ساختمان داده ها

مدرس: یاسر اکبری
دانشگاه فنی و حرفه ای قم
بهار ۱۳۹۶

محتویات درسی:
computer8.blogfa.ir
ایمیل:
271akbari@gmail.com

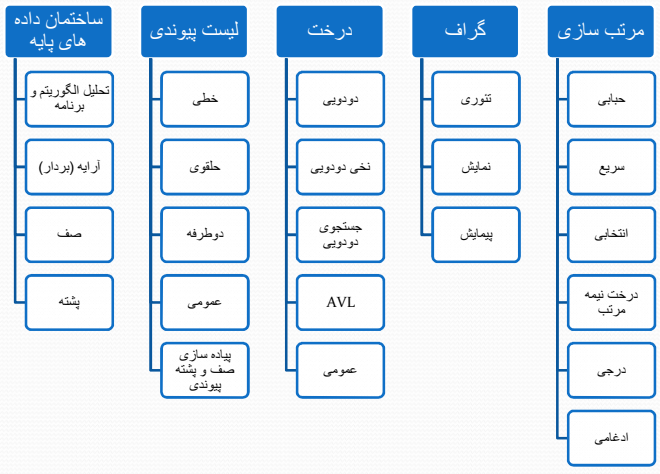
- **بارم بندی**
  - امتحانات کتبی (۱۶ نمره):
  - پایانترم ۱۰ نمره + میانترم ۶ نمره
  - تمرینات (۴ نمره)

عنوان مرجع:

- Fundamental of data Structure in C++ *Ellis Horowitz, Sartaj Sahni, Dinesh Mehta*
- ساختمان داده ها در C++ - حسین ابراهیم زاده قلزم - انتشارات سیمای دانش
- ساختمان داده ها به زبان C - امیر علیخانزاده - انتشارات باغانی
- کتاب "اصول ساختمان داده ها در C++"، مولف حسن علیزاده، انتشارات ناقوس، سال ۸۹.
- جزوات خانم لیلا معظمی دانشگاه جامع علمی کاربردی
- کتاب مهندسی کامپیوتر مهندس سمیع و کاجی

پس از پایان این درس از فراگیر انتظار می‌رود که:

ساختارهای خطی و غیرخطی را توضیح دهد	شناختی	درک و فهم	۱-تعریف ساختارهای خطی (LINEAR STRUCTURE) و غیرخطی (NON-LINEAR STRUCTURE)
نحوه نمایش رشته ها را شرح دهد	شناختی	درک و فهم	۲- رشته ها (STRINGS) نحوه نمایش رشته ها
عملیات روی رشته ها را تشریح کند	شناختی	درک و فهم	۲-۱- عملیات روی رشته ها (تعیین رشته ،مجاورسازی ،جستجوی یک زیررشته درداخل رشته دیگر،انتخاب زیررشته ای از رشته دیگر،مقایسه رشته ها،حذف زیررشته ای از داخل رشته ،زیررشته ای درداخل رشته دیگر،جایگزین کردن قسمتی از یک رشته با زیررشته دیگر،تهیه کپی از روی یک رشته)
رکورد ها را توصیف کند	شناختی	دانش	۳- رکورد ها -نحوه توصیف رکورد ها - انجام عملیات روی رکورد ها
عملیات رایج را روی رکورد ها توضیح دهد	شناختی	درک و فهم	SEARCH ،SORT، حذف و اضافه
نحوه نمایش آرایه ها را در حافظه کامپیوتر نشان دهد	شناختی	کاربرد	۲- آرایه ها - نحوه نمایش آرایه های یک ،دو... n بعدی در حافظه کامپیوتر
آدرس یک عنصر در آرایه های یک ،دو... n بعدی را محاسبه کند	شناختی	کاربرد	- نحوه محاسبه آدرس عنصری در آرایه های یک ،دو... n بعدی
روی آرایه ها عملیات معمول را انجام دهد	شناختی	کاربرد	- انجام عملیات SORT ،SEARCH، حذف و درج در آرایه ها
محاسن ،معایب و محدودیتهای آرایه ها را توضیح دهد	شناختی	درک و فهم	- محاسن ،معایب و محدودیتهای آرایه ها و کاربرد آنها
			- ماتریس ها



