ساختمان های داده لیستی!؟

ساختمان های داده و الگوریتم

سجاد هاشمیان، بهار ۱۴۰۰

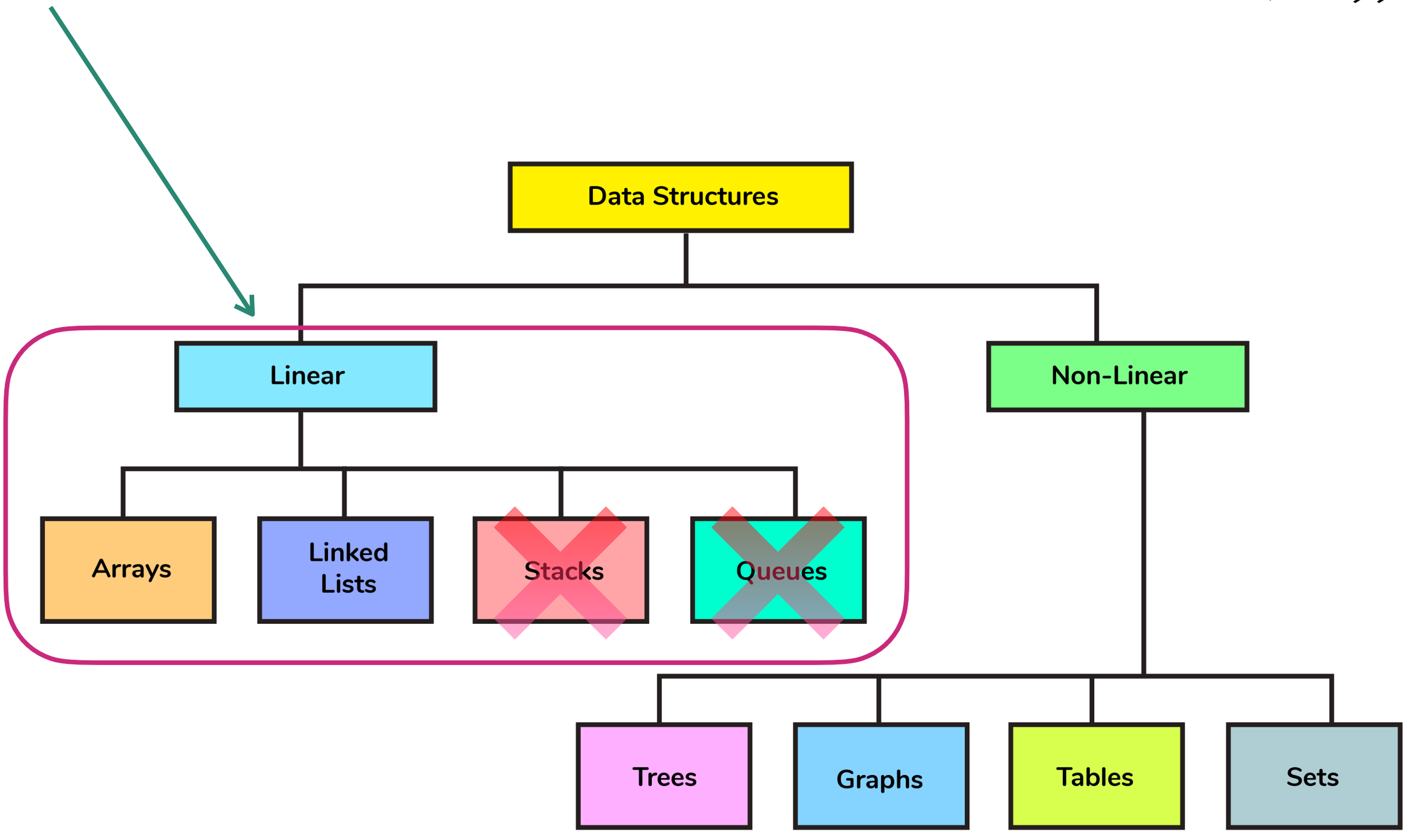
لیست اصلا چیه؟

برگرفته از ویکی‌پدیا

- یک نوع داده انتزاعی است که نمایانگر تعداد **شمارش‌پذیری** از مقادیر مرتب است، به‌طوری که یک مقدار ممکن است بیش از یک بار مشاهده شود.
- یک لیست نمایش کامپیوتری از مفهوم دنباله‌های متناهی در ریاضیات است.
- به طور غیردقیق منظورمان یک فهرست از عناصر است.
- به طور غیردقیق‌تر منظورمان همان ساختمان‌داده های خطی است.

