مرکز آموزش الکترونیکی دانشگاه علم و صنعت ایران

طراحی و تحلیل الگوریتمها

فصل سوم: روشهای تقسیم و غلبه

(جلسه سوم)

انتخاب (selection)

*آرایه n عنصری و نامرتب A را در نظر بگیرید. هدف مشخص کردن k امین عنصر کوچکتر آرایه A می باشد.

*به عنوان مثال سومین (k=3) عنصر کوچکتر آرایه مقابل 6 می باشد. 9,3,7,1,8,6

*یک راه حل ساده عبارت است از مرتب کردن آرایه (به صورت صعودی). پس از مرتب سازی k امین خانه آرایه حاوی پاسخ است.

*اما زمان راه حل فوق از $\theta(n \log n)$ می باشد.

انتخاب با استفاده از افراز

*الگوریتم partition که در الگوریتم مرتب سازی سریع مورد استفاده قرار گرفت را می توان برای این مسئله نیز به کار برد

*ایده:

فرض کنید عنصر افراز(v) در خانه A[j] قرار داشته باشد

در اینصورت عناصر A[1..j-1] کوچکتر از v هستند و اگر k<j آنگاه k امین عنصر زیر آرایه A[1..j-1] جواب است

در غیر اینصورت k-j امین عنصر زیر آرایه A[j+1..n] جواب است

الگوریتم select1

*بر اساس ایده فوق می توان تا زمانی که j!=k عملیات افراز را تکرار کرد

*در هر مرحله هم طول بازه (آرایه) کمتر می شود و هم اگر k>j باشد مقدار k کمتر می شود

*در بهترین حالت اولین عنصر (یعنی عنصر افراز) پاسخ است

*در بدترین حالت همه عناصر یکی یکی به عنوان عنصر افراز انتخاب می شوند و آخرین عنصر پاسخ مسئله است

الگوریتم select1 (۲)

```
int  select1(int A[],int n,int k)
{  low=0; up=n-1;
   do{
      j=partition(A,low,up);
      if(k==j)  return A[j];
      else if(k<j) up=j;
      else  low=j+1;
      }while(1)
}
```

مثال

در آرایه زیر هفتمین عنصر را بیابید

i	0	1	2	3	4	5	6	7	8
A[i]	65	70	75	80	85	60	55	50	45

Partition(A,0,8) ⇒ j=4

i	0	1	2	3	4	5	6	7	8
A[i]	60	45	50	55	65	85	80	75	70

مثال(۲)

i	5	6	7	8
A[i]	85	80	75	70

Partition(A,5,8) ⟹ j=8

i	5	6	7	8
A[i]	70	80	75	85

مثال(۳)

i	5	6	7
A[i]	70	80	75

Partition(A,5,7) ⟹ j=5

i	5	6	7
A[i]	70	75	80

مثال(۳)

i	6	7
A[i]	75	80

Partition(A,6,7) → j=6

i	6	7
A[i]	75	80

تحلیل الگوریتم select1

در اولین فراخوانی تابع partition اگر j=k باشد، بلافاصله اجرای تابع پایان می یابد. در این حالت زمان اجرای تابع select1 برابر زمان اجرای partition می باشد یعنی زمان آن $\Omega(n)$

در بدترین حالت هر عنصر آرایه یکی یکی به عنوان عنصر افراز انتخاب می شوند. یعنی حلقه do-while n بار تکرار می شود و در نتیجه partition نیز n بارفراخوانی می گردد.پس $O(n^2)$

تمرین: با مثالی نشان دهید بدترین حالت اجرای این الگوریتم چه موقع می تواند باشد؟

تحلیل الگوریتم select1(ادامه)

- مانند الگوریتم مرتب سازی سریع می توان ثابت کرد زمان اجرای میانگین این الگوریتم $\theta(n)$ می باشد
- بسته به اینکه k چه مقداری دارد زمان اجرای تابع متفاوت است
- برای محاسبه حالت میانگین باید همه مقادیر را با هم جمع و بر تعداد حالات یعنی n تقسیم کرد
- پس از محاسبه تابع میانگین با استفاده از استقرا قوی می توان ثابت کرد که تابع از $\theta(n)$ می باشد

الگوریتم بهینه برای مسئله انتخاب

- اگر عنصر افراز با دقت بیشتری انتخاب شود، می توان به الگوریتمی رسید که مرتبه زمانی آن $O(n)$ باشد
- عنصر افراز باید به گونه ای انتخاب شود که چند درصد از عناصر کوچکتر از آن و چند درصد بزرگتر از آن باشند
- برای این منظور از قاعده میانه میانه ها استفاده می شود