تعاریف

انجام کار ها در کامپیوتر

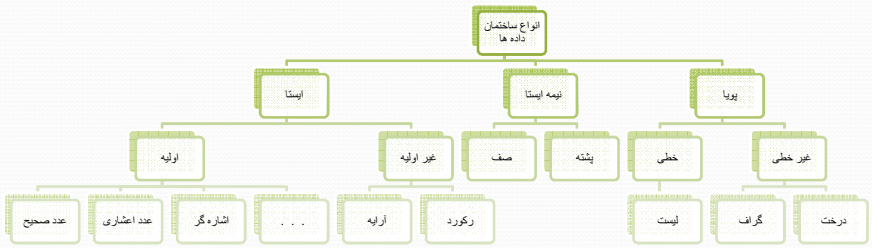
تعریف ساختمان داده

- ساختارهایی که جهت دریافت داده های خام به شکل مناسب توسط کامپیوتر مورد استفاده قرار می گیرند.
- پیاده سازی و اجرای الگوریتمهای مختلف روی آنها انجام می شود.
- مدل ریاضی سازماندهی داده ها

تحلیل الگوریتم ها – داده های ایستا – داده های نیمه ایستا – داده های پویای خطی – داده های پویای غیرخطی – الگوریتم های مرتب سازی

تحلیل الگوریتم ها

انواع ساختمان داده ها



تفاوت داده های ایستا و پویا

- طول حیات ساختار ساختمان داده ها
  - ایستا تغییر نمی کند.
  - پویا تغییرات مجاز و نامحدود می کند.

تعاریف الگوریتم

- گرفته شده از ریشه خوارزمی
- روش انجام کار یا حل مسئله
- روشی گام به گام برای حل مسئله
- الگوریتم گاه دارای مراحلی است که تکرار می‌شود و یا در مرحله‌ای نیازمند تصمیم‌گیری است.
- بیان الگوریتم با شبه کد

مجموعه ای محدود از دستورالعملهای مرتب که با اجرا شدن هدف خاصی محقق می شود.

ویژگی یک الگوریتم

1. ورودی:
   ورود صفر یا چند کمیت از محیط خارج
2. خروجی:
   خروج حداقل یک کمیت
3. قطعیت:
   تمام دستورالعملهای آن کاملا بدون ابهام باشند
4. محدودیت:
   خاتمه یافتن پس از طی مراحل محدود
5. کارایی:
   قابل اجرا بودن هر دستورالعمل (حتی دستی و با قلم و کاغذ)

تفاوت الگوریتم و برنامه

الگوریتم باید پایان پذیر باشد. ولی برنامه ممکن است پایان پذیر نباشد. مانند سیستم عامل که همواره در سیکل انتظار اجرای برنامه جدید است.

معیارهای کارایی یک الگوریتم

- عوامل زیادی در ارزیابی یک برنامه موثرند.
- زمان مصرفی
  - با شمارش و محاسبه تعداد عملیات کلیدی الگوریتم اندازه‌گیری می‌شود.
- حافظه مصرفی
  - با شمارش و محاسبه ماکزیمم حافظه مورد نیاز الگوریتم اندازه‌گیری می‌شود.

معیارهای کارایی یک الگوریتم

- عوامل زیادی در ارزیابی یک برنامه موثرند.
- زمان مصرفی
  - با شمارش و محاسبه تعداد عملیات کلیدی الگوریتم اندازه‌گیری می‌شود.
- حافظه مصرفی
  - با شمارش و محاسبه ماکزیمم حافظه مورد نیاز الگوریتم اندازه‌گیری می‌شود.

پیچیدگی الگوریتم

- M یک الگوریتم و n تعداد داده‌های ورودی است.
  - پیچیدگی الگوریتم M برابر با f(n) است که زمان اجرای مورد نیاز الگوریتم را بر حسب تعداد داده ورودی یعنی n می دهد.
- حافظه اغلب اوقات فقط مضربی از n است بنابراین در پیچیدگی لحاظ نمی شود.
- بهترین حالت، حالت میانگین، بدترین حالت
  - پیچیدگی یک الگوریتم یعنی زمان اجرای آن در بدترین حالت.

۱۴

محاسبه پیچیدگی الگوریتم ها

- بهترین حالت
  - حداقل مقدار ممکن f(n)
- حالت میانگین
  - مقدار مورد انتظار یا امید ریاضی f(n) است.
  - فرض کنید اعداد n_1 و ... و n_k به ترتیب با احتمال p_1 و ... و p_k ظاهر می شوند. امید ریاضی این اعداد برابر است با:
  - $E = n_1*p_1 + n_2*p_2 + ... + n_k*p_k$
- **بدترین حالت**
  - ماکزیمم مقدار f(n) برای هر ورودی ممکن است.

