشته سیستم می‌رویم و دستور $\text{Hanoi}(n-1, B, A, C)$ را اجرا می‌کنیم. از آنجایی که مقدار n برابر یک می‌باشد با اجرا شدن این دستور عبارت `move B to C` چاپ می‌شود.

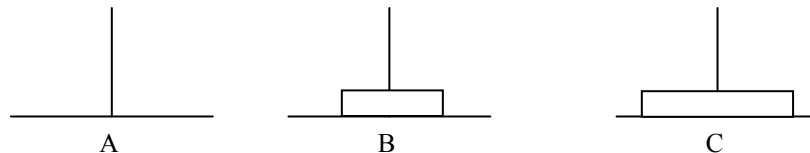


بنابراین به ترتیب عبارت‌های زیر چاپ می‌شوند:

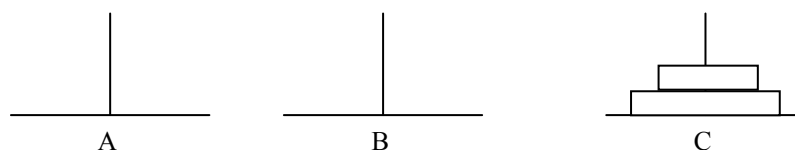
`move A to B`



`move A to C`

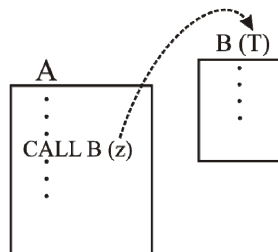


`move B to C`



مراحل فراخوانی یک زیربرنامه (CALL)

- در هر فراخوانی یک زیربرنامه، مراحل زیر انجام می‌شود:
- انتقال متغیرهای محلی، پارامترها و آدرس بازگشت داخل حافظه پشته (stack). منظور از آدرس بازگشت (return address) آدرس اولین دستور بعد از CALL می‌باشد. متغیرهای محلی (local variables) متغیرهایی هستند که داخل یک بلوک خاص تعریف شده‌اند و هنگامی که کنترل اجرای یک برنامه وارد یک بلوک گردید، حافظه به آنها تخصیص می‌یابد. حافظه stack از محل سگمنت پشته (stack segment) به یک برنامه یا رویه بازگشتی اختصاص می‌یابد. پارامترها همان مقداری است که در آرگومان‌های رویه قرار می‌گیرد. در هر CALL علاوه بر مقادیر ذکرشده در بالا برخی مقادیر سیستمی مانند ثبات Flags و... داخل پشته قرار می‌گیرند. در هر CALL یک قاب پشته‌ای یا stack-frame حاوی مقادیر ذکرشده در بالا در stack قرار می‌گیرد.
 - جانشینی پارامترها (parameter substitution)، انتقال نظیر به نظیر مقادیر پارامترها به داخل آرگومان‌ها، به این مرحله اصطلاحاً parameter passing گویند.
 - رجوع به آدرس رویه فراخوانده‌شده



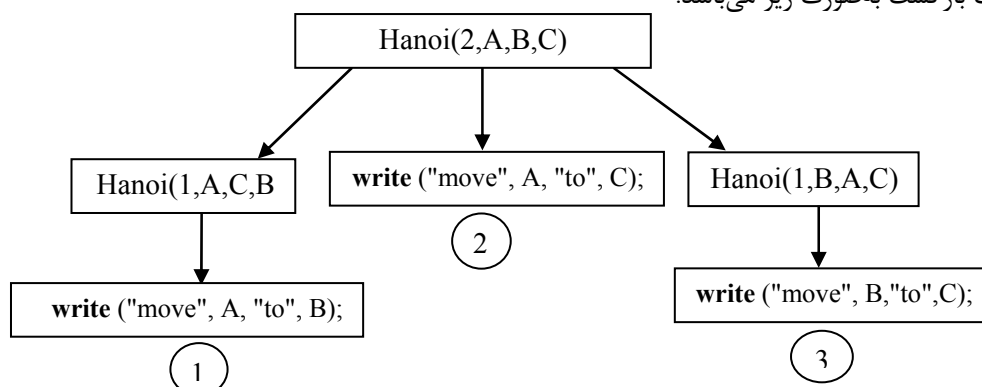
عملکرد دستور Return

- در هر رویه باید یک یا چند دستور return به منظور بازگشت از رویه به برنامه فراخواننده (caller) وجود داشته باشد که این دستور باعث می‌شود که:
- مقادیر پارامترها و متغیرهای محلی و غیره از stack بازیابی شوند.
 - رجوع به آدرس بازگشت (return address) انجام شود.

درخت فراخوانی‌های بازگشتی (Recursive Call Tree)

- توسط این درخت که هر رأس آن نشان‌دهنده یک CALL می‌باشد می‌توان عملکرد یک رویه بازگشتی را بررسی کرد. تعداد فرزندان هر گره می‌تواند نشان‌دهنده تعداد آدرس‌های بازگشت در هر CALL باشد، یعنی اینکه مثلاً اگر در یک رویه تعداد فراخوانی‌های بازگشتی در هر مرحله دو تا باشد، آنگاه درخت فراخوانی‌های بازگشتی یک درخت دودویی خواهد بود.
- مثال: درخت فراخوانی را برای $Hanoi(2, A, B, C)$ رسم کنید.

