ساختمان داده ها

تحلیل سرشکن شده
(Amortized Analysis)

مدرس: غیاثی‌شیرازی
دانشگاه فردوسی مشهد

تعریف پیچیدگی سرشکن شده

- فرض کنیم بر روی یک ساختمان داده n عمل انجام می شود که در مجموع زمان انجام آنها در بدترین حالت برابر $T(n)$ است. در این صورت می گوییم زمان اجرای هر عمل به طور سرشکن شده برابر $T(n)/n$ است.
- توجه: این تحلیل با تحلیل حالت متوسط متفاوت است زیرا در این تحلیل نه الگوریتم الزاما تصادفی فرض می شود و نه داده های ورودی تصادفی فرض می شوند.

مثال ۱: ساختمان داده Stack

- فرض کنیم در صورت پر شدن آرایه، طول آرایه دو برابر می شود.
- برای سادگی، فرض کنیم مکانیزمی برای کاهش طول آرایه نداریم.

- Push (x)
  - یک عنصر را به پشته اضافه می کند.
  - در بدترین حالت ممکن است نیاز به افزایش طول آرایه باشد بنابراین بدترین زمان اجرا $\Theta(n)$ است.
- Pop ()
  - یک عنصر را از بالای پشته حذف می کند.
  - بدترین زمان اجرا $\Theta(1)$ است.
  - بنابراین زمان n عمل push و pop متوالی در بدترین حالت $O(n^2)$ است.

مثال ۲: افزایش یک شمارنده دودویی

- فرض کنیم یک شمارنده k بیتی داریم.
- هزینه افزایش مقدار شمارنده = تعداد بیت های تغییر یافته $\Theta(k)$ –
- بنابراین طبق تحلیل بدترین حالت زمان n بار افزایش از مرتبه $O(nk)$ است.

روش های انجام تحلیل سرشکن شده

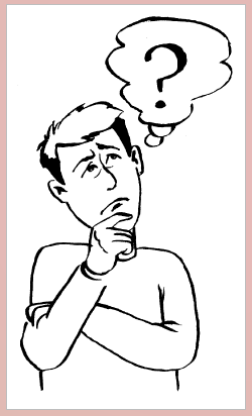
- Aggregate method (روش جمع زدن)
- Accounting method (روش حسابداری)
- Potential function method (روش تابع پتانسیل)

روش اول: روش جمعی (Aggregate)

- به طریقی یک کران بالا برای زمان اجرای n عمل به دست می آوریم.
- با تقسیم این کران بالا بر n، زمان سرشکن شده هر عمل به دست می آید.

مثال ۱: ساختمان داده Stack

- اگر n عمل push و pop انجام شود،



- زمان سرشکن شده هر عمل از مرتبه ..... است.

مثال ۱: ساختمان داده Stack

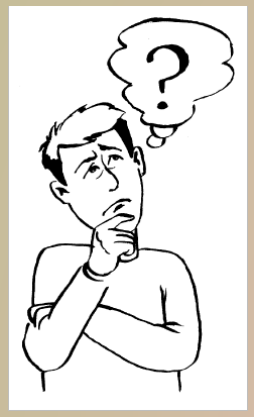
- فرض کنیم n عمل push و pop انجام شده است.
- با توجه به اینکه افزایش طول در زمان هایی که اندازه آرایه به مقادیر ۱، ۲، ۴، ۸، ۱۶ و ... می رسد انجام می شود و زمان اجرای عمل کپی داده ها هنگام افزایش طول، متناسب با طول فعلی آرایه است، حداکثر زمان کپی داده‌ها برابر است با:

$$1 + 2 + \cdots + 2\text{^}(\lfloor \log n \rfloor)$$

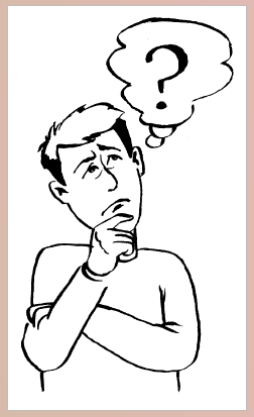
$$< n\left(1 + \frac{1}{2} + \frac{1}{4} + \cdots\right) = 2n$$

- با احتساب زمان n عمل push و pop، زمان کل برابر 3n می شود.
- زمان سرشکن شده هر عمل از مرتبه $\Theta(1)$ است.

مثال ۲: افزایش یک شمارنده دودویی



- فرض کنیم n عمل افزایش شمارنده انجام شده است.



- تعداد کل تغییرات بیتی برابر است با:

- زمان سرشکن شده تعداد تغییرات بیتی از مرتبه ..... است.