۱۵

پیچیدگی الگوریتم مثال

تعداد عملیات تکرار حلقه در ادامه آمده است:

- بهترین حالت
  - C(n)=1
- بدترین حالت
  - C(n)=n
- حالت میانگین
  - C(n)= (n+1)/2

C(n) = 1.(1/n) + 2.(1/n) + ... + n.(1/n) =
1/n * (1+2+...+n) =
(n+1)/2

۱۶

مرتبه اجرایی

- **ملاک مقایسه چند الگوریتم ارائه شده برای یک مسئله**
  - **مرتبه (سرعت) اجرا**

- اگر همه الگوریتم ها بر روی یک ماشین اجرا شوند.
- اگر سرعت اجرای کلیه دستورات معمولی برابر واحد فرض شود.

- زمان اجرای الگوریتمها بصورت **تابعی** از تعداد ورودی های مسئله محاسبه می شود.

عمل کلیدی الگوریتم

به دستوراتی گفته می‌شود که کل کار انجام شده توسط الگوریتم متناسب با تعداد دفعاتی باشد که این دستورات انجام می‌شود.

تعیین مرتبه اجرایی الگوریتم ها

مثال مرتبه اجرا | **سه مرتبه متفاوت**

1. کد

```
X:=X+1;
```

2. کد

```
For i:=1 to n do
   X:=X+1;
```

3. کد

```
For i:=1 to n do
   For i:=1 to n do
      X:=X+1;
```


تعیین مرتبه اجرایی الگوریتم ها

- مرتبه اجرایی تابع $f(n)$ ، تابع $g(n)$ است هر گاه:

- این رابطه دو شرطی بدین معنی است که اجرای تابع $f(n)$ بیش از $g(n)$ زمان مصرف نمی کند.

برای تعیین مرتبه اجرایی الگوریتم ها فرمولهای زیر مفید هستند:

1+1+1+1
4+3+2+1
16+9+4+1

آهنگ رشد توابع

آهنگ رشد توابع

- پیچیدگی الگوریتم M یعنی f(n) با زیاد شدن تعداد داده‌های ورودی یعنی n افزایش می‌یابد. برای تعیین آهنگ این افزایش، f(n) را با تعدادی از توابع مانند توابع زیر بیان و مقایسه می کنیم:

- این توابع به ترتیب آهنگ رشدشان نوشته شده‌اند.

۲۳

آهنگ رشد (Big O)

- برای مقایسه f(n) با توابع استاندارد از نماد O استفاده می‌کنیم که به صورت زیر تعریف می‌شود:

f(n) و g(n) دو تابع روی اعداد صحیح مثبت هستند و این خاصیت را دارند که f(n) به ازای تمام مقادیر n توسط مضربی از g(n) محدود شده است. به عبارت دیگر فرض کنید عدد صحیح مثبت n_0 و عدد مثبت C وجود دارند طوریکه به ازای تمام nهای بزرگتر از n_0 داریم:

|f(n)|<= C|g(n)|

در این صورت می‌توان نوشت: f(n) از مرتبه g(n) است یا:

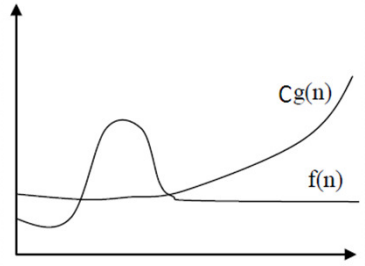
f(n) = O(g(n))

- f(n)=O(g(n)) است اگر و فقط اگر مقادیر ثابت n_0 و c وجود داشته باشد طوریکه برای تمامی مقادیر n که $n \geq n_0$، f(n) کوچکتر یا مساوی cg(n) باشد.

۲۴

BIG O (O)

$$f(n) \in O(g(n)) \; if \; \exists \; (C > 0, n_0 > 0) \; s.t. \; \forall n \geq n_0 : |f(n)| \leq C|g(n)|$$



- O(1) : زمان محاسبه ثابت است.
- O(n) : یک تابع خطی
- $O(n^2)$: یک تابع درجه دو

قضیه

چند جمله ای P(x) درجه n یعنی

$$P(x) = a_0 + a_1x + a_2x^2 + \ldots + a_nx^n$$

از مرتبه $O(x^n)$ است.

نکات

- چنانچه بین توابع استاندارد عمل + یا – باشد، مرتبه جمله برابر با مرتبه قویترین تابع آن است.
- $n + \text{Log } n + n^2 = O(n^2)$
- $2^n + n^3 + n = O(2^n)$
- همیشه باید کوچکترین O ممکن را برای تعیین پیچیدگی به کار برد.
- $3n+3 = O(n^2)$
- $3n+3 = O(n)$ $\rightarrow 3n+3 = O(n)$

Omega (Ω)

- $f(n)=\Omega(g(n))$ است اگر و فقط اگر مقادیر مثبت n_o و c وجود داشته باشد طوریکه برای تمامی مقادیر n که $n \geq n_o$، $|f(n)|$ بزرگتر یا مساوی $|cg(n)|$ باشد.
- همیشه باید بزرگترین $g(n)$ ممکن انتخاب شود.
- $3n^2+2 = \Omega(n^2)$
- $3n^2+2 = \Omega(n)$ → $3n^2+2 = \Omega(n^2)$

قضیه

چند جمله ای $P(x)$ درجه n یعنی

$$P(x) = a_o + a_1x + a_2x^2 + \dots + a_nx^n$$

که در آن $a_n > o$ است آنگاه $P(x)=\Omega(x^n)$ است.