لیست اصلا چیه؟

برگرفته از وبسایت interviewbit

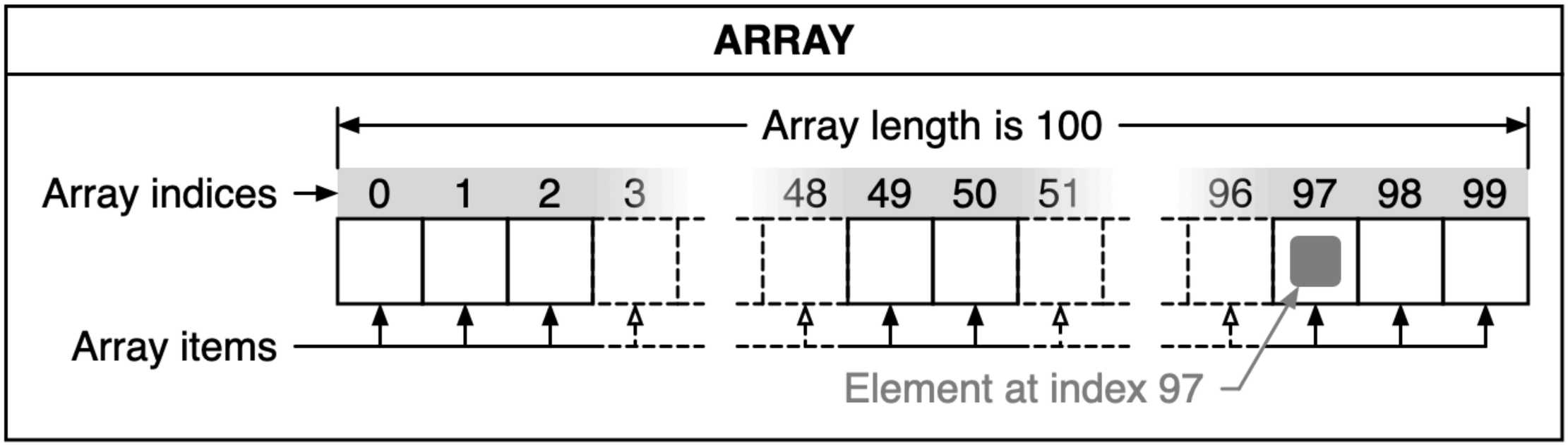


انواع لیست

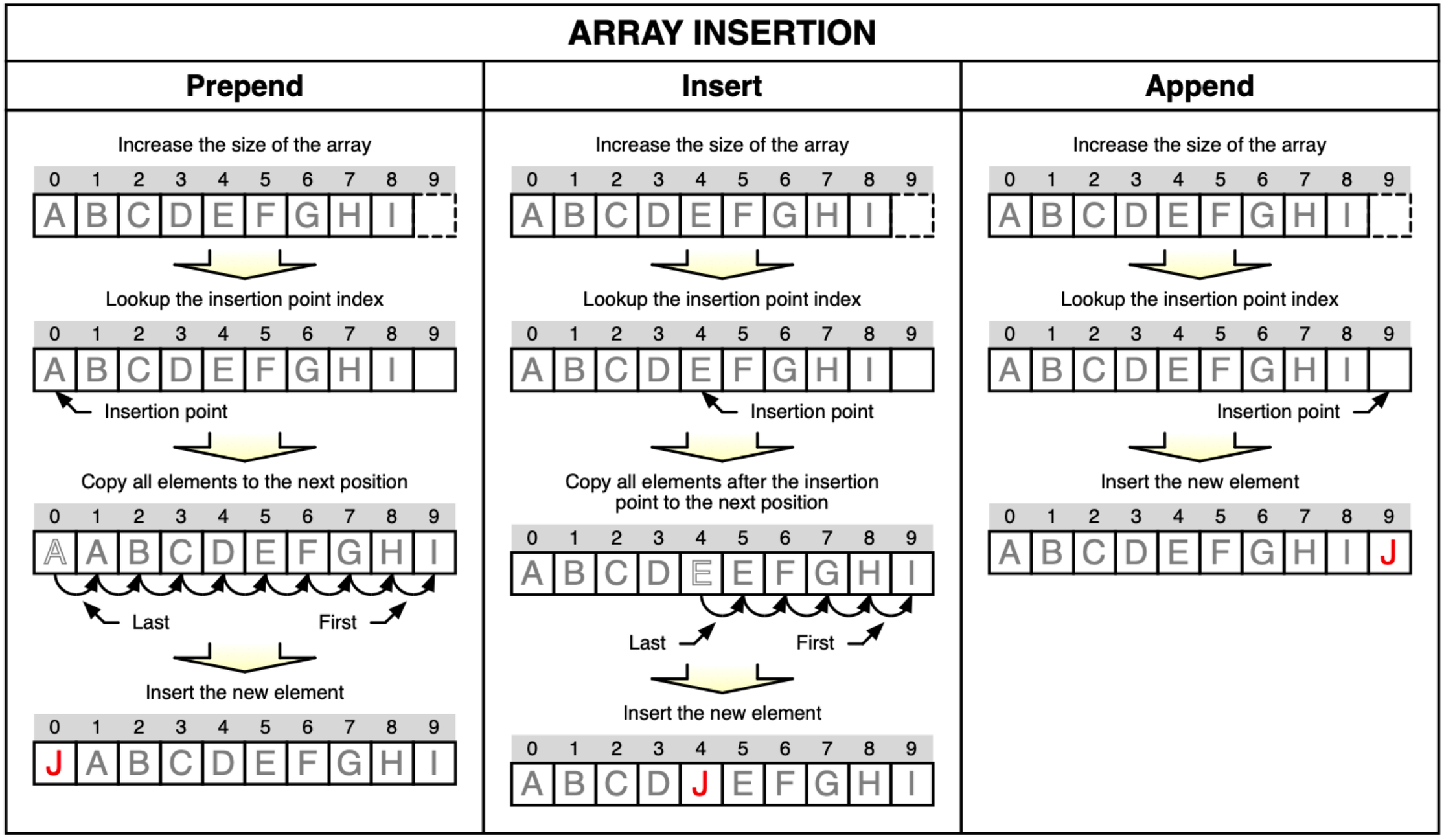
Armando Giuseppe Bonatto Minella blog, linked-in

- آرایه
- لیست‌های پیوندی
  - لیست‌های دو طرفه
  - لیست‌های حلقوی
- **لیست‌های پویا**
- لیست‌های درختی (البته اگر ترجمه دقیق رو بیخیال شیم، باید بگیم ساختمان‌داده‌های درختی)
- لیست با دسترسی تصادفی (Universal Data Structures)
- ...

