۱۳۹۷

حل تمرین

طراحی الگوریتم‌ها

ریچارد نئوپولیتن

تهیه کننده: محمد مهدی صفاری

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 خواهم 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 از 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 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 «شمارنده‌ی 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.

فضای فکر

لیست اعداد:	Array
شمارنده حلقه:	i
بزرگترین عدد:	Max

برای نوشتن این الگوریتم به فضایی برای نگهداری آرایه‌ی مورد اشاره نیاز داریم که نامش را Array میگذاریم؛ همچنین فضایی برای نگهداری شمارنده‌ی 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 روبرو 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 استفاده‌اش را خواهید برد؛ چون تقریباً همه زبانهای برنامه‌نویسی (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 میگرداند.

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

۰ - شروع

۱ - لیست اعداد را در متغیر Array قرار بده؛

۲ - عنصرِ صفرمِ Array را در Max قرار بده؛

۳ - 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:

۳-۱ - اگر Max کوچکتر از عنصرِ i اُمِ Array بود، عنصرِ i اُمِ Array را در Max قرار بده؛

۴ - Max را به عنوان خروجی برگردان؛

۵ - پایان.

اگر بخواهیم با Pseudo-Code (شِبهِ کُد) 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.Lenght; i++)
   3.1. if (Max < Array[i])
            Max = Array[i];
4. return Max;
5. End.
```

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- 2^{31} (یعنی ۲،۱۴۷،۴۸۳،۶۴۸-) تا 2^{31} (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 (Max < Array[i]) 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 هر گاه شرط Max < Array[i] درست بوده، مقدار Max همانطور که در الگوریتم خواسته ایم، به مقدار 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

همانطور که دیدید، بزرگترین عدد لیست، در انتها بعنوان جواب روی خروجی رفت. حالا فکر میکنم که با همین منطق بتوانید یک الگوریتم بنویسید که کوچکترین عدد یک لیست n عنصری را پیدا کند.

در ادامه این الگوریتم را خواهیم نوشت؛ چرا که برای حل تمرین بعدی به این الگوریتم هم نیاز خواهیم داشت. اما قبل از اینکه به خواندن ادامه بدهید، تلاش کنید تا خودتان الگوریتمش را بنویسید. ساده است ☺

؟

☺ ؟

:] ؟

موفق شدید؟ امیدوارم که توانسته باشید با یک تغییر کوچک، الگوریتم پیدا کردن بزرگترین عدد لیست را طوری اصلاح کنید که به جای بزرگترین عدد، کوچکترین عدد را پیدا کند.

کافی است شرطی که در خط ۳-۱ الگوریتم نوشتیم را برعکس کنید. البته به منظور منطقی تر شدن الگوریتم برای خواننده‌ی انسانی، بهتر است که نام متغیر Max را هم به Min تغییر دهیم؛ هرچند که از نظر کامپیوتر، این نام ها فقط برچسب به شمار می آیند و معنایی ندارند.

به اینصورت:

فضای فکر ۲

لیست اعداد:	Array
شمارنده حلقه:	i
کوچکترین عدد:	Min

۰ - شروع
۱ - لیست اعداد را در متغیر Array قرار بده؛
۲ - عنصرِ صفرمِ Array را در Min قرار بده؛
۳ - 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:
۳-۱ - اگر Min بزرگتر از عنصرِ i اُمِ Array بود، عنصرِ i اُمِ Array را در Min قرار بده؛
۴ - Min را به عنوان خروجی برگردان؛
۵ - پایان.

این هم کد C# مربوط به پیدا کردن کوچکترین عدد در یک لیست:

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

این هم Trace الگوریتم کوچکترین عدد:

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

End

خوب، الان دیگه وقتشه که اگر با مفهوم تابع آشنا نیستید، باعث آشناییتون بشم!

مفهوم تابع در واقع یک مفهوم ریاضی است که در برنامه نویسی هم کاربرد دارد. خوب دقت کنید: تابع را، در حالت کلی، یک ماشین فرض کنید که صفر یا بیش از صفر مقدار را بعنوان ورودی میگیرد، عملیاتی را انجام می دهد و صفر یا بیش از صفر مقدار را بعنوان خروجی برمیگرداند.

خوب این به چه کار ما در الگوریتم و برنامه نویسی می آید؟ بدانید و آگاه باشید که زندگی را ساده می کند! در برنامه نویسی، می توانیم یک بار که یک الگوریتم را نوشتیم، آن را بصورت یک تابع در آوریم و هر گاه که خواستیم عملیات آن الگوریتم را روی یک ورودی اجرا کنیم، فقط اسم تابع آن الگوریتم را بیاوریم و ورودی مورد نظر را به آن تابع ارسال کنیم.

نوشتن تابع (یا بهتر است عادت کنیم بگوییم متد Method) در زبان C# به اینصورت است:

```
{ (پارامترهای ورودی) <نام تابع> <نوع اطلاعات خروجی تابع> <سطح دسترسی تابع>
        .
        .       بدنه تابع – دستوراتی که عملیات تابع را انجام میدهند
        .
}
```

یا به شبه کد:

```
<access level> <return data-type> <function name> ( Input paramaters ) {
        .
        .       Function's Body – Instructions to carry out the function's operations
        .
}
```

مثلا این سوال را در نظر بگیرید:

الگوریتمی بنویسید که بزرگترین و کوچکترین عدد هر یک از ۳ لیست مجزای زیر را پیدا کند.

