دانشگاه فردوسی مشهد

الگوریتم پیشرفته

تحلیل سرشکن

دکتر حامد فهیمی

دانشکده علوم ریاضی

تحلیل سرشکن‌شده (Amortized Analysis)

مقدمه:

در ادامه می‌خواهیم مفهومی مهم در تحلیل الگوریتم‌ها به نام **«تحلیل سرشکن‌شده» (Amortized Analysis)** را بررسی کنیم.

در تحلیل سرشکن‌شده، ما زمان مورد نیاز برای انجام دنباله‌ای از عملیات روی یک ساختمان داده را بر کل عملیات انجام شده، میانگین می‌گیریم. هدف این است که نشان دهیم، اگرچه ممکن است یک عملیات واحد در یک دنباله بسیار گران باشد، اما میانگین هزینه هر عملیات در آن دنباله، کوچک است.

تفاوت تحلیل سرشکن‌شده و تحلیل حالت میانگین (Average-Case)

ممکن است این تحلیل شبیه به تحلیل حالت میانگین (Average-Case Analysis) به نظر برسد، اما یک تفاوت اساسی وجود دارد:

- **تحلیل سرشکن‌شده** هیچ احتمالی را در بر نمی‌گیرد.
- این تحلیل، **عملکرد میانگین هر عملیات را در بدترین حالت (worst-case) تضمین می‌کند**. یعنی ما یک مرز بالای قطعی (Deterministic) برای هزینه کل یک دنباله از عملیات به دست می‌آوریم.

سه تکنیک اصلی تحلیل سرشکن‌شده

در این فصل، ما سه تکنیک رایج برای تحلیل سرشکن‌شده را پوشش خواهیم داد:

1. **تحلیل تجمعی (Aggregate Analysis):** که امروز به تفصیل به آن خواهیم پرداخت. در این روش، ما یک کران بالای $T(n)$ را برای هزینه کل یک دنباله از n عملیات تعیین می‌کنیم. سپس هزینه سرشکن‌شده (Amortized Cost) هر عملیات $T(n)/n$ خواهد بود. در این روش، همه عملیات هزینه سرشکن‌شده یکسانی دریافت می‌کنند.
2. **روش حسابداری (Accounting Method):** در این روش، ما به عملیات‌های مختلف، هزینه‌های سرشکن‌شده متفاوتی اختصاص می‌دهیم. ما برخی عملیات‌ها را بیش از هزینه واقعی‌شان شارژ می‌کنیم (overcharge) و این اضافه پرداخت را به عنوان "اعتبار پیش‌پرداخت" (prepaid credit) در ساختمان داده ذخیره می‌کنیم تا بعداً هزینه عملیاتی را که کمتر از هزینه واقعی‌شان شارژ می‌شوند، پرداخت کنیم.
3. **روش پتانسیل (Potential Method):** این روش شبیه به روش حسابداری است، با این تفاوت که اعتبار پیش‌پرداخت را به عنوان "انرژی پتانسیل" کل ساختمان داده در نظر می‌گیرد، نه اینکه اعتبار را به اشیاء خاصی در داخل آن مرتبط کند.

مثال‌هایی که بررسی خواهیم کرد:
برای بررسی این سه روش، از دو مثال کلاسیک استفاده خواهیم کرد:

1. پشته (Stack) با عملیات اضافی `MULTIPOP` (خالی کردن چند عنصر با هم).
2. شمارنده دودویی (Binary Counter) با عملیات `INCREMENT` (یکی افزودن).

> **نکته مهم:** هزینه‌هایی که در طول تحلیل سرشکن‌شده تخصیص داده می‌شوند، **فقط برای اهداف تحلیلی** هستند و نباید در کد واقعی پیاده‌سازی شوند.

بخش ۱۷.۱: تحلیل تجمعی (Aggregate Analysis)

در تحلیل تجمعی، ما نشان می‌دهیم که برای هر n دلخواه، یک دنباله از n عملیات در **کل**، زمان $T(n)$ (در بدترین حالت) طول می‌کشد.

