دانشکده مهندسی کامپیوتر

جزوه درس

ساختمان‌های داده

استاد درس: سید صالح اعتمادی

پاییز ۱۳۹۸

جلسه ۱۶

آرایه های پویا و اصل سرشکن کردن

هادی شیخی - ۱۳۹۸/۸/۲۰

جزوه جلسه ۱۶ام مورخ ۱۳۹۸/۸/۲۰ درس ساختمان‌های داده تهیه شده توسط هادی شیخی. در جهت مستند کردن مطالب درس ساختمان های داده، بر آن شدیم که از دانشجویان جهت مکتوب کردن مطالب کمک بگیریم. هر دانشجو می‌تواند برای مکتوب کردن یک جلسه داوطلب شده و با توجه به کیفیت جزوه از لحاظ کامل بودن مطالب، کیفیت نوشتار و استفاده از اشکال و منابع کمک آموزشی، حداکثر یک نمره مثبت از بیست نمره دریافت کند.

۱.۱۶ آرایه های پویا [۱] [۲]

یکی از مشکلات آرایه های عادی اندازه ثابت آنهاست. آرایه های پویا نوعی ساختمان داده با قابلیت تغییر سایز هستند. یک آرایه پویا معمولا با ایجاد یک آرایه ابتدایی در حافظه ایجاد میشود. عناصر آرایه پویا (اگر با ساختمان داده دیگری پیاده سازی نشده باشد) به صورت پیوسته در حافظه قرار دارند و هنگامی که دیگر جایی برای ذخیره عناصر نبود با توجه به پیاده سازی، حافظه جدیدی با اندازه بزرگتر برای ذخیره سازی تخصیص میدهیم و همه مقادیر قبلی را در آن مقداردهی میکنیم.

```
Pushback(value):
 if size = capacity then
    allocate newArray[2 * capacity];
    for i from 0 to size - 1 do
        newArray[i] = arrary[i];
    end
    free array;
    array = newArray;
    capacity = 2 * capacity;
end
array[size] = value;
size = size + 1;
```

Algorithm 1: Add a new item to dynamic array

۱.۱.۱۶ نرخ رشد

نرخ رشد در ارایه های پویا به عوامل متعددی وابسته است مانند اهمیت و پیچیدگی فضا و زمان، الگوریتم های تخصیص حافظه و ... نرخ رشد ایده آل برای ایجاد آرایه های جدید عدد طلایی ۱.۶۱۸۰۳۳ است که برای ساده سازی محاسبات در اکثر پیاده سازی ها نرخ رشد را ۲ در نظر گرفته اند. [۳]

۲.۱.۱۶ پیچیدگی زمان و کارایی

پیچیدگی زمان ارایه های پویا مشابه ارایه های عادی است:

- مقدار دهی یا تغییر عنصر در مکان خاصی از لیست (constant time)
- پیمایش لیست (linear time)
- وارد کردن یا حذف کردن عنصر در میان لیست (linear time)
- وارد یا حذف کردن عنصر در آخر لیست (constant amortized time)

نکته مهم که باید در هنگام کار با لیست ها و ارایه ها در نظر گرفت اینست که هنگام پاس دادن آنها به توابع باید آدرس ابتدای آنها را پاس داد نه کل لیست را، در غیر اینصورت پیچیدگی زمان الگوریتم برای لیست های به اندازه کافی بزرگ بالا میرود و الگوریتم کارایی خود را از دست میدهد.

۲.۱۶ اصل سرشکن

برای بیان اصل سرشکن روش ها متفاوتی ارائه شد به طور خلاصه درباره آن بحث شد.

۱.۲.۱۶ بیان میانگین

برای محاسبه هزینه هر عمل در دنباله ای از اعمال میتوان مجموع هزینه کل را بر تعداد اعمال انجام شده تقسیم کرد و هزینه میانگین برای انجام هر عمل را بدست آورد. با فرض c به عنوان هزینه میانگین برای هر عمل میتوان نوشت :

$$c = \frac{Cost(AllOperations)}{n}$$

حال از این متد برای محاسبه هزینه میانگین برای اضافه کردن عنصر جدید در یک آرایه پویا استفاده میکنیم. اینگونه بیان میکنیم که از آنجایی که نرخ رشد ما ۲ و اندازه ابتدایی آرایه ۱ است، برای عنصر هایی که در جایگاه بعدی توان های دو هستند باید یک آرایه جدیدی ساخته شود و طبق کد هایی که قبل تر ارائه شد برای ساخت آرایه جدید با طول n به طور تقریب باید تعداد n عملیات انجام شود.

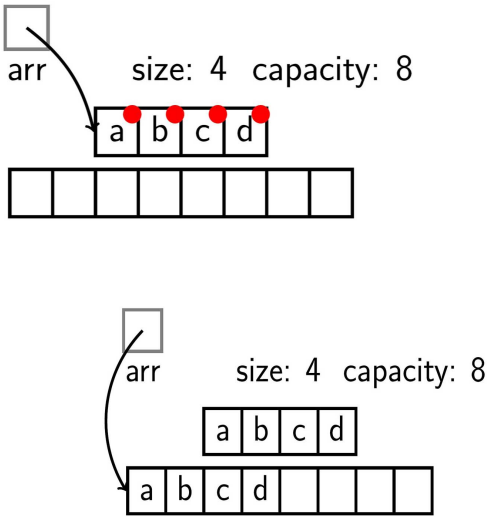
$$\text{let } c_i = \text{cost of i'th insertion}$$

$$c_i = 1 + \begin{cases} i-1 & \text{if i - 1 is a power of 2} \\ 0 & \text{otherwise} \end{cases}$$

$$\frac{\sum_{i=1}^{n} c_i}{n} = \frac{n + \sum_{j=1}^{\lfloor \log_2^n \rfloor} 2^j}{n} = \frac{O(n)}{n} = O(1)$$

۲.۲.۱۶ بیان صندوقدار

[۴] اگر انجام هر عمل واحد را به عنوان خرج کردن یک سکه در نظر بگیریم ، با اضافه کردن یک عنصر یک سکه خرج میکنیم، حال فرض کنید در حین اضافه کردن عنصر جدید ما سه سکه خرج کنیم یعنی یکی برای اضافه کردن و دو سکه برای مکان جدید اضافه شده و یکی از مکان های نیمه اول آرایه ذخیره کنیم . برای ساختن آرایه جدید از سکه های ذخیره شده استفاده میکنیم.



شکل ۱.۱۶: ذخیره کردن و استفاده کردن از سکه ها

Bibliography

[1] GeekForGeeks, "How dynamic array works." https://www.geeksforgeeks.org/how-do-dynamic-arrays-work/.

[2] InterviewCake, "What is dynamic array." https://www.interviewcake.com/concept/java/dynamic-array.

[3] Wikipedia, "Golden ratio." https://en.wikipedia.org/wiki/Golden_ratio.

[4] A. S. Kulikov, "Data structures and algorithms specialization." https://www.coursera.org/specializations/data-structures-algorithms.