الگوریتم
تحلیل

محسن هوشمند
دانشکده تکنولوژی اطلاعات و علم رایانه
دانشگاه تحصیلات تکمیلی علوم پایه زنجان

الگوریتم- سفر پیدایش؟

الگوریتم منشا از نام خوارزمی

کتاب جبر و مقابله در دوازده قرن پیش

فأما **الأموال والجذور التی تعدل العدد** فمثل قولک مال و عشرة أجذاره یعدل تسعة و ثلاثین درهماً و معناه أی مال إذا زدت علیه مثل عشرة أجذاره بلغ ذلک کله تسعة وثلاثین.

فقیاسه أن تنصف الأجذار و هی فی هذه المسألة خمسة فتضربها فی مثلها فتکون خمسة وعشرین فتزیدها علی التسعة والثلاثین فتکون أربعة وستین فتأخذ جذرها و هو ثمانیة فتنقص منه نصف الأجذار و هو خمسة فیبقی ثلاثة و هو جذر المال الذی ترید و المال تسعة.

شش- محمد بن موسی الخوارزمی، جبر و مقابله ۲۱۵ قمری، ۲۰۹ خورشیدی، ۸۳۰ میلادی

جذر: x، مال: x^2، مفرد: عدد ثابت

اما مال‌ها و جذرهایی که با عددی برابر می‌شوند:

اگر بگویی: یک مال، به اضافهٔ ده جذر از آن مال، با سی و نه درهم برابر می‌شود، مقصود آن است که اگر به مالی به اندازهٔ ده جذر از آن مال افزوده شود مجموع آن می‌شود سی و نه درهم.

راه حل آن چنین است: باید جذرها را نصف کنی – مقدار نصف آن در این مسئله پنج می‌شود – و آن نصف را در مانند خودش ضرب کنی، در این صورت حاصل‌ضرب بیست و پنج می‌شود. آن‌گاه این عدد را بر سی و نه بیفزایی، مجموعه شصت و چهار می‌شود، سپس جذر این عدد را می‌گیری، هشت می‌شود، آن‌گاه نیمی از شمارهٔ جذرها را که عبارت باشد از پنج، از آن کم می‌کنی که در نتیجه سه باقی می‌ماند، و همین عدد سه، جذر مال مورد نظر است، و آن مال نه است.

تلخیص سخن شیخ: توصیف الگوریتم برای حل نوع خاصی از معادلات درجه دو با استفاده از مثال $x^2 + 10x = 39$

$$x^2 + 10x = 39$$
$$\Longrightarrow (x+5)^2 = 39 + 25$$
$$= 64;$$

$$x + 5 = 8 \Longrightarrow x = 3; x^2 = 9$$

م خوارزمی، جبر مقابله، حسین خدیوجم، ۱۳۶۳

روش حل بالا را می‌توان از نمونه‌های حل الگوریتمی دانست.

الگوریتم- سیر تحول

اما وجود الگوریتم پیش از اطلاق نام الگوریتم

- مثال؟

انتقال روش حل خوارزمی با ترجمه به لاتین و اروپا

منجر به تلاش افرادی با حساب عددی

در سده هفدهم، به تأثیر از واژه «لگاریتم» (جان نپر، ۱۶۱۴) و نیز تمایل به یونانی‌نمایی، تغییر صورت واژه از Algorism به Algorithm

در ادامه اندک اندک به معنای هر گونه روش محاسباتی روشمند

۱۲۲۲ ش، کمی پیشتر از ساخت هر رایانه‌ای

- آدا لاولیس طراح نخستین الگوریتم برای اجرا روی «ماشین تحلیلی» چارلز بابیج
- نخستین برنامه‌نویس تاریخ
- عدم ساخته‌شدن ماشین تحلیلی
- یادداشت‌های او حاوی نخستین الگوریتم منتشر شده برای اجرا با ماشین

«روح حساب» به منازعهٔ بین «الگوری‌های» نو که از نوشتن اعداد بهره می‌برند و «چرتکه‌اندازان» سنتی با جداول شمارش‌شان می‌نگرد. برگفته از مروارید فلسفه Margarita Philosophica، اثر گرگور رایش، ۱۵۰۴ مسیحی

الگوریتم–سیر تحول

دهه ۱۳۱۰ شمسی

- انقلابی بنیادین در مفهوم الگوریتم
- تورینگ: هر الگوریتم قابل تعریف به عنوان ماشین تورینگ
- همزمان، آلونزو چرچ و استیون کلینKleene معرف تعریف‌های معادلی با جبر لامبدا و توابع بازگشتی
- این سه تعریف اساس «تز چرچ–تورینگ» و بنیان نظری علم رایانش است.

ماشین تورینگ

- بدل شدن به مدلی انتزاعی اما دقیق از محاسبه
- امکان شبیه‌سازی هر رایانش مکانیکی قابل تصور با مجموعه‌ای متناهی از دستورالعمل‌ها
- الگوریتم دیگر فقط «روش» نبود؛ بلکه «شیء ریاضی» شد که می‌توان آن را مطالعه، تحلیل، و حتی اثبات کرد.

الگوریتم–سیر تحول

دهه ۱۳۱۰ شمسی

- انقلابی بنیادین در مفهوم الگوریتم
- تورینگ: هر الگوریتم قابل تعریف به عنوان ماشین تورینگ
- همزمان، آلونزو چرچ و استیون کلینKleene معرف تعریف‌های معادلی با جبر لامبدا و توابع بازگشتی
- این سه تعریف اساس «تز چرچ–تورینگ» و بنیان نظری علم رایانش است.

ماشین تورینگ

- بدل شدن به مدلی انتزاعی اما دقیق از محاسبه
- امکان شبیه‌سازی هر رایانش مکانیکی قابل تصور با مجموعه‌ای متناهی از دستورالعمل‌ها
- الگوریتم دیگر فقط «روش» نبود؛ بلکه «شیء ریاضی» شد که می‌توان آن را مطالعه، تحلیل، و حتی اثبات کرد.

الگوریتم-سیر تحول

معنی لغوی آن: روش مخصوص حل دسته‌ای از مسائل

«مسئله»: پرسشی که به دنبال پاسخی برای آن هستیم.

- مرتب کردن افزایشی فهرستی از چند عدد
- بررسی حضور عددی در فهرست داده شده

مسئله دارای متغیرهایی است که مقدار معینی نگرفته‌اند و معروف به پارامتر

- پارامترهای مرتب‌سازی: فهرست و طول آن
- پارامترهای جستجو: فهرست و طول آن و عدد منظور

به دلیل داشتن پارامتر

- ، هر مسئله دارای رده‌ای است که که هر یک با تخصیص مقداری به پارامتر تعیین می‌گردد.
- نمونه: هر تخصیص خاص از مسئله
- راه‌حل نمونه‌ای از یک مسئلهْ: پاسخ به پرسش مسئله در رابطه با آن نمونه

الگوریتم–سیر تحول

۱۳۲۰ تا ۱۳۴۰– اختراع رایانه‌های الکترونیکی

- انتقال الگوریتم‌ها از روی کاغذ به درون ماشین‌ها
- پدیدار شدن طوفانی از الگوریتم‌های بنیادین
- ۱۳۲۴ جان فون نویمان مبدع الگوریتم «مرتب‌سازی ادغامی»
- ۱۳۲۶ جرج دانتزیگ ارائه دهندهٔ روش «سیمپلکس» برای حل مسائل بهینه‌سازی خطی
- ۱۳۳۱ دیوید هافمن مخترع الگوریتم «کدگذاری هافمن» برای فشرده‌سازی داده‌ها
- ۱۳۳۵دیکسترا معرف الگوریمت یافتن کوتاه‌ترین مسیر در گراف
- ۱۳۴۱ تونی هور معرف الگوریتم مرتب‌سازی سریع را معرفی کرد

الگوریتم‌های این‌چنینی هستهٔ بسیاری از سیستم‌های نرم‌افزاری

دهه‌های ۱۳۴۰ و ۱۳۵۰

- پرسش اساسی «الگوریتم چقدر باید سریع باشد؟»
- تعریف مفاهیمی مانند نماد O–بزرگ (کنوث)
- مفهوم زمان اجرا یا پیچیدگی زمانی
- مسئلهٔ P در برابر NP (کوک–لوین)
- مطالعه الگوریتم نه فقط از نظر ماهیت، بلکه از نظر کارآمدی

الگوریتم–سیر تحول

از ۱۳۵۰ تا ۱۳۸۰

- افزایش پیچیدگی و اهمیت الگوریتم‌ها
- ۱۳۵۶ الگوریتم‌های رمزنگاری مانند RSA
  - امنیت ارتباطات دیجیتال
- ۱۳۷۷ موتورهای جستجو مانند گوگل با الگوریتم رتبه‌بندی PageRank
  - انقلابی در دسترسی به اطلاعات
- ۱۳۷۹ الگوریتم‌های توزیع‌شده مانندMapReduce
  - جهت پردازش داده‌های عظیم

چنین الگوریتم‌هایی به زندگی روزمرهٔ میلیون‌ها حتی میلیاردها تن انسان وارد شد.