بنابراین، هزینه میانگین یا **هزینه سرشکن‌شده** (Amortized Cost) برای هر عملیات $T(n)/n$ است.

توجه داشته باشید که این هزینه سرشکن‌شده به هر عملیات اعمال می‌شود، حتی اگر انواع مختلفی از عملیات‌ها در دنباله وجود داشته باشند. (این برخلاف دو روش دیگر است که می‌توانند هزینه‌های سرشکن‌شده متفاوتی به عملیات‌های مختلف اختصاص دهند).

مثال ۱: عملیات پشته (Stack Operations)

بیایید تحلیل تجمعی را با یک پشته که عملیات جدیدی به آن اضافه شده، بررسی کنیم.

عملیات استاندارد پشته:
ما دو عملیات اساسی پشته را می‌شناسیم که هر دو در زمان $O(1)$ اجرا می‌شوند:

- `PUSH(S, x)` : شیء x را به پشته S اضافه می‌کند.
- `POP(S)` : عنصر بالای پشته S را برمی‌گرداند.

فرض کنید هزینه واقعی هر یک از این عملیات‌ها 1 باشد. هزینه کل یک دنباله از n عملیات `PUSH` و `POP` برابر n و زمان اجرای کل $\Theta(n)$ است.

عملیات جدید: `MULTIPOP`
حالا عملیات `MULTIPOP(S, k)` را اضافه می‌کنیم که k عنصر بالای پشته S را حذف می‌کند (یا اگر پشته کمتر از k عنصر داشت، کل پشته را خالی می‌کند).

```
MULTIPOP(S, k)
    while not STACK-EMPTY(S) and k > 0
        POP(S)
        k = k - 1
```

هزینه `MULTIPOP` :
هزینه واقعی `MULTIPOP` برابر است با $min(s, k)$ که s تعداد عناصر پشته و k آرگومان تابع است.

تحلیل ساده‌لوحانه (Naive Analysis):
بیایید یک دنباله از n عملیات `POP` ، `PUSH` و `MULTIPOP` را روی یک پشته خالی تحلیل کنیم.

- هزینه بدترین حالت (Worst-case) یک عملیات `MULTIPOP` چقدر است؟ $O(n)$. (مثلاً پشته n عنصر داشته باشد و ما `MULTIPOP(S, n)` را صدا بزنیم).
- چون بدترین حالت یک عملیات $O(n)$ است ، اگر n عملیات داشته باشیم (که $O(n)$ تای آن‌ها می‌توانند `MULTIPOP` باشند)، هزینه کل $O(n^2)$ خواهد بود.
- این تحلیل $O(n^2)$ اگرچه **صحیح** است، اما **دقیق (tight) نیست**.

تحلیل تجمعی (Aggregate Analysis):
بیایید با استفاده از تحلیل تجمعی، مرز بهتری پیدا کنیم.

- **بینش کلیدی:** اگرچه یک `MULTIPOP` می‌تواند گران باشد، اما هر دنباله از n عملیات `POP` ، `PUSH` و `MULTIPOP` روی یک پشته خالی، حداکثر $O(n)$ هزینه دارد.
- **چرا؟** یک شیء تنها زمانی می‌تواند از پشته `POP` شود (چه توسط `POP` عادی و چه درون `MULTIPOP`) که قبلاً `PUSH` شده باشد.
- تعداد کل فراخوانی‌های `POP` (شامل فراخوانی‌های درون `MULTIPOP`) حداکثر برابر با تعداد کل عملیات‌های `PUSH` است.
- در یک دنباله n عملیاتی، ما حداکثر n عملیات `PUSH` می‌توانیم داشته باشیم.
- بنابراین، تعداد کل `POP` ها (شامل `MULTIPOP`) حداکثر n است. هزینه کل `PUSH` ها نیز حداکثر n است.
- **نتیجه:** هزینه کل ($T(n)$) برای هر دنباله n عملیاتی، $O(n)$ است.
- **هزینه سرشکن‌شده:** $T(n)/n = O(n)/n = O(1)$.