حل: درخت بازگشت به صورت زیر می‌باشد:



تحلیل زمانی الگوریتم هانوی

از آنجایی که در الگوریتم هانوی به ازای اندازه ورودی n دو بار خود را با مقدار $n-1$ به صورت بازگشتی فراخوانی می کند و یک جابجایی انجام می دهد، می توان پیچیدگی زمانی آن را به صورت تابع بازگشتی زیر نوشت:

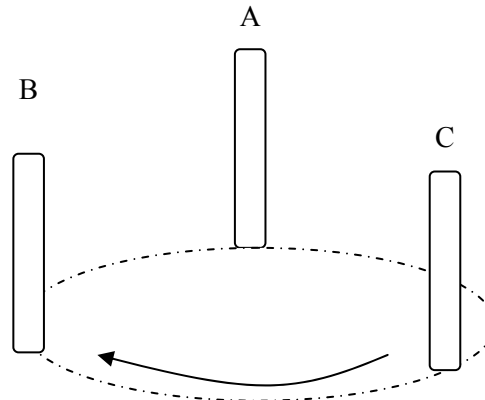
$$T(n) = \begin{cases} \theta(1) & n = 1 \\ 2T(n-1) + \theta(1) & n > 1 \end{cases}$$

اگر $n=1$ آنگاه زمان الگوریتم مقداری ثابت و از مرتبه $\theta(1)$ است اما اگر $n > 1$ آنگاه الگوریتم دارای دو فراخوانی بازگشتی با $n-1$ می باشد، که زمان هر کدام $T(n-1)$ می باشد و زمان ثابت $\theta(1)$ برای مقایسه n با 1 و عمل Move و عمل return صرف می شود.

الگوریتم غیر بازگشتی برج هانوی

اگر n (تعداد دیسک) زوج باشد جهت حرکت را ساعتگرد و در غیر این صورت پادساعتگرد در نظر می گیریم. حال مراحل زیر را تا هنگامی که همه دیسک ها در میله مقصد قرار بگیرد تکرار می کنیم.

- کوچکترین دیسک را به میله بعدی در جهت حرکت منتقل کنید.
- در صورت امکان تنها حرکتی را که شامل کوچکترین دیسک نباشد انجام بدهید.



نکته: تعداد جابجایی این الگوریتم برابر تعداد جابجایی های روش بازگشتی می باشد. برای اثبات داریم:

پایه استقرا: $n=1$: روشن است و این کار با یک حرکت انجام می شود.

فرض استقرا: برای $n-1$ دیسک درست کار می کند. یعنی $n-1$ دیسک به صورت مستقل و با $2^{n-1}-1$ حرکت از میله 1 به میله بعدی در جهت ساعتگرد (برای $n-1$ زوج) یا پادساعتگرد (برای $n-1$ فرد) به درستی و مطابق مقررات مسأله منتقل می کند.

گام استقرا: فرض کنید n زوج است. این الگوریتم مطابق فرض استقرا پس از $2^{n-1}-1$ حرکت $n-1$ دیسک بالای میله را به میله بعدی در جهت ساعتگرد (یعنی میله ۲) منتقل می کند. در این حرکت ها بزرگترین دیسک نقشی ندارد و در انتها این دیسک در میله ۱ باقی می ماند. سپس تنها حرکت ممکن را انجام می دهیم که بزرگترین دیسک را از میله 1 به میله 3 می بریم سپس در جهت حرکت که ساعتگرد است کاری که الگوریتم به صورت استقرایی انجام می دهد. حرکت $n-1$ دیسک موجود در میله ۲ به میله بعدی (میله ۳) است و این کار با فرض استقرا پس از $2^{n-1}-1$ حرکت دیگر به درستی انجام می شود. مشابه این استدلال برای اعداد فرد هم برقرار است. جمع کل $2^{n-1}-1 = 1 + 2(2^{n-1}-1)$ است. اگر طبق استقرا تعداد حرکت ها زیر مسأله ها بهینه است. پس تعداد کل حرکت های این الگوریتم بهینه است.