الگوریتم-سیر تحول

از دهه هشتاد شمسی به بعد

- روزگار «الگوریتم‌های هوشمند»
- ۱۳۷۶ پیروزی ابر رایانه دیپ بلو از شرکت آی‌بی‌ام بر گری کاسپاروف، قهرمان شطرنج جهان
  - نخستین پیروزی ماشین در مسابقه‌ای رسمی بر قهرمان جهان
  - عدم استفاده از یادگیری ماشین، بلکه مبتنی بر جستجوی گسترده و ارزشیابی دست‌ساخت
- از سال ۱۳۸۹ یادگیری ماشین و یادگیری عمیق
  - بر پایه الگوریتم‌هایی مانند شبکه‌های عصبی با ایده اولیه ازه دهه ۱۳۲۰ شمسی
  - کارآمدی فراوانی در حوزه‌های بینائی ماشین، پردازش سیگنال و زبان طبیعی
  - الگوریتم‌ها دیگر صرفاً پیرو دستورات صریح نیستند، بلکه یادگیری از داده‌ها،کشف الگوها و تصمیم‌گیری تقریبی و کارآمد
- ۱۴۰۱ انتشار چت‌بات ChatGPT
- وجود داده‌های عظیم و بزرگ و جریان داده‌ها
  - منجر به الگوریتم‌هایی که در ازای میزان قابل کنترلی از خطا فضا و زمان کمتری را در پردازش داده‌ها برای استخراج و مدیریت داده دارا باشند.

الگوریتم

«مسئله» به معنای پرسشی که به دنبال پاسخی برای آن هستیم

- دارای متغیرهایی بدون مقدار معین و معروف به پارامتر
- پارامترهای مرتب‌سازی: فهرست و طول آن. پارامترهای جستجو: فهرست و طول آن و عدد منظور

به دلیل داشتن پارامتر، هر مسئله دارای رده‌ای است

هر تخصیص خاص یک نمونه.

راه‌حل نمونه‌ای از یک مسئلهْ پاسخ به پرسش مسئله در رابطه با آن نمونه است.

نمونه‌ای از مسئله مرتب‌سازی و راه‌حل آن

- فهرست $S = [10,12,1,5,14,15,7]$ و $n = 7$. پاسخ آن $[1,5,7,10,12,14,15]$

نمونه‌ای از مسئلهٔ جستجو و راه‌حل آن

- فهرست $S = [10,12,1,5,14,15,7]$ و $n = 7$ و x=5. پاسخ آن «آری، x در S موجود است»

الگوریتم

حال فرض کنید فهرست دارای هزاران عضو

عدم امکان یافتن پاسخ به شیوهٔ معمول

عدم امکان اعمال شیوهٔ اقتضایی انسانی در رایانه

تولید برنامه رایانه‌ای جهت حل نمونه‌های مسئله

- نیاز به تدوین رویهٔ عمومی و گام به گام جهت پاسخ‌دهی به هر نمونه
- رویه گام به گام را الگوریتم خوانیم و اظهار می‌کنیم که الگوریتم مسئله را حل می‌کند.

الگوریتم

الگوریتم جستجو مقدار در فهرست

با شروع از عضو اول فهرست x را با هر عضو فهرست مقایسه کن تا x یافته شود یا فهرست به اتمام رسد. اگر یافت شد پاسخ آری وگرنه پاسخ نه.

معایب الگوریتم بالا

- نبودن به رویه فرمال

تعریف غیرصوری: الگوریتم در رایانش «دستورالعملی است که مراحل مختلف انجام کاری را به زبان دقیق و با جزئیات کافی به نحوی بیان کند که ترتیب مراحل روشن و شرط خاتمهٔ عملیات نیز آشکار باشد.»

- پس مجموعه‌ای متناهی از مراحل،
- غیرمبهم،
- اجراپذیر باشد،
- شرط خاتمه داشته باشد.

زمینه‌های مطالعه الگوریتم

طراحی الگوریتم: اعم از استقراء ریاضی، تقسیم و غلبه، حریصانه، برنامه‌نویسی پویا، پسگرد، انشعاب و تحدید.

- البته می‌توان نگاه به طراحی الگوریتم را به دو صورت دید، یا طبقه‌بندی بر اساس فنون همچون برنامه‌نویسی پویا و یا توجه به مسئله‌ای خاص مانند مرتب‌سازی و پیمایش گراف.

اعتبارسنجی (اثبات درستی الگوریتم‌ها): الگوریتم در صورتی درست است که به ازای هر ورودیْ (هر نمونهْ) خروجی متناظر درست را پاسخ دهد.

تحلیل الگوریتم (تحلیل مقدم، ارزیابی کارایی): الگوریتم در زمان اجرا روی پردازنده از حافظه جهت ذخیره‌سازی برنامه و داده‌ها استفاده می‌کند. تحلیل الگوریتم به فرایندی اطلاق می‌شود که میزان پیچیدگی زمان و پیچیدگی فضا اجرای الگوریتم را بررسی می‌کند.

پیاده‌سازی

آزمون برنامه شامل اشکال‌زدایی و پروفایلینگ برنامه (اندازه‌گیری کارایی و تحلیل موخر)

- پروفایلینگ: اجرای صحیح برنامه بر مجموعه داده ورودی و تعیین زمان و فضای لازم برای محاسبه نتیجه

الگوریتم

بازگشتی
- مستقیم
- غیرمستقیم

غیربازگشتی

مثال فیبوناتچی- بازگشتی مستقیم

```
Alg. Fib(n):
    if n <= 1:
        return n
    else:
        return Fib(n – 1) + Fib(n – 2)
```

مثال- ب‌م‌م دو عدد صحیح و مثبت و a>b>0

```
Alg. BMM(a, b):
    If b == 0:
        return a
    else:
        return BMM(b, a % b)
```

مثال- جستجوی کلید x در آرایه X[0,...,n-1]

```
Alg. Jostejo(X, n, i):
    if i >= n:
        return Hich
    else if X[i] == x:
        return i
    else:
        return Jostejo(X, n, i + 1)
```

الگوریتم کارا و تحلیل الگوریتم

```
def jostejo(mqadir, kelid):
    """yaftan andis mqdar kelid dar fehrest vorodi."""
    andis = -1
    for i in range(len(mqadir)):
        if mqadir[i] == kelid:
            andis = i
            break
    if andis == -1:
        print('dar majmoe nist')
    return andis
```

جستجوی ترتیبی در مقابل جستجوی دودوئی

جستجوی ترتیبی

- در بدترین حالت جستجو به تعداد ورودی‌ها

فهرست مرتب و استقرار مرتب مقادیر از کوچک به بزرگ مرتب

- امکان کاهش بدترین حالت

الگوریتم جستجو دودوئی

- یکی از الگوریتم‌های مهم جستجو
- دارای بدترین زمان جستجوی کمتر
- در ابتدا تعداد فهرست مشخص و تعیین اندیس عدد میانه
- مقایسهٔ مقدار کلید با عدد متناظر اندیس میانه مقایسه
  - در صورتی تساوی برگرداندن اندیس برگردانده می‌شود
  - اگر کوچکتر جستجو در نیمه چپ فهرست و اگر کلید بزرگتر جستجو در نیمهٔ راست فهرست
- در واقع هر دفعه جستجو در نیمی از فهرست ادامه می‌یابد. می‌توان نشان داد که در بدترین حالت لگاریتمی در مبنای دو مقایسه انجام خواهد پذیرفت. به عنوان مثال اگر طول فهرست ۱۰۰۰ باشد، در جستجوی معمولی در بدترین حالت ۱۰۰۰ مقایسه، ولی در جستجوی درختی حداکثر ۱۰ مقایسه انجام خواهد یافت که بهبودی بسیار مناسب است.

