عنوان درس:

ساختمان داده ها

جلسه ۱: مفایم اصلی

مدرس:
امیر امیدی
(Amir Omidi)
Omidi.students@gmail.com

omidi.students@gmail.com

www.jozve.org

فهرست مطالب

- مقدمه
- آرایه ها
- پشته ها و صف ها
- لیست های پیوندی
- درخت ها
- مرتب سازی
- گراف ها

omidi.students@gmail.com
www.jozve.org

ارزیابی درس

- آزمون پایان ترم ۱۴ نمره
- میان ترم ۶ نمره
- تکالیف + آزمون میان ترم + حضورغیاب = ۳ نمره
- پروژه = ۳ نمره

برای نمرات زیر ۷ در آزمون پایان ترم، نمره میان ترم با ضرب نمره پایان ترم در ۱۰/۷ محاسبه خواهد شد.

omidi.students@gmail.com
www.jozve.org

موضوع درس در باره چیست؟

- این درس در مورد
  - ذخیره سازی کارای داده ها
  - به کار گیری ساده داده ها

  بحث می کند.

- کاربرد این ساختارهای داده ای کارا و ساده در طراحی الگوریتم های کارا می باشد.

omidi.students@gmail.com
www.jozve.org

چرا به این الگوریتم های کارا نیاز مندیم؟

- کامپیوتر ها کار های خیلی پیچیده را انجام می دهند.
  - یک فهرست ۸,۰۰۰,۰۰۰,۰۰۰ صفحه ای را در نظر بگیرید.(Google)
- در اختیار داشتن راه حلهای مختلف برای حل مسائل سخت
- کدنویسی به درد بخور و صحیح

omidi.students@gmail.com
www.jozve.org

چرا به این الگوریتم های کارا نیاز مندیم؟ (ادامه)

- نیاز به ایجاد نرم افزار های قابل قبول
  - طرحی درست
  - نگهداری ساده
  - قابل اعتماد
  - سادگی استفاده
  - الگوریتم های سریع

omidi.students@gmail.com
www.jozve.org

مثال

- مجموعه ای شامل ۳۰۰۰ متن مختلف با میانگین ۲۰ سطر در هر متن و ۱۰ کلمه در هر خط را در نظر بگیرید.(یعنی ۶۰۰,۰۰۰ کلمه)
- می خواهیم تعداد کلمات *"happy"* را در این مجموعه بدست آوریم.
- فرض می کنیم که مقایسه هر کلمه در این مجموعه با کلمه مورد نظر ۱ ثانیه طول می کشد.
- حال چه باید بکنیم؟

omidi.students@gmail.com
www.jozve.org

مثال(ادامه)

- راه حل ۱: تطبیق ترتیبی
- $1_{sec.} * 600{,}000_{words} = 166_{Hours}$
- راه حل ۲: جستجوی باینری
  - ابتدا کلمات را مرتب کنید
  - حال در هر مرحله نصف کلمات را حذف کنید.
- $\text{Log}_{2}\ 600{,}000 = 19_{sec.}$
- ۱۹ ثانیه را با ۱۶۶ ساعت مقایسه کنید

omidi.students@gmail.com
www.jozve.org

طراحی شیء گرا و ساخت یافته

در هر دو روش هدف تجزیه پروژه طراحی نرم افزار به زیر پروژه های ساده تر و حل هر کدام است

طراحی شیء گرا(تجزیه شیء گرایی)	**طراحی ساخت یافته**(تجزیه الگوریتمی)
• تقسیم برنامه به چند قطعه • تهیه کد و داده برای هر قطعه • تبدیل قطعات به واحدها(*Units*) یا اشیاء(*Objects*) • ایجاد رابطه متقابل بین این اشیاء	• تصور برنامه به عنوان یک فرایند • ایجاد قطعه برنامه های در قالب توابع برای نمایش مراحل مختلف این فرایند.

omidi.students@gmail.com
www.jozve.org

تجرید داده ها و محصور سازی

- در ارتباط بین کامپیوتر و انسان مورد استفاده قرار می گیرند.