روش‌های حل روابط بازگشتی

۱) روش درخت بازگشت

مراحل درخت بازگشت عبارتند از:

- ۱- ساخت درخت بازگشت
 - ۲- تعیین ارتفاع درخت بازگشت برحسب اندازه ورودی
 - ۳- تعیین هزینه هر سطح از درخت بازگشت
 - ۴- محاسبه مجموع هزینه‌های تمامی سطوح
 - ۵- اضافه کردن تعداد برگ‌ها (هر کدام با هزینه $O(1)$) به هزینه به‌دست آمده در گام قبل.
- نکته:** در اکثر مواقع مقدار محاسباتی که به کمک درخت بازگشتی حاصل می‌شود دقیق نمی‌باشد و برای محاسبه دقیق آن از استقرای قوی ریاضی یا روش جایگزینی استفاده می‌شود.
- نکته:** اگر درخت بازگشتی کامل باشد، می‌توان متوسط زمان اجرا را به‌دست آورد ولی اگر درخت بازگشتی کامل نباشد فقط می‌توان big-O یا امگا را محاسبه نمود.
- ✓ تست: مرتبه زمانی تابع زیر کدام است؟

$$T(n) = \begin{cases} c & n = 1 \\ 2T(\frac{n}{2}) + cn & n > 1 \end{cases}$$

$$T(n) = \theta(n) \quad (۴)$$

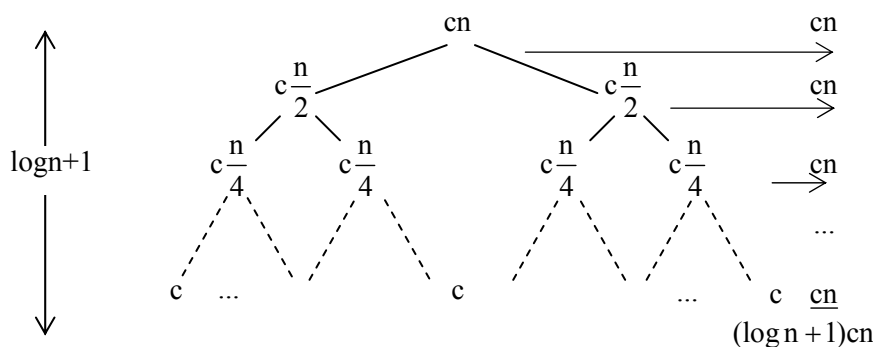
$$T(n) = \theta(\log n) \quad (۳)$$

$$T(n) = \theta(n^2) \quad (۲)$$

$$T(n) = \theta(n \log n) \quad (۱)$$

حل: گزینه «۱»

درخت فراخوانی بازگشتی به‌صورت زیر می‌باشد:



بنابراین داریم:

$$T(n) = (\log n + 1)cn = cn \log n + cn = O(n \log n)$$

✓ تست: مرتبه زمانی رابطه بازگشتی زیر کدام است؟

$$T(n) = \begin{cases} 1 & n = 1 \\ 3T(\lfloor \frac{n}{4} \rfloor) + n^2 & n > 1 \end{cases}$$

$$T(n) = \theta(n) \quad (۴)$$

$$T(n) = \theta(\log n) \quad (۳)$$

$$T(n) = \theta(n^2) \quad (۲)$$

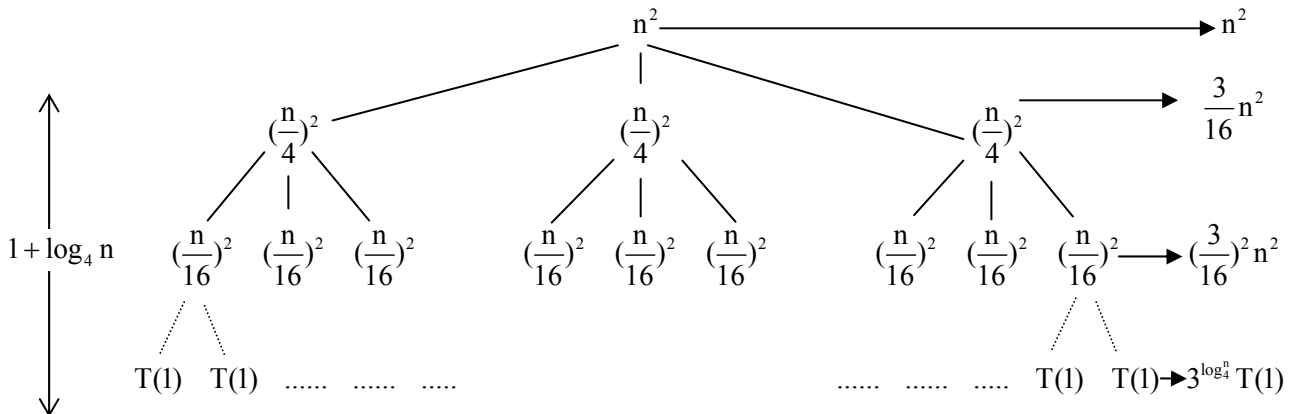
$$T(n) = \theta(n \log n) \quad (۱)$$

حل: گزینه «۲»

معمولاً کران بالا و پایین در حل معادلات بازگشتی قابل توجه نیستند، بنابراین رابطه بازگشتی را به صورت زیر می نویسیم:

$$T(n) = \begin{cases} 1 & n = 1 \\ 3T\left(\frac{n}{4}\right) + n^2 & n > 1 \end{cases}$$

درخت فراخوانی بازگشتی به صورت زیر می باشد:



با توجه به درخت فراخوانی فوق داریم:

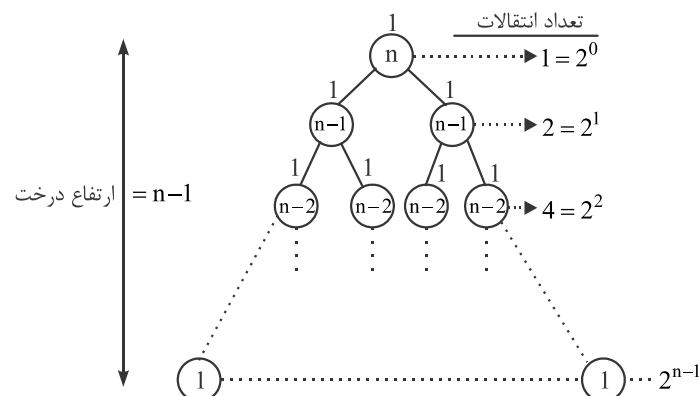
$$T(n) = n^2 + \sum_{i=1}^{\log_4 n - 1} \left(\frac{3}{16}\right)^i n^2 + 3^{\log_4 n} T(1) = n^2 + \left(\frac{\left(\frac{3}{16}\right)^{\log_4 n} - 1}{\frac{3}{16} - 1} \right) n^2 + n^{\log_4 3}$$

هنگامی که مقدار n به سمت بی نهایت میل می کند، داریم $\left(\frac{3}{16}\right)^{\log_4 n} \approx 0$ بنابراین:

$$T(n) = n^2 + \left(\frac{-1}{\frac{3}{16} - 1} \right) n^2 + n^{\log_4 3} = \left(\frac{1}{1 - \frac{3}{16}} \right) n^2 + n^{\log_4 3} = \frac{13}{16} n^2 + n^{0.79} = O(n^2)$$

مثال: برای الگوریتم Hanoi بازگشتی، با استفاده از درخت فراخوانی مرتبه زمانی را به دست آورید.

حل: ابتدا درخت فراخوانی بازگشتی را به صورت زیر رسم می کنیم:



$$T(n) = \sum_{i=0}^{n-1} 2^i = 2^n - 1 \Rightarrow T(n) = 2^n - 1$$

با توجه به این درخت داریم:

در درخت فراخوانی‌های بازگشتی بالا، در سطح i ام، 2^i گره داریم. هر گره در این درخت، یک فراخوانی بازگشتی را مشخص می‌کند. بدیهی است که زمان اجرای الگوریتم $\theta(2^n)$ خواهد بود و عمق پشته فراخوانی بازگشتی نیز $\theta(n)$ می‌باشد.

۲) روش معادلات بازگشتی کلاسیک

برای مثال می‌خواهیم با این روش معادله بازگشتی $T(n) = 2T(n-1) + 1$ با شرایط اولیه $T(1) = 1$ و $T(2) = 3$ ، ابتدا معادله مشخصه را به صورت زیر به دست آورده و سپس ریشه‌های آن را به دست می‌آوریم.

$$(x-2)(x-1) = 0 \Rightarrow \begin{cases} 1 = T(1) = 2c_1 + c_2 \\ 3 = T(2) = 4c_1 + c_2 \end{cases} \Rightarrow \begin{cases} c_1 = 1 \\ c_2 = -1 \end{cases}$$

بنابراین: $T(n) = 2^n - 1$

✓ تست: مسأله برج هانوی را با سه میله A ، B و C در نظر می‌گیریم. در اینجا هیچ حلقه‌ای را نمی‌توان مستقیماً از A به B یا از B به A منتقل کرد، یعنی چنین انتقال‌هایی تنها به کمک میله C انجام می‌شوند.

اگر در ابتدا n حلقه در میله A داشته باشیم و $T(n)$ حداقل تعداد عملیات لازم برای انتقال n حلقه از A به B باشد، آنگاه $T(n)$ با کدام رابطه مشخص می‌شود؟ ($T(1) = 2$) (مهندسی کامپیوتر سراسری ۸۱)

$$\begin{aligned} T(n) &= 3T(n-1) + 2 & (۲) \\ T(n) &= 6T(n-1) + 3 & (۱) \\ T(n) &= T(n-1) + T(n-2) + 1 & (۳) \\ T(n) &= T(n-1) + T(n-2) + 2 & (۴) \end{aligned}$$

✓ حل: گزینه «۲»

برای انتقال n مهره از A به B طبق شرایط گفته شده، باید مراحل زیر را طی کنیم:

- $n-1$ مهره را به صورت بازگشتی از A به B منتقل می‌کنیم.
- مهره n ام را از A به C منتقل می‌کنیم.
- $n-1$ مهره را به صورت بازگشتی از B به A منتقل می‌کنیم.
- مهره n ام را از C به B منتقل می‌کنیم.
- $n-1$ مهره را به صورت بازگشتی از A به B منتقل می‌کنیم.