Theta (Θ)

- $f(n) = \theta(g(n))$ اگر و فقط اگر ثابتهای مثبت C_1 و C_2 و n_o وجود داشته باشند که برای تمام $n \geq n_o$ داشته باشیم:

$$|C_1g(n)| <= |f(n)| <= |C_2g(n)|$$

- θ دقیقتر از O و Ω می باشد و مرتبه دقیق $f(n)$ را نشان می دهد.
- $f(n) = \theta(g(n))$ اگر و فقط اگر $g(n)$ هم کران بالا هم کران پایین $f(n)$ باشد.

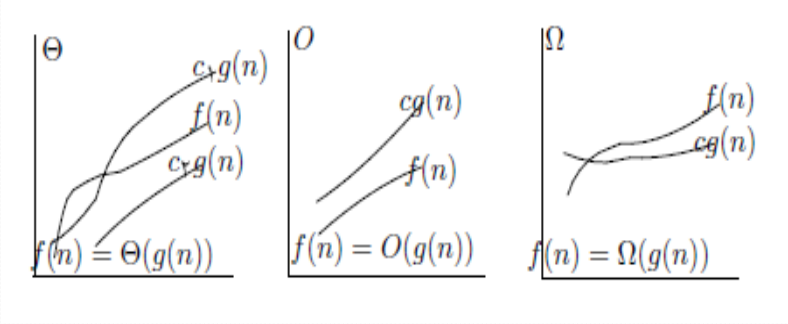
قضیه

چند جمله ای $P(x)$ درجه n یعنی

$$P(x) = a_o + a_1x + a_2x^2 + \dots + a_nx^n$$

که در آن $a_n > o$ است آنگاه $P(x)=\Theta(x^n)$ است.

مقایسه



۲۹

مثال

پیچیدگی الگوریتم زیر را تعیین کنید.

```
X=0
i=n
While (i>1)
{
        X--;
        i%=2
}
```

۳۰

مثال

پیچیدگی الگوریتم زیر را تعیین کنید.
(m=n)

```
For i:=1 to n
        For j:=1 to m
                X++;
```

مثال

پیچیدگی الگوریتم زیر را تعیین کنید.

```
X=0
i=n
While (i>1)
{
        X--;
        i/=2
}
```

قضیه

راه حل مسائل

عموما روابطی شبیه جدول زیر بدست میآید. با روش:

مثال

1- در الگوریتم رو برو در صورتی که n=m باشد مرتبه اجرایی کدام است؟

```
for i:=1 to n do
   for j:=1 to m do
         for k:=1 to j do
               x:=x+1
```


مثال

-2

مثال زمان اجرایی؟

$$T(n) = \begin{cases} ۱, & n = ۱ \\ n + T(n - ۱), & n \geq ۲ \end{cases}$$

مثال

3- اگر n=8 باشد. چند عمل ضرب انجام میشود؟ (هر عمل square نیز یک عمل ضرب نیاز دارد.)

```
Function count (n)
  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) ) )
```

با هر فراخوانی تابع بجز مرحله آخر یکبار شمارش عمل ضرب مستقیم و یک بار شمارش عمل ضرب غیر مستقیم انجام می شود. = هشت عمل ضرب

تست26 ارشد 1378

مثال

4- مقدار ACK(1,1) در تابع رو به رو چقدر می شود؟

```
int ACK (int m, int n)
  {
  if (m<0 or n<0)
        return 0 :
  else if (m==0)
        return n+1
        else if (n==0)
              return ACK (m-1 , 1);
              else
              return ACK (m-1 , ACK (m , n-1)  );
  }
```

مقدار سه جواب تابع است.

