ساختمان داده

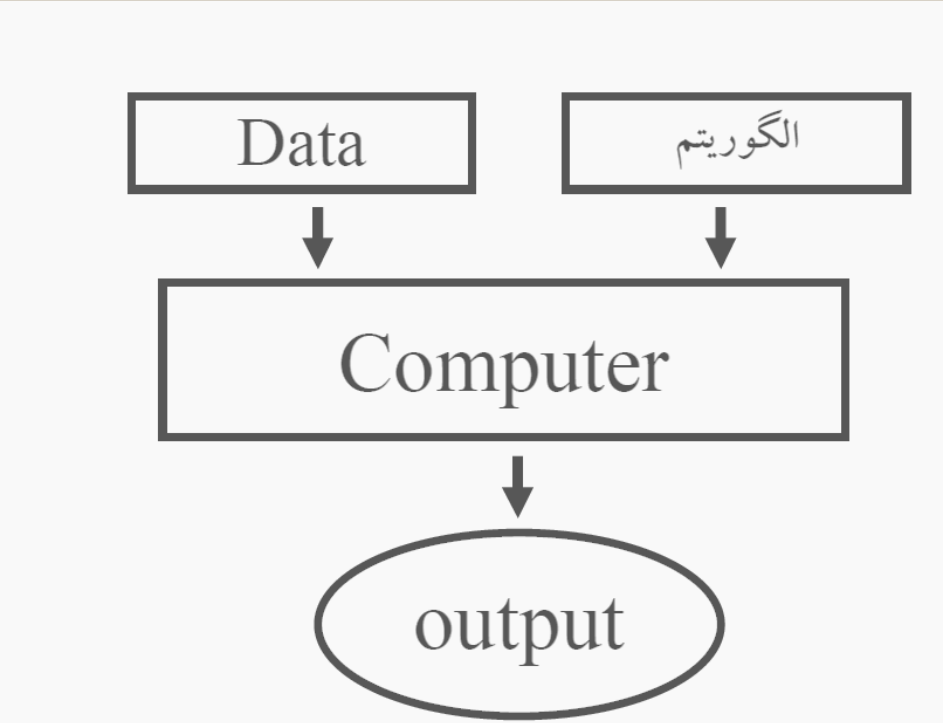
سلمان آسوده
عضو هيات علمي دانشگاه ولايت

روش های تحلیل الگوریتم

اهداف این فصل

- خصوصیات کلی یک الگوریتم را تعریف کنید.
- مرتبه زمانی یک الگوریتم را تعیین کنید.
- مقادیر بازگشتی یک الگوریتم بازگشتی را محاسبه کنید.

ساختار کامپیوتر



سوال

- به نظر شما معیار برتری یک الگوریتم نسبت به الگوریتم دیگر چیست؟

- زمان اجراي الگوريتم
- ميزان حافظه مصرفي الگوريتم

عوامل موثر در زمان اجرا

- سرعت سخت افزار
- نوع کامپایلر
- اندازه داده ورودی مسئله
- ترکیب داده های ورودی
- مرتبه (پیچیدگی) زمانی الگوریتم

مرتبه زمانی الگوریتم

- تابع زمانی الگوریتم $T_{(n)}$
- n اندازه ورودی مسئله است.
- تابع ممکن است شامل چند داده ورودی باشد.

محاسبه تابع زمانی $T_{(n)}$

مقداري ثابت مي باشد

- ✓ زمان مربوط به اعمال جایگزینی
- ✓ زمان مربوط به انجام اعمال محاسبات
- ✓ زمان مربوط به تکرار تعدادی دستور (حلقه ها)
- ✓ زمان مربوط به توابع بازگشتی

تعداد مراحل

- توضیحات comments، تعاریف زیر برنامه و توابع، { }، begin، end
  - تعداد مراحل اجرایی صفر

Procedure f(…) ;	0
void f(…);	0

- دستورهای تعیین نوع
  - تعداد مراحل اجرایی صفر مگر آنکه برای آنها مقدار دهی اولیه صورت گیرد در اینصورت یک

int x;	0
int x=3;	1
float a,b=5;	1

- دستور اجرایی
  - به ازای هر بار اجرا دارای گام ۱

y=x*y+z;	1
write (y)	1
return p;	1

تعداد مراحل

- دستور شرطی if

عبارت شرط ۱ گام و گام کل دستور وابسته به درست و غلط بودن شرط

```
If (x<y)          1
   S=2;           1
Else
   S=5;           1
```

