سلام

جستجوی هوشمند فراگیر

دو: روش های جستجو

ساختمان داده ها و الگوریتم

مدرس: دکتر نجمه منصوری

نگارنده: سجاد هاشمیان

جستجو در آرایه

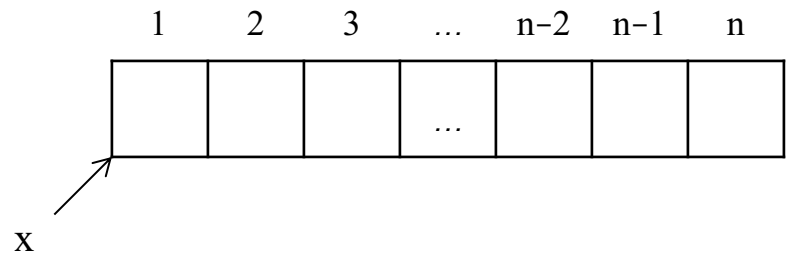
به طور کلی برای جستجو یک عنصر در آرایه ۲ روش اصلی داریم:

- جستجو خطی(ترتیبی)
- جستجو دودویی(باینری)
- جستجو سه‌تایی
- جستجوی درون‌یابی
- جستجوی پرشی
- جستجوی فیبوناچی
- ...

جستجو خطی

در این روش برای جستجوی داده x در آرایه n تایی A[1..n] جستجو را از یکی از دو طرف آرایه (پیش فرض از ابتدا) آغازمی‌کنیم و داده مورد نظر را پشت سر هم با عناصر آرایه مقایسه می‌کنیم. تا این‌که یا داده مورد نظر پیدا شود یا به طرف دیگر آرایه(انتهای آرایه) برسیم.

```
def linear_search(A,x):
    flag=False
    for i in range(0,len(A)):
        if(A[i]==x):
            flag=True
            break
    return flag
```



دقت کنید: در روش جستجوی خطی چون هیچ استراتژی خاصی جز جستجوی پشت سر هم از ابتدا تا انتها وجود ندارد، در نتیجه مرتب بودن یا نبودن آرایه تأثیری در روند جستجو ندارد.

تحلیل جستجو خطی

بهترین حالت	حالت متوسط	بدترین حالت
حداقل ۱ مقایسه، O(1)	تقریبا n/2 مقایسه، O(n)	حداکثر n مقایسه، O(n)

در الگوریتم جستجوی خطی مهمترین پارامتر، مقایسه است. در نتیجه تعداد مقایسات مهمترین عامل در محاسبه مرتبه اجرایی الگوریتم جستجوی خطی به حساب می‌آید.

الف) بهترین حالت: داده مورد جستجو (x) در ابتدای لیست باشد. در این حالت حداقل مقایسه را داریم.

ب) بدترین حالت: داده مورد جستجو (x) در انتهای لیست باشد. در این حالت حداکثر مقایسه را داریم.

ج) حالت متوسط: متوسط مقایسات برابر است با: $\frac{\text{مجموع مقایسه ها}}{\text{تعداد عناصر آرایه}}$

در حالت کلی اگر داده x عنصر اول آرایه باشد با ۱ مقایسه، اگر عنصر دوم باشد با ۲ مقایسه و ... و اگر عنصر nام باشد با n مقایسه بدست می‌آید، در نتیجه:

مجموع مقایسه $1 + 2 + ... + n = \frac{n(n+1)}{2} \longrightarrow$ متوسط $\frac{\frac{n(n+1)}{2}}{n} = \frac{n+1}{2} \in O(n)$

جستجو دودویی

شرط اولیه در این روش جستجو مرتب بودن آرایه (پیش‌فرض صعودی) است. در غیر این صورت این روش غیرقابل استفاده و تعریف نشده خواهد بود.

روش جستجو: داده مورد جستجو (x) را با خانه میانی A[mid] آرایه مقایسه می‌کنیم. در صورتی که با آن خانه برابر باشد جستجو پایان می‌پذیرد. در غیر این صورت در صورتی که x<A[mid] باشد به نیمه بالای آرایه می‌رویم و در صورتی که x>A[mid] باشد به نیمه پایین آرایه می‌رویم و دوباره با قسمت میانی آن نیمه عمل مقایسه را انجام می‌دهیم این عمل را تا زمانی انجام می‌دهیم که یا به داده موردنظر برسیم که محل آن mid خواهد بود یا این‌که داده x در آرایه وجود ندارد که در این صورت low (اندیس پایین نیمه آرایه) از high (اندیس بالای نیمه آرایه) بیشتر می‌شود.