تست 70 ارشد 77

تحلیل الگوریتم ها

مثال

5- مقدار چقدر F(3,6) است؟

```
int F (int m, int n)
   {
   If (m==1 ll n==0 ll m==n)
         return 1;
   else
   return F(m-1,n)+F(m-1,n-1)
}
```


تحلیل الگوریتم ها

مثال

6- زمان اجرای تابع باز گشتی زیر چیست؟

```
int test (int n)
   {
   if (n<= 2)
         return 1;
   else
         return test (n-2) * test (n-2);
   }
```

مثال

7- مقدار تابع برگشتی recursive(5) چقدر است؟

```
function recursive (n);
begin
  if n=1 then
        recursive :=1
        else
              recursive:= recursive(n-1) + recursive(n-1)
end
```


ارزیابی

1- ماتریس پراکنده یا خلوت چیست؟

ارزیابی

4- مقدار تابع برگشتی recursive(5) چقدر است؟

```
function recursive (n);
begin
   if n=1 then
         recursive :=1
         else
                recursive:= recursive(n-1) + recursive(n-1)
end
```

a) 13
b) 14
c) 15
d) 16

ارزیابی

5- مقدار تابع F(3,6) چقدر است؟

```
int F (int m, int n)
  {
   If (m==1 ll n==0 ll m==n)
       return 1;
   else
   return F(m-1,n)+F(m-1,n-1)
  }
```

a) 3
b) 4
c) 5
d) 6

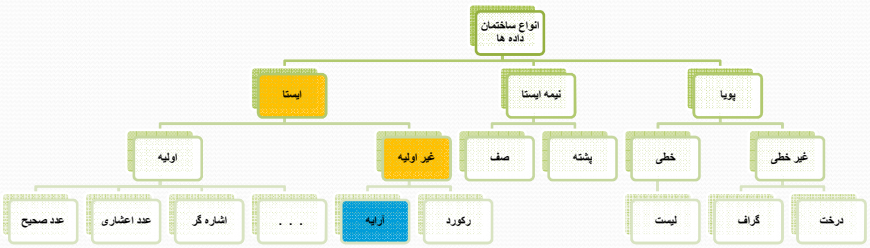
تحلیل الگوریتم ها – داده های ایستا – داده های نیمه ایستا – داده های پویای خطی – داده های پویای غیرخطی – الگوریتم های مرتب سازی

داده های ایستا

یادآوری مفاهیم برنامه نویسی

- متغیر
- ساختارهای تصمیم و گزینش
  - if else if else
- عبارت های شرطی
  - if (a<=s && e>2) {دستورات}
- ساختارهای تکرار
  - for i=1 to 100 do { دستورات }
  - while (عبارت شرطی) { دستورات }
- حلقه های تو در تو
  - for (i=0; i<20; i++) { for (j=0; j<100; j++) { دستورات } }

آرایه



تعریف آرایه یک بعدی

- آرایه:

A[5]

A

31	25	40	26	19

- A[0]=31
- A[1]=25
- A[2]=40
- A[3]=26
- A[4]=19

آرایه دو بعدی

B

31	25	40	26	19
23	34	14	56	67
70	54	10	41	98

B[3][5]

- B[0][0]=31
- B[0][1]=25
- B[0][2]=40
- B[0][3]=26
- B[0][4]=19
- B[1][0]=23
- B[1][1]=34
- B[1][2]=14
- B[1][3]=56
- B[1][4]=67
- B[2][0]=70
- B[2][1]=54
- B[2][2]=10
- B[2][3]=41
- B[2][4]=98

شماره سطر

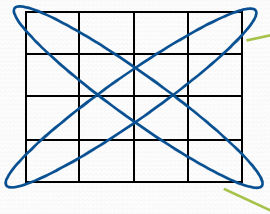
شماره ستون

آدرس یک خانه از آرایه سه بعدی

- آدرس خانه (i, j, k) را از آرایه A[p][q][m] (یک آرایه p*q*m عنصری) بدست آورید. (فرض شده است آدرس خانه اول آرایه صفر است و همه بعدها از اندیس صفر شروع شده‌اند.)
- address A[i][j][k] = i*q*m + j*m + k
- مثال از آرایه دو بعدی صفحه قبل
- address B[1][2] = 1*5 + 2 = 7
- همین محاسبات را میتوان برای آرایه n بعدی تعمیم داد.

آرایه

- تعداد عناصر آرایه: (تعداد عناصر بعد اول) * (تعداد عناصر بعد دوم) * ... * (تعداد عناصر بعد nام)
- آرایه دو بعدی را ماتریس گویند.
- ماتریس مربعی
  - ماتریسی که تعداد سطر و ستون برابری داشته باشد.
- ماتریس مربعی دارای دو قطر است.



- عناصری که شماره سطر (i) و شماره ستون (j) آنها برابر باشد روی قطر اصلی قرار دارند.

ماتریس‌های خاص

1. ماتریس بالا مثلثی:
   - اگر عناصر زیر قطر اصلی ماتریس مربعی صفر باشد ماتریس بالا مثلثی است.
2. ماتریس پایین مثلثی:
   - اگر عناصر بالای قطر اصلی ماتریس مربعی صفر باشد ماتریس پایین مثلثی است.