- مثال(ضبط صوت)
  - همیشه از دکمه هایی مانند PLAY, REC, FFW, REW استفاده می کنیم و نمیتوانیم با مدارات داخلی آن سرو کار داشته باشیم. (محصور سازی)

  - کتابچه راهنمای ضبط صوت در باره کار هر دکمه توضیح می دهد ولی در مورد طرز کار هر کدام از آنها چیزی نمی گوید. (تجرید یا جدا سازی)

omidi.students@gmail.com
www.jozve.org

تجرید داده ها و محصور سازی(ادامه)

- **محصور سازی یا پنهان سازی**

  عبارت است از مخفی کردن جزئیات پیاده سازی اشیاء داده های از دنیای خارج

- **تجرید یا جدا سازی**

  جداسازی یک شیء داده ای از پیاده سازی آن

omidi.students@gmail.com
www.jozve.org

انواع داده ای

- متغیرها

بازه قابل قبول	اندازه به بایت	نوع
127 To -127	1	Char
0 To 255	1	Unsigned char
127 To -127	1	Signed char
32767 To -32767	2 or 4	Int
0 To 65535	2 or 4	Unsigned int
32767 To -32767	2 or 4	Signed int
32767 To -32767	2	Short int
0 To 65535	2	Unsigned Short int
32767 To -32767	2	Signed Short int
2147483647 To -2147483647	4	Long int
2147483647 To -2147483647	4	Signed long int
0 To 4294967295	4	Unsigned long int
10^{38} To 10^{-38}	4	Float
10^{308} To 10^{-308}	8	Double
10^{4932} To 10^{-4932}	10	Long double

omidi.students@gmail.com
www.jozve.org

انواع داده ای(ادامه)

تعریف ثوابت

- #Define M 100
- Const int M = 100;

نوع داده شمارشی

- مانند enum day {sat, sun, mon, tus, wed, thu, fri}

اشاره گر ها

- int i=50;
- int *p;
- p = &i;

omidi.students@gmail.com
www.jozve.org

عملگر ها

- **عملگر های محاسباتی**

-	+	*	/	%	--	++
منفی یوناری و منفی یکانی	جمع	ضرب	تقسیم	باقیمانده تقسیم	کاهش	افزایش
-x or x-y	x+y	X*y	x/y	x%y	--x or x--	++y or ++y

- **عملگر های رابطه ای**

>	>=	<	<=	==	!=
بزرگتر	بزرگتر یا مساوی	کوچکتر	کوچکتر یا مساوی	متساوی	نا مساوی
x>y	x>=y	x<y	x<=y	x==y	x!=y

- **عملگر های منطقی**

!	&&	||
Not (نقیض)	And (و)	Or (یا)
!x	x>y || m<p	x>y && m<p

omidi.students@gmail.com
www.jozve.org

عملگر ها(ادامه)

- **عملگر های بیتی**

&	|	^	~	>>	<<
و	یا	یای انحصاری	نقیض	شیفت به راست	شیفت به چپ
AND	OR	XOR	NOT	Right shift	Left shift

- **عملگر های ترکیبی**

+=	-=	*=	/=	%=
انتساب جمع	انتساب تفریق	انتساب ضرب	انتساب تقسیم	انتساب باقی مانده تقسیم
x+=y	x-=y	x*=y	x/=y	x%=y
x=x+y	x=x-y	x=x*y	x=x/y	x=x%y

omidi.students@gmail.com
www.jozve.org

عملگر ها(ادامه)

- عملگرهای &(آدرس) و *(محتوی)
- عملگر ؟

```
y = x>5 ? x*2 : x*5
```

- عملگر ,

```
y = ( x=2 , x*4/2)
```

- عملگر sizeof

```
m=Sozeof(int)
```

- عملگر ()

omidi.students@gmail.com
www.jozve.org

ورودی و خروجی داده ها

- **چاپ اطلاعات**

C

```
Printf("The Answer Is: %d",x);
```

```
Cout<<"The Answer Is:"<<x<<endl;
```

C++

کاراکتر های فرمت در دستور printf()

کاراکتر	نوع اطلاعاتی	کاراکتر	نوع اطلاعاتی
%c	یک کاراکتر	%G	اعداد اعشاری ممیز شناور
%d	اعداد صحیح دهدهی مثبت و منفی	%o	اعداد مبنای ۸ مثبت
%i	اعداد صحیح دهدهی مثبت و منفی	%s	رشته ای از کاراکتر ها
%e	نمایش علمی عدد همراه با حرف e	%u	اعداد صحیح بدون علامت
%E	نمایش علمی عدد همراه با حرف E	%x	اعداد مبنای ۱۶ مثبت
%f	عدد اعشاری ممیز شناور	%p	poiter
%g	عدد اعشاری ممیز شناور	%%	علامت %

www.jozve.org

ورودی و خروجی داده ها

- نوشتن کاراکتر با تابع putch()