بنابراین داریم:

$$T(n) = T(n-1) + 1 + T(n-1) + 1 + T(n-1)$$

🔴 نکته: فرض کنید دو الگوریتم A و B عمل یکسانی را انجام می‌دهند و معادله زمانی آنها با یکدیگر برابر است اما یکی بازگشتی و دیگری غیربازگشتی است، در این صورت الگوریتم غیر بازگشتی سریع‌تر از الگوریتم بازگشتی اجرا می‌شود زیرا الگوریتم بازگشتی در هر فراخوانی بازگشتی یک سربار زمانی ناشی از عملیات روی $stack$ انجام می‌پذیرد.

🔴 نکته: با توجه به نکته بالا معمولاً بهتر است که برای اجرای الگوریتم روی ماشین، آن را به غیربازگشتی تبدیل نماییم.

✓ تست: اعداد فیبوناچی (Fibonacci numbers) توسط دنباله بازگشتی زیر تعریف می‌شود و تابع بازگشتی زیر نیز جمله n ام دنباله فیبوناچی را ارائه می‌دهد. ($F_0 = 0, F_1 = 1, F_i = F_{i-1} + F_{i-2}, i \geq 2$)

```
int Fibo(int n){
    if (n <= 1)
        return n;
    else return (Fibo(n-1) + Fibo(n-2));
}
```

معادله زمانی این الگوریتم عبارتست از:

$$\begin{aligned} T(n) &= 2T(n-1) + \theta(1) & (۱) \\ T(n) &= 2T(n-2) + \theta(1) & (۳) \\ T(n) &= T(n-1) + T(n-2) + \theta(1) & (۲) \end{aligned}$$

(۴) هیچ‌دام

✓ حل: گزینه «۲»

این الگوریتم به روش بالا به پایین نوشته شده است و مرتبه زمانی آن به صورت $T(n) = O(2^n)$ می‌باشد.

نکته: با توجه به اینکه جمله n ام دنباله فیبوناچی از رابطه $F_n = \frac{\phi^n + \hat{\phi}^n}{\sqrt{5}}$ به دست می آید و همچنین $\frac{1}{\sqrt{5}} < \frac{1}{2} < \frac{1}{\sqrt{5}}$ بنابراین F_n با روند کردن $\phi^n / \sqrt{5}$ به نزدیکترین عدد صحیح به دست می آید.
✓ تست: برنامه زیر برای محاسبه سری فیبوناچی ارائه شده است. گزینه صحیح را انتخاب کنید.

(مهندسی کامپیوتر سراسری ۸۰)

Function fibo (n:integer):integer;
Var f,f₁,f₂,I :integer;

begin

f₁:=1;

f₂:=1;

for i:=1 to n do

begin

f:= f₁ + f₂;

f₁:= f₂;

f₂:=f;

end;

fibo:=f;

end;

(۱) الگوریتم این برنامه از رده برنامه ریزی پویا و مرتبه آن خطی است.

(۲) الگوریتم این برنامه از رده تقسیم و حل و مرتبه آن خطی است.

(۳) الگوریتم این برنامه از رده برنامه ریزی پویا و مرتبه آن بیش از خطی است.

(۴) الگوریتم این برنامه از رده تقسیم و حل و مرتبه آن بیش از خطی است.

✓ حل: گزینه «۱»

همان طور که ملاحظه می کنید این الگوریتم به صورت پایین به بالا طراحی شده است و این کار سبب بهبود بسیار زیاد در زمان الگوریتم شده است.

✓ تست: مرتبه زمانی یک الگوریتم با رابطه بازگشتی $T(n) = T(\frac{n}{2}) + \frac{T(\frac{n}{2})}{T(\frac{n}{2}) - 1}$ بیان شده است. مرتبه زمانی

(مهندسی کامپیوتر سراسری ۸۱)

الگوریتم کدام است؟

$$O(n^{\frac{3}{2}}) \quad (۴)$$

$$O(n^{\log_2 3}) \quad (۳)$$

$$O(3^n) \quad (۲)$$

$$O(2^n) \quad (۱)$$

✓ حل: گزینه «۳»

با توجه به رابطه زمانی بازگشتی داده شده معلوم است که الگوریتم با هر مقدار n خود را سه بار به صورت بازگشتی با مقدار $n/2$ فراخوانی می کند، بنابراین در درخت فراخوانی، فاکتور شاخه برابر ۳ و عمق درخت برابر $\log_2 n$ می باشد. بنابراین داریم:

$$T(n) = 3^{\log_2 n} = n^{\log_2 3}$$

۳) روش جایگذاری برای حل معادلات بازگشتی

این روش دارای دو مرحله است:

(۱) حدس و گمان در مورد فرم جواب

(۲) استفاده از استقرای ریاضی جهت پیدا کردن ثابت ها و نشان دادن اینکه جواب صحیح است.

• مثال: معادله بازگشتی $T(n) = 2T(\frac{n}{2}) + n$ را در نظر بگیرید، نشان دهید: $T(n) = O(n \log n)$.