تأکید مجدد: ما از استدلال احتمالی استفاده نکردیم. ما یک مرز بدترین حالت $O(n)$ را برای *کل دنباله* n عملیاتی نشان دادیم و سپس این هزینه کل را بر n تقسیم کردیم تا هزینه سرشکن‌شده به دست آید.

مثال ۲: افزایش‌دهنده شمارنده دودویی (Incrementing a Binary Counter)

به عنوان مثال دوم تحلیل تجمعی، پیاده‌سازی یک شمارنده دودویی (binary counter) k-بیتی را در نظر بگیرید که از 0 به بالا می‌شمارد. ما از یک آرایه $A[0..k-1]$ استفاده می‌کنیم.

```
INCREMENT(A)
    i = 0
    while i < A.length and A[i] == 1
        A[i] = 0
        i = i + 1
    if i < A.length
        A[i] = 1
```

هزینه `INCREMENT` :
هزینه این عملیات، خطی با تعداد بیت‌هایی است که فلیپ (Flip) می‌شوند (از 0 به 1 یا 1 به 0).

تحلیل ساده‌لوحانه:

- بدترین حالت (Worst-case) برای یک `INCREMENT` چه زمانی است؟ زمانی که تمام آرایه A پر از 1 باشد. (مثلاً 1111...1). در این حالت، هزینه $\Theta(k)$ است.
- بنابراین، یک دنباله از n عملیات `INCREMENT` در بدترین حالت $O(nk)$ هزینه خواهد داشت.
- باز هم، این تحلیل صحیح است، اما دقیق (tight) نیست.

تحلیل تجمعی:
ما می‌توانیم تحلیل خود را دقیق‌تر کنیم و نشان دهیم هزینه n عملیات `INCREMENT` (با شروع از صفر) $O(n)$ است.

- **مشاهده کلیدی:** همه بیت‌ها در هر بار `INCREMENT` فلیپ نمی‌شوند. (به جدول شکل 17.2 در PDF مراجعه کنید).
- بیت $A[0]$ هر بار `INCREMENT` فلیپ می‌شود (در n عملیات، n بار فلیپ می‌شود).
- بیت $A[1]$ یک در میان فلیپ می‌شود (در n عملیات، $\lfloor n/2 \rfloor$ بار فلیپ می‌شود).
- بیت $A[2]$ هر چهار بار یکبار فلیپ می‌شود (در n عملیات، $\lfloor n/4 \rfloor$ بار فلیپ می‌شود).
- به طور کلی، بیت $A[i]$ در یک دنباله n عملیاتی، $\lfloor n/2^i \rfloor$ بار فلیپ می‌شود.

محاسبه کل فلیپ‌ها ((T(n)):
تعداد کل فلیپ‌ها در n عملیات برابر است با:

$$T(n) = \sum_{i=0}^{k-1} \lfloor \frac{n}{2^i} \rfloor$$

ما می‌توانیم این مجموع را با یک سری هندسی بی‌نهایت کران‌دار کنیم:

$$T(n) = \sum_{i=0}^{k-1} \lfloor \frac{n}{2^i} \rfloor \sum_{i=0}^{\infty} \frac{n}{2^i} = n \sum_{i=0}^{\infty} (\frac{1}{2})^i = n \cdot 2 = 2n$$

- **نتیجه:** هزینه کل بدترین حالت برای n عملیات `INCREMENT` (با شروع از صفر) $O(n)$ است.
- **هزینه سرشکن‌شده:** $T(n)/n = O(n)/n = O(1)$.