```
A = { 342, 329, 616, 131, 249, 254, 898, 398, 968, 403, 143, 288, 313, 636, 267 }
B = { 81, 470, 831, 232, 432, 989, 58, 998, 587, 321, 725, 330, 632 }
C = { 890, 313, 786, 354, 508, 787, 515, 608, 51, 15, 116 }
```

یعنی حالا باید هر دو الگوریتم قبلی را ترکیب کنید و مجموعاً ۳ بار، یعنی یکبار برای هر کدام از این لیست ها بنویسیدش.
یعنی هر کدام از آن ۲ الگوریتم، ۵ خط اجرایی دارد که باید برای ۳ لیست بنویسید که میشود ۳۰ خط مطلب تکراری!
اینجاست که اون جمله ی طلایی رو بهتون میگم:

برنامه‌نویس باید خیلی بیشتر از اینکه از دستش استفاده میکنه، از مغزش استفاده کنه!

یعنی میشود بعنوان یک استثناء، اینطور گفت که تنبلی همیشه هم بد نیست! فکر میکنم با من موافق باشید که تنبلی ای که نتیجه‌ی مثبت بده، دیگه نور علی نوره! پس بیاید یک کم تنبل باشیم. در واقع ما، تا پیش از این، ۲ تا الگوریتم نوشتیم که توانایی پیدا کردن بزرگترین و کوچکترین عدد در یک لیست رو دارند. کافیست که یک خرده بسته بندیشان را بهتر کنیم و تبدیلشان کنیم به تابع؛ به اینصورت:

۰ - شروع

۱ - تابع Max را تعریف کن که یک لیست از اعداد با نام Array بعنوان ورودی میگیرد و پس از اجرای عملیات زیر، یک عدد برمیگرداند:

۱-۰ - عنصرِ صفرمِ Array را در Max قرار بده؛

۱-۱ - 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:

۱-۲ - اگر Max کوچکتر از عنصرِ iاُمِ Array بود، عنصرِ iاُمِ Array را در Max قرار بده؛

۱-۳ - Max را به عنوان خروجی برگردان؛

۲ - تابع Min را تعریف کن که یک لیست از اعداد در متغیری با نام Array بعنوان ورودی میگیرد و پس از اجرای عملیات زیر، یک عدد بر میگرداند:

۲-۰ - عنصرِ صفرمِ Array را در Min قرار بده؛

۲-۱ - 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:

۲-۲ - اگر Min بزرگتر از عنصرِ iاُمِ Array بود، عنصرِ iاُمِ Array را در Min قرار بده؛

۲-۳ - Min را به عنوان خروجی برگردان؛

۳ - تابع Max را برای لیست A فراخوانی کن و مقدار خروجی اش را بعنوان بزرگترین عدد لیست A نمایش بده.

۴ - تابع Min را برای لیست A فراخوانی کن و مقدار خروجی اش را بعنوان کوچکترین عدد لیست A نمایش بده.

۵ - تابع Max را برای لیست B فراخوانی کن و مقدار خروجی اش را بعنوان بزرگترین عدد لیست B نمایش بده.

۶ - تابع Min را برای لیست B فراخوانی کن و مقدار خروجی اش را بعنوان کوچکترین عدد لیست B نمایش بده.

۷ - تابع Max را برای لیست C فراخوانی کن و مقدار خروجی اش را بعنوان بزرگترین عدد لیست C نمایش بده.

۸ - تابع Min را برای لیست C فراخوانی کن و مقدار خروجی اش را بعنوان کوچکترین عدد لیست C نمایش بده.

۹ - پایان

حالا اگر بخواهیم این الگوریتم را به کد بنویسیم، اینطور میشود:

```
static void Main(string[] args) {
            //Functions Introduction:
            // معرفی توابع
            int[] A = new int[] { 342, 329, 616, 131, 249, 254, 898, 398, 968, 403, 143, 288, 313, 636, 267 };
            int[] B = new int[] { 81, 470, 831, 232, 432, 989, 58, 998, 587, 321, 725, 330, 632 };
            int[] C = new int[] { 890, 313, 786, 354, 508, 787, 515, 608, 51, 15, 116 };

            Console.WriteLine("Max(A) = {0}", Max(A));
            Console.WriteLine("Min(A) = {0}", Min(A));
            Console.WriteLine("Max(B) = {0}", Max(B));
            Console.WriteLine("Min(B) = {0}", Min(B));
            Console.WriteLine("Max(C) = {0}", Max(C));
            Console.WriteLine("Min(C) = {0}", Min(C));
}

static int Max(int[] Array) {
            int Max = Array[0];
            for (int i = 0; i < Array.Length; i++) {
                        if (Array[i] > Max) {
                                    Max = Array[i];
                        }
            }
            return Max;
}

static int Min(int[] Array) {
            int Min = Array[0];
            for (int i = 0; i < Array.Length; i++) {
                        if (Array[i] < Min) {
                                    Min = Array[i];
                        }
            }
            return Min;
}
```