الگوریتم
۱. دو متغیر i و j به ترتیب مقادیر اولیه 0 و n-1 را می گیرند. ۲. تا زمانیکه $i \neq j$ است مراحل زیر را اجرا می کنیم: ۲.۱. متغیر k مقدار سقف میانگین i و j را می گیرد. ۲.۲. اگر A[k]>x باشد، j مقدار k-1 می‌گیرد. ۲.۳. اگر A[k]≤x باشد، i مقدار k می‌گیرد. ۳. حال i=j است،اگر A[i]=x بود،i را خروجی میدهیم. ۴. در غیر این صورت خروجی نخواهیم داشت.

پیاده سازی

پیاده‌سازی بازگشتی	پیاده‌سازی غیربازگشتی

def binary_search(A,x,low,high):
    mid=(low+high)//2
    if(low>high):
        return False
    if(A[mid]==x):
        return True
    elif(x<A[mid]):
        return binary_search(A,x,low,mid-1)
    else:
        return binary_search(A,x,mid+1,high)
	low=1
	high=n
	flag=False
	while((low<high)and(flag!=True)):
	    mid=(low+high)//2
	    if(A[mid]==x):
	        flag=True
	    elif(x<A[mid]):
	        high=mid-1
	    else:
	        low=mid+1
	
	return flag

مثال

جستجوی دودویی.

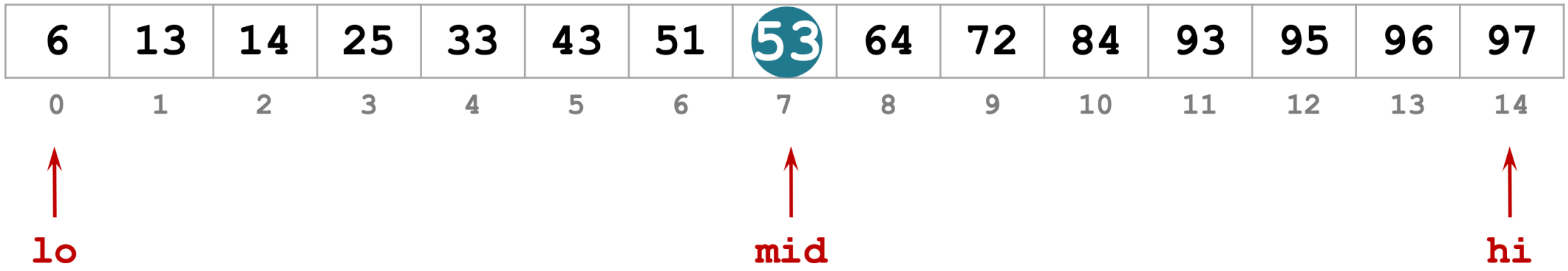
کلید مورد نظر را با کلید وسط مقایسه کن.

اگر کوچکتر است، نیمه‌ی چپ را جستجو کن.

اگر بزرگتر است، نیمه‌ی راست را جستجو کن.

اگر مساوی است، کلید پیدا شده است.

33



مثال

جستجوی دودویی.

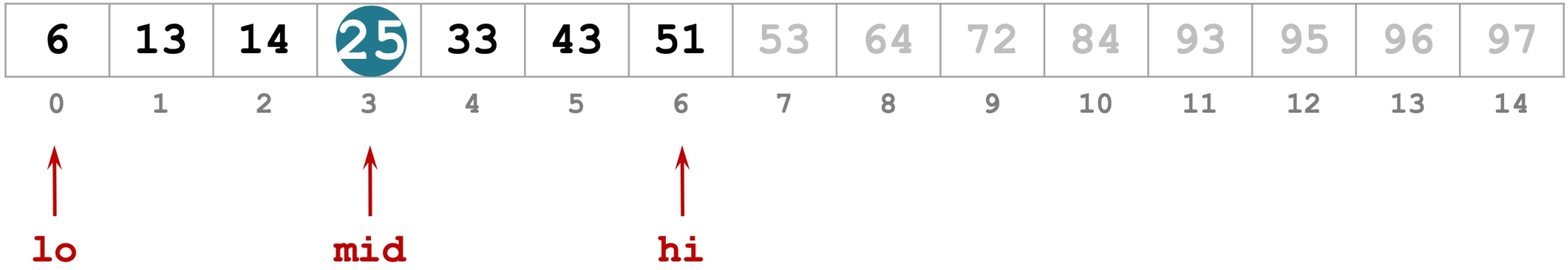
کلید مورد نظر را با کلید وسط مقایسه کن.