الگوریتم کارا و تحلیل الگوریتم

```
def jostejo_dodoee(adadha, kelid):
    """Jostejoye derkhti majomoe morattab jahat yaftan andis kelid"""
    chap = 0
    rast = len(adadha) - 1
    while chap <= rast:
        i_miani = int((chap + rast) / 2 )
        if kelid > adadha[i_miani]:
            chap = i_miani + 1
        elif kelid < adadha[i_miani]:
            rast = i_miani - 1
        else:
            return i_miani
    payam = 'dar fehrest nabood'
    return payam
```

جستجوی ترتیبی در مقابل جستجوی دودوئی

الگوریتم کارا و تحلیل الگوریتم

جستجوی ترتیبی در مقابل جستجوی دودوئی

اندازه فهرست	تعداد مقایسه‌های جستجو ترتیبی	تعداد مقایسه‌های جستجو دودوئی
۱۲۸	۱۲۸	۸
۱۰۲۴	۱۰۲۴	۱۱
۱۰۴۸۵۷۵	۱۰۴۸۵۷۵	۲۱
۴۴۲۹۴۹۶۷۲۹۶	۴۴۲۹۴۹۶۷۲۹۶	۳۳

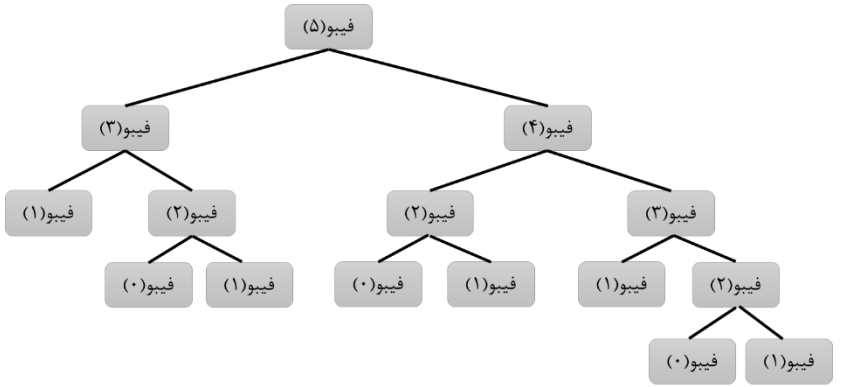
الگوریتم کارا و تحلیل الگوریتم

```
def Fibonacci(n):
    if n == 0:
        return 0
    elif n == 1:
        return 1
    else:
        return Fibonacci(n-1) + Fibonacci(n-2)

for i in range(11):
    print('Fib(', i, ')=', Fibonacci(i))
```

محاسبهٔ بازگشتی عدد فیبوناتچی

- زمان‌بر بودن
- فرقش با جستجو چیست؟



الگوریتم کارا و تحلیل الگوریتم

محاسبهٔ غیربازگشتی عدد فیبوناتچی

```
def Fibo(n):
    f = [0]*(n + 1)
    f[0] = 1
    f[1] = 1
    for i in range(2, n+1):
        f[i] = f[i - 1] + f[i - 2]
    return f[n]
```

الگوریتم کارا و تحلیل الگوریتم

```
def Fibo(n):
    f_qabli = 1
    f_2qabli = 1

    if n == 0 or n == 1:
        return 1
    for i in range(2, n+1):
        f = f_qabli + f_2qabli
        f_2qabli = f_qabli
        f_qabli = f

    return f
```

محاسبهٔ غیربازگشتی عدد فیبوناتچی

- امکان اجتناب از فهرست

- الگوریتم بازگشتی فیبوناتچی و الگوریتم جستجو دودوئی از نوع تقسیم و غلبه
- الگوریتم تکراری فیبوناتچی از نوع برنامه‌ریزی پویا
- اهمیت فنون متفاوت طراحی الگوریتم و آشنایی با آن‌ها
- مناسبتر بودن هر یک در حل بعضی مسائل نسبت به دیگر شیوه‌ها

تحلیل الگوریتم

جهت تعیین میزان کارایی الگوریتم در حل مسئله، نیاز به تحلیل الگوریتم

در کار قبلی تحلیل غیرصوری

روش صوری آن استفاده از تحلیل پیچیدگی

تحلیل الگوریتم نیازمند موارد:

- تعیین عملیات مورداستفاده شامل چهار عمل اصلی، دستورات مقایسه‌ای، خواندن و نوشتن، فراخوانی پردازه‌ها، دستورات انتساب
- تعیین مجموعه داده جهت بررسی همه حالات اعم از بهترین، متوسط، و بدترین زمان اجراء

در تحلیل متقدم:

- ذخیره و بازیابی هر مقدار با دیگر مقادیر برابر است
- زبان برنامه‌نویسی را وارد ماجرا نمی‌کنیم
- افزودگی‌های مقداردهی‌های اولیه یا خواندن و نوشتن را در نظر نمی‌گیریم
- به تعداد حقیقی چرخه‌های پردازنده توجه نمی‌کنیم. زیرا تعداد چرخه‌ها به رایانه خاص مرتبط است.
- علاوه بر آن به دنبال شمارش تمامی عملیات‌های انجام یافته نیز نیستیم.
  - صرفا به دنبال آنهایی هستیم که از رایانه، زبان، و برنامه‌نویسی مستقل است.

تحلیل الگوریتم

تأثیر اندازه مسئله روی پاسخ تاثیر

- استفاده از اندازه ورودی

روش معیار در تحلیل الگوریتم

- افزایش زمان اجرای الگوریتم با افزایش اندازه ورودی
- کل زمان اجرا تقریبا متناسب با تعداد اجرای عملیات پایه (مثلا مقایسه)
- تحلیل کارایی الگوریتم با تعیین تعداد دفعاتی که عملیات پایه‌ای بر اساس تابعی از اندازه ورودی
- اگر تعداد ورودی‌های مسئله n باشد سعی بر گزارش تابعی از میزان زمان اجرا بر اساس مقدار مذکور

الف	ب	ج

x = x + y	for i ∈ n:
	  x = x + y
		for i ∈ n:
		  for j ∈ n:
		  x = x + y

تحلیل الگوریتم

وابستگی انتخاب ورودی به نوع مسئله

انواع تعاریف اندازه ورودی

- تعداد اعضای فهرست مثل جستجو، مرتب‌سازی. یا تعداد سطر و ستون در ضرب ماتریسی.
- اندازه ورودی با لحاظ کردن دو مقدار
  - مثال، برای تعیین پیچیدگی الگوریتم‌های گراف بررسی دو مقدار تعداد راس و تعداد یال
- تعیین اندازه ورودی با تعداد علائم موردنیاز
  - تعداد بیت‌ها در فیبوناتچی بازگشتی

تحلیل پیچیدگی زمان الگوریتم برابر با تعداد تکرار عملیات پایه بر اساس مقدار اندازه ورودی

تحلیل الگوریتم

الگوریتم جمع اعداد: n تعداد اعداد. عمل اصلی جمع.