Putch(متغیر);

Putch('کاراکتر');

- ورود اطلاعات

Sscanf("%d%d",&x&y);

C

cin>>x>>y;

C++

- خواندن کاراکتر ها با توابع getch(), getche(), getchar();

X=getch();// عکس العملی در صفحه نمایش ندارد

Y=getche();//.پس از ورود کاراکتر آن را در صفحه نمایش نشان می دهد

Z=getchar();//.پس از ورود کاراکتر باید کلید اینتر را فشرد

omidi.students@gmail.com

www.jozve.org

حلقه های تکرار

- ساختار تکرار For

```
For (مقدار اولیه اندیس حلقه; شرط حلقه; گام حرکت )
{ دستور یا دستورت;}
```

- ایجاد حلقه بی نهایت با for

```
For (;;)
{دستور یا دستورات;}
```

- حلقه های تو در تو

```
For(i=0;i<10;I++)
{For (j=0;j<20;j++)
  {
      دستور یا دستورات;
}}
```


omidi.students@gmail.com
www.jozve.org

حلقه های تکرار(ادامه)

- ساختار تکرار while

```
While(شرط)
{
دستور یا دستورات;
}
```

- ساختار تکرار do...while

```
do
{
دستور یا دستورات;
}
While(شرط);
```


omidi.students@gmail.com
www.jozve.org

ساختار های تصمیم

- ساختار تصمیم if

- If (شرط)
  دستور;
  Else
  دستور;

- If (شرط)

```
{
  دستور یا دستورات;
}
Else
{
دستور یا دستورات;
}
```


omidi.students@gmail.com
www.jozve.org

ساختار های تصمیم(ادامه)

- دستور Break
- دستور Continue
- دستور goto
- ساختار تصمیم switch

```
Switch(عبارت){
  Case  مقدار۱:
          دستور یا دستورات;
          break;
  Case  مقدار۲:
          دستور یا دستورات;
          break;
  …
  Default:
          دستور یا دستورات;
```