اگر کوچکتر است، نیمه‌ی چپ را جستجو کن.

اگر بزرگتر است، نیمه‌ی راست را جستجو کن.

اگر مساوی است، کلید پیدا شده است.

33



مثال

جستجوی دودویی.

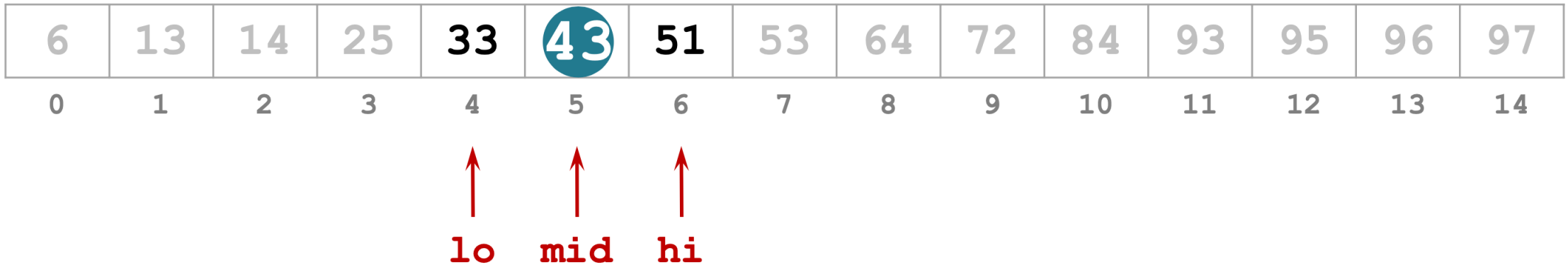
کلید مورد نظر را با کلید وسط مقایسه کن.

اگر کوچکتر است، نیمه‌ی چپ را جستجو کن.

اگر بزرگتر است، نیمه‌ی راست را جستجو کن.

اگر مساوی است، کلید پیدا شده است.

33



مثال

جستجوی دودویی.

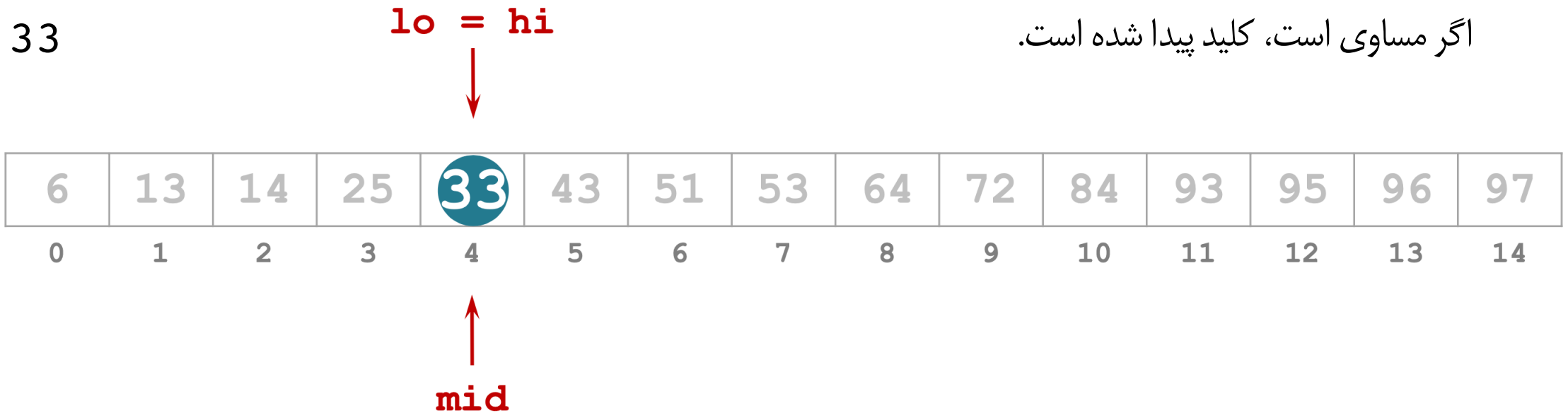
کلید مورد نظر را با کلید وسط مقایسه کن.