- فارغ از مقدار ورودی‌ها n تکرار حلقه رخ می‌دهد. پس $Z(n) = n$

مرتب‌سازی تبادلی

```
Alg. MorattabSazi_Tabadoli(X[0,…,n-1], n):
    for i ∈ [0,…, n-1]:
      for j ∈ [i+1,…,n-1]:
        if X[i] < X[j]:
          X[i] ↔ X[j]
```

مقایسه عمل اصلی الگوریتم مرتب‌سازی تبادلی است.

$$Z(n) = (n-۱) + (n-۲) + (n-۳) + \cdots + ۱ = \frac{(n-۱)n}{۲}$$

تحلیل الگوریتم

مثال‌های قبلی صرفا وابسته به تعداد ورودی

اغلب اوقات وباسته نبودن صرف پیچیدگی زمانی به اندازه ورودی مسئله

امکان وابستگی به مقدار ورودی علاوه بر طول ورودی

- ختم الگوریتم در موقعیتی در ابتدا در انتها یا در وسط مشاهده داده‌ها ختم شود.
- الگوریتم جستجو ترتیبی
- الگوریتم جستجوی ترتیبی:

بهترین زمان اجرا $z(n) = ۱$ است. چرا؟ بدترین زمان اجرا آن برابر $z(n) = n$

متوسط زمان اجرا (امید ریاضی)

$$z(n) = \sum_{k=0}^{n-۱} \left(k \times \frac{۱}{n}\right) = \frac{۱}{n} \sum_{k=۱}^{n} k = \frac{۱}{n} \frac{(n-۱)n}{۲} = \frac{n-۱}{۲}$$

مرتبه

عدم لزوم محاسبه دقیق

کفایت محاسبهٔ کران جهت تعیین پیچیدگی

اهمیت توان بزرگتر چندجمله‌ای

n	۱۰	۲۰	۵۰	۱۰۰	۱۰۰۰
$۰.۱n^۲$	۱۰	۴۰	۲۵۰	۱۰۰۰	۱۰۰۰۰۰
$۰.۱n^۲ + n + ۱۰۰$	۱۲۰	۱۶۰	۴۰۰	۱۲۰۰	۱۰۱۱۰۰

اُمیکرون بزرگ O (بدترین زمان اجرا)

$$f(n) = O\big(g(n)\big) \Leftrightarrow \exists c, n_0 : \forall n \geq n_0 \; 0 \leq f(n) \leq cg(n)$$

مثال- $f(n) = ۲n^۳ + ۲n^۲ = O(n^۳)$؟ $c = ۴$ و $n_0 = ۲$

مثال- $f(n) = ۲n^۳ + ۲n^۲ = O(n^۴)$؟ $c = ۴$ و $n_0 = ۲$

معمولا در پی کوچکترین کران بالایی

قضیه- اگر $a_m, a_0 + \cdots + a_{m-۱} n^{m-۱} + a_m n^m > 0$، $f(n) = O(n^m)$ آن‌گاه:

نتایج O

اگر تعداد اجرا دستوری در الگوریتمی f(n) باشد آن‌گاه زمان اجرای آن از مرتبهٔ $O(n^m)$ است.

اگر برنامه دارای k بخش با مرتبه بزرگی $O(nm_1)$، $O(nm_2)$، ...، $O(nm_k)$ باشد آن‌گاه مرتبه بزرگی برنامه $O(nm)$ است به طوری که

$$m = \text{بیش}(m_1, m_2, \dots, m_k)$$

قانون جمع (حالت دوتائی)

قانون حاصل‌ضرب: اگر $f_1(n) = O(g_1(n))$ و $f_2(n) = O(g_2(n))$، آن‌گاه $f_1(n)f_2(n) = O(g_1(n)g_2(n))$

$$O(f(n))O(g(n)) = O(f(n)g(n))$$

$$f(n) = O(f(n))$$

$$O(cf(n)) = O(f(n)), c > 0$$

$$O(f(n)g(n)) = f(n)O(g(n))$$

$$O(f(n)) + O(g(n)) = O(f(n) + g(n))$$

مثال- $f(n) = (4n^3 + 5n^2 + 7n)(8logn)$ از $O(n^3 logn)$

اُمگا بزرگ Ω (بهترین زمان اجرا)

$$f(n) = \Omega\big(g(n)\big) \Leftrightarrow \exists c, n_0 : \forall n \geq n_0 \; 0 \leq cg(n) \leq f(n)$$

مثال- $f(n) = ۳n^۳ + ۲n^۲ = \Omega(n^۳)$؟ $c = ۳$ و $n_0 = ۱$

همچنین $۳n^۲$و $۳n$ نیز در مثال اخیر صدق می‌کنند ولی معمولا به دنبال بزرگترین کران پائین هستیم.

تتا بزرگ Θ

$$f(n) = \Theta\big(g(n)\big) \Leftrightarrow \exists c_1, c_2, n_0 : \forall n \geq n_0 \; 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$$

قضیه $f(n) = \Theta\big(g(n)\big)$ اگر و تنها اگر $f(n) = O\big(g(n)\big)$ و $f(n) = \Omega\big(g(n)\big)$

مثال- $f(n) = 3n^3 + 2n^2 = \theta(n^3)$؟

نماد o

$$f(n) = o\big(g(n)\big) \Leftrightarrow \exists c, n_0 : \forall n \geq n_0 \; 0 \leq f(n) < cg(n)$$

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$$

مثال- $۷n^۲ + ۸n + ۳ = o(n^۳)$

مثال- $۷n^۲ + ۸n + ۳ = O(n^۲)$ است ولی $۷n^۲ + ۸n + ۳ \neq o(n^۲)$

مثال- $\log(n) = o(n)$

نماد ω

$$f(n) = \omega\big(g(n)\big) \Leftrightarrow \exists c, n_0 : \forall n \geq n_0\ 0 \leq cg(n) < f(n)$$

قضیه $f(n) = \omega\big(g(n)\big)$ اگر و فقط اگر $g(n) = o\big(f(n)\big)$

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$$

مثال- $\omega(n) = ۳ + ۸n + ۷n^۲$ است ولی $\omega\ (n^۲) \neq ۳ + ۸n + ۷n^۲$

مثال- $\omega(\log n) = n$

مجانب‌ها

الف- $O(n^۲)$

- $۳nlogn + ۸,۴n^۲,۵n + ۷,۶n^۲ + ۹,۲nlogn,n^۲ + ۲n$

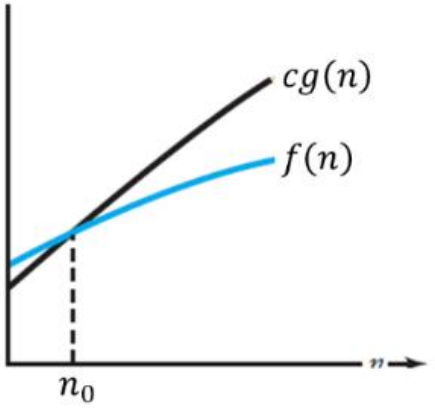
ب- $\Omega(n^۲)$

- $۴n^۲,۴n^۳ + ۳n^۲,n^۲ + ۹,۶n^۶ + n^۴,۵n^۲ + ۲n,۲^n + ۴n$

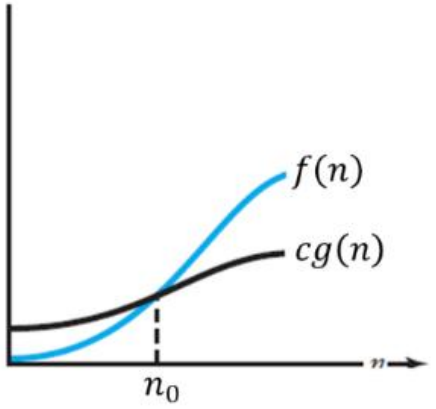
ج- $\Theta(n^۲) = O(n^۲) \cap \Omega(n^۲)$