قاعده میانه میانه ها

- میانه n عنصر، عنصری است که از نیمی از عناصر بزرگتر و از نیمی دیگر کوچکتر باشد
- در این قاعده n عنصر به [n/r] دسته r تایی تقسیم می شوند و از n-r[n/r] عناصر استفاده نمی شود
- میانه هر کدام از این [n/r] دسته محاسبه می شود. تعداد این میانه ها برابر با تعداد دسته ها است ([n/r])
- سپس میانه این میانه ها را محاسبه می کنیم

مثال

*میانه میانه ها را برای عناصر زیر محاسبه کنید. N=25, r=5

34, 12, 23, 9, 24, 1, 56, 22, 11, 90, 32, 2, 46, 37, 52, 81,29, 30, 88, 39, 7, 49, 71, 66, 92

* این عناصر به ۵ دسته ۵ تایی تقسیم می شوند

34, 12, 23, 9, 24

1, 56, 22, 11, 90

32, 2, 46, 37, 52

81,29, 30, 88, 39

7, 49, 71, 66, 92

میانه ها ← 9, 22, 37, 39, 66

میانه میانه ها ← 37

الگوریتم بهینه برای مسئله انتخاب(ادامه)

* اگر میانه میانه ها به عنوان عنصر افراز انتخاب شود، نتیجه می شود حداقل$\lceil r/2 \rceil [[n/r]/2]$تعداد عنصر کوچکتر مساوی (یا بزرگتر مساوی) میانه میانه ها می باشند
* حالا با یک الگوریتم بازگشتی می توان الگوریتم انتخاب را نوشت که در آن میانه میانه ها به عنوان عنصر افراز انتخاب می شود

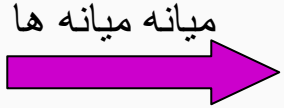
الگوریتم بهینه برای مسئله انتخاب(ادامه)

Select2(int A[], int k, int low, int up)
n=up-low+1;
if(n<=r) sort A[low..up] and return A[k] else
Divide A into [n/r] sub array with size r
Insert the median elements in array m
v=select2(m,[n/r]/2,1,[n/r])
Call partition for A with v as pivot element
If k=j-low+1 then return v
else if k<j-low+1 then return select2(A,k,low,j-1)
else return select2(A,k-(j-low+1),j+1,up)

مثال

* ۴ امین کوچکترین عنصر را با الگوریتم انتخاب (بهینه) در میان عناصر زیر بیابید

34, 12, 23, 9, 24, 1, 56, 22, 11, 90, 32, 2, 46, 37,
52, 81,29, 30, 88, 39, 7, 49, 71, 66, 92



37, 12, 23, 9, 24, 1, 56, 22, 11, 90, 32, 2, 46,34,
52, 81,29, 30, 88, 39, 7, 49, 71, 66, 92

مثال (ادامه)

12, 23, 9, 24, 1, 22, 11 , 32, 2, 34,29, 30, 7, 37,
56, 90, 46, 52, 81, 88, 39, 49, 71, 66, 92

j=14

12, 23, 9, 24, 1, 22, 11 , 32, 2, 34,29, 30, 7

12 22

میانه میانه ها

12

مثال (ادامه)

12, 23, 9, 24, 1, 22, 11 , 32, 2, 34,29, 30, 7

9, 1, 11, 2, 7, 12, 23, 24, 22, 32, 34, 29, 30

j=6

9, 1, 11, 2, 7

9 چهارمین کوچکترین عنصر

تحلیل الگوریتم select2

- فرض کنید t(n) زمان اجرای select2(A,k,1,n) باشد
- زمان یافتن هر میانه از O(1) می باشد
- با فرض اینکه r=5 زمان select2(m,[n/5]/2,1,[n/5]) برابر t(n/5) می باشد
- اندازه A[low..j-1] و A[j+1..up] حداکثر 1.2+0.7n می باشد (چرا؟)

تحلیل الگوریتم select2 (ادامه)

- مقدار $0.7n+1.2$ برای $n>=24$ از $3n/4$ بیشتر نیست پس
- $t(n)<=t(n/5)+t(3n/4)+cn$
- با استفاده از استقراء قوی می توان نشان داد که $t(n)=O(n)$
- پس می توان گفت در بدترین حالت زمان اجرای select2 از $O(n)$ می باشد

مسئله ضرب دو ماتریس

* فرض کنید A و B دو ماتریس به ترتیب با اندازه های n*p و p*m باشند. حاصلضرب آنها C=A*B یک ماتریس n*m خواهد بود
* حالتی را در نظر بگیرید که ماتریسها مربعی باشند، یعنی دو ماتریس A و B هردو n*n باشند. در نتیجه ماتریس حاصلضرب نیز n*n می باشد
* برای ضرب این دو ماتریس باید هر سطر از A را در هر ستون از B ضرب کرد

الگوریتم معمولی ضرب دو ماتریس

```
void mul(A,B,C)
{  for(i=1;i<=n;i++)
      for(j=1;j<=n;j++)
        {  C[i,j]=0;
           for(k=1;k<=n;k++)
              C[i,j]=C[i,j]+A[i,k]*B[k,j];
        }
}
```