✓ حل: برای این کار باید نشان دهیم که رابطه $T(n) \leq cn \log n$ (*) به ازای برخی مقادیر مناسب $c > 0$ برقرار است. ابتدا با

فرض اینکه برای تمام مقادیر کوچکتر از n (از جمله $\frac{n}{2}$)، در رابطه * صدق می کنند داریم:

$$T\left(\frac{n}{2}\right) \leq c \left[\frac{n}{2}\right] \log\left(\frac{n}{2}\right)$$

حال برای $T(n)$ رابطه را به صورت زیر به دست می آوریم:

$$\begin{aligned} T(n) &= 2T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n \leq 2c\left\lfloor \frac{n}{2} \right\rfloor \log\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n \\ &\leq cn \log\left(\frac{n}{2}\right) + n = cn \log n - cn \log 2 + n \\ &= cn \log n - cn + n = c \log n - n(c-1) \end{aligned}$$

بدیهی است که اگر $c \geq 1$ باشد آنگاه $n(c-1)$ جمله‌ای مثبت است بنابراین بدیهی است

$$\begin{aligned} T(n) &= cn \log n - n(c-1) < cn \log n \quad \text{as } c \geq 1 \\ \Rightarrow T(n) &< cn \log n \quad \text{as } c \geq 1 \end{aligned}$$

نکته: پیدا کردن یک حدس اولیه خوب چندان ساده نیست بلکه نیاز به تجربه و خلاقیت دارد و روش درخت بازگشتی می تواند برای ایجاد یک حدس اولیه خوب استفاده شود.

نکته: برخی اوقات به دلیل وجود برخی ثابت‌ها با وجود اینکه حدس و گمان برای جواب صحیح است اما نمی توان با استقرا کار را ادامه داد زیرا استقرا به حد کافی برای اثبات جزئیات کران‌های قوی نیست. در این صورت می توان یک جمله از مرتبه پایین را از حدس جواب کسر نمود.

مثال: معادله $T(n) = T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + T\left(\left\lceil \frac{n}{2} \right\rceil\right) + 1$ را با حدس $O(n)$ در نظر بگیرید. در این صورت:

$$T(n) = T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + T\left(\left\lceil \frac{n}{2} \right\rceil\right) + 1 \leq c\left\lfloor \frac{n}{2} \right\rfloor + c\left\lceil \frac{n}{2} \right\rceil + 1 = cn + 1$$

بنابراین شرط $T(n) \leq cn$ برقرار نیست. در این مواقع بهتر است حدس را به صورت $T(n) \leq cn - b$ ($b \geq 0$) در نظر بگیریم. پس

$$T(n) \leq (c\left\lfloor \frac{n}{2} \right\rfloor - b) + (c\left\lceil \frac{n}{2} \right\rceil - b) + 1 = cn - 2b + 1$$

با به کار بردن استقرا داریم:

واضح است که اگر $b \geq 1$ انتخاب شود؛ آنگاه جمله $-b + 1$ منفی بوده و در نتیجه داریم:

$$T(n) \leq cn - b + (-b + 1) \leq cn - b \Rightarrow T(n) \leq cn - b \quad \text{as } b \geq 1$$

✓ تست: مرتبه زمانی رابطه بازگشتی زیر کدام است؟

$$T(n) = \begin{cases} 1 & n = 1 \\ T\left(\frac{n}{3}\right) + T\left(\frac{2n}{3}\right) + O(n) & n > 1 \end{cases}$$

$$T(n) = \theta(n) \quad (۴)$$

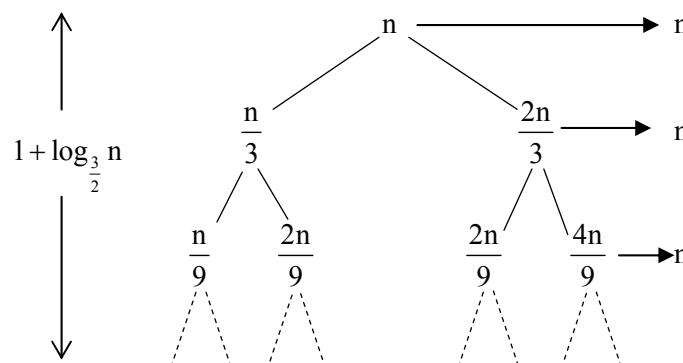
$$T(n) = \theta(\log n) \quad (۳)$$

$$T(n) = \theta(n^2) \quad (۲)$$

$$T(n) = \theta(n \log n) \quad (۱)$$

هـ حل: گزینه «۱»

روش اول: درخت فراخوانی بازگشتی را به صورت زیر رسم می کنیم:



برای به دست آوردن حداکثر ارتفاع، شاخه‌ای که تغییرات کمتری دارد را به صورت زیر در نظر می‌گیریم:

$$n \rightarrow \frac{2}{3}n \rightarrow \frac{4}{9}n = \left(\frac{2}{3}\right)^2 n \rightarrow \left(\frac{2}{3}\right)^3 n \rightarrow \dots \rightarrow 1$$

با توجه به رابطه فوق:

$$\left(\frac{2}{3}\right)^k n = 1 \Rightarrow n = \left(\frac{3}{2}\right)^k \Rightarrow k = \log_{\frac{3}{2}} n$$

و ارتفاع درخت برابر $k+1$ می‌باشد.