- $۴n^۲,۶n^۲ + ۹,۵n^۲ + ۲n$

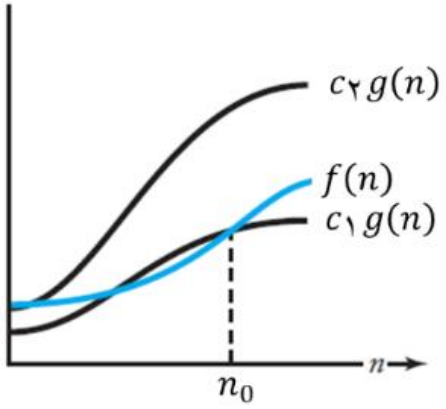
مجانب‌ها



$f(n) \in O(g(n))$



$f(n) \in \Omega(g(n))$



$f(n) \in \Theta(g(n))$

قضیه

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \begin{cases} c, c > 0 , f(n) = \Theta(g(n) \\ 0, f(n) = o(g(n) \\ \infty, f(n) = \omega(g(n) \end{cases}$$

$$O(1) < O(\log n) < O(n) < O(nlogn) < O(n^2) < O(n^3) < O(2^n)$$

خواص توابع مقایسه مرتبه

- تعدی

$$f(n) = \Theta\big(g(n)\big), g(n) = \Theta\big(h(n)\big) \Longrightarrow f(n) = \Theta\big(h(n)\big)$$

$$f(n) = O\big(g(n)\big), g(n) = \mathrm{O}\big(h(n)\big) \Longrightarrow f(n) = \mathrm{O}\big(h(n)\big)$$

$$f(n) = \Omega\big(g(n)\big), g(n) = \Omega\big(h(n)\big) \Longrightarrow f(n) = \Omega\big(h(n)\big)$$

$$f(n) = \mathrm{o}\big(g(n)\big), g(n) = \mathrm{o}\big(h(n)\big) \Longrightarrow f(n) = \mathrm{o}\big(h(n)\big)$$

$$f(n) = \omega\big(g(n)\big), g(n) = \omega\big(h(n)\big) \Longrightarrow f(n) = \omega\big(h(n)\big)$$

خواص توابع مقایسه مرتبه

- بازتابی

$$f(n) = \Theta\big(f(n)\big)$$
$$f(n) = O\big(f(n)\big)$$
$$f(n) = \Omega\big(f(n)\big)$$

- تقارن

$$f(n) = \Theta\big(g(n)\big) \Leftrightarrow g(n) = \Theta\big(f(n)\big)$$

- تقارن ترانهاده

$$f(n) = \mathrm{O}\big(g(n)\big) \Leftrightarrow g(n) = \Omega\big(f(n)\big)$$
$$f(n) = o\big(g(n)\big) \Leftrightarrow g(n) = \omega\big(f(n)\big)$$

روش‌های تحلیل پیچیدگی الگوریتم‌های بازگشتی

روش حدسی

با معادله مشخصه

- بازگشت خطی همگن
- بازگشت خطی ناهمگن
- تغییر متغیر (تبدیل دامنه)
- حل معادله بازگشتی با جانشینی

روش درختی

روش درختی

$$z(n) = ۲z\left(\frac{n}{۲}\right) + \theta(n)$$

روش درختی

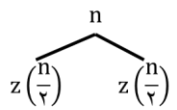
$$z(n) = ۲z\left(\frac{n}{۲}\right) + \theta(n)$$

$z(n)$

روش درختی

$$z(n) = ۲z\left(\frac{n}{۲}\right) + \theta(n)$$

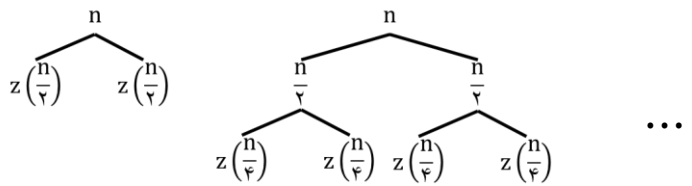
$z(n)$



روش درختی

$$z(n) = ۲z\left(\frac{n}{۲}\right) + \theta(n)$$

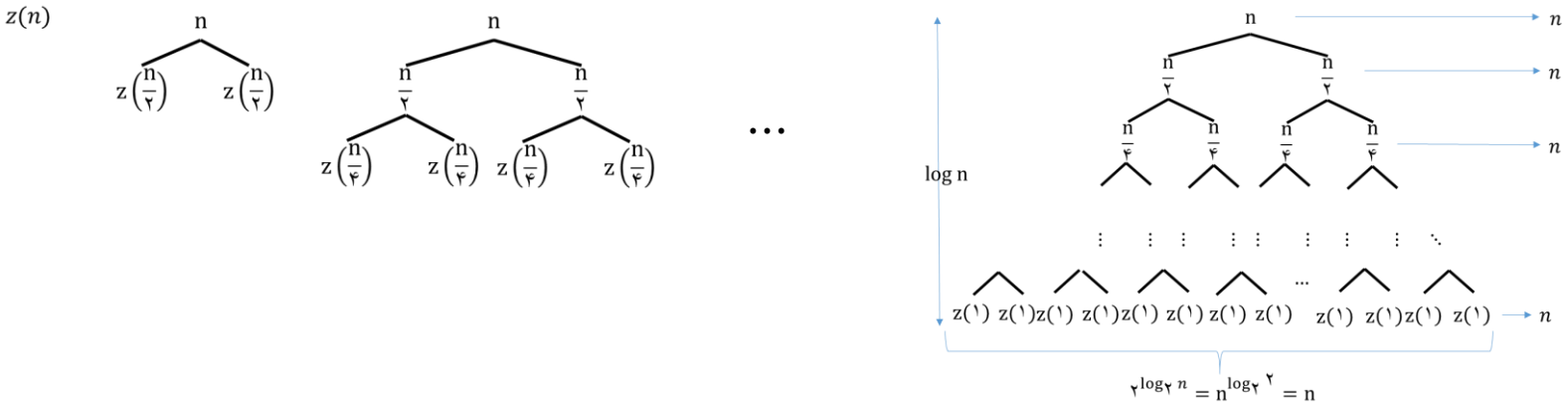
$z(n)$



...