آرایه

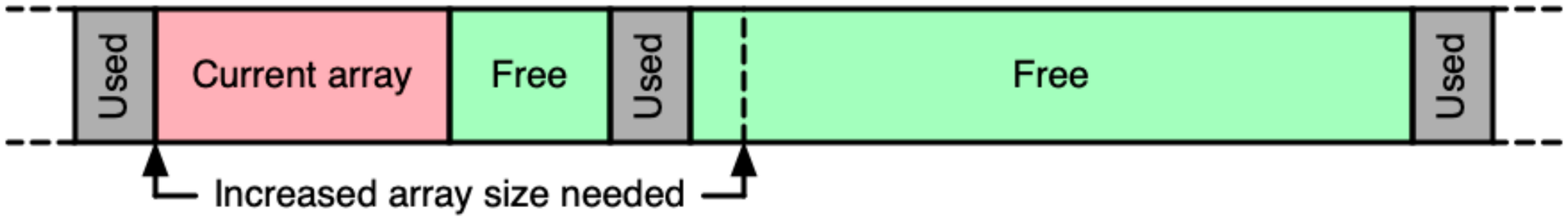

ARRAY
Array length is 100
Array indices
0
1
2
3
48
49
50
51
96
97
98
99
Array items
Element at index 97

آرایه

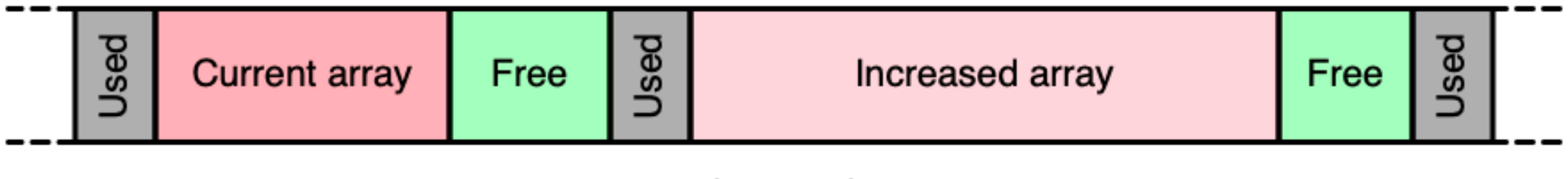

ARRAY INSERTION
Prepend
Increase the size of the array
0 1 2 3 4 5 6 7 8 9
A B C D E F G H I
Lookup the insertion point index
Insertion point
Copy all elements to the next position
A A B C D E F G H I
Last
First
Insert the new element
J A B C D E F G H I
Insert
Increase the size of the array
Lookup the insertion point index
Insertion point
Copy all elements after the insertion point to the next position
A B C D E E F G H I
Last
First
Insert the new element
A B C D J E F G H I
Append
Increase the size of the array
Lookup the insertion point index
Insertion point
Insert the new element
A B C D E F G H I J

ARRAY REALLOCATION

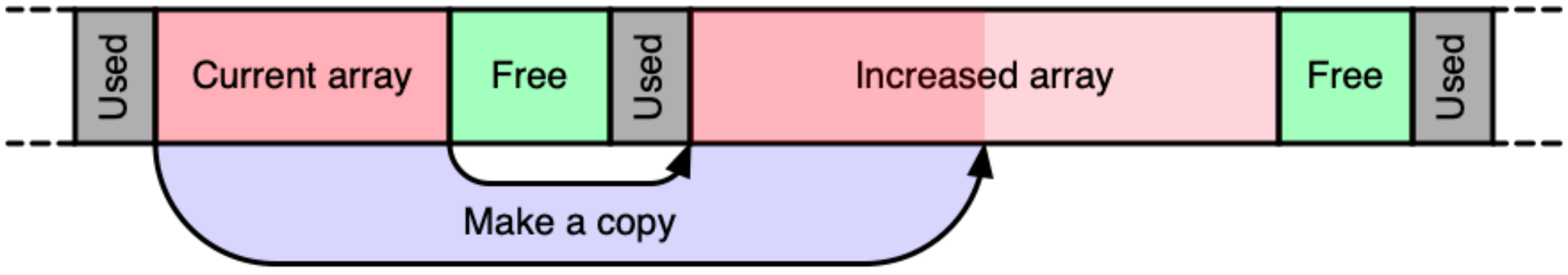
Occours when there's not enough contiguous space to fit the increased array size