omidi.students@gmail.com
www.jozve.org

توابع

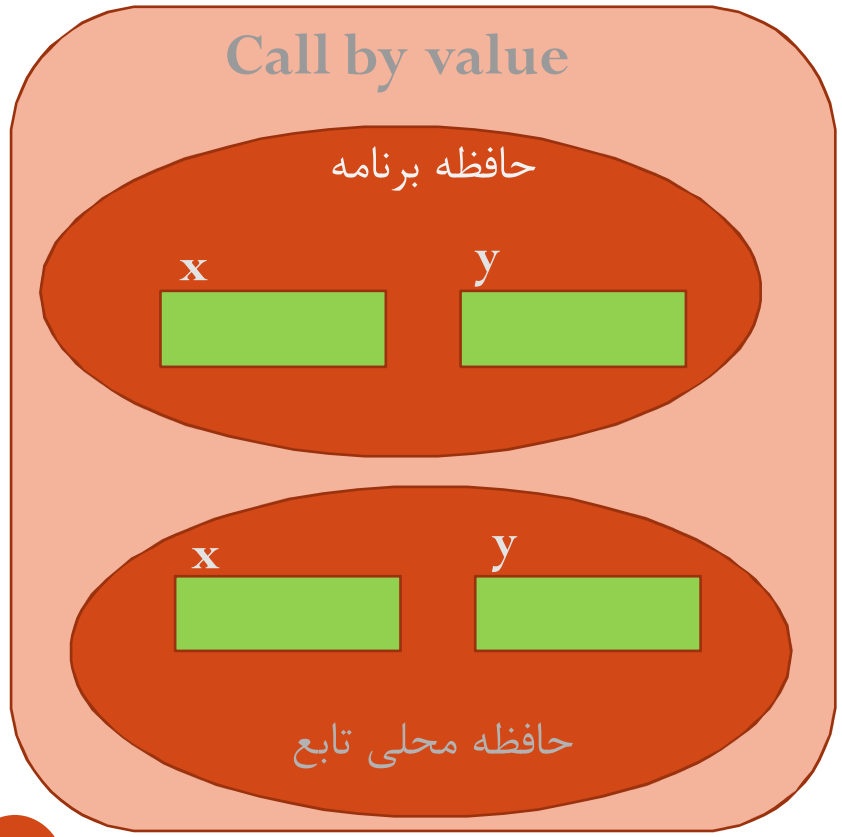
- نوشتن تابع

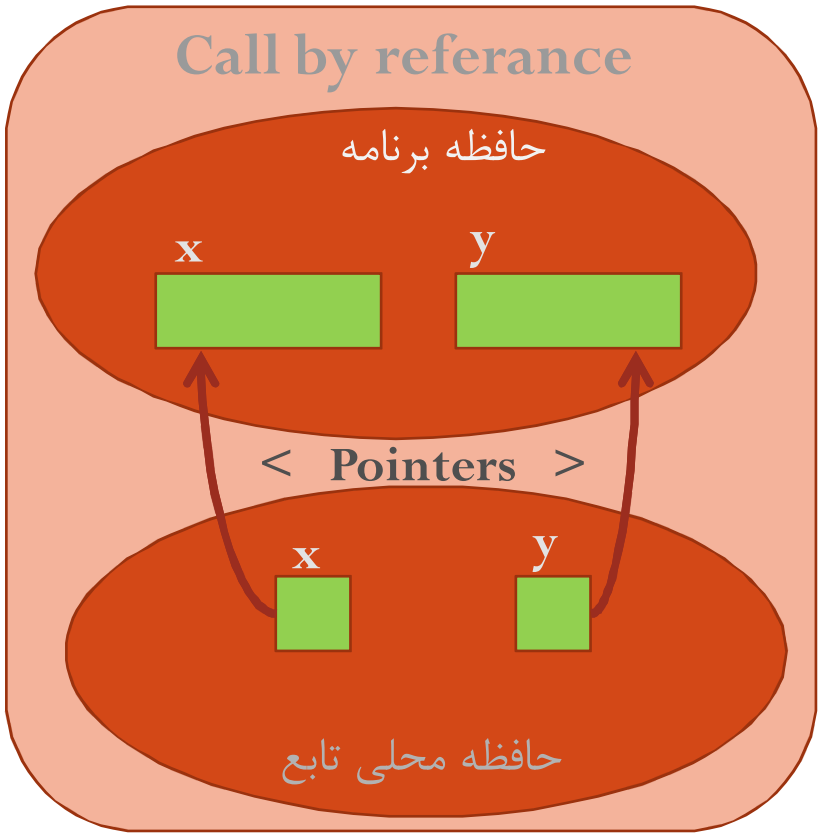
```
#include<stdio.h>
...
<نوع تابع>     نام تابع (لیست پارامتر ها)//الگوی تابع

Int main();
{
.....
فراخوانی تابع//(لیست پارامتر ها)نام تابع
}

<نوع تابع>     نام تابع (لیست پارامتر ها)// تعریف تابع
{دستورات تابع;}
```


omidi.students@gmail.com
www.jozve.org

توابع(ادامه)

- روشهای ارسال پارامترها به توابع





omidi.students@gmail.com
www.jozve.org

آرایه ها

- تعریف آرایه
  - آرایه یک بعدی

نوع آرایه نام آرایه[طول آرایه]

int x[5];

x

x[0]	X[1]	X[2]	X[3]	x[4]

- آرایه چند بعدی

نوع آرایه نام آرایه[طول بعد اول][طول بعد دوم]...[طول بعد آخر]

int x[3][4]

x

x[0][0]	x[0][1]	x[0][2]	x[0][3]
x[1][0]	x[1][1]	x[1][2]	x[1][3]
x[2][0]	x[2][1]	x[2][2]	x[2][3]

omidi.students@gmail.com
www.jozve.org

آرایه ها(ادامه)

در آرایه یک بعدی طول آرایه ذکر نمی شود، و در آرایه دوبعدی طول سطر ذکر نمی شود ولی طول ستون باید حتما ذکر شود.

- آرایه های یک بعدی به عنوان آرگومان تابع
  - الگوی تابع
    void func1(int x[])
  - فراخوانی تابع
    Int x[10]
    ....
    func1(x)
  - تعریف تابع
    void func1(int x[]) ...