ماتریس پایین مثلثی

x	o	o	o
x	x	o	o
x	x	x	o
x	x	x	x

ماتریس بالا مثلثی

x	x	x	x
o	x	x	x
o	o	x	x
o	o	o	x

3. ماتریس sparse یا خلوت:
   - اگر تعداد عناصر صفر ماتریس بیشتر از عناصر غیر صفر آن باشد.

داده های ایسنا

ماتریس‌های خاص

ماتریس بالا مثلثی یا پایین مثلثی

x	o	o	o
x	x	o	o
x	x	x	o
x	x	x	x

x	x	x	x
o	x	x	x
o	o	x	x
o	o	o	x

- ماتریس n×n
- تعداد کل عناصر: n^2
- تعداد عناصر غیرصفر: $n(n+1)/2$
- تعداد عناصر صفر: $n(n-1)/2$

داده های ایسنا

ماتریس‌های خاص

۱-نحوه ذخیره ماتریس پایین مثلثی n×n

- در این ماتریس خانه مخالف صفر
  - ✓ $n(n+1)/2$ خانه مخالف صفر وجود دارد.
  - ✓ برای همین یک آرایه یک بعدی با $n(n+1)/2$ خانه در نظر می گیریم
- برای هر خانه مخالف صفر ماتریس پایین مثلثی، رابطه های زیر محل آن خانه را در آرایه یک بعدی مشخص می کند.
  - ✓ فرمول سطری
  - ✓ فرمول ستونی

داده های ایسا

ماتریس‌های خاص

۲-نحوه ذخیره ماتریس بالا مثلثی n×n

- در این ماتریس خانه مخالف صفر
  - ✓ $n(n+1)/2$ خانه مخالف صفر وجود دارد.
  - ✓ برای همین یک آرایه یک بعدی با $n(n+1)/2$ خانه در نظر می گیریم.
- برای هر خانه مخالف صفر ماتریس بالا مثلثی، رابطه های زیر محل آن خانه را در آرایه یک بعدی مشخص می کند.
  - ✓ فرمول سطری
  - ✓ فرمول ستونی

داده های ایسا

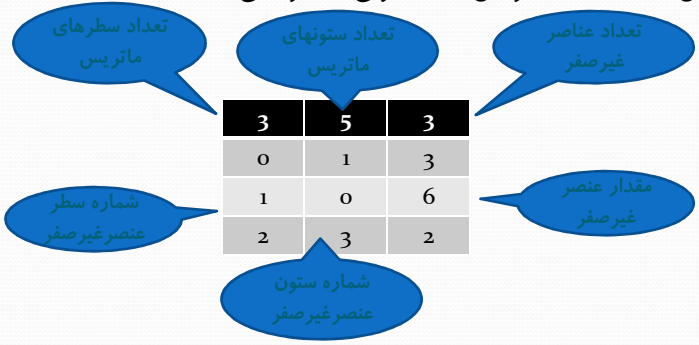
ماتریس‌های خاص

۳-نحوه ذخیره‌سازی ماتریس‌های خلوت

0	3	0	0	0
6	0	0	0	0
0	0	0	2	0

- اشغال بسیار زیاد و بیهوده حافظه
- برای صرفه‌جویی در حافظه،

این ماتریس را در یک ماتریس سه ستونی ذخیره می‌کنند.



ماتریس‌های خاص

۳-نحوه ذخیره‌سازی ماتریس‌های خلوت

- برای مثال اگر یک ماتریس خلوت 6000×5000 دارای 5000 عضو غیر صفر باشد، ماتریس خلوت، 30 میلیون خانه از حافظه اشغال می‌کند اما ذخیره آن در یک ماتریس سه ستونی، تنها 15003 خانه از حافظه اشغال می‌کند.
- این روش مقرون به صرفه خواهد بود. اگر ...
  -
  -

پردازش روی آرایه‌ها

1. پیمایش عناصر آرایه :
   -
2. اضافه کردن
   -
   -
3. حذف کردن
   -
   -

پردازش روی آرایه‌ها ۱-پیمایش آرایه

- آرایه A[n] را پیمایش کنید.

```
i = 0
while (i<n)
    apply process to A[i]
    i++
```

```
for i = 0 to n-1
    apply process to A[i]
```


پردازش روی آرایه‌ها ۲-الگوریتم اضافه کردن یک عنصر

- عنصر item را به انتها و خانه kام آرایه A[n] اضافه کنید (در این مساله **منظور از n تعداد عناصر درج شده در آرایه است و نه اندازه آرایه**).