Allocate a new memory area which can fit the increased array size



Copy all elements from the current array to an increased array



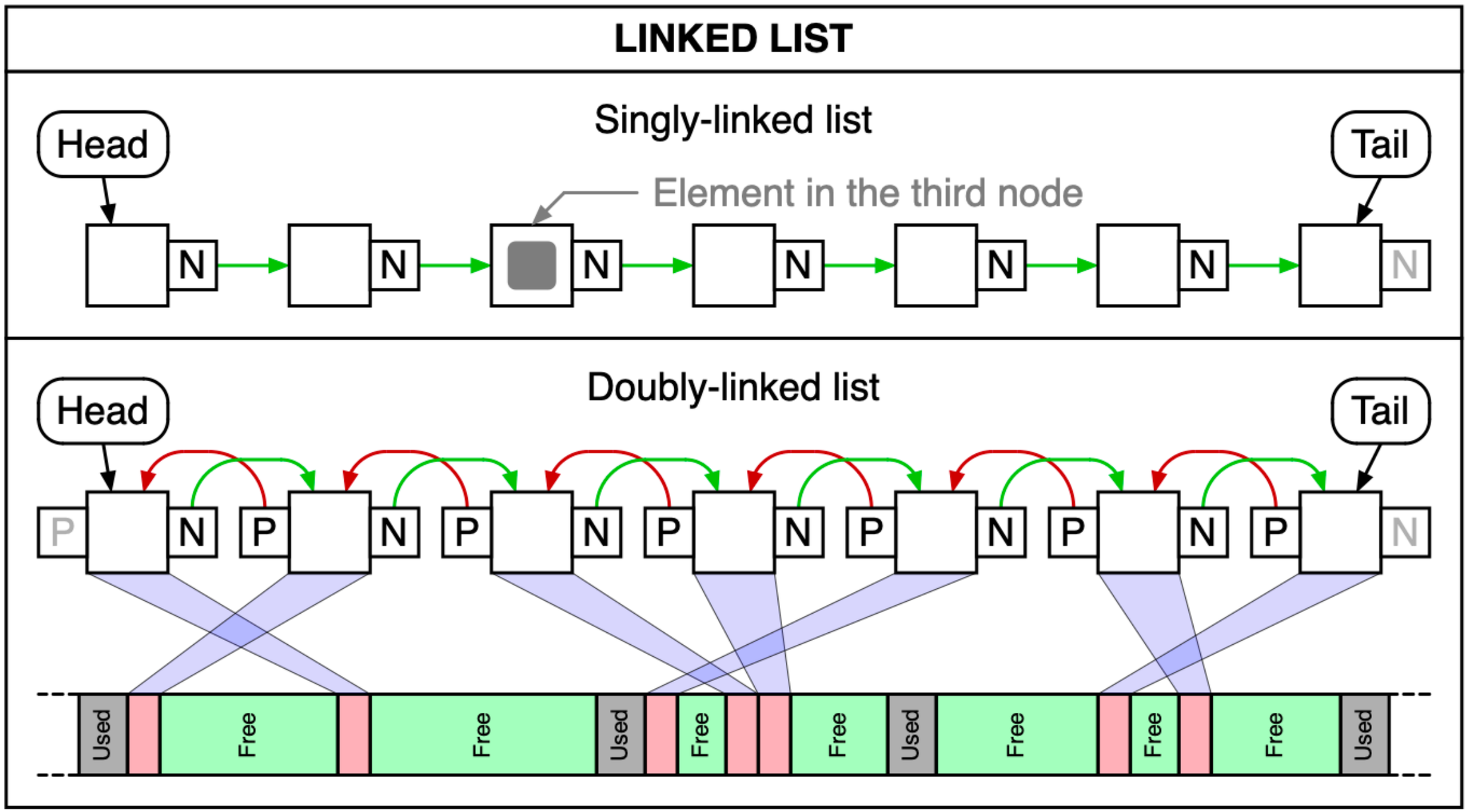
Free the memory area previously occupied by the array before the size increment

انواع لیست

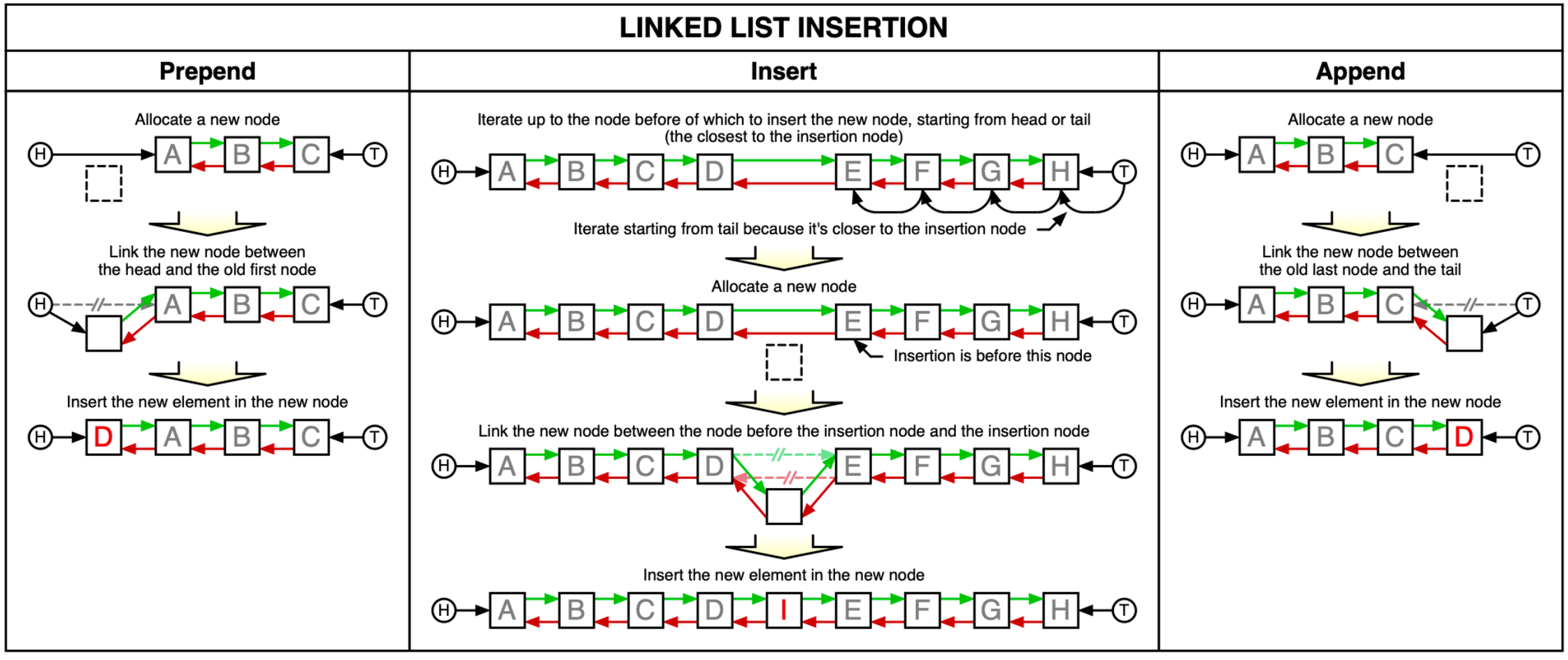
Armando Giuseppe Bonatto Minella blog, linked-in

- آرایه
- لیست‌های پیوندی
  - لیست‌های دو طرفه
    - لیست‌های حلقوی
  - **لیست‌های پویا**
  - لیست‌های درختی
  - لیست با دسترسی تصادفی
  - ...