- آرایه های دو بعدی به عنوان آرگومان تابع
  - الگوی تابع
    void func1(int x[5][10])
  - فراخوانی تابع
    Int x[5][10]
    ....
    func1(x)
  - تعریف تابع
    void func1(int x[][10]) ...

omidi.students@gmail.com
www.jozve.org

اشاره گرها

- متغیر های اشاره گر
  - نوع *متغیر;
  - Int *a;
- تخصیص حافظه پویا
  - اشاره گر = (نوع) malloc(size)
  - Int *p;

C

P = (int*) malloc(sizeof(int));

C++

P = New Int;

- برگرداندن حافظه به سیستم

C

Free(اشاره گر);

- Free(p);

C++

Delete(p)

omidi.students@gmail.com

www.jozve.org

ساختمان ها

- تعریف نوع ساختمان

```
Struct <ساختمان>  {
    عناصر ساختمان
}<اسامی متغیر های ساختمان>;

Struct personel{
   char name[31];
   int persno;
   int salary
};
```

- تعریف متغیر نوع ساختمان

```
Struct  نوع ساختمان  <اسامی متغیر های ساختمان>
Struct personel p1,p2;
```


omidi.students@gmail.com
www.jozve.org

ساختمان ها(ادامه)

- دسترسی به عناصر ساختمان

<نام عنصر> .<اسم متغیر از نوع ساختمان>

P1.name

P1.persno

- آرایه ای از ساختمان ها
  - Struct personel p[100];
- اشاره گر به ساختمان
  - Struct personel *p;

omidi.students@gmail.com

www.jozve.org

ارزیابی کارایی الگوریتم ها

- میزان حافظه لازم در زمان اجرای الگوریتم

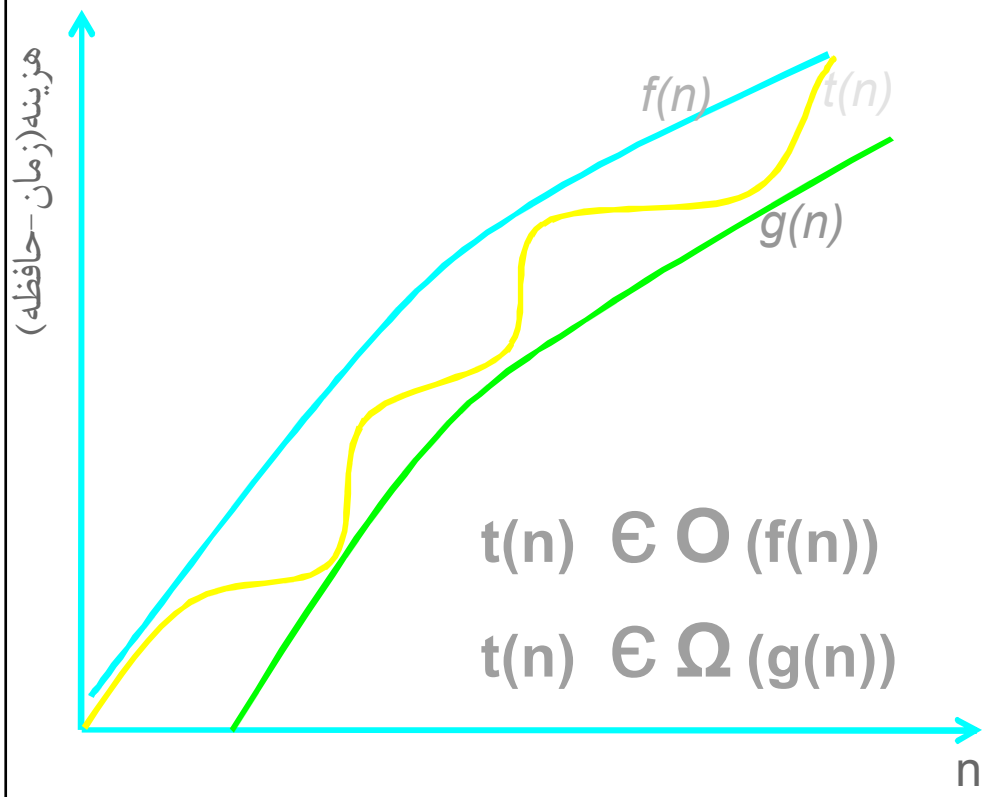
- زمان لازم برای اجرای کامل الگوریتم
  - زمان کامپایل
  - زمان اجرا