حل تمرین‌های بخش ۱۷.۱

تمرین ۱۷.۱-۱ (MULTIPUSH)

> **سوال:** اگر مجموعه عملیات پشته شامل عملیات `MULTIPUSH` نیز باشد که **k** آیتم را روی پشته PUSH می‌کند، آیا مرز **O(1)** برای هزینه سرشکن‌شده عملیات پشته همچنان پابرجا خواهد بود؟

پاسخ: خیر، مرز $O(1)$ لزوماً پابرجا نخواهد بود. هزینه سرشکن‌شده می‌تواند $O(n)$ باشد.

تحلیل تجمعی:

1. بیایید هزینه عملیات `MULTIPUSH(S, k)` را برابر k در نظر بگیریم (زیرا k آیتم را PUSH می‌کند).
2. تحلیل تجمعی قبلی ما (برای `MULTIPOP`) متکی بر این واقعیت بود که تعداد کل `POP` ها نمی‌تواند از تعداد کل `PUSH` ها بیشتر باشد و تعداد کل `PUSH` ها توسط n (تعداد کل عملیات) محدود می‌شد.

3. با `MULTIPUSH` ، این فرض دوم دیگر برقرار نیست.
4. **بدترین حالت:** دنباله‌ای از n عملیات را در نظر بگیرید که همگی `MULTIPUSH(S, n)` باشند.
- عملیات اول: `MULTIPUSH(S, n)` . هزینه = n.
- عملیات دوم: `MULTIPUSH(S, n)` . هزینه = n.
- ...
- عملیات n-ام: `MULTIPUSH(S, n)` . هزینه = n.
5. **هزینه کل ($T(n)$):** هزینه کل برای این n عملیات $n \times n = n^2$ است.

$$T(n) = O(n^2)$$

6. **هزینه سرشکن‌شده:**

$$\text{Amortized Cost} = T(n)/n = O(n^2)/n = O(n)$$

بنابراین، هزینه سرشکن‌شده $O(1)$ نخواهد بود.

تمرین ۱۷.۱-۲ (DECREMENT)

> **سوال:** نشان دهید که اگر عملیات `DECREMENT` (کاهش شمارنده) در مثال شمارنده **k**-بیتی گنجانده شود، **n** عملیات می‌تواند تا **Θ(nk)** زمان هزینه داشته باشد.

پاسخ: تحلیل تجمعی $O(1)$ برای `INCREMENT` به این واقعیت بستگی داشت که شمارنده از 0 شروع می‌کند و بیت $A[i]$ تنها زمانی فلیپ می‌شود که همه بیت‌های $A[0]$ تا

$$A[i-1]$$

برابر 1 باشند. `DECREMENT` این الگو را می‌شکند.

اثبات با ارائه دنباله بدترین حالت:
ما می‌توانیم یک دنباله از n عملیات `INCREMENT` و `DECREMENT` بسازیم که هر عملیات $\Theta(k)$ هزینه داشته باشد.

1. **حالت اولیه:** شمارنده را روی حالتی تنظیم کنید که `INCREMENT` بدترین هزینه را دارد. مثلاً $0111\ldots1$ (یک 0 و $k-1$ تا 1).
2. **عملیات ۱:** `INCREMENT(A)` را روی $0111\ldots1$ اجرا کنید.
- این عملیات $k-1$ بیت 1 را به 0 فلیپ می‌کند و بیت 0 را به 1 فلیپ می‌کند.
- نتیجه: $1000\ldots0$.
- هزینه: $\Theta(k)$.
3. **عملیات ۲:** `DECREMENT(A)` را روی $1000\ldots0$ اجرا کنید. (عملیات `DECREMENT` برعکس `INCREMENT` عمل می‌کند: بیت‌ها را از 0 به 1 فلیپ می‌کند تا به 1 برسد و آن را 0 کند).
- این عملیات $k-1$ بیت 0 را به 1 فلیپ می‌کند و بیت 1 را به 0 فلیپ می‌کند.
- نتیجه: $0111\ldots1$.
- هزینه: $\Theta(k)$.
4. **ادامه دنباله:** ما می‌توانیم این دو عملیات (`INCREMENT` و `DECREMENT`) را $n/2$ بار تکرار کنیم.