لیست پیوندی


LINKED LIST
Singly-linked list
Head
Tail
Element in the third node
N
Doubly-linked list
Head
Tail
P
N
Used
Free

لیست پیوندی


LINKED LIST INSERTION
Prepend
Allocate a new node
Link the new node between the head and the old first node
Insert the new element in the new node
Insert
Iterate up to the node before of which to insert the new node, starting from head or tail (the closest to the insertion node)
Iterate starting from tail because it's closer to the insertion node
Allocate a new node
Insertion is before this node
Link the new node between the node before the insertion node and the insertion node
Insert the new element in the new node
Append
Allocate a new node
Link the new node between the old last node and the tail
Insert the new element in the new node

تقابل مشکلات

آرایه در برابر لیست پیوندی

	Linked list	Array	Dynamic array	Balanced tree	Random-access list	Hashed array tree
Indexing	Θ(*n*)	Θ(1)	Θ(1)	Θ(log n)	Θ(log n)[11]	Θ(1)
Insert/delete at beginning	Θ(1)	N/A	Θ(*n*)	Θ(log n)	Θ(1)	Θ(*n*)
Insert/delete at end	Θ(1) when last element is known; Θ(*n*) when last element is unknown	N/A	Θ(1) amortized	Θ(log *n*)	N/A [11]	Θ(1) amortized
Insert/delete in middle	search time + Θ(1)[12][13]	N/A	Θ(*n*)	Θ(log *n*)	N/A [11]	Θ(*n*)
Wasted space (average)	Θ(*n*)	0	Θ(*n*)[14]	Θ(*n*)	Θ(*n*)	$\Theta(\sqrt{n})$

برگرفته از ویکی‌پدیا

انواع لیست

Armando Giuseppe Bonatto Minella blog, linked-in

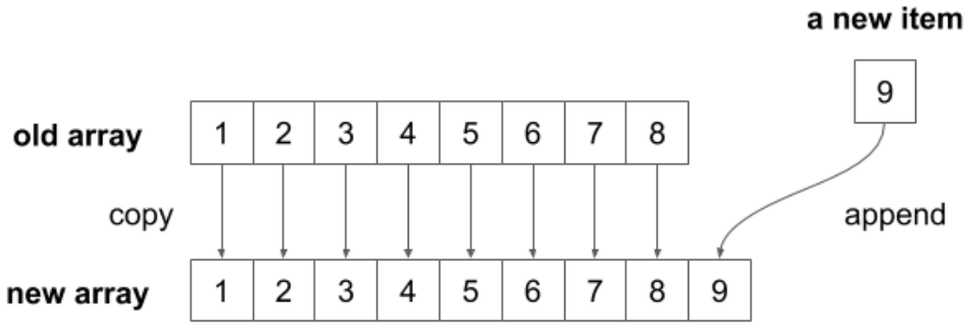
- آرایه
- لیست‌های پیوندی
  - لیست‌های دو طرفه
    - لیست‌های حلقوی
  - **لیست‌های پویا**
  - لیست‌های درختی
  - لیست با دسترسی تصادفی
  - ...

آرایه‌های پویا

بلقوه آرایه، بلفطره لیست

آرایه‌ای است که می‌تواند تغییر اندازه دهد و اجازه دهد عناصری به آن(انتها) اضافه یا از آن حذف شود. در یک آرایه پویا در ابتدای کار حافظه اختصاص داده نمی‌شود؛ به‌طوری‌که اندازه‌اش غیرقابل تغییر باشد.