مثال ۲: افزایش یک شمارنده دودویی

- فرض کنیم n عمل افزایش شمارنده انجام شده است. در این صورت رقم اول هر بار، رقم دوم حداکثر n/2 بار، ... و رقم k ام حداکثر $\frac{n}{2^{k-1}}$ بار تغییر علامت می دهد.
- بنابراین تعداد کل تغییرات بیتی برابر است با:

$$n\left(1+\frac{1}{2}+\frac{1}{4}+\cdots+\frac{1}{2^{k-1}}\right)<2n$$

- بنابراین تعداد سرشکن شده تغییرات بیتی برابر ۲ است که از مرتبه $\Theta(1)$ است.

Counter value	A[7]	A[6]	A[5]	A[4]	A[3]	A[2]	A[1]	A[0]	Total cost
0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	1	1
2	0	0	0	0	0	0	1	0	3
3	0	0	0	0	0	0	1	1	4
4	0	0	0	0	0	1	0	0	7
5	0	0	0	0	0	1	0	1	8
6	0	0	0	0	0	1	1	0	10
7	0	0	0	0	0	1	1	1	11
8	0	0	0	0	1	0	0	0	15
9	0	0	0	0	1	0	0	1	16
10	0	0	0	0	1	0	1	0	18
11	0	0	0	0	1	0	1	1	19
12	0	0	0	0	1	1	0	0	22
13	0	0	0	0	1	1	0	1	23
14	0	0	0	0	1	1	1	0	25
15	0	0	0	0	1	1	1	1	26
16	0	0	0	1	0	0	0	0	31

روش دوم: روش حسابداری (Accounting Method)

- زمان سرشکن شده هر عمل را حدس می زنیم.
- هرگاه هزینه اجرای واقعی از هزینه سرشکن شده کمتر باشد، مابه التفاوت این دو مقدار را به اشیائی در ساختمان داده به عنوان اعتبار (credit) قرض می دهیم.
- هرگاه هزینه اجرای واقعی از هزینه سرشکن شده بیشتر باشد، از اعتباراتی که قبلا در اشیاء ساختمان داده ذخیره کرده ایم بر می داریم و این هزینه را می پردازیم.
- هزینه سرشکن شده عملیات مختلف می توانند متفاوت باشند.
- در این روش لازم است همواره اعتبار ذخیره شده نامنفی باشد.

مثال ۱: تحلیل ساختمان داده Stack

- زمان سرشکن شده هر عمل مقداری است که تعیین آن بر عهده تحلیل گر است. این مقادیر را چنین انتخاب می کنیم:

  pop=1 , push=3

- هزینه سرشکن شده و واقعی برای عمل pop یکسان است.