۱+۱ شرط درست

۱+۱ شرط نا درست

۲

۲

```
If (x<y)          1
   S=S+1;         1
Else
   t=t+1;         1
   r=r+1          1
```

۱+۱ شرط درست

۱+۲ شرط نا درست

۲

۳

مثال- تابع زمانی-مرتبه زمان اجرا

```
(1) x = 0 ;
(2) for (i = 0 ;  i < n ; i++)
(3)       x++ ;
```

سطر	زمان	تعداد
1	C_1	۱
2	C_2	$n+1$
3	C_3	n

$$T(n) = C_1 + C_2(n+1) + C_3 n$$

حال C را بیشترین مقدار C_1، C_2، C_3 در نظر می‌گیریم بنابراین:

$$T(n) = C(2n+2)$$

تعداد مراحل

- تعداد گام در حلقه های تو در تو
- از بیرونی ترین حلقه شروع کرده و تعداد تکرار هر حلقه را برای تمام حلقه ها و دستورات تکرار شونده پایین آن در "تعداد تکرار+۱" را برای خود حلقه در نظر می گیریم

```
procedure add( var a,b,c: matrix; m,n: integer);          0
var i,j : integer;                                        0
begin                                                     0
for i:=1 to m do                                          m+1
   for j:=1 to n do                                       m(n+1)
      c[i,j]:= a[i,j]+ b[i,j];                            mn
end                                                       0
                                                        +
                                                          2mn+2m+1
```

تعداد مراحل

void sum (int m, int n , float s[][])	0
{ int i,j	0
for (j=0;j<m; j++)	m+1
{ S[n-1][j]=0;	m
for (i=0;i<n-1;i++)	mn
S[n-1][j]+=S[i][j];	m(n-1)=mn-m
}	0
}	0
	+
	2mn+m+1

تعداد مراحل

float sum (float *a, const int n)	0
{	0
float s=0;	1
for (int i=0; i< n ; i++)	n+1
S+=a[i];	n
return s;	1
}	0
	+
	2n+3

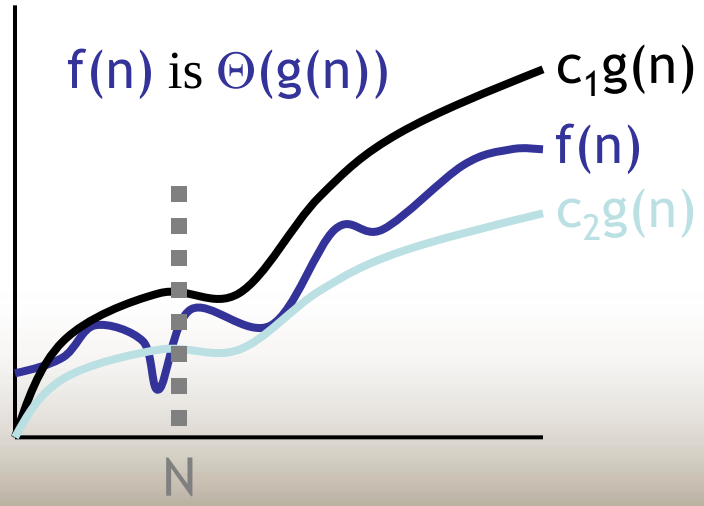
علامت گذاری مجانبی O، Ω، Θ

- انگیزه ما برای تعیین شماره مراحل، توانایی مقایسه پیچیدگی زمانی دو برنامه است که یک عمل را انجام می دهند و نیز پیش بینی رشد زمان اجرا با تغییر مشخصه ورودی است.

- برای بررسی دقیق تر توابع از از تابع های رشد با نمادهای Ω ، Θ و O استفاده می شود.

علامت گذاری مجانبی Θ

- $f(n) = \theta(g(n))$ می باشد اگر و فقط اگر به ازای مقادیر ثابت c1 و c2 و n0، برای تمام مقادیر $n>=n0$ داشته باشیم $c1g(n)<= f(n) <= c2g(n)$.
- وقتی n به سمت بینهایت میل می کند رفتار f(n) برابر از g(n) خواهد بود.