از آنجایی که تمام عوامل شناخته شده نیستند، پس ما فقط تخمینی از زمان را خواهیم داشت.

omidi.students@gmail.com
www.jozve.org

ارزیابی کارایی الگوریتم ها(ادامه)

- حالتهای ارزیابی الگوریتم ها
  - بدترین حالت o O
  - حالت متوسط θ
  - بهترین حالت ω Ω



$t(n) \in O(f(n))$

$t(n) \in \Omega(g(n))$

T(n) = (زمان انجام یک عمل جمع * تعداد جمع ها)* (زمان انجام یک عمل تفریق * تعداد کل تفریق ها) . . . .

omidi.students@gmail.com
www.jozve.org

ارزیابی کارایی الگوریتم ها(ادامه)

تعریف ریاضی O

$$O(f(n)) = \{t : N \to R^* \mid (\exists c \in R^+)(\exists n_0 \in N)(\forall n \geq n_0)[t(n) \leq cf(n)]\}$$

تعریف ریاضی Ω

$$\Omega(g(n)) = \{t : n \to R^* \mid (\exists d \in R^+)(\exists n_0 \in N)(\forall n \geq n_0)[t(n) \geq dg(n)]\}$$

تعریف ریاضی θ

$$\theta(f(n)) = \{t : n \to R^* \mid (\exists c, d \in R^+)(d \leq c)(\exists n_0 \in N)(\forall n \geq n_0)[df(n) \leq t(n) \geq cf(n)]\}$$

omidi.students@gmail.com
www.jozve.org

ارزیابی کارایی الگوریتم ها(ادامه)

- عملگر ها و دستورات مختلف در C++ که دارای هزینه زمانی $O(1)$ می باشند.

انتساب	And	Sizeof()
جمع	Or	Delete
تفریق	Not	Goto
ضرب	Shift	Break
تقسیم	>	Continue
توان	<	Call
دستورات مدیریت حافظه	>=	Return
دسترسی به خانه های آرایه	<=	
دسترسی به اعضای ساختمان	==	
	!=	

omidi.students@gmail.com

www.jozve.org

ارزیابی کارایی الگوریتم ها(ادامه)

$$f(n) \in O(s(n)) \qquad g(n) \in O(r(n)) \quad \Rightarrow \quad f(n)+g(n) \in O(s(n)+r(n)) \subseteq O(\max\{s(n), r(n))$$

مثال

$$f(n) \in O(n^2) \qquad g(n) \in O(n) \quad \Rightarrow \quad f(n)+g(n) \in O(n^2+n) \in O(n^2)$$

omidi.students@gmail.com

www.jozve.org

ارزیابی کارایی الگوریتم ها(ادامه)

$$f(n) \in O(s(n)) \quad g(n) \in O(r(n)) \Rightarrow f(n) * g(n) \in O(s(n) * r(n))$$

مثال

$$f(n) \in O(n^2) \quad g(n) \in O(n) \Rightarrow f(n) * g(n) \in O(n^2 * n) \in O(n^3)$$

omidi.students@gmail.com
www.jozve.org

ارزیابی کارایی الگوریتم ها(ادامه)

- هزینه هایی که بیشتر مورد استفاده قرار می گیرد

$$O(1) - O(\log_2^n) - O(n) - O(n\log_2^n) - O(n^2) \ldots O((\log_2^n)^2) - O(n!) - O(e^n)$$

$$\lim_{n\to\infty}\frac{g(n)}{f(n)}\begin{cases} L \Rightarrow g(n)\in\theta(f(n)) & (c>0) \\ 0 \Rightarrow g(n)\in o(f(n)) \quad or \quad f(n)\in\Omega(g(n)) \\ \infty \Rightarrow f(n)\in O(g(n)) \quad or \quad g(n)\in\Omega(f(n)) \end{cases}$$

omidi.students@gmail.com

www.jozve.org

ارزیابی کارایی الگوریتم ها(ادامه)

- چند مثال

```
for ( i=1 ; i<=n ; i++)
    /* o(1) */
```

O(n)

```
for ( i=1 ; i<=n ; i++)
    /* o(1) */
for ( j=1 ; j<=m ; j++)
    /* o(1) */
```