افزودن به انتها

```
A[n] = item
n = n+1   یا   n++
```

افزودن عنصر به خانه kام

```
for i = n-1 to k
    A[i+1] = A[i]
A[k] = item
n++
```

O(n)

43	22	35	90	51	62	
43	22	35	66	90	51	62

داده های ایسا

پردازش روی آرایه‌ها ۲-الگوریتم حذف یک عنصر

- عنصر item را از انتها و خانه kام آرایه A[n] حذف کنید. (در این مساله **منظور از n تعداد عناصر درج شده در آرایه است و نه اندازه آرایه**).

حذف عنصر از خانه kام

```
item = A[k]
for i = k+1 to n-1
        A[i-1] = A[i]
n--
```

حذف از انتها

```
item = A[n-1]
A[n-1] = null
n--
```

43	22	35	66	90	51	62

43	22	66	90	51	62

O(n)

داده های ایسا

کارایی آرایه‌ها

- بسیار سریع و آسان است.
  - دسترسی به عناصر آرایه‌ها
- آرایه‌ها برای ذخیره این مجموعه داده‌ها، کارایی مطلوبی ندارند
  - اگر در یک مجموعه داده، دائما عمل حذف یا اضافه صورت گیرد
  - چون این عملیات به طور متوسط منجر به انتقال نصف عناصر آرایه می‌شوند.

انواع روشهای جستجو

- جستجو در آرایه نامرتب
  - جستجوی خطی یا ترتیبی (linear search)
- جستجو در آرایه مرتب
  - جستجوی خطی یا ترتیبی (linear search)
  - جستجوی دودویی (binary search)
- این جستجوها برای هر نوع ساختمان داده خطی به (علاوه بر آرایه) به کار می‌رود.

جستجوی خطی

- محل عنصر x را در آرایه نامرتب A[n] پیدا کنید.

```
Loc = -1
for i = 0 to n-1 {
    if (A[i] == x){
        Loc=i
        break
    }
}
if (Loc == -1)    write: x is not in the data
else              write: Loc is the location of x
```

43	22	35	90	51	62

O(n)

جستجوی دودویی

- الگوریتم جستجوی دودویی روشی برای جستجوی یک مقدار درون یک لیست مرتب است. عنصر وسط لیست انتخاب شده و با عنصر جستجو مقایسه می‌شود تا تعیین شود از آن بزرگتر، کوچکتر یا مساوی است. اگر عنصر جستجو از عنصر وسط بزرگتر باشد جستجو در نیمه بالایی و اگر کوچکتر باشد جستجو در نیمه پایینی لیست ادامه پیدا می‌کند.
- پیچیدگی الگوریتم جستجوی دودویی $O(\log n)$ است.
- محدودیت جستجوی دودویی این است که نیاز به لیست مرتب شده دارد.

الگوریتم غیر بازگشتی جستجوی دودویی

- محل عنصر x را در آرایه مرتب A[n] پیدا کنید.

```
int BinarySearch(A, x, low, high) {
 low = 0
 high = n-1
 while (low < n) {
        mid = (low + high) / 2
        if (x<A[mid])    high = mid-1
        else if (x>A[mid])    low = mid+1
        else    return mid  //found
 }
 return -1    //not found
}
```

O(log n)

داده های ایسنا

مثال از کارآمدی جستجوی دودویی

- جستجوی دودویی برای یافتن عدد ۱۶۰ در آرایه زیر، چند عنصر را بررسی می‌کند؟ جستجوی خطی چند عنصر؟

84	22	96	149	12	35	190	104	65	102

12	22	35	65	84	96	102	104	149	190

0+9/2=4 → 84

5+9/2=7 → 104

8+9/2=8 → 149

9+9/2=9 → 190

- جستجوی دودویی با بررسی ۴ عنصر و جستجوی خطی با بررسی ۱۰ عنصر.

داده های ایسنا

حداکثر تعداد مراحل جستجوی دودویی

- جستجوی دودویی برای یافتن یک عدد در آرایه‌ای با ۱۰۲۴ عنصر، حداکثر چند مرحله طی می‌کند؟
- تعداد عناصر در هر مرحله نصف می‌شود.

1024
512
256
128
64
32
16
8
4
2
1

- حداکثر تعداد مراحل برابر با است.