در این گونه مسائل که درخت کامل نیست حد بالا را در نظر می‌گیرید و فرض می‌کنید درخت کامل است تا بتوان حد بالا را به دست آورد.

$$T(n) = (\log_{\frac{3}{2}} n + 1)n = O(n \log_{\frac{3}{2}} n) = O(n \log n)$$

بنابراین داریم:

روش دوم: می‌توان از روش جایگزینی نیز استفاده نمود. در ابتدا حدس می‌زنیم $T(n) = O(n \log n)$ می‌باشد سپس باید حدس خود را اثبات کنیم:

$$T(n) \leq dn \log n$$

$$T(n) = T\left(\frac{n}{3}\right) + T\left(\frac{2n}{3}\right) + n$$

$$T(n) \leq d \frac{n}{3} \log \frac{n}{3} + d \frac{2n}{3} \log \frac{2n}{3} + n = d \frac{n}{3} \log n - d \frac{n}{3} \log 3 + d \frac{2n}{3} \log n + d \frac{2n}{3} \log 2 - d \frac{2n}{3} \log 3 + n$$

$$= dn \log n - dn \log 3 + d \frac{2n}{3} + n = dn \log n + n(1 + 0.66d - 1.58d)$$

با فرض $d > 2$ عبارت $1 + 0.66d - 1.58d < 0$. بنابراین ضریب n منفی می‌شود و می‌توان آن را از نامساوی حذف نمود.

$$T(n) \leq dn \log n$$

برای به دست آوردن مقدار دقیق d باید از تعریف big-O استفاده نمود.

۴) روش تغییر متغیر برای حل معادلات بازگشتی

در این روش سعی می‌شود با یک تغییر متغیر مناسب معادله را به فرم مناسب تغییر داد.

به مثال زیر توجه نمایید:

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

با قرار دادن $m = \log n$ داریم $n = 2^m$ و با قرار دادن در معادله بالا داریم:

$$T(2^m) = 2T(\sqrt{2^m}) + m \Rightarrow T(2^m) = 2T(2^{\frac{m}{2}}) + m$$

حال اگر قرار دهیم $S(m) = T(2^m)$ در آن صورت فرم نهایی معادله بالا به فرم زیر خواهد بود:

$$S(m) = 2S\left(\frac{m}{2}\right) + m$$

به راحتی می‌توان دید که $S(m) = O(m \log m)$ می‌باشد (طبق مثال‌های قبلی)، در نتیجه:

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

نکته مهم: اگر $a \geq 1$ و $c > 0$ آنگاه معادله بازگشتی

$$T(n) = T(n-a) + T(a) + cn$$

دارای جواب $T(n) = O(n^2)$ می‌باشد.

(برای اثبات می‌توان از درخت فراخوانی‌های بازگشتی استفاده نمود)

نکته مهم: معادله بازگشتی $T(n) = T(\alpha n) + T((1-\alpha)n) + cn$ با شرایط $0 < \alpha < 1$ و $c > 0$ دارای جوابی به فرم

$$T(n) = O(n \log n) \text{ می‌باشد که مبنای لگاریتم عبارتست از } \min\left\{\frac{1}{\alpha}, \frac{1}{1-\alpha}\right\}.$$

به عنوان مثال با به کار بردن روش درخت بازگشتی نشان دهید که معادله بازگشتی $T(n) = T(\frac{n}{3}) + T(2\frac{n}{3}) + cn$ دارای حد پایین مجانبی $T(n) = \Omega(n \log n)$ می باشد.

۵) قضیه اصلی (Master theorem)

رابطه بازگشتی زیر را در نظر بگیرید که در آن $\frac{n}{b}$ می تواند به فرم های $\left\lfloor \frac{n}{b} \right\rfloor$ یا $\left\lceil \frac{n}{b} \right\rceil$ باشد.

$$T(n) = aT\left(\frac{n}{b}\right) + f(n), a \geq 1, b > 1$$

در این صورت داریم:

$$1) \text{ if } \exists \varepsilon > 0; f(n) = O(n^{\log_b a - \varepsilon}) \Rightarrow T(n) = \theta(n^{\log_b a})$$

$$2) \text{ if } f(n) = \theta(n^{\log_b a} \times \log_2^k n) \Rightarrow T(n) = \theta(n^{\log_b a} \times \log_2^{k+1} n)$$

$$3) \text{ if } \exists \varepsilon > 0; f(n) = \Omega(n^{\log_b a + \varepsilon}), a f\left(\frac{n}{b}\right) < c f(n), c < 1, n \geq 0 \Rightarrow T(n) = \theta(f(n))$$

نکته: در به کار بردن قضیه master باید دقت کرد که شرایط به طور کامل برقرار باشد مخصوصاً در مورد سوم که ممکن است برخی اوقات همه شرایط به طور کامل برقرار نباشد که در سؤال های چهارگزینه ای نمونه به آن اشاره خواهیم داشت.

