مرکز آموزش الکترونیکی دانشگاه علم و صنعت ایران

طراحی و تحلیل الگوریتمها

فصل سوم: روشهای تقسیم و غلبه

(جلسه دوم)

کلیه حقوق مادی و معنوی این اثر متعلق به دانشگاه علم و صنعت ایران می باشد

ضرب چند جمله ای ها

- دو چند جمله ای از درجه n-1 به صورت زیر تعریف شده اند:

$$P(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_{n-1} x^{n-1}$$

$$Q(x) = b_0 + b_1 x + b_2 x^2 + \ldots + b_{n-1} x^{n-1}$$

- هدف محاسبه R(x)=P(x).Q(x)، R(x) یک چند جمله ای از درجه 2n-2 خواهد بود. چرا؟)
- فرض کنید از یک آرایه یک بعدی برای نمایش هر یک از چند جمله ای ها استفاده کرده باشیم
- A[0..n-1], B[0..n-1] ➔ C[0..2n-2]

کلیه حقوق مادی و معنوی این اثر متعلق به دانشگاه علم و صنعت ایران می باشد

الگوریتم معمولی برای ضرب چند جمله ای ها (الگوریتم اول)

```
Void simple_mult(float A[],float B[],float C[])
{
    for(int i=0;i<=2n-2;i++)
        C[i]=0;
    for(i=0;i<=n-1;i++)
        for(int j=0;j<=n-1;j++)
            C[i+j]=C[i+j]+A[i]*B[j];
}
```

$\Theta(n^2)$

کلیه حقوق مادی و معنوی این اثر متعلق به دانشگاه علم و صنعت ایران می باشد

الگوریتم ضرب دو چند جمله ای با روش تقسیم وغلبه (الگوریتم دوم)

- ایده: هرچند جمله ای به دو قسمت سمت چپ و راست به صورت زیر تقسیم می شود:

$$P(x)=P_l(x)+x^{n/2}P_r(x)$$

- هر یک از این چند جمله ای ها از درجه n/2 است
- مثال:

$$P(x)=1+x+3x^2-4x^3=1+x+(3-4x)x^2=P_l(x)+x^2P_r(x)$$

کلیه حقوق مادی و معنوی این اثر متعلق به دانشگاه علم و صنعت ایران می باشد

الگوریتم دوم(ادامه)

$$P(x)=1+x+3x^2-4x^3=1+x+(3-4x)x^2=P_l(x)+x^2P_r(x)$$

$$Q(x)=1+2x-5x^2-3x^3=1+2x+(-5-3x)x^2=Q_l(x)+x^2Q_r(x)$$

$$R(x)=P(x).Q(x)=[P_l(x)+x^2P_r(x)].[Q_l(x)+x^2Q_r(x)]=$$
$$P_l(x).Q_l(x)+[P_l(x)Q_r(x)+Q_l(x)P_r(x)]x^2+P_r(x)Q_r(x)x^4$$

برای محاسبه $R(x)=P(x).Q(x)$ (ضرب دو چند جمله ای از درجه n) نیاز به ۴ ضرب دو چند جمله ای از درجه $n/2$ می باشد

کلیه حقوق مادی و معنوی این اثر متعلق به دانشگاه علم و صنعت ایران می باشد

الگوریتم دوم (ادامه)

- پس از محاسبه ضربها نیاز به ۳ عمل جمع (برای چند جمله ای های حاصل) و ۲ عمل ضرب می باشد (برای ضرب کافی است توان x با توان تک تک جملات جمع شود)

$$R(x)=P(x).Q(x)=[P_l(x)+x^{n/2}P_r(x)].[Q_l(x)+x^{n/2}Q_r(x)]=$$

$$P_l(x).Q_l(x)+[P_l(x)Q_r(x)+Q_l(x)P_r(x)]x^{n/2}+P_r(x)Q_r(x)x^n$$

- پیچیدگی زمانی این ۳ عملیات جمع و ۲ عمل ضرب از $\theta(n)$ است

کلیه حقوق مادی و معنوی این اثر متعلق به دانشگاه علم و صنعت ایران می باشد

محاسبه پیچیدگی زمانی الگوریتم دوم

* اگر T(n) زمان ضرب دو چند جمله ای از درجه n باشد

$T(n)=4T(n/2)+5n$

$T(n)=4T(n/2)+n \Rightarrow \Theta(n^2)$

کلیه حقوق مادی و معنوی این اثر متعلق به دانشگاه علم و صنعت ایران می باشد

الگوریتم سریع برای ضرب دو چند جمله ای (الگوریتم سوم)