مثال –

$g(n)=n$ $f(n) = 3n+2$ –

$3n <= 3n + 2 <= 4n$, for all $n >= 2$, $\therefore$ $3n + 2 = \Theta(n)$

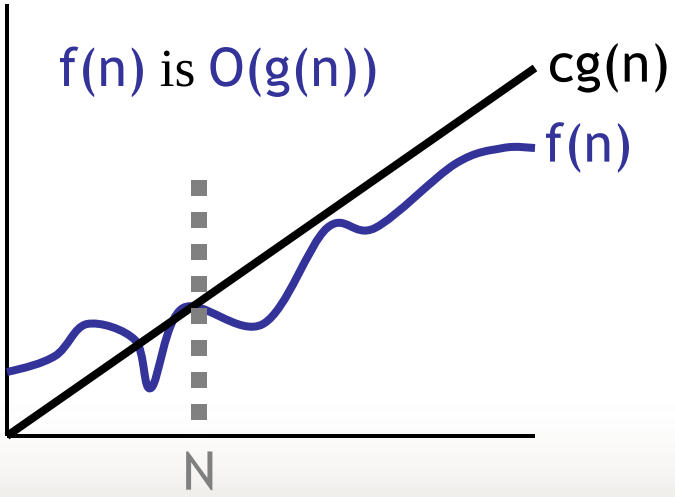
$g(n)=n^2$ $f(n) = 10n^2+4n+2$ –

$n^2 <= 10n^2+4n+2 <= 11n^2$, for all $n >= 5$, $\therefore$ $10n^2+4n+2 = \Theta(n^2)$

تعریف O (Big "oh"):

f(n)=O(g(n)) اگر و فقط اگر ثابتهای مثبتی مانند C و N وجود داشته باشند به طوری که به ازای تمامی مقادیر n و n>=N، f(n)<=cg(n) باشد.

وقتی n به سمت بینهایت میل می کند رفتار f(n) حداکثر (کوچکتر یا مساوی) g(n) خواهد بود.



- $f(n) = 3n+2$

$3n + 2 <= 4n$, for all $n >= 2$, $\therefore 3n + 2 = O(n)$

- $f(n) = 10n^2+4n+2$

$10n^2+4n+2 <= 11n^2$, for all $n >= 5$, $\therefore 10n^2+4n+2 = O(n^2)$

مثال

i) $T(n) = (۲n + ۱) \in O(n^۲)$

i) $T(n) = (۲n + ۱)$

$\leq ۲n^۲$

که در آن اگر $C = ۲$ و $n_۰ = ۲$ باشد آنگاه $T(n) \in O(n^۲)$ خواهد بـود. بنـابراین رابطـه بالا یک رابطه صحیح می‌باشد.

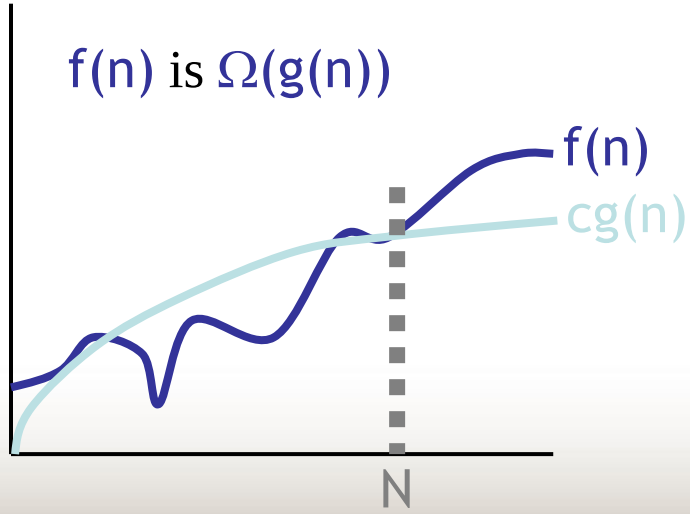
ii) $T(n) = (5n^2 + n + 1) \in O(n)$

همانطور که ملاحظه می‌کنید نمی توان C و n_0 معرفی کرد که رابطه $5n^2 + n + 1 \le Cn$ به ازای $n \ge n_0, C$ همواره برقرار باشد به عبارت دیگر $T(n)$ به هیچ وجه در حالت کلی نمی‌تواند کمتر از Cn باشد. در نتیجه رابطه بالا یک رابطه نادرست می‌باشد.