مثال ۱: ساختمان داده Stack
حسابداری در عمل push

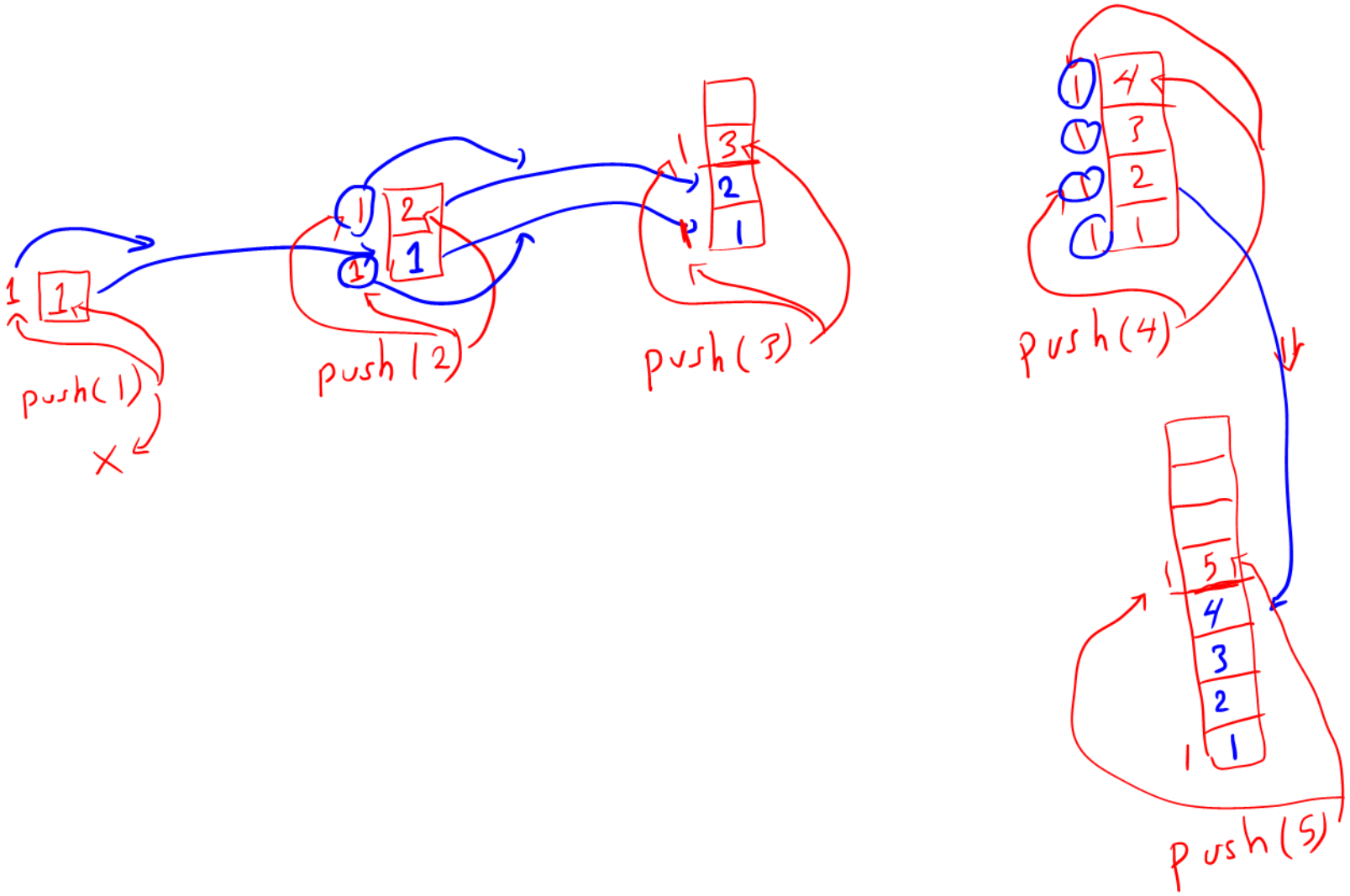
- فرض کنیم پس از افزایش طول آرایه اعتبار ذخیره شده صفر است.
- بنابراین در لحظه شروع (اعتبار صفر) آرایه ای به طول 2n داریم که n خانه آن پر است
- باید هزینه های سرشکن شده را به نحوی به خانه ها اختصاص دهید که هزینه عمل بعدی افزایش طول آرایه تامین شود. در گام بعدی طول آرایه 2n است و باید 4n شود و بنابراین به 2n اعتبار نیاز داریم.

مثال ۱: ساختمان داده Stack
حسابداری در عمل push

- فرض کنیم از آرایه ای به طول 2n تعداد n+i-1 خانه آن پر است ولی هنوز آرایه پر نیست. در عمل push بعدی ما ۳ هزینه سرشکن شده را چنین مصرف می کنیم.
  - ۱ هزینه برای push
  - ۱ هزینه را در خانه n+i ذخیره می کنیم.
  - ۱ هزینه را در خانه i ذخیره می کنیم.
- بنابراین هنگام پر شدن آرایه هر خانه به مقدار یک واحد اعتبار دارد.

مثال ۱: ساختمان داده Stack
حسابداری در عمل push

- اگر آرایه پر باشد و طول آن n باشد بنابراین نیاز به افزایش طول آرایه است که هزینه آن n است. این هزینه را از n اعتباری که قبلا در عناصر آرایه قرار داده ایم برمی داریم.
- پس از این عمل اعتبار ما صفر می شود.
- این تحلیل نشان می دهد که هزینه سرشکن شده عملیات push و pop از مرتبه $\Theta(1)$ است.


1
1
push(1)
X
1
2
1
push(2)
1
3
2
1
push(3)
1
4
3
2
1
push(4)
1
5
4
3
2
1
push(5)

مثال ۲: افزایش یک شمارنده دودویی

- فرض کنیم شمارنده از صفر شروع می شود.
- هزینه واقعی برابر تعداد تغییرات بیتی است.
- هزینه سرشکن شده را برای هر عمل ۲ در نظر می گیریم.
- در عمل افزایش شمارنده، تمام بیت های ۱ سمت راست ۰ و اولین بیت ۰ از سمت راست ۱ می شود.
- بنابراین در هر عمل افزایش شمارنده دقیقا یک تبدیل بیتی از ۰ به ۱ داریم.