- ایده: مانند الگوریتم قبل عمل کنیم، اما به جای ۴ عمل ضرب (از درجه n/2) از ۳ عمل ضرب (از درجه n/2) استفاده کنیم

$$R(x)=P(x).Q(x)=[P_l(x)+x^{n/2}P_r(x)].[Q_l(x)+x^{n/2}Q_r(x)]=$$

$$P_l(x).Q_l(x)+[P_l(x)Q_r(x)+Q_l(x)P_r(x)]x^{n/2}+P_r(x)Q_r(x)x^n$$

$$R_l(x)=P_l(x).Q_l(x) \qquad R_r(x)=P_r(x)Q_r(x)$$

$$R_m(x)=[P_l(x)+P_r(x)][Q_l(x)+Q_r(x)]$$

$$R_l(x)+[P_l(x)Q_r(x)+P_r(x)Q_l(x)]+R_r(x)$$

$$R(x)=R_l(x)+[R_m(x)-R_l(x)-R_r(x)]x^{n/2}+R_r(x)x^n$$

$$[P_l(x)Q_r(x)+P_r(x)Q_l(x)]=R_m(x)-R_l(x)-R_r(x)$$

کلیه حقوق مادی و معنوی این اثر متعلق به دانشگاه علم و صنعت ایران می باشد

تحلیل زمانی الگوریتم دوم

$$R(x)=R_l(x)+[R_m(x)-R_l(x)-R_r(x)]x^{n/2}+R_r(x)x^n$$

$$R_l(x)=P_l(x).Q_l(x) \qquad R_r(x)=P_r(x)Q_r(x)$$

$$R_m(x)=[P_l(x)+P_r(x)][Q_l(x)+Q_r(x)]$$

تعداد ضربها برای دو چند جمله ای از درجه n/2، ۳ عدد می باشد

تعداد جمع و تفریقهای دو چندجمله ای از درجه n/2، ۶عدد می باشد

تعداد ضرب چند جمله ای ها درx (با پیچیدگی O(n)) ۲ تا می باشد

$T(n)=3T(n/2)+8n \Rightarrow T(n)=\theta(n^{1.58})$

کلیه حقوق مادی و معنوی این اثر متعلق به دانشگاه علم و صنعت ایران می باشد

ضرب اعداد صحیح بزرگ

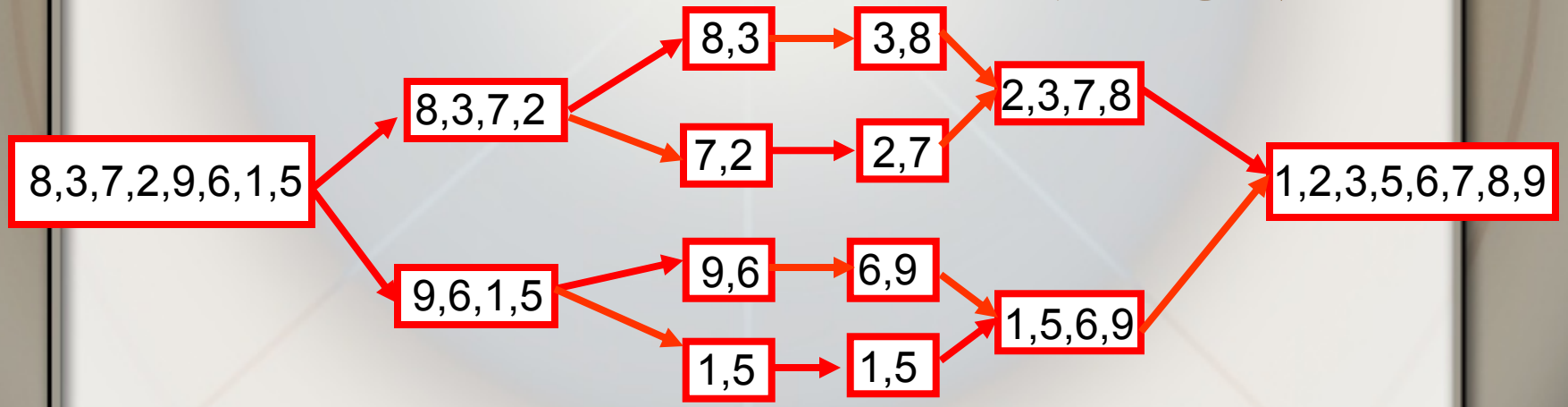
- عدد بزرگ عددی است که توسط انواع داده ای استاندارد، در زبانهای برنامه نویسی قابل تعریف نباشد (مانند یک عدد ۱۰۰ رقمی)
- برای ضرب دو عدد بزرگ می توان از الگوریتم ضرب چندجمله ای ها (الگوریتم سوم) استفاده نمود
- به جای X از ۱۰ استفاده می کنیم:

$$34552345 = 3455 \times 10^4 + 2345$$

کلیه حقوق مادی و معنوی این اثر متعلق به دانشگاه علم و صنعت ایران می باشد

مرتب سازی ادغامی(Merge Sort)

