

حل تمرین

طراحی الگوریتمها

ریچارد نئوپولیتن

مقدمه

با اینکه کتاب Foundations of Algorithms که باید «مبانی الگوریتمها» ترجمه میشد، یکی از منابع خوب فراگیری روشهای بهبود و طراحی الگوریتمها است، ترجمه بسیار پر اشتباه و احتمالاً بدون ویراستاری این کتاب، می تواند به سادگی خواننده را گمراه کند. در این نوشته قصد داریم تا ضمن ارائه حل برخی تمرین های کتاب، به اشتباهات آن نیز اشاره کنیم. این اشتباهات آنچنان گسترده اند که به احتمال زیاد تعدادی از آنها، ناگفته بمانند. به منظور فهرست کردن اشتباهات چاپی کتاب، لطفاً مشکلاتی که میبینید را [از طریق این Wiki گزارش کنید](#)؛ در ضمن اگر مورد اشتباه یا مشکوک به اشتباه در این نوشته مشاهده کردید، لطفاً [اطلاع رسانی کنید](#).

در هنگام حل مسایل، بعد از توضیح و نوشتن الگوریتم، قطعه کد پیاده سازی الگوریتم، به زبان C# را هم در اینجا خواهیم آورد که البته به جهت اختصار، همه ی کد پروژه نخواهد بود؛ هرچند که کل سورس پروژه های این نوشته از آدرس زیر قابل دریافت است.

<https://github.com/mmsaffari/Foundations-of-Algorithms>

برای اینکه بتوانید الگوریتم بنویسید، نیازی نیست که برنامه نویسی بدانید؛ اما اگر کد نویسی به یک زبان برنامه نویسی را هم یاد بگیرید، هم الگوریتم را بهتر متوجه خواهید شد، هم میتوانید الگوریتم هایتان را به برنامه تبدیل کنید و امتحان شان کنید؛ که تجربه ای بسیار لذت بخش خواهد بود!

فصل اول

۱ - الگوریتمی بنویسید که بزرگترین عدد را در یک لیست (یک آرایه) n عنصری پیدا کند.

گام صفر!

پیش از اینکه بخواهید به فضای فکر و الگوریتم و کد و غیره فکر کنید، باید به فکر چگونگی حل مساله باشید. اصلاً فکر کنید کامپیوتر و برنامه و اینها وجود ندارند و قرار است مساله را با دست حل کنید؛ البته این به شرطی است که مساله، شما را وادار به استفاده از الگوریتم خاصی نکند یا به روشی محدودیتی بر شما اعمال نکند.

برای حل این مساله به روش دستی چه کار می کنید؟ احتمالاً خواهید گفت:

یک بار، اعداد لیست را از اول تا آخر نگاه میکنید و بزرگترین عدد را جایی در ذهن خودتان نگه میدارید که همین عدد جواب مساله است.

اگر بخواهیم با بیان بهتری - که به الگوریتم نزدیک باشد - راه حل بالا را بیان کنیم، باید عبارتهایی را استفاده کنیم که شفاف تر باشند و آسانتر بشود به الگوریتم تبدیل شان کرد. برای مثال وقتی گفتیم «بزرگترین عدد را جایی در ذهن خودتان نگه میدارید» در حال استفاده از حافظه هستیم که در الگوریتم، به جاهایی در حافظه که در

آنها مقدار یا مقادیری را ذخیره میکنیم، «متغیر» میگوییم که با توجه به نوع اطلاعاتی که میخواهیم در آنها نگه داریم، تعریف می شوند.

یا هنگامی که میگوییم «یک بار، اعداد لیست را از اول تا آخر نگاه میکنیم»، بهتر می بود اگر میگفتیم «عدد های لیست را، یک به یک مورد بررسی قرار میدهیم» یا «عدد های لیست را، یک به یک، می شماریم و مورد بررسی قرار میدهیم».

هر گاه کاری را تکرار کردید، چه تعداد دفعات تکرار مشخص باشد و چه نباشد، در حال استفاده از مفهوم حلقه هستید. برای اینکه از مفهوم حلقه به درستی پیاده شود، باید شرطی برای پایان آن در نظر بگیریم. در راه حل بالا، رسیدن به آخرین عضو لیست، شرط پایان حلقه است. پس باید تکرار های «یک به یک» مان را بشماریم؛ یعنی باید جایی در حافظه - که به آن «شمارنده ی حلقه» میگوییم، دفعات تکرار حلقه را نگه داریم و هر بار که عدد جدیدی از آن لیست را نگاه میکنیم، این شمارنده را یکی زیاد کنیم و اگر شمارنده به آخرین عضو آن لیست رسید، شمارش را متوقف کنیم.

حالا با دانستن این ۲ مورد، بیایید الگوریتم بنویسیم؛ یعنی دستورالعملی بنویسیم که با اجرای دقیقش بشود بزرگترین عدد یک لیست را پیدا کرد.

گام اول (بعد از گام صفرم!) ایجاد فضای فکر است. یعنی با توجه به آنچه بعنوان دستورالعمل در ذهن داریم، فضاهایی که برای ذخیره اطلاعات در حافظه مان به آنها نیاز داریم را مشخص کنیم و گوشه ای یادداشت کنیم.