مثال ۲: افزایش یک شمارنده دودویی
نحوه مصرف اعتبار سرشکن شده

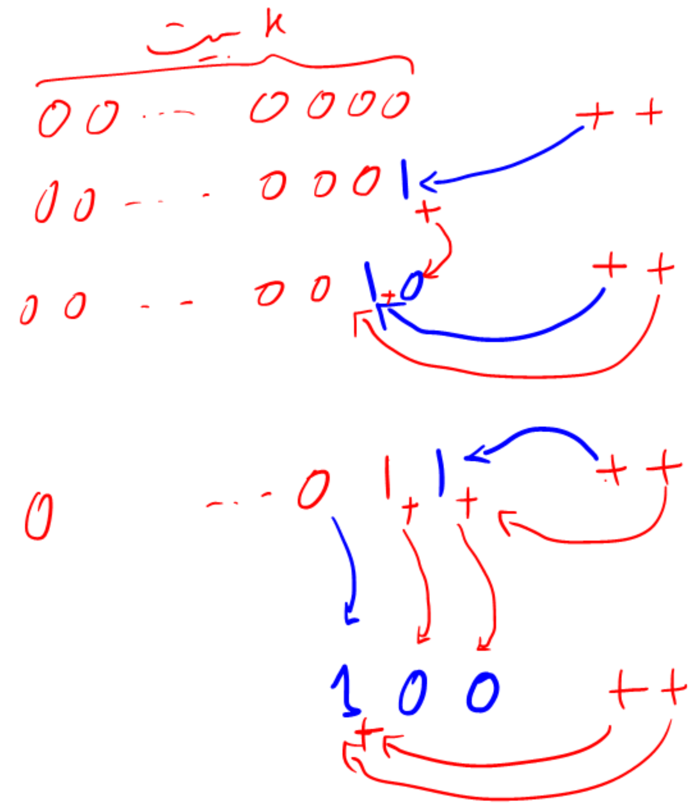
- در هر عمل افزایش شمارنده، یک واحد از دو واحد هزینه سرشکن شده را به هزینه واقعی تبدیل یک بیت از صفر به ۱ اختصاص می دهیم و **تصور می کنیم** که یک واحد دیگر را در همان بیت ذخیره کرده ایم.
- بنابراین همواره هر بیت ۱ دارای یک واحد اعتبار می باشد.

مثال ۲: افزایش یک شمارنده دودویی
اثبات امکان پرداخت هزینه واقعی از اعتبارات

- هزینه واقعی= هزینه تبدیل یک بیت از ۰ به ۱

  +هزینه تبدیل تمام بیتهای ۱ سمت راست به ۰
- یک واحد هزینه تبدیل یک بیت از ۰ به ۱ را از دو واحد هزینه سرشکن شده می پردازیم.
- یک واحد اعتباری که در بیت تغییر وضعیت داده به ۱ باید ذخیره شود را نیز از هزینه سرشکن شده می پردازیم.
- هزینه تبدیل بیت های ۱ به صفر را از اعتبار ذخیره شده در بیت های ۱ می پردازیم.

زمان سرشکن شده = 2



??? 0 1111
10000

در هر افزایش دقیقاً یک بیت
صفر به یک تغییر می‌کند

صرف تغییر 0 به 1 ++
کنار 1 جهت ذخیره می‌شود.

روش سوم: روش تابع پتانسیل (Potential Function Method)

- این روش مشابه روش حسابداری است با این تفاوت که اعتبارات را به جای آنکه به اشیاء درون ساختمان داده اختصاص دهیم، به کل ساختمان داده نسبت می دهیم.
- به اعتباری که در ساختمان داده ذخیره شده است، پتانسیل گفته می شود.
- معمولا برخلاف روش حسابداری که اعتبار اشیاء در تصور ما بود، در روش تابع پتانسیل مقدار اعتبار موجود در ساختمان داده مقداری است که از وضعیت کنونی ساختمان داده قابل محاسبه است.

روش تابع پتانسیل

- فرض کنیم بر روی یک ساختمان داده n عمل انجام شده است.
- فرض کنید هزینه واقعی عمل i ام برابر c_i و هزینه سرشکن شده آن برابر a_i باشد. همچنین فرض کنید مقدار تابع پتانسیل از Φ_{i-1} به Φ_i تغییر کرده است.
- برای آنکه تحلیل سرشکن شده صحیح باشد باید
$$c_i + \Phi_i - \Phi_{i-1} \leq a_i$$
- یعنی هزینه سرشکن شده عمل i ام باید از مجموع هزینه واقعی عمل i ام و اعتبار ذخیره شده در تابع پتانسیل بیشتر باشد.
- معمولا پتانسیل اولیه را صفر می گیریم. در این صورت برای درست بودن تحلیل تابع پتانسیل نباید هیچگاه منفی شود.