علامت گذاری مجانبی Ω

$f(n)= \Omega(g(n))$ می باشد اگر و فقط اگر به ازای مقادیر ثابت مثبت c و $n0$ ، برای تمام مقادیر n به شرطی که $n \geq n0$ باشد داشته باشیم $f(n) \geq cg(n)$

وقتی n به سمت بینهایت میل می کند رفتار $f(n)$ حداقل (بزرگتر یا مساوی) $g(n)$ خواهد بود.



- $f(n) = 3n+2$
  - $3n + 2 >= 3n$, for all $n >= 1$, $\therefore 3n + 2 = \Omega(n)$
- $f(n) = 10n^2+4n+2$
  - $10n^2+4n+2 >= n^2$, for all $n >= 1$, $\therefore 10n^2+4n+2 = \Omega(n^2)$

علامت گذاری مجانبی O، Ω، Θ

- قضیه

$$f(n) = a_m n^m + \ldots + a_1 n + n_0 \Rightarrow f(n) = O(n^m)$$

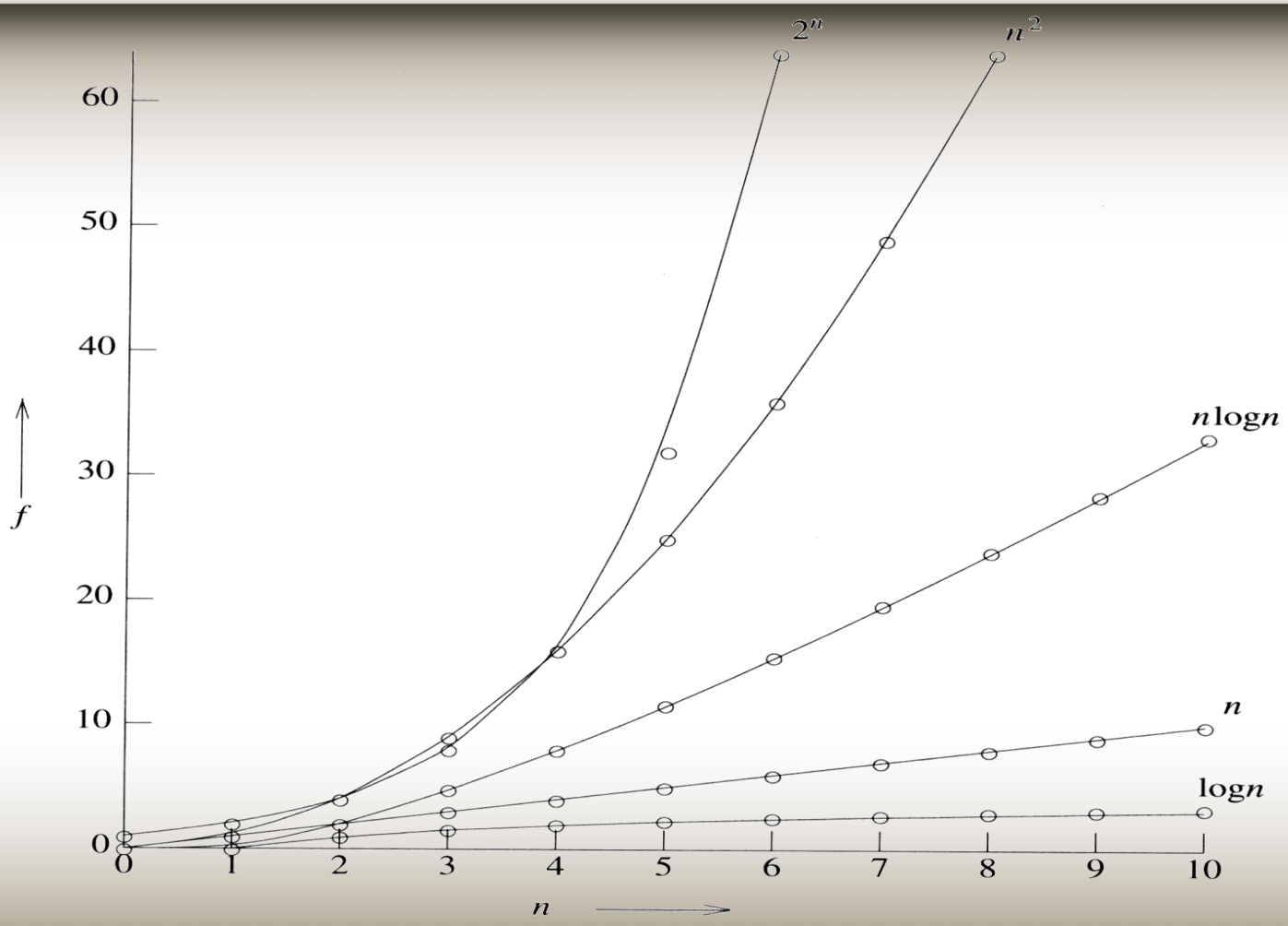
$$f(n) = a_m n^m + \ldots + a_1 n + a_0 \Rightarrow f(n) = \Omega(n^m)$$
$$a_m > 0$$