روش درختی

$$z(n) = ۲z\left(\frac{n}{۲}\right) + \theta(n)$$

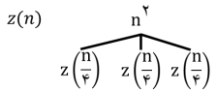


روش درختی

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

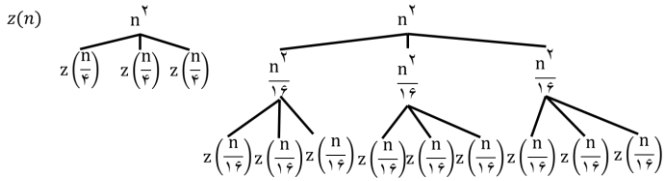
روش درختی

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

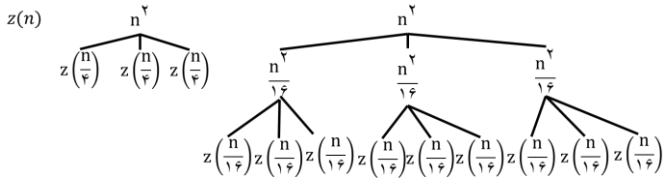


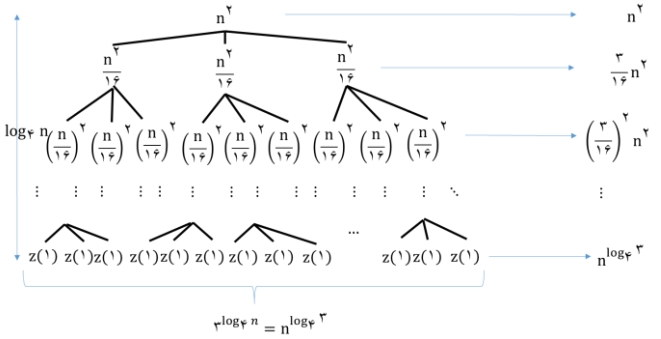
روش درختی

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



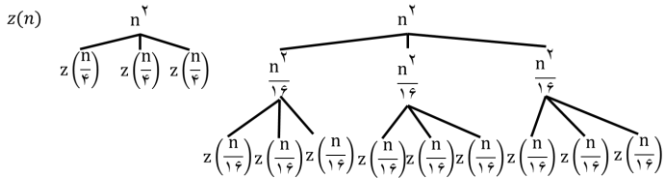
روش درختی

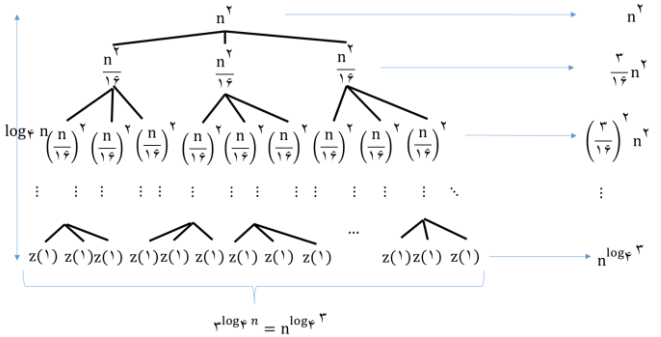




$$z(n) = ۳z\left(\frac{n}{۴}\right) + \theta(n^۲)$$

روش درختی

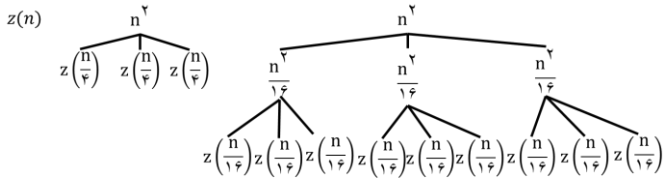


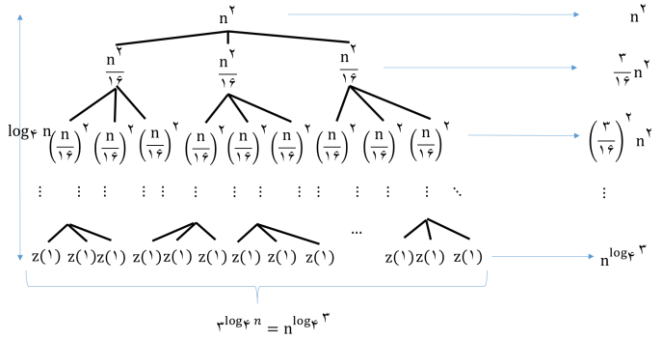


$$z(n) = ۳z\left(\frac{n}{۴}\right) + \theta(n^۲)$$

$$z(n) =$$

روش درختی

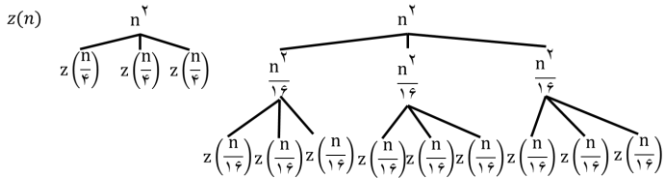


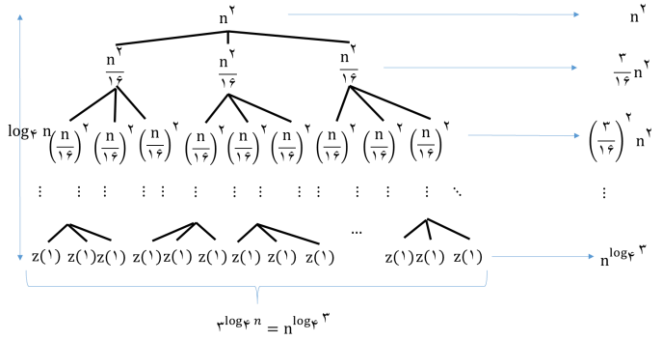


$$z(n) = ۳z\left(\frac{n}{۴}\right) + \theta(n^۲)$$

$$z(n) = n^۲ + \frac{۳}{۱۶}n^۲ + \left(\frac{۳}{۱۶}\right)^۲ n^۲ + \cdots + \left(\frac{۳}{۱۶}\right)^{\log_۴ n} n^۲$$

روش درختی

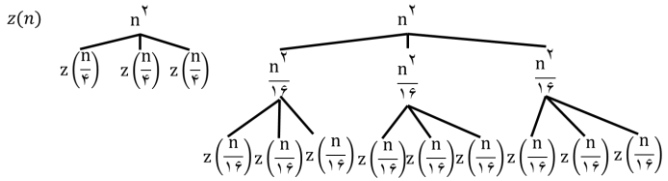


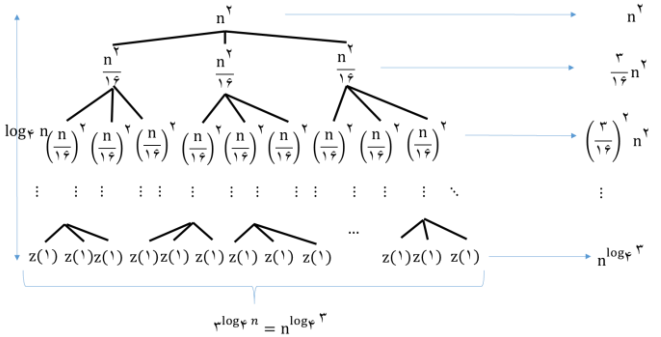


$$z(n) = ۳z\left(\frac{n}{۴}\right) + \theta(n^۲)$$

$$z(n) = n^۲ + \frac{۳}{۱۶}n^۲ + \left(\frac{۳}{۱۶}\right)^۲ n^۲ + \cdots + \left(\frac{۳}{۱۶}\right)^{\log_۴ n} n^۲ + \theta\left(n^{\log_۴ ۳}\right)$$

روش درختی



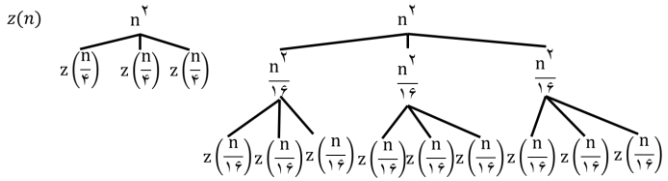


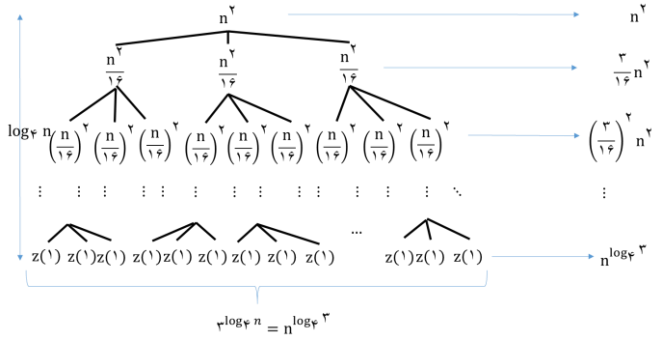
$$z(n) = ۳z\left(\frac{n}{۴}\right) + \theta(n^۲)$$

$$z(n) = n^۲ + \frac{۳}{۱۶}n^۲ + \left(\frac{۳}{۱۶}\right)^۲ n^۲ + \cdots + \left(\frac{۳}{۱۶}\right)^{\log_۴ n} n^۲ + \theta\left(n^{\log_۴ ۳}\right)$$

$$= \sum_{i=0}^{\log_۴ n} \left(\frac{۳}{۱۶}\right)^i n^۲ + \theta\left(n^{\log_۴ ۳}\right)$$