$\theta(n^3)$

روش تقسیم و غلبه معمولی

* در این روش هر کدام از ماتریسهای A و B به ۴ زیر ماتریس با اندازه های n/2*n/2 تقسیم می شوند به صورت زیر:

$$\begin{array}{c} \\ n/2 \\ n/2 \end{array}\begin{array}{c} \begin{array}{cc} n/2 & n/2 \end{array} \\ \left[\begin{array}{c|c} A_{11} & A_{12} \\ \hline A_{21} & A_{22} \end{array}\right] \end{array} \times \left[\begin{array}{c|c} B_{11} & B_{12} \\ \hline B_{21} & B_{22} \end{array}\right] = \left[\begin{array}{c|c} C_{11} & C_{12} \\ \hline C_{21} & C_{22} \end{array}\right]$$

* در اینصورت می توان نشان داد:

$$C_{11} = A_{11}B_{11} + A_{12}B_{21}$$
$$C_{12} = A_{11}B_{12} + A_{12}B_{22}$$
$$C_{21} = A_{21}B_{11} + A_{22}B_{21}$$
$$C_{22} = A_{21}B_{12} + A_{22}B_{22}$$

تحلیل الگوریتم تقسیم و حل

* فرض کنید t(n) زمان ضرب دو ماتریس n*n باشد. در نتیجه زمان ضرب دو ماتریس n/2*n/2 برابر t(n/2) خواهد بود

$$C_{11} = A_{11}B_{11} + A_{12}B_{21}$$
$$C_{12} = A_{11}B_{12} + A_{12}B_{22}$$
$$C_{21} = A_{21}B_{11} + A_{22}B_{21}$$
$$C_{22} = A_{21}B_{12} + A_{22}B_{22}$$

$\Rightarrow$ $t(n)=8t(n/2)+4n^2$

$\Rightarrow$ $t(n)=O(n^3)$

روش استراسن برای ضرب دو ماتریس

* ولکر استراسن نشان داد که C_{ij} را می توان به صورت زیر به دست آورد:

$C_{11}=P+S-T+V$

$C_{12}=R+T$

$C_{21}=Q+S$

$C_{22}=P+R-Q+U$

که در این فرمولها ۷ متغیر P,Q,R,S,T,U,V ماترسهای n/2*n/2 هستند که نحوه محاسبه آن ها توضیح داده می شود

روش استراسن (ادامه)

$$P = (A_{11} + A_{22})(B_{11} + B_{22})$$
$$Q = (A_{21} + A_{22})B_{11}$$
$$R = A_{11}(B_{12} - B_{22})$$
$$S = A_{22}(B_{21} - B_{11})$$
$$T = (A_{11} + A_{12})B_{22}$$
$$U = (A_{21} - A_{11})(B_{11} + B_{12})$$
$$V = (A_{12} - A_{22})(B_{21} + B_{22})$$

* محاسبه هر یک از این ۷ ماتریس نیاز به یک عمل ضرب n/2*n/2 دارد
* در این فرمولها تعداد جمع و تفریقهای ماتریسهای n/2*n/2 برابر ۱۰ می باشد
* ضمناً تعداد ۸ عمل جمع و تفریق ماتریسهای n/2*n/2 برای محاسبه C_{ij} ها لازم است

تحلیل الگوریتم ضرب استراسن

* اگر $t(n)$ زمان ضرب دو ماتریس $n*n$ با روش استراسن باشد:

$$t(n)=7t(n/2)+18n^2$$

$$T(n)=O(n^{\log 7})=O(n^{2.81})$$

تمرین حل شده (۱)

* با استفاده از روش استراسن برای محاسبه حاصلضرب دو ماتریس 8*8 چند عمل ضرب نیاز است؟ تعداد ضربها در حالت کلی برای یک ماتریس n*n چقدر می باشد؟
* پاسخ: در این سوال فقط تعداد ضربها مد نظر است. اگر فرض کنیم F(n) تعداد ضربها برای ضرب دو ماتریس n*n با روش استراسن باشد:

$F(n)=7F(n/2)$
$F(1)=1$ ⟹ $F(n)=7^{\log n}$ ⟹ $F(8)=7^3$

تمرین حل شده (۲)

❊ الگوریتم هوارد برای مرتب سازی عناصر آرایه A به صورت زیر است: پیچیدگی آنرا حساب کنید.

```
Algorithm stooge_sort(low,high)
if A[low]>A[high] then swap(A[low],A[high]);
if low+1<high then
{ k=[high-low+1/3]
  stooge_sort(low,high-k);
  stooge_sort(low+k,high);
  stooge_sort(low,high-k);
}
```

$T(n)=3T(2n/3)+1$

$T(1)=1$

$a=3$

$b=3/2$

$T(n)=O(n^{2.7})$