برای نوشتن این الگوریتم به فضایی برای نگهداری آرایه ی مورد اشاره نیاز داریم که نامش را **Array** میگذاریم؛ همچنین فضایی برای نگهداری شمارنده ی حلقه و نیز فضای دیگری برای نگهداری «بزرگترین عدد» مورد اشاره که نام شمارنده ی حلقه را **i** و نام فضایی که در آن «بزرگترین عدد» را نگه میداریم را **Max** میگذاریم و بصورت زیر نمایش داده و یادداشت میکنیم.

قبل از اینکه شروع به نوشتن دستورالعمل کنیم، شاید خوب باشد که چند نکته را متذکر بشویم:

- ۰ - شمارش لیست ها را با صفر شروع کنید. این عادت خوبی است که بعداً موقع کد نوشتن استفاده اش را خواهید برد؛ چون تقریباً همه زبانهای برنامه نویسی (به غیر از زبانهای خانواده BASIC)، اینطور هستند که اولین عضو یک لیست را عضو صفرم میدانند.
- ۱ - برای حل هر یک از تمرین های منتخب، یک (یا بعضاً چند) تابع (که همان Method در برنامه نویسی شی گرا باشد) خواهیم نوشت که مفروضات سوال را بعنوان پارامتر (آرگومان) ورودی دریافت میکند و آنچه در سوال خواسته شده را بر میگردداند.
- ۲ - بعضی از متغیر ها را که برای یک تمرین استفاده کردیم، شاید برای تمرین های دیگر هم استفاده کنیم. برای مثال، لیست اعدادی که در این تمرین استفاده میکنیم را در تمرین بعد هم استفاده خواهیم کرد.

خوب، به حل تمرین برسیم :

فضای فکر ۱

لیست اعداد: Array

شمارنده حلقه: i

بزرگترین عدد: Max

۳ - شروع

۴ - لیست اعداد را در متغیر Array قرار بده؛

۵ - عنصر صفرم Array را در Max قرار بده؛

۶ - با استفاده از متغیر i به عنوان شمارنده حلقه، از ۱ تا تعداد اعداد موجود

در لیست بشمار و به ازای هر بار شمارش، کارهای زیر را انجام بده:

۶-۱ - اگر Max کوچکتر از عنصر iام Array بود، عنصر iام Array را در

Max قرار بده؛

۷ - Max را به عنوان خروجی برگردان؛

۸ - پایان.

اگر بخواهیم با Pseudo-Code (شبه کد) این الگوریتم را بنویسیم، اینطور میشود:

0. Start
1. `int[] Array = new int[] {<List of Numbers>};`
2. `int Max = Array[0];`
3. `for (int i=1; i<Array.Length; i++)`
 - 3.1. `if (Max < Array[i])`
`Max = Array[i];`
4. `return Max;`

نکته ها:

- برای تعریف هر متغیر، باید نوع اطلاعاتی که در آن ذخیره میشود را بدانیم. برای مثال در این تمرین میدانیم که یک لیست از «اعداد» را قرار است در Array نگه داریم. در زبانهای برنامه نویسی، چندین نوع داده‌ی عددی وجود دارند. مثلاً byte هست که از ۰ تا ۲۵۵ را میتواند در خودش نگه دارد؛ یا int که بسیار بیشتر از byte گنجایش دارد و از ۱-۲۳۱ - (یعنی ۲،۱۴۷،۴۸۳،۶۴۸) تا ۲۳۱ (یعنی ۲،۱۴۷،۴۸۳،۶۴۷) را میتواند در خودش نگه دارد؛ بزرگتر از int هم داریم.
- برای تعریف آرایه ها که ساده ترین نوع لیست ها هستند (بله! انواع مختلف لیست هم داریم) از علامت [] استفاده می کنیم. مثلاً `int[]` یعنی یک آرایه (یا لیست) از نوع int یا مثلاً `string[]` یعنی آرایه ای از نوع string. نوع داده‌ی string برای ذخیره اطلاعات نوشتاری استفاده می شود.

حالا کد C# را ببینید: (ستون اعدادی که اول هر خط میبینید، فقط برای خوانایی نوشته شده و جزء کد نیست)

```

1 int[] Array = new int[] { 11, 8, 70, 84, 65, 19, 39, 35, 88, 87, 86, 3, 74, 90, 2 };
2 int Max = Array[0];
3 for (int i = 0; i < Array.Length; i++) {
4     if (Array[i] > Max) Max = Array[i];
5 }
6 return Max;

```

حالا کد را Trace کنیم. یعنی خط به خط اجرا کنیم:

این آرایه ای است که بعنوان ورودی در Array قرار خواهد گرفت. هر عضوش با یک شماره‌ی اندیس که از ۰ شروع شده و تا ۱۴ ادامه پیدا میکند، شماره گذاری شده. (۱۵ عضو دارد)

Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Array:	11	8	70	84	65	19	39	35	88	87	86	3	74	90	2

در جدول زیر مشاهده میکنید که هرگاه شرط $\text{Max} < \text{Array}[i]$ درست بوده، مقدار Max همانطور که در الگوریتم خواسته ایم، به مقدار $\text{Array}[i]$ تغییر داده شده (فلش های نارنجی).

Max	i	Array[i]	Max < Array[i]	Output
11	1	8	FALSE	
	2	70	TRUE	
70	3	84	TRUE	
84	4	65	FALSE	
	5	19	FALSE	
	6	39	FALSE	
	7	35	FALSE	
	8	88	TRUE	
88	9	87	FALSE	
	10	86	FALSE	
	11	3	FALSE	
	12	74	FALSE	
	13	90	TRUE	
90	14	2	FALSE	90

End