- ایده: آرایه نا مرتب A را به دو قسمت مساوی تقسیم می کنیم، نیمه سمت چپ را مرتب می کنیم (با همین روش)، سپس نیمه سمت راست را مرتب می کنیم (با همین روش) و در نهایت دو نیمه مرتب شده را با هم ادغام می کنیم



کلیه حقوق مادی و معنوی این اثر متعلق به دانشگاه علم و صنعت ایران می باشد

الگوریتم مرتب سازی ادغامی

```
Void  merge_sort(int low, int high)
{
   if(low<high)
     {  int mid=(low+high)/2;
        merge_sort(low,mid);
        merge_sort(mid+1,high);
        merge(low,mid,high);
     }
}
```

Merge_sort(1,n)

کلیه حقوق مادی و معنوی این اثر متعلق به دانشگاه علم و صنعت ایران می باشد

الگوریتم ادغام دو زیر آرایه مرتب شده

- فرض کنید نیمه سمت چپ آرایه A و همچنین نیمه سمت راست آن جداگانه مرتب شده اند، هدف ادغام این دو نیمه است به قسمی که آرایه A در کل مرتب شده باشد
- مرتبه این الگوریتم در بدترین حالت $O(n)$
- پیچیدگی فضایی این الگوریتم $O(n)$ می باشد (به خاطر آرایه B)

```
for(k=low;i<=high;k++)
{ if(A[i]<=A[j])
  { B[k]=A[i]; i++;}
  else
  { B[k]=A[j]; j++;}
  k++;
}
for(i=low;i<=high;i++)
  A[i]=B[i];
```

کلیه حقوق مادی و معنوی این اثر متعلق به دانشگاه علم و صنعت ایران می باشد

محاسبه پیچیدگی الگوریتم مرتب سازی ادغامی

```
Void  merge_sort(int low, int high)        → T(n)
{
   if(low<high)
    {  int mid=(low+high)/2;
       merge_sort(low,mid);                → T(n/2)
       merge_sort(mid+1,high);             → T(n/2)
       merge(low,mid,high);                → O(n)
    }
}
```

→ $T(n)=2T(n/2)+n$ → $T(n)=\theta(n \log n)$

کلیه حقوق مادی و معنوی این اثر متعلق به دانشگاه علم و صنعت ایران می باشد

مرتب سازی سریع(Quick Sort)

- ایده: در روش مرتب سازی سریع، اعداد به دو قسمت (معمولاً نا مساوی) تقسیم می شوند. این تقسیم به گونه ای صورت می گیرد که دیگر نیازی به ادغام دو زیر آرایه نباشد
- ابتدا یک عنصر از میان آرایه به دلخواه انتخاب می شود (V)
- سپس اعداد آرایه طوری چیده می شوند که همه عناصر کوچکتر از V در سمت چپ آن و همه عناصر بزرگتر از V در سمت راست آن قرار می گیرند (افراز)
- به این صورت آرایه به دو قسمت تقسیم می شود و این فرایند برای هر قسمت انجام می شود

کلیه حقوق مادی و معنوی این اثر متعلق به دانشگاه علم و صنعت ایران می باشد

مثال

6,2,5,3,9,1,8

2,5,3,1,6,9,8

2,5,3,1

9,8

کلیه حقوق مادی و معنوی این اثر متعلق به دانشگاه علم و صنعت ایران می باشد

الگوریتم افراز

- اولین عنصر آرایه به عنوان عنصر افراز در نظر گرفته می شود (V)
- از دومین عنصر آرایه شروع به مقایسه کرده و به راست حرکت می کنیم تا به اولین عدد بزرگتر از V برسیم، فرض کنید اندیس چنین خانه ای i باشد
- از آخرین عنصر آرایه به سمت چپ حرکت می کنیم تا به اولین عنصر کوچکتر از V برسیم، فرض کنید اندیس چنین خانه ای j باشد

کلیه حقوق مادی و معنوی این اثر متعلق به دانشگاه علم و صنعت ایران می باشد

الگوریتم افراز(ادامه)

- اگر j>i، محتوای خانه i ام با j ام عوض شود و مراحل ۲ و ۳ از خانه های i و j به بعد ادامه پیدا کند
- اگر i>=j الگوریتم پایان یافته و j اندیس خانه ای که نهایتاً v باید در آن قرار گیرد را نشان می دهد