روش درختی



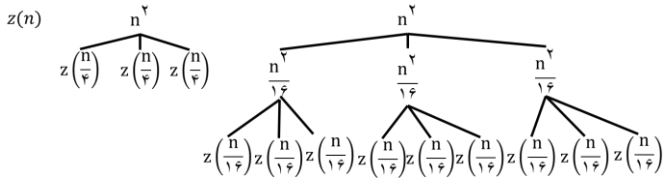


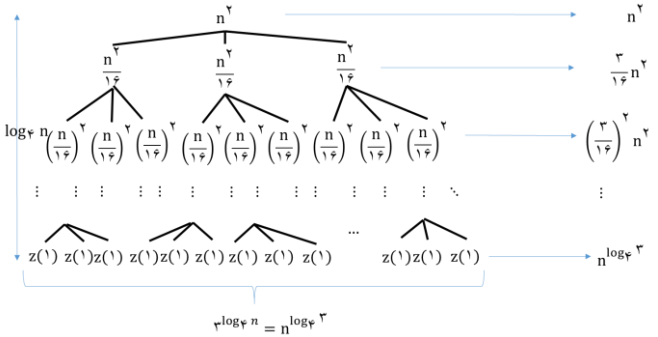
$$z(n) = ۳z\left(\frac{n}{۴}\right) + \theta(n^۲)$$

$$z(n) = n^۲ + \frac{۳}{۱۶}n^۲ + \left(\frac{۳}{۱۶}\right)^۲ n^۲ + \cdots + \left(\frac{۳}{۱۶}\right)^{\log_۴ n} n^۲ + \theta\left(n^{\log_۴ ۳}\right)$$

$$= \sum_{i=0}^{\log_۴ n} \left(\frac{۳}{۱۶}\right)^i n^۲ + \theta\left(n^{\log_۴ ۳}\right) < \sum_{i=0}^{\infty} \left(\frac{۳}{۱۶}\right)^i n^۲ + \theta\left(n^{\log_۴ ۳}\right)$$

روش درختی





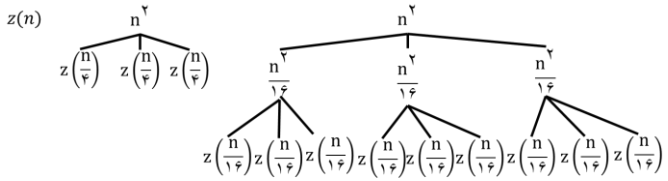
$$z(n) = ۳z\left(\frac{n}{۴}\right) + \theta(n^۲)$$

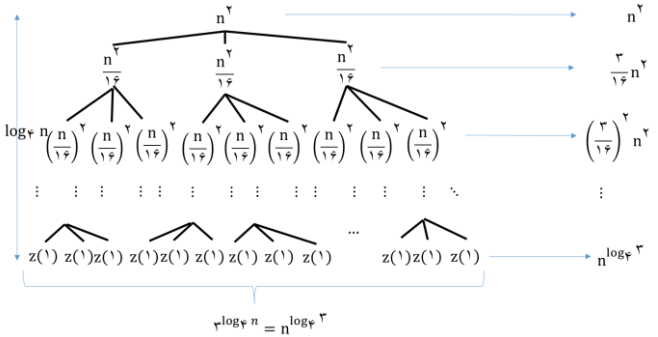
$$z(n) = n^۲ + \frac{۳}{۱۶}n^۲ + \left(\frac{۳}{۱۶}\right)^۲ n^۲ + \cdots + \left(\frac{۳}{۱۶}\right)^{\log_۴ n} n^۲ + \theta\left(n^{\log_۴ ۳}\right)$$

$$= \sum_{i=0}^{\log_۴ n} \left(\frac{۳}{۱۶}\right)^i n^۲ + \theta\left(n^{\log_۴ ۳}\right) < \sum_{i=0}^{\infty} \left(\frac{۳}{۱۶}\right)^i n^۲ + \theta\left(n^{\log_۴ ۳}\right)$$

$$= \frac{۱}{۱ - \left(\frac{۳}{۱۶}\right)} n^۲ + \theta\left(n^{\log_۴ ۳}\right)$$

روش درختی





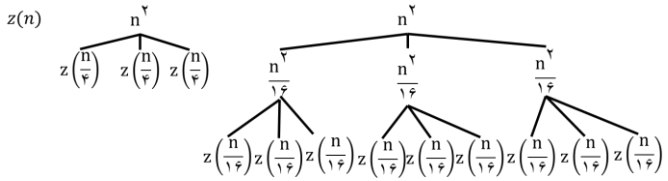
$$z(n) = ۳z\left(\frac{n}{۴}\right) + \theta(n^۲)$$

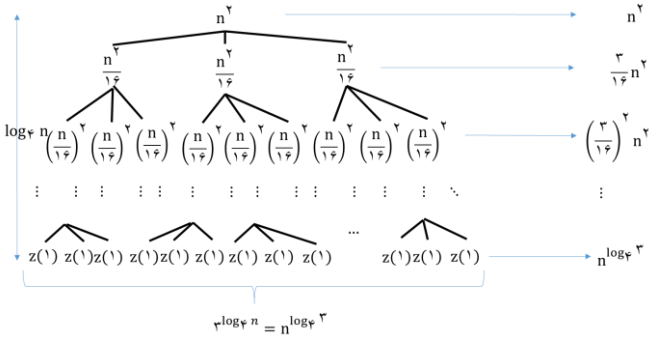
$$z(n) = n^۲ + \frac{۳}{۱۶}n^۲ + \left(\frac{۳}{۱۶}\right)^۲ n^۲ + \cdots + \left(\frac{۳}{۱۶}\right)^{\log_۴ n} n^۲ + \theta\left(n^{\log_۴ ۳}\right)$$

$$= \sum_{i=0}^{\log_۴ n} \left(\frac{۳}{۱۶}\right)^i n^۲ + \theta\left(n^{\log_۴ ۳}\right) < \sum_{i=0}^{\infty} \left(\frac{۳}{۱۶}\right)^i n^۲ + \theta\left(n^{\log_۴ ۳}\right)$$

$$= \frac{۱}{۱ - \left(\frac{۳}{۱۶}\right)} n^۲ + \theta\left(n^{\log_۴ ۳}\right) = \frac{۱۶}{۱۳} n^۲ + \theta\left(n^{\log_۴ ۳}\right)$$

روش درختی





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

$$z(n) = n^۲ + \frac{۳}{۱۶}n^۲ + \left(\frac{۳}{۱۶}\right)^۲ n^۲ + \cdots + \left(\frac{۳}{۱۶}\right)^{\log_۴ n} n^۲ + \theta\left(n^{\log_۴ ۳}\right)$$

$$= \sum_{i=0}^{\log_۴ n} \left(\frac{۳}{۱۶}\right)^i n^۲ + \theta\left(n^{\log_۴ ۳}\right) < \sum_{i=0}^{\infty} \left(\frac{۳}{۱۶}\right)^i n^۲ + \theta\left(n^{\log_۴ ۳}\right)$$

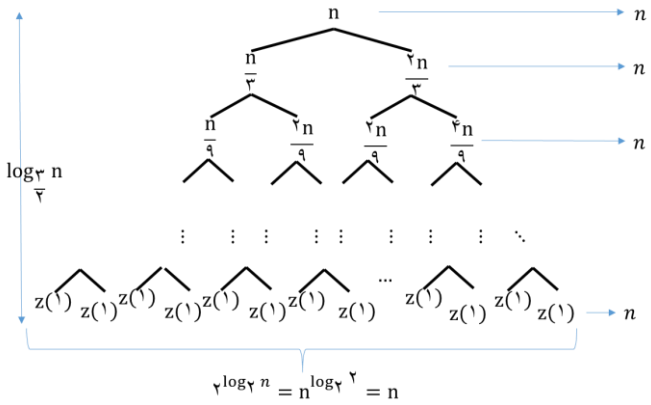
$$= \frac{۱}{۱ - \left(\frac{۳}{۱۶}\right)} n^۲ + \theta\left(n^{\log_۴ ۳}\right) = \frac{۱۶}{۱۳}n^۲ + \theta\left(n^{\log_۴ ۳}\right) = O\left(n^۲\right)$$

روش درختی

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

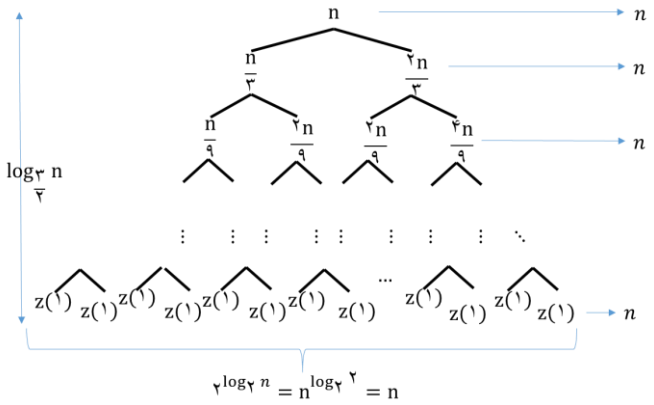
روش درختی

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



روش درختی

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



$$nlogn$$

قضیه اصلی حل روابط تقسیم و غلبه

فرض می‌کنیم

$$\begin{cases} z(n) = az\left(\frac{n}{b}\right) + cn^k \\ t(۱) = \theta(۱) \end{cases}$$

که a و b و c اعداد مثبت و $a \geq ۱$ و $b > ۱$، آن‌گاه،

الف- اگر $b^k < a$ آن‌گاه $z(n) = \theta\left(n^{\log_b a}\right)$

ب- اگر $b^k = a$ آن‌گاه $z(n) = \theta\left(n^k \log n\right)$

ج- اگر $b^k > a$ آن‌گاه $z(n) = \theta\left(n^k\right)$.