مسئله) چگونه با یک آرایه به صورت پویا برخورد کنیم؟



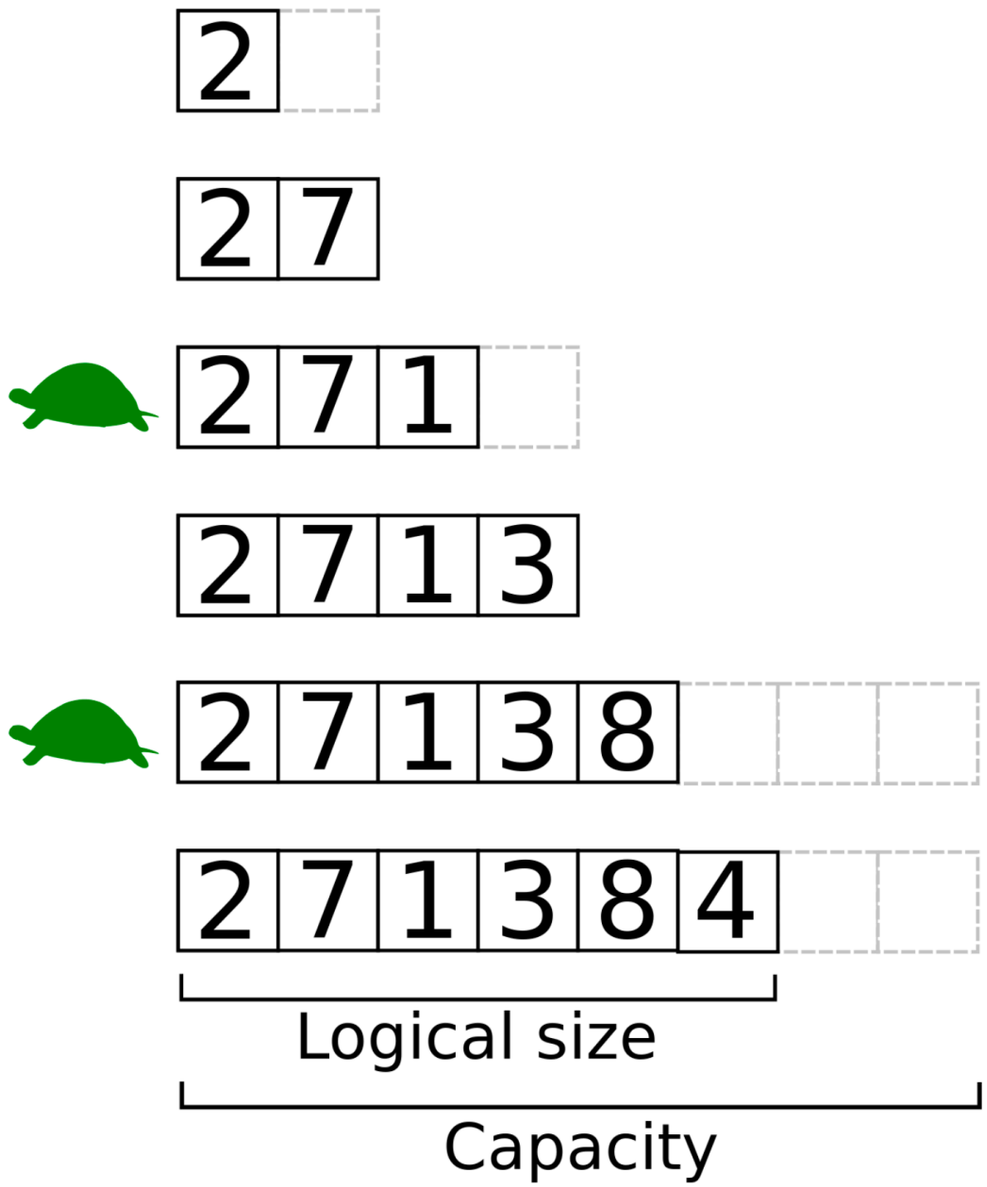
آرایه پویا

آرایه ها و ظرفیت پویا با اندازه محدود

- ساده‌ترین آرایه پویا با اختصاص دادن آرایه‌ای به طول ثابت (معمولاً بزرگتر از تعداد عناصر در لحظه) ساخته می‌شود که بعد آن را به دو قسمت تقسیم می‌کنند:
  - عناصر آرایه پویا در ابتدا **آرایه زیرین** به صورت پیوسته ذخیره می شوند و موقعیت های باقی مانده نسبت به انتهای آرایه اصلی ذخیره می شوند یا استفاده نمی شوند.
  - با استفاده از فضای ذخیره شده می توان عناصر را در پایان یک آرایه پویا در زمان ثابت اضافه کرد تا این فضا کاملاً مصرف شود.
- هنگامی که تمام فضا مصرف می شود و یک عنصر جدید باید اضافه شود، آرایه اصلی با **اندازه ثابت** افزایش یابد.
- این تغییر اندازه شامل اختصاص یک آرایه زیرین جدید و کپی کردن هر عنصر از آرایه اصلی است.
- تعداد عناصر استفاده شده توسط محتویات آرایه پویا اندازه یا اندازه منطقی آن است، در حالی که اندازه آرایه اصلی را ظرفیت آرایه پویا یا اندازه فیزیکی می نامند، که حداکثر اندازه ممکن بدون تغییر مکان داده‌ها است.