تحلیل تجمعی:

- هر جفت عملیات (یک `INCREMENT` و یک `DECREMENT`) $\Theta(k) + \Theta(k) = \Theta(k)$ هزینه دارد.
- برای n عملیات ($n/2$ جفت)، هزینه کل $T(n)$ خواهد بود:

$$T(n) = (n/2) \times \Theta(k) = \Theta(nk)$$

- بنابراین، هزینه سرشکن‌شده $T(n)/n = \Theta(k)$ است، نه $O(1)$.

تمرین ۳-۱۷.۱ (هزینه i اگر i توان ۲ باشد)

> **سوال:** فرض کنید **n** عملیات روی یک ساختمان داده انجام می‌دهیم که هزینه عملیات **i**-ام برابر **i** است اگر **i** توان دقیقی از 2 باشد، و در غیر این صورت 1 است. از تحلیل تجمعی برای تعیین هزینه سرشکن‌شده هر عملیات استفاده کنید.

پاسخ: هزینه سرشکن‌شده هر عملیات $O(1)$ است.

تحلیل تجمعی:

1. ما باید هزینه کل $T(n)$ برای n عملیات را محاسبه کنیم.

$$T(n) = \sum_{i=1}^{n} c_i$$

2. هزینه c_i به صورت زیر تعریف شده است:

$$c_i = \begin{cases} i & \text{if } i \text{ is a power of } 2 \\ 1 & \text{otherwise} \end{cases}$$

3. می‌توانیم $T(n)$ را به دو بخش تفکیک کنیم: عملیات‌هایی که توان 2 نیستند و عملیات‌هایی که توان 2 هستند.

$$T(n) = \sum_{i \text{ not power of } 2}^{n} 1 + \sum_{i \text{ is power of } 2}^{n} i$$

4. **بخش اول (هزینه‌های 1):** تعداد کل عملیات n است. تعداد عملیات‌هایی که توان 2 هستند برابر $\lfloor \lg n \rfloor + 1$ است. بنابراین، تعداد عملیات‌هایی که هزینه 1 دارند، $n - (\lfloor \lg n \rfloor + 1)$ است. هزینه این بخش دقیقاً $n - (\lfloor \lg n \rfloor + 1)$ است که قطعاً $< n$ و $O(n)$ است.
5. **بخش دوم (هزینه‌های i):** ما باید مجموع توان‌های 2 تا n را محاسبه کنیم.

$$\sum_{i \text{ is power of } 2}^{n} i = \sum_{k=0}^{\lfloor \lg n \rfloor} 2^k$$

6. این یک سری هندسی است. ما می‌دانیم که $\sum_{k=0}^{m} 2^k = 2^{m+1} - 1$.
با جایگذاری $m = \lfloor \lg n \rfloor$، داریم:

$$\sum_{k=0}^{\lfloor \lg n \rfloor} 2^k = 2^{\lfloor \lg n \rfloor + 1} - 1$$

7. ما می‌دانیم که $\lfloor \lg n \rfloor \leq \lg n$.
بنابراین $2^{\lfloor \lg n \rfloor} \leq 2^{\lg n} = n$.
در نتیجه $2^{\lfloor \lg n \rfloor + 1} = 2 \cdot 2^{\lfloor \lg n \rfloor} \leq 2n$.
پس، مجموع بخش دوم $\leq 2n - 1$ و $O(n)$ است.
8. **هزینه کل ($T(n)$):**

$$T(n) = (\text{Cost of non-powers of } 2) + (\text{Cost of powers of } 2)$$

$$T(n) < n + (2n - 1) = 3n - 1$$

بنابراین، $T(n) = O(n)$.
9. **هزینه سرشکن‌شده:**

$$\text{Amortized Cost} = T(n)/n = O(n)/n = O(1)$$