درصورت جانشینی $z(n) = az\left(\frac{n}{b}\right) + cn^k$ با $z(n) \leq at\left(\frac{n}{b}\right) + cn^k$ ($z(n) \geq at\left(\frac{n}{b}\right) + cn^k$) نتایج با O یا Ω جانشین خواهد شد.

قضیه اصلی حل روابط تقسیم و غلبه

مثال- $z(n) = ۸z\left(\frac{n}{۴}\right) + ۵n^{۲}$

$$۸ < ۴^{۲} \Longrightarrow z(n) \in \Theta\left(n^{۲}\right)$$

مثال- $z(n) = ۹z\left(\frac{n}{۳}\right) + ۵n^{۱}$

$$۹ > ۳^{۱} \Longrightarrow z(n) \in \Theta\left(n^{\log_{۳} ۹}\right) = \Theta\left(n^{۲}\right)$$

قضیه عمومی حل روابط تقسیم و غلبه

فرض کنیم که معادله بازگشتی زمان اجرای الگوریتمی عبارت است از

$$\begin{cases} z(n) = az\left(\frac{n}{b}\right) + f(n), n = b^m \\ z(۱) = \theta(۱) \end{cases}$$

که a و b و c اعداد مثبت و $a \geq ۱$ و $b > ۱$، C>0 و f(n)>0 آن‌گاه

الف- اگر $f(n) = o\left(n^{\log_b a}\right)$ آن‌گاه $z(n) = \theta\left(n^{\log_b a}\right)$

ب- اگر $f(n) = \theta\left(n^{\log_b a}\right)$ آن‌گاه $z(n) = \theta\left(n^{\log_b a} \log n\right)$

ج- اگر $f(n) = \omega\left(n^{\log_b a}\right)$ آن‌گاه $z(n) = \theta\left(f(n)\right)$

قضیه عمومی حل روابط تقسیم و غلبه

لم- اگر در رابطه $z(n) = az\left(\frac{n}{b}\right) + f(n)$ داشته باشیم $f(n) = \theta\left(n^{\log_b a} \log^j n\right)$ آن‌گاه $z(n)$
$= \theta\left(n^{\log_b a} \log^{j+1} n\right)$

تحلیل احتمالی الگوریتم‌ها

در بسیاری از الگوریتم‌های جدید، رفتار الگوریتم نه تنها به ورودی بلکه وابسته به انتخاب‌های تصادفی الگوریتم

عدم کفایت تحلیل کلاسیکِ زمانی و فضایی برای درک کامل رفتار چنین الگوریتم‌هایی

نیاز به تحلیل تحلیل

ارزیابی عملکرد الگوریتم با استفاده از ابزارهای احتمال و آمار

تحلیل احتمالی الگوریتم‌ها

احتمال صحت در الگوریتم‌های تصادفی

- عدم اهمیت صرف زمان اجرا مهم نیست
- بلکه نیاز به بررسی «الگوریتم با چه احتمالی جواب درست تولید می‌کند؟»

الف) الگوریتم‌های مونت‌کارلو

- همیشه سریع هستند،
- اما احتمال خروجی نادرست باشد،
- میزان خطا معمولاً کوچک و قابل کنترل
- مثال: فیلتر بلوم امکن برگرداندن مثبت کاذب، اما عاری از منفی کاذب

ب) الگوریتم‌های لاس‌وگاس

- تولید همیشگی جواب درست تولید
- اما زمان اجرای آن‌ها تصادفی
- مثال: انتخاب سریع تصادفی که میانگین اجرا سریع، ولی در موارد نادر امکان کند بودن

تحلیل احتمالی الگوریتم‌ها

کران خطا: تعیین دقیق محدود خطا یا دقت خروجی

- نامساوی مارکوف، نامساوی چبیشف، کران چرنوف، و قانون اعداد بزرگ (LLN)

الف) نامساوی مارکوف: برای متغیر تصادفی غیرمنفی X و $a > 0$:

$$P(X \geq a) \leq \frac{E[X]}{a}$$

مثال- اگر میانگین زمان اجرا $O(n\log n)$ باشد، احتمال اینکه زمان اجرا از $100n\log n$ بیشتر شود حداکثر $\frac{1}{100}$ است.

تحلیل احتمالی الگوریتم‌ها

ب) نامساوی چبیشف:

$$P(|\ X - E[X]\ | \geq a) \leq \frac{Var(X)}{a^2}$$

کاربرد: در الگوریتم‌های مونت‌کارلو برای کنترل وردایی کاربرد دارد.

تحلیل احتمالی الگوریتم‌ها

ب) نامساوی چبیشف:

$$P(|\ X - E[X]\ | \geq a) \leq \frac{Var(X)}{a^2}$$

کاربرد: در الگوریتم‌های مونت‌کارلو برای کنترل وردایی کاربرد دارد.

تحلیل احتمالی الگوریتم‌ها

اطمینان و تقویت

در بسیاری از الگوریتم‌ها، کاهش چشم‌گیر خطا با تکرار مستقل الگوریتم و ترکیب نتایج

- تقویت
- تقویت پیروزی بر این اساس است که اگر اجرای الگوریتم احتمال موفقیت pداشته باشد، تکرار آن چندین بار می‌تواند احتمال موفقیت را به مقدار دلخواه نزدیک به ۱ برساند.
- مثال- الگوریتمی دارای ۸۰ درصد احتمال تولید جواب صحیح
  - با اجرای ۱۰ بار و رأی اکثریتْ
  - کوچک شدن نمائی احتمال خطای نهایی

سخن کوتاه، اگر درستی پاسخ الگوریتم با احتمال $p > 0.5$

- با تکرار مستقل و رأی اکثریت، کاهش احتمال خطا به صورت نمایی

تحلیل احتمالی الگوریتم‌ها

قضیه (تقویت چرنوف-هوفدینگ): اگر الگوریتمی با احتمال $p = \frac{1}{2} + \gamma$ پاسخ درست برگرداند، با t بار تکرار و رأی اکثریت:

$$P(\text{خطا}) \leq e^{-2\gamma^2 t}$$

مثال- برای $p = 0.6$ یا $\gamma = 0.1$ و $t = 100$:

$$P(\text{خطا}) \leq e^{-2\times 0.01\times 100} = e^{-2} \approx 0.135$$

با $t = 200$

$$pr \leq e^{-4} \approx 0.0.$$

منابع درس

[Medjedovic22] D, Medjedovic, E. Tahirovic, “Algorithms and Data Structures for Massive Datasets,” Simon and Schuster, 2022.

[Gakhov20] A. Gakhov, Probabilistic Data Structures And Algorithms For Big Data Applications, BoD, 2020.

[Rocca21] M. La Rocca, Advanced Algorithms and Data Structures, Simon and Schuster, 2021.

[Neapolitan]

[CLRS]

[Anand2011] J. Leskovec, A. Rajaraman, J.D. Ullman, Mining of Massive Datasets. Cambridge university press, 2020.