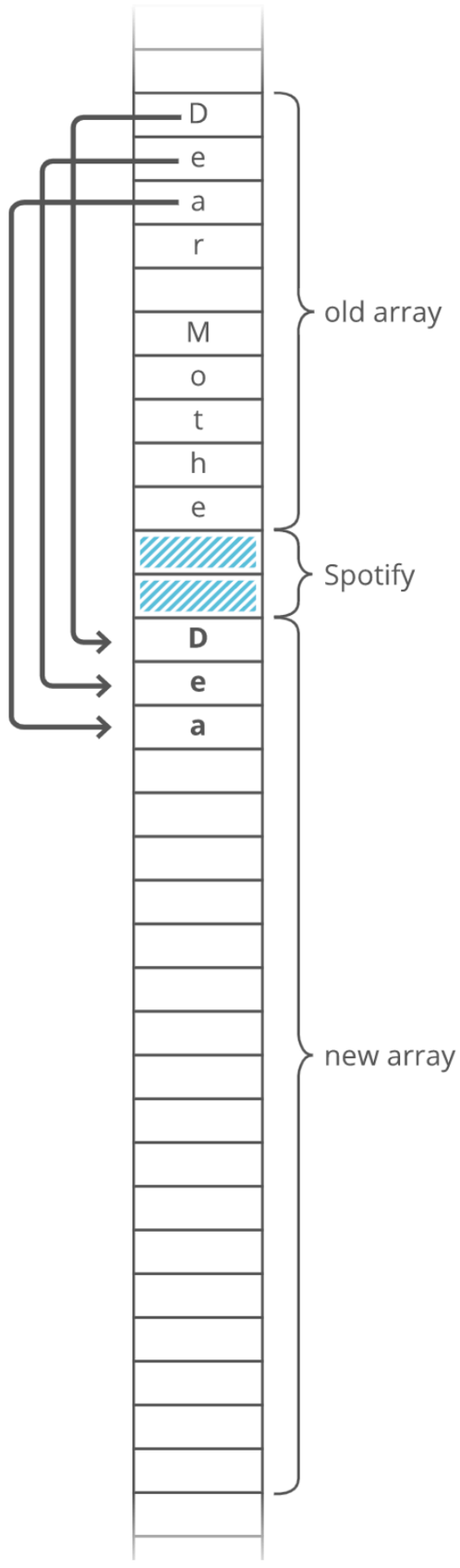
آرایه پویا

آرایه ها و ظرفیت پویا با اندازه محدود



مسئله

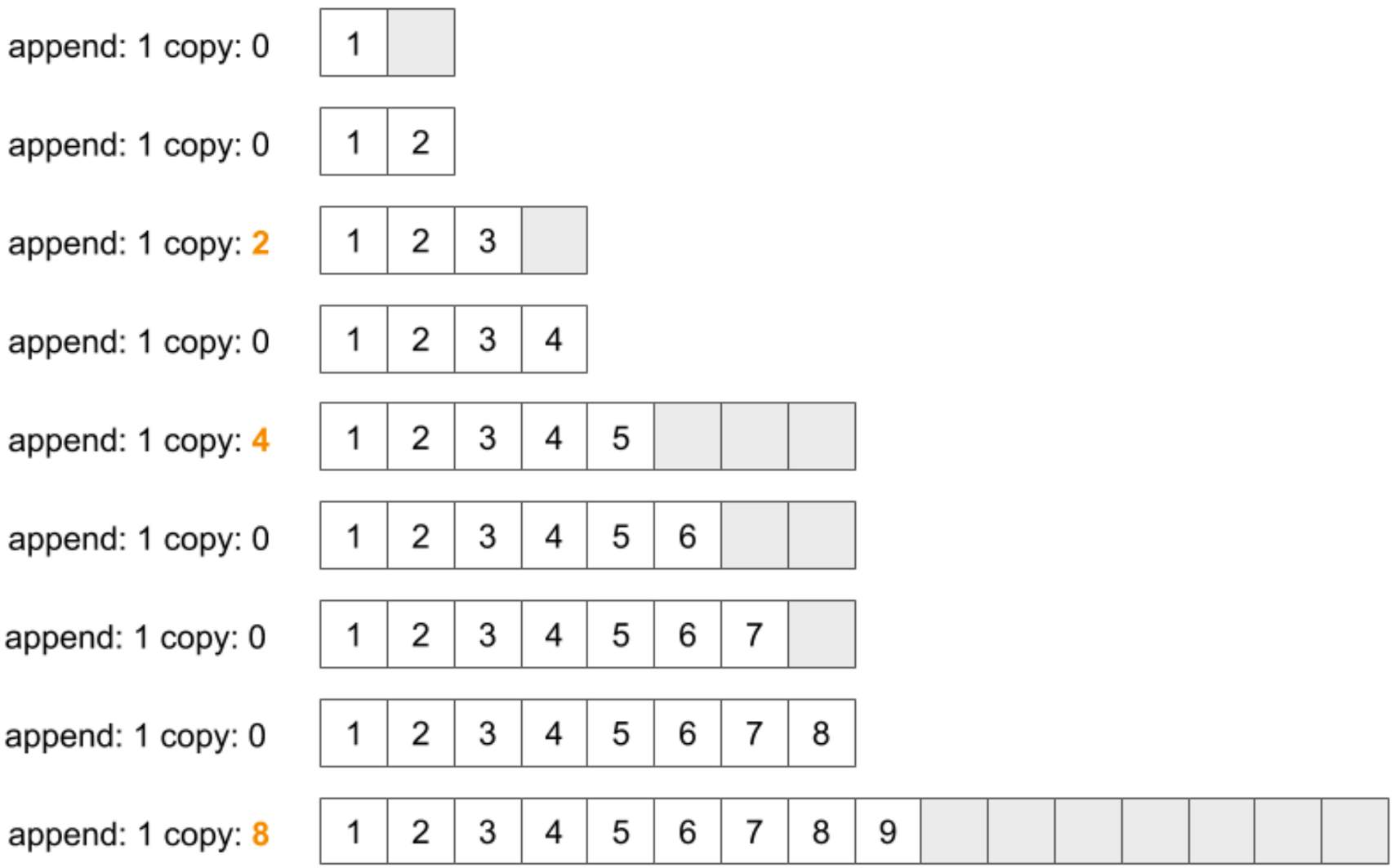
آیا این تغییر اندازه و کپی کردن، گران تمام نمی‌شود؟



تحلیل آرایه پویا

زمان اجرا افزودن n عنصر

با فرض ثابت بودن نرخ رشد آرایه و برابر بودن آن با ۲ داریم:



تحلیل آرایه پویا

زمان اجرا افزودن n عنصر

$$O(\underbrace{(1+1+\cdots+1)}_{\text{درج}}+\underbrace{(2+4+\cdots+n)}_{\text{کپی}})$$

$$\begin{aligned}
&O(n+1+1\cdot 2^1+1\cdot 2^2+\cdots+1\cdot 2^{\log n}-1)\\
&=O(n+\frac{1-2^{\log n+1}}{1-2}-1)\\
&=O(n+2\cdot 2^{\log n})\\
&=O(n)
\end{aligned}$$

آرایه پویا

کارنامه، بررسی قدرت و ضعف

مزایا:

- سرعت بالا دسترسی به عناصر
- اندازه پویا آرایه
- پیوسته بودن حافظه مصرفی

معایب:

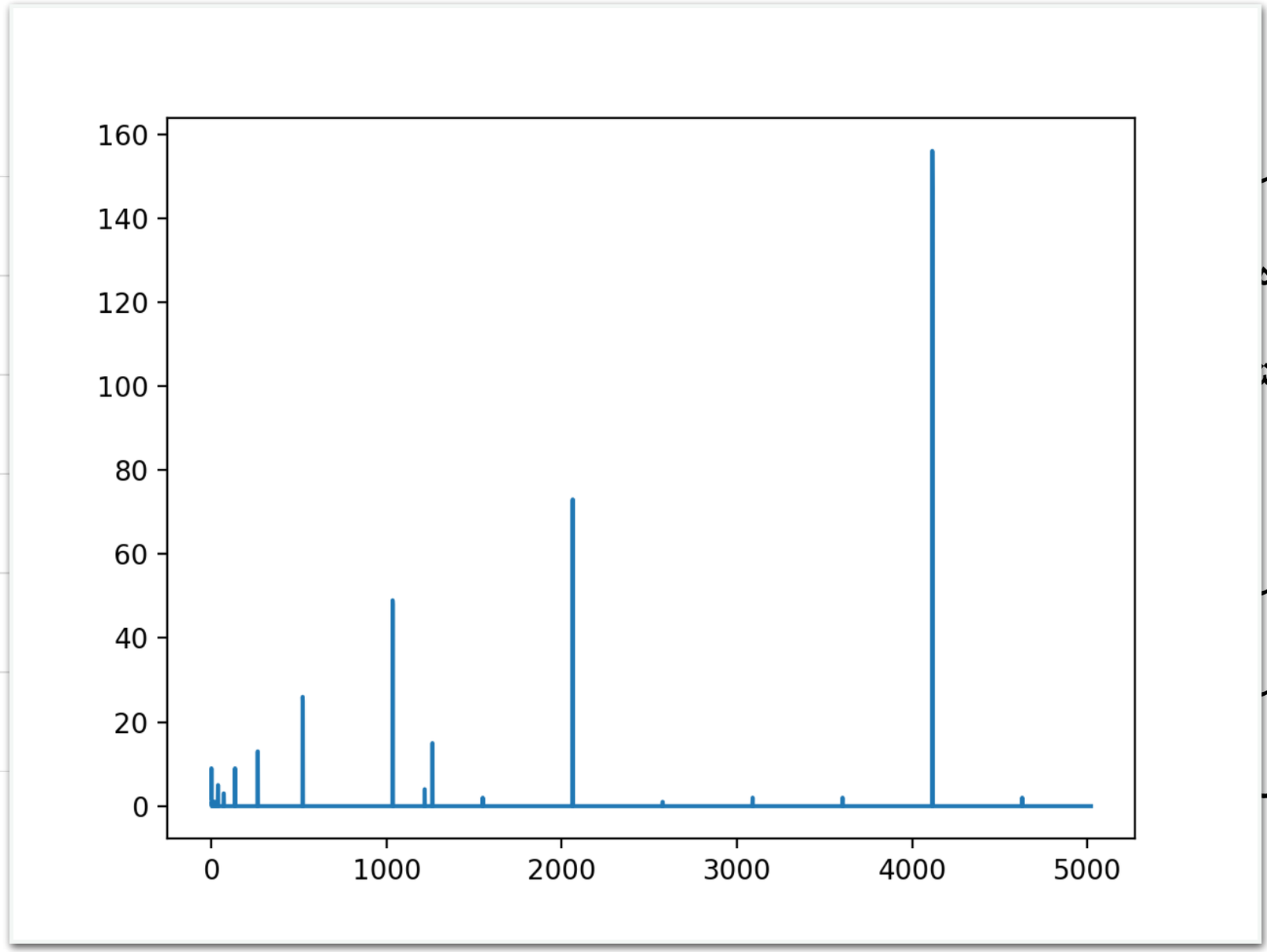
- سرعت پایین درج در بدترین حالت
- سرعت پایین حذف و درج در میانه
- حافظه بلااستفاده زیاد

	Average Case	Worst Case
space	$O(n)$	$O(n)$
lookup	$O(1)$	$O(1)$
append	$O(1)$	$O(n)$
insert	$O(n)$	$O(n)$
delete	$O(n)$	$O(n)$

آرایه پویا

کارنامه، بررسی قدرت و ضعف

space
lookup
append
insert
delete



مزایا:

- سرع
- اندازه
- پیوست

معایب:

- سرع
- سرع
- حافظ

نرخ رشد(g)

اهمیت یک ضریب بی‌اهمیت!

- در مثال‌ها ما برای سادگی(البته واقعا اینطور نیست!) نرخ رشد آرایه پویا را برابر با ۲ در نظر گرفتیم؛ اما در واقع چه مقداری بهینه است؟
- به طور کلی افزایش مقدار g باعث افزایش سرعت رشد می‌شود، اما همین باعث افزایش مقدار حافظه مصرفی بلا استفاده آرایه‌پویا نیز می‌شود [مثلا با $g = 2$، عملیات درج 2^{30}ام]

Implementation	Growth factor
Java	1.5 (3/2)
Python	~1.125
Microsoft Visual C++	1.5 (3/2)
G++	2
Rust	2

ما اخذ

منابع و مراجع، هر آنچه که برآمد اما نیامد

1. MIT 6.006 Introduction to Algorithms, Fall 2011, Table Doubling, Karp-Rabin, MIT OpenCourseWare

2. Number crunching: Why you should never, ever, EVER use linked-list in your code again at kjellkod.wordpress.com

3. Brodnik, Andrej, et al. "Resizable arrays in optimal time and space." Workshop on Algorithms and Data Structures. Springer, Berlin, Heidelberg, 1999.

4. Armando Giuseppe Bonatto Minella, Random Access List, available at Linkedin.com

سوال؟

هرچه دل تنگ‌ت می‌خواند، بپرس.