کلیه حقوق مادی و معنوی این اثر متعلق به دانشگاه علم و صنعت ایران می باشد

مثال

خروجی الگوریتم j=3

کلیه حقوق مادی و معنوی این اثر متعلق به دانشگاه علم و صنعت ایران می باشد

الگوریتم افراز (Partition)

```
int partition(int A[], int m,int p)
{ int v=A[m],i=m,j=p;
  do{
     do{
        i++;}while(A[i]<v)
     do{
        j--;}while(A[j]>v)
     if(i<j) swap(A[i],A[j]);
     }while(i<j)
 A[m]=A[j]; A[j]=v; return j;
 }
```

کلیه حقوق مادی و معنوی این اثر متعلق به دانشگاه علم و صنعت ایران می باشد

الگوریتم مرتب سازی سریع

```
Void Quick_sort(int A[],int p,int q)
{
   if (p<q)
    { int j=partition(A,p,q+1);
      Quick_sort(A,p,j-1);
      Quick_sort(A,j+1,q);
    }
}
```

کلیه حقوق مادی و معنوی این اثر متعلق به دانشگاه علم و صنعت ایران می باشد

تحلیل الگوریتم مرتب سازی سریع

- فرض کنید C(n) تعداد مقایسه ها برای مرتب سازی آرایه n تایی A به روش مرتب سازی سریع باشد
- در این صورت:

$$C(n)=n+1+C(x)+C(y)$$

- بدترین حالت زمان اجرای الگوریتم:

x=0 and y=n-1 (x=n-1 and y=0) ➔

C(n)=n+1+C(n-1) ➔ $C(n)=O(n^2)$

- بدترین حالت زمانی است که آرایه به صورت صعودی یا نزولی مرتب شده باشد

کلیه حقوق مادی و معنوی این اثر متعلق به دانشگاه علم و صنعت ایران می باشد

تحلیل الگوریتم مرتب سازی سریع(۲)

- بهترین حالت زمان اجرای الگوریتم وقتی اتفاق می افتد که عنصر افراز در وسط آرایه قرار گیرد.

$$C(n)=n+1+2T(n/2) \rightarrow C(n)=\Omega(n \log n)$$

- بررسی حالت متوسط :
  - عنصر افراز ممکن است هر یک از n عنصر آرایه باشد
  - اگر عنصر افراز k امین عنصر آرایه باشد، دو زیر آرایه به صورت A[1..k-1] و A[k+1..n] تقسیم می شود
  - حالت متوسط میانگین این حالتهاست (جمع مقایسات در همه این حالات تقسیم بر n) یعنی:

$$C(n)=n+1+\frac{1}{n}\sum_{k=1}^{n}[C(k-1)+C(n-k)]$$

کلیه حقوق مادی و معنوی این اثر متعلق به دانشگاه علم و صنعت ایران می باشد

تحلیل الگوریتم مرتب سازی سریع(۳)

- با تبدیل k به k+1 خواهیم داشت:

$$\sum_{k=1}^{n}[C(k-1)+C(n-k)]=2\sum_{k=0}^{n-1}C(k)$$

- رابطه قبل به صورت زیر نوشته می شود:(با ضرب طرفین در n)

$$nC(n)=n(n+1)+2\sum_{k=0}^{n-1}C(k) \qquad (A)$$

- با تبدیل n به n-1 در رابطه قبل داریم:

$$(n-1)C(n-1)=n(n-1)+2\sum_{k=0}^{n-2}C(k) \qquad (B)$$

- با کم کردن رابطه (B) از رابطه (A) داریم:

$$nC(n)-(n-1)C(n-1)=2n+2C(n-1)$$

- با ساده سازی رابطه فوق داریم:

$$\frac{C(n)}{n+1}=\frac{C(n-1)}{n}+\frac{2}{n+1}$$

کلیه حقوق مادی و معنوی این اثر متعلق به دانشگاه علم و صنعت ایران می باشد

تحلیل الگوریتم مرتب سازی سریع(۴)

- با حل رابطه بازگشتی فوق داریم:

$$\frac{C(n)}{n+1}=\frac{2}{n+1}+\frac{2}{n}+\frac{2}{n-1}+\ldots+\frac{2}{3}+\frac{C(1)}{2}$$

- با توجه به اینکه C(1)=0 داریم:

$$\frac{C(n)}{n+1}=2\sum_{k=3}^{n+1}\frac{1}{k}\leq 2(\log(n+1))$$

- در نهایت:

C(n)<=(n+1).2.log(n+1) ⟹ C(n)=θ(n logn)

کلیه حقوق مادی و معنوی این اثر متعلق به دانشگاه علم و صنعت ایران می باشد