اگر کوچکتر است، نیمه‌ی چپ را جستجو کن.

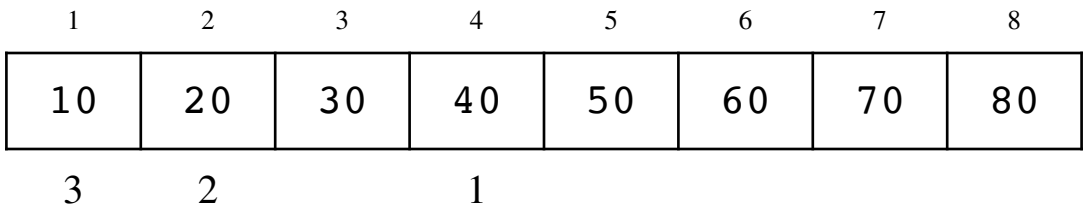
اگر بزرگتر است، نیمه‌ی راست را جستجو کن.

اگر مساوی است، کلید پیدا شده است.

33

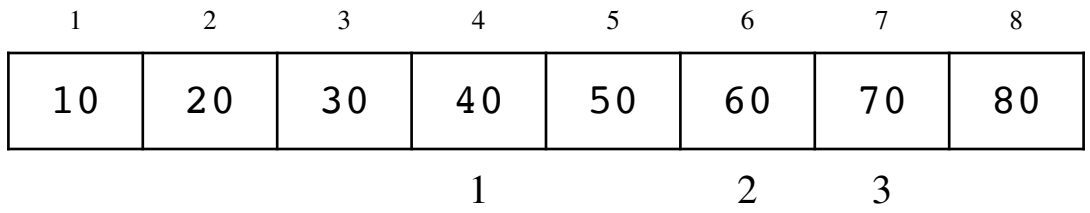


مثال



Low	High	Mid	A[mid]	x	Flag
1	8	4	40	10	False
1	3	2	20	10	False
1	1	1	10	10	True

مثال



Low	High	Mid	A[mid]	x	Flag
1	8	4	40	75	False
5	8	6	60	75	False
7	8	7	70	75	False
8	8	8	80	75	False
8	7	(Low>High)	NA	75	False

تحلیل

به آرایه زیر و تعداد مقایسه بدست آمده از جستجو های **موفق** و **ناموفق** دقت کنید:

	1	2	3	4	5	6	7	8	9	10
	12	30	34	40	45	50	56	60	64	70
تعداد مقایسات برای جستجو موفق	3	2	3	4	1	3	4	2	3	4
تعداد مقایسات برای جستجو موفق	3	3	3	4	4	3	4	4	3	4	4

به طور مثال $x=45$ با ۱ مقایسه و $x=40,56,70$ با $\lg 10 + 1 = 4$ مقایسه به عنوان جستجو موفق بدست می‌آیند.

$x=20$ بین $12,30$ با $\lg 10 = 3$ مقایسه ناموفق تمام می‌شود.

$x=66$ بین $64,70$ با $\lg 10 + 1 = 4$ مقایسه ناموفق تمام می‌شود.

تحلیل

جستجوی موفق

حداقل تعداد مقایسه : `1`

حداکثر تعداد مقایسه : `lg n +1`

اگر x در `A[1:n]` باشد. همواره با **حداکثر** `O(lg n)` پیدا می‌شود.

جستجوی ناموفق

حداقل تعداد مقایسه : `lg n`

حداکثر تعداد مقایسه : `lg n +1`

تحلیل

گزاره. جستجوی دودویی برای جستجو در یک آرایه‌ی مرتب با اندازه‌ی n حداکثر $\lg n + 1$ مقایسه انجام می‌دهد.

برای اثبات $T(n)$ را حداکثر تعداد مقایسه‌های جستجوی دودویی در یک زیر آرایه‌ی مرتب با اندازه‌ی n تعریف می‌کنیم. حال داریم:

$$T(N) \le T(N/2) + 1, \quad T(1) = 1$$

$$\begin{aligned} T(N) &\le T(N/2) + 1 \\ &\le T(N/4) + 1 + 1 \\ &\le T(N/8) + 1 + 1 + 1 \\ &\quad \cdots \\ &\le T(N/N) + 1 + 1 + \ldots + 1 \\ &= 1 + \lg N \end{aligned}$$