نکته مهم: در معادله $T(n) = aT(\frac{n}{b}) + f(n)$ ، $a \geq 1$ و $b > 1$ چنانچه داشته باشیم:

$$f(n) = \theta(n^{\log_b a} \log_n^k)$$

$\forall k \geq 0$

آنگاه داریم:

$$T(n) = \theta(n^{\log_b a} \log_n^{k+1})$$

مثال: تابع آکرمن را به صورت زیر تعریف می کنیم، بهترین زمان اجرای آن را به دست آورد.

$$A(i, j) = \begin{cases} A(1, j) = 2^j & j \geq 1 \\ A(i, 1) = A(i-1, 2) & i \geq 2 \\ A(i, j) = A(i-1, A(i, j-1)) & i, j \geq 2 \end{cases}$$

حل: تابع آکرمن را برای مقادیر کوچک i و j به صورت زیر محاسبه می کنیم:

	j=1	j=2	j=3	j=4
i=1	2^1	2^2	2^3	2^4
i=2	2^2	2^{2^2}	$2^{2^{2^2}}$	$2^{2^{2^{2^2}}}$
i=3	2^{2^2}	$2^{\left\{ \begin{smallmatrix} 2 \\ 2 \end{smallmatrix} \right\}_{16}}$	$2^{\left\{ \begin{smallmatrix} 2 \\ 2 \end{smallmatrix} \right\}_{16}^2}$	$2^{\left\{ \begin{smallmatrix} 2 \\ 2 \end{smallmatrix} \right\}_{16}^2}$

$$A(2, 2) = A(1, A(2, 1)) = A(1, 2^2) = 2^{2^2}$$

$$A(2, 3) = A(1, A(3, 1)) = A(1, 2^{2^2}) = 2^{2^{2^2}}$$

$$A(2, 4) = A(1, A(4, 1)) = A(1, 2^{2^{2^2}}) = 2^{2^{2^{2^2}}}$$

$$A(3,1)=A(2,2)=2^{2^2}$$

$$A(3,2)=A(2,A(3,1))=A(2,2^{2^2})=A(2,16)=2^{2^{16}}$$

$$A(3,3)=A(2,A(3,2))=A(2,2^{2^{16}})=2^{2^{2^{16}}}$$

$$A(3,4)=A(2,A(3,3))=A(2,2^{2^{2^{16}}})=2^{2^{2^{2^{16}}}}$$

$$A(4,1)=A(3,2)=2^{2^{16}} > 10^{80}$$

$$(n^n) \ll A(n,n) \text{ یا } A(n,n)=\Omega(n^n)$$

این تابع از تمام توابع که تاکنون مشاهده کرده‌اید بدتر است و داریم:

✓ تست: مرتبه زمانی تابع بازگشتی زیر کدام است؟

$$T(n) = \begin{cases} 1 & n = 0 \\ \frac{1 + n(n-1)T(n-1)}{n^2 + 1} & n > 0 \end{cases}$$

$$T(n) = \theta(n \log n) \quad (۲)$$

$$T(n) = \theta\left(\frac{\log n}{n!}\right) \quad (۴)$$

$$T(n) = \theta\left(\frac{\log n}{n}\right) \quad (۱)$$

$$T(n) = \theta(\log n) \quad (۳)$$

✓ حل: گزینه «۱»

$$n T(n) = \left[\frac{1}{n} + (n-1)T(n-1) \right] \frac{n^2}{n^2 + 1}$$

$$n T(n) = g(n)$$

$$g(n) = \frac{n^2}{n^2 + 1} \left[\frac{1}{n} + g(n-1) \right]$$

$$g(n) < \frac{1}{n} + g(n-1)$$

$$g(n) = 1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n} = \log n$$

$$g(n) \in O\left(\frac{\log n}{n}\right)$$

$$T(n) = \frac{1 + n(n-1)T(n-1)}{n^2 + 1} \Rightarrow T(n) = \frac{n^2 - 1}{n^2 - 1} \times \frac{1 + n(n-1)T(n-1)}{n^2 + 1}$$

$$T(n) = \frac{n^2 - 1}{n^4 - 1} \times [1 + n(n-1)T(n-1)] \Rightarrow \frac{T(n)}{n+1} = \frac{n-1}{n^4 - 1} \times [1 + n(n-1)T(n-1)]$$

$$\frac{T(n)}{n+1} = \frac{1}{n^4 - 1} \times \left[\frac{n-1}{1} + n(n-1)^2 T(n-1) \right]$$

$$\frac{T(n)}{n+1} = \frac{n^2}{n^4 - 1} \times \left[\frac{n-1}{n^2} + \frac{(n-1)^2}{n} T(n-1) \right]$$