صحت تحلیل سرشکن شده

- از رابطه

$$c_i \leq a_i + \Phi_{i-1} - \Phi_i$$

- نتیجه می گیریم که:

$$\sum_{i=1}^{n} c_i \leq \sum_{i=1}^{n} (a_i + \Phi_{i-1} - \Phi_i) = \sum_{i=1}^{n} a_i + \Phi_0 - \Phi_n$$

$$\leq \sum_{i=1}^{n} a_i$$

- بنابراین مجموع هزینه سرشکن شده n عمل از هزینه واقعی آنها بیشتر است.

مثال ۱: تحلیل ساختمان داده Stack

- مقدار تابع پتانسیل Φ را بر روی ساختمان داده پشته چنین تعریف می کنیم.

ماکسیمم صفر و تفاضل تعداد خانه های پر و خالی

$\max(0, \text{تعداد خانه‌های پر} - \text{تعداد خانه‌های خالی})$

مثال ۱: تحلیل ساختمان داده Stack

تحلیل عمل pop

- فرض کنیم در گام i-1 ظرفیت پشته s است و تعداد n عنصر در پشته قرار دارد. داریم:

$$\Phi_{i-1} = \max\big(0, n - (s - n)\big) = \max(0, 2n - s)$$

- پس از عمل pop داریم:

$$\Phi_i = \max(0, 2(n-1) - s) \leq \max(0, 2n - s) = \Phi_{i-1}$$

- بنابراین در عمل pop هزینه سرشکن شده کافی است در رابطه زیر صدق کند

$$a_i \geq c_i \geq c_i + \Phi_i - \Phi_{i-1}$$

- بنابراین هزینه سرشکن شده pop را 1 (c_i) در نظر می گیریم.

مثال ۱: تحلیل ساختمان داده Stack
هزینه سرشکن شده push

- فرض کنیم در گام i-1 ظرفیت پشته s است و تعداد n عنصر در پشته قرار دارد. داریم:

$$\Phi_{i-1} = \max\big(0, n - (s - n)\big) = \max(0, 2n - s)$$

- در صورت وجود فضای خالی برای عمل push داریم:

$$\Phi_i = \max(0, 2(n+1) - s) \leq \Phi_{i-1} + 2$$

- همچنین هزینه واقعی push برابر است با: $c_i = 1$
- هزینه سرشکن شده push کافی است در رابطه زیر صدق کند.

$$a_i \geq 1 + 2 \geq c_i + \Phi_i - \Phi_{i-1}$$

مثال ۱: تحلیل ساختمان داده Stack
هزینه سرشکن شده push

- فرض کنیم در گام i-1 ظرفیت پشته s است و تعداد n عنصر در پشته قرار دارد. داریم:

$$\Phi_{i-1} = \max\big(0, n - (s - n)\big) = \max(0, 2n - s)$$

- در صورت عدم وجود فضای خالی برای عمل push داریم (s=n):

$$\Phi_{i-1} = 2n - n = n$$

$$\Phi_i = \max(0, 2(n+1) - 2s) = 2$$

- همچنین هزینه واقعی push برابر است با: $c_i = n + 1$
- هزینه سرشکن شده عمل push کافی است در رابطه زیر صدق کند.

$$a_i \geq c_i + \Phi_i - \Phi_{i-1} = n + 1 + 2 - n$$

- پس هزینه سرشکن شده عمل push برابر 3 است، یعنی $\Theta(1)$.

مثال ۲: افزایش یک شمارنده دودویی

- فرض کنیم شمارنده از صفر شروع می شود.
- هزینه واقعی برابر تعداد تغییرات بیتی است.
- تابع پتانسیل را برابر تعداد بیت های ۱ تعریف می کنیم.
- تابع پتانسیل همواره نامنفی است.
- فرض کنیم در عمل افزایش شمارنده b بیت از ۱ به صفر تغییر کنند. در این صورت هزینه سرشکن شده باید در رابطه زیر صدق کند.

$$a_i \geq c_i + \Phi_i - \Phi_{i-1} = b + 1 - (b-1) = 2$$

مطالعه بیشتر

- فصل ۱۷ کتاب CLRS که از ص ۴۵۱ شروع می شود.
- اسلاید های درس آقای Sartaj Sahni از دانشگاه فلوریدا
- توجه: مثال پشته ارائه شده در این اسلاید با مثال پشته کتاب متفاوت است. با توجه به واقعی نبودن مثال پشته کتاب، مدرس مثال را تغییر داده و تحلیل آن نیز به تناسب تغییر کرده است.