$$f(n) = a_m n^m + \ldots + a_1 n + a_0 \Rightarrow f(n) = \Theta(n^m)$$
$$a_m > 0$$

نمونه هایی از توابع رشد

تابع رشد
$O(1)$
$O(\log_2 N)$
$O(N)$
$O(N \log_2 N)$
$O(N^2)$
$O(2^N)$
$O(N!)$
$O(N^k)$

مقایسه توابع رشد



مثال

پیچیدگی الگوریتم جستجوی خطی در همه حالات را بدست آورید.

توابع بازگشتی

- هر تابعی که بصورت مستقیم و غیر مستقیم خودش را فراخوانی کند،تابع بازگشتی است.

خواص:

- ✓ باید معیاری به عنوان معیار پایه(base case) وجود داشته باشد که تابع در این حالت خودش را فراخوانی می کند.
- ✓ در هر بار فراخوانی، به معیار پایه نزدیکتر می شود.

تابع فاكتوريل

- $f(n)=\begin{cases} 1 & \text{if } n=0 \\ n*f(n-1) & \text{if } n>1 \end{cases}$

مثال

```
int Rec(int n)
{
    if (n==1)
        return(1)
    else
        return(Rec(n-
1) + Rec(n-1))
}
```

- Rec(6)=?

تابع بازگشتی

```
f(n)
{
  if(n≤1)   return 2*n;
  return f(n-1)+f(n-1)+n;
 }
```

f(4) چند است؟

f(4) چه تعداد فراخوانی دارد؟

f(4) چه تعداد فراخوانی بازگشتی دارد؟

f(4) چه تعداد عمل + را انجام می دهد؟

f(4) چه تعداد عمل * انجام می دهد؟

تعداد تکرار if را بدست آورید؟

مثال

رابطه بازگشتی در برنامه زیر را بدست آورید؟

اگر n=8 باشد، چند عمل ضرب در تابع زیر انجام می شود؟

```
function count(n)
begin
    if n<=0 then return(1)
    if n=1 then return(2)
    if n=2 then return(3)
    return (count (n-2) * square (count (n-
4));
end
```

مثال

در این زیر برنامه، برای تعداد دفعات چاپ پیام message یک رابطه بازگشتی بنویسید. به ازای ۳۶ تعداد دفعات چاپ را بدست آورید؟

```
Time(x){
If(x<8)
 wirte(“message”);
else{
Time([x/2]);
Time([x/4]);
Time([x/8]);
}
}
```

مثال

با توجه به تابع زیر برای تعداد ستاره های که چاپ می شوند فرمول بازگشتی بنویسید و بگویید f(4) چندتا ستاره چاپ می کند؟

```
F(int n)
{ if(n<=2) write(“***”);
Else(
Write(“**”);
F(n-1);
Write(“*”);
F(n-3);
Write(“****”);
F(n-2);
}
```

مثال

- در کد زیر f(4) چه تعداد عدد و به چه ترتیبی چاپ می کند؟

```
F(int n)
{
    if(n>1)
    {
     write(n);
     F(n-2);
     Write(n-1);
     F(n-1)
     Write(n);
}
}
```