O(n)

O(m)

O(max(n,m))

omidi.students@gmail.com

www.jozve.org

ارزیابی کارایی الگوریتم ها(ادامه)

- چند مثال

```
for ( i=1 ; i<=n ; i++)
    for ( j=1 ; j<=m ; j++)
        /* o(1) */
```

O(n*m)

```
for ( i=1 ; i<=n ; i++)
    for ( j=i ; j<=n ; j++)
        /* o(1) */
```

$O(n^2)$

omidi.students@gmail.com

www.jozve.org

ارزیابی کارایی الگوریتم ها(ادامه)

- مثال

```
Func   sort(int s[] , int n)
{
if  (n==1)
O(1)    O(1)

        return(s)
            O(1)

    else

    {   break  s into tow halfs s1,s2 length of n/2
            O(1)

        return (sort(s1[],n/2) , sort(s2[],n/2))
            O(1)        T(n/2)              T(n/2)
    }
}
```

$$T(n) \in \begin{cases} O(1) & n = 1 \\ O(n) + 2T(n/2) & n > 1 \end{cases}$$

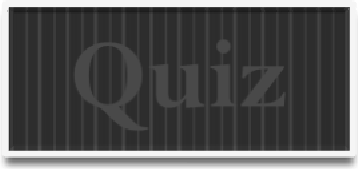
$$T(n) \in \begin{cases} 1 & n = 1 \\ 2T(n/2) + n & n > 1 \end{cases}$$

omidi.students@gmail.com

www.jozve.org

ارزیابی کارایی الگوریتم ها(ادامه)

- مثال



حال این معادله بازگشتی را با بسط دادن حل کنید

$$T(n) \in \begin{cases} 1 & n = 1 \\ 2T(n/2) + n & n > 1 \end{cases}$$

یادآوری

$$s(n) = \frac{a_1(q^n - 1)}{q - 1}$$

omidi.students@gmail.com

www.jozve.org

ارزیابی کارایی الگوریتم ها(ادامه)

- **یک رابطه مهم**

اگر داشته باشیم

$$T(n)=\begin{cases}1 & n=1\\ aT(\frac{n}{b})+d(n) & n>1\end{cases} \quad , \quad a,b\in N \quad , \quad b>1 \ , \quad d(n*m)=d(n)*d(m)$$

آنگاه

$$t(n)\in\begin{cases}O(n^{\log_b^{d(b)}}) & a<d(b)\\ O(n^{\log_b^{a}}*\log_b^{n}) & a=d(b)\\ O(n^{\log_b^{a}}) & a>d(b)\end{cases}$$

omidi.students@gmail.com

www.jozve.org

ارزیابی کارایی الگوریتم ها(ادامه)

- مثال

$$T(n)=\begin{cases}1 & n=1\\ 2T(\frac{n}{2})+n) & n>1\end{cases}$$

- حل

a=2
b=2
d(b)=2

a = d(b)

$$T(n)\in O(n^{\log_b^a}*\log_b^n)\Rightarrow t(n)\in O(n\log_2^n)$$

omidi.students@gmail.com

www.jozve.org

تکلیف منزل

- ۲ مورد از برنامه های بازگشتی را که درکتاب برنامه نویسی با آنها آشنا شدید، از نظر پیچیدگی زمانی بررسی کنید.(بهتر است رابطه ای بازگشتی از روی الگوریتم استخراج کرده و سپس آن را حل کنید)

- برنامه ای بنویسید که دو ماتریس A_{m*n} , B_{n*p} را در هم ضرب کند، سپس این برنامه را از نظر پیچیدگی زمانی بررسی کنید.

زمان تحویل تکلیف
هفته آینده می باشد

omidi.students@gmail.com

www.jozve.org

بحث جلسه بعد

- آرایه ها
  - بدست آوردن تعداد عناصر آرایه های یک بعدی و چند بعدی
  - بدست آوردن آدرس خانه های آرایه های یک بعدی و چند بعدی
  - معرفی ماتریس خلوت
  - ماتریس های بالا مثلثی و پایین مثلثی
  - چند نمونه از کاربرد های ماتریسها
  - تطبیق الگو

omidi.students@gmail.com
www.jozve.org