توابع بازگشتی

- تابع بازگشتی اجازه بیان راه‌حل یک مسئله را به طور مختصر و مفید می‌دهد. مسئله‌ای که به صورت بازگشتی حل می‌شود باید بتواند به مسائل کوچکتر تقسیم بشود و حل مسائل کوچک به همان روش مسئله بزرگ قابل انجام باشد. مسئله کوچکتر به مسئله کوچکتری شکسته می‌شود تا سرانجام به کوچکترین اندازه مسئله برسد که حالت پایه نامیده می‌شود که می تواند بدون استفاده از بازگشتی حل شود.
- مزایا:
  1. برنامه کوتاهتر (تعداد کم دستورات)
  2. راحتی پیاده سازی
- معایب:
  1. اتلاف حافظه
  2. سرعت اجرای کمتر

مثال : تابع فاکتوریل

- الگوریتم غیر بازگشتی تابع فاکتوریل را به صورت زیر تعریف می‌کنند.

```
t=1
for  i = 1 to n  {
        t = t*i
}
```

مثال : تابع فاکتوریل

- الگوریتم بازگشتی تابع فاکتوریل را به صورت زیر تعریف می‌کنند.

- تابع !3 با حل بازگشتی به صورت زیر حل می‌شود:

3*2!
2*1!
1*0!
1
1*1
2*1
3*2=6

مثال : تابع ضرب

- تابع بازگشتی و غیربازگشتی ضرب را با استفاده از جمع به صورت زیر تعریف می‌کنند.

```
int Mul (int a, int b)
{
 if (b == 1)
   return a
 else
   return a + Mul (a,b-1)
}
```

```
int Mul (int a, int b)
{
   s=0
 for i=1 to b
        s=s+a
}
```

الگوریتم بازگشتی جستجوی دودویی

- محل ع

```
int BinarySearch(A, x, low, high) {
  if (high < low)
    return -1     // not found
  mid = (low + high) / 2
  if (A[mid] > x)
    return BinarySearch(A, x, low, mid-1)
  else if (A[mid] < x)
    return BinarySearch(A, x, mid+1, high)
  else
    return mid    // found
}
```

O(log n)

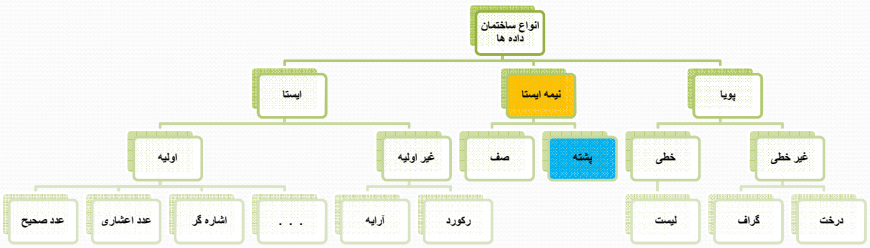
داده های نیمه ایستا

تحلیل الگوریتم ها – داده های ایستا – داده های پویای خطی – داده های پویای غیرخطی – الگوریتم های مرتب سازی

پشته

داده های نیمه ایستا

پشته



داده های نیمه ایستا

تعریف پشته

- پشته (stack) یک ساختمان داده خطی از عناصر است.
- عمل حذف و اضافه عناصر تنها می‌تواند از یک انتهای آن موسوم به بالای پشته (top) انجام گیرد.
- آخرین ورودی اولین خروجی (Last In First Out) (LIFO)
- ترتیبی که عنصرها از پشته خارج می‌شوند برعکس ترتیبی است که وارد آن می‌شوند.
- پشته بشقابها مثالی از پشته است.

داده های نیمه ایستا

پیاده سازی پشته

- برای اضافه کردن عنصر در پشته از اصطلاح push استفاده می‌گردد.
- برای حذف یک عنصر در پشته از اصطلاح pop استفاده می‌گردد.

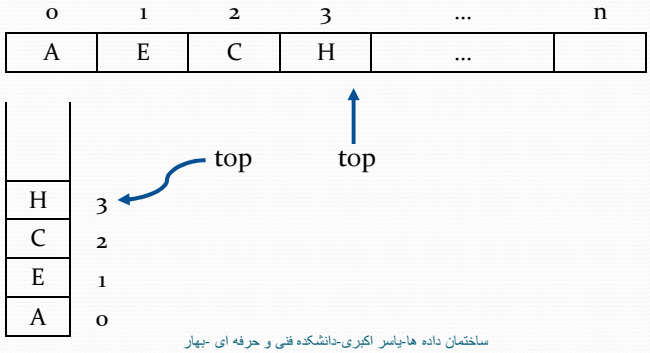
- هر پشته توسط عناصر زیر پیاده سازی می شود:
  - آرایه خطی Stack
  - متغیر اشاره‌گر Top: مکان عنصر بالای پشته
  - متغیر MAXSTK: بیشترین تعداد عنصر قابل نگهداری توسط پشته مثلا در یک آرایه n خانه ای n عنصر می‌تواند وجود داشته باشد.

داده های نیمه ایستا

نمایش پشته

- معمولا به کمک آرایه نمایش داده می شود. پشته را میتوان به صورتهای زیر نمایش داد.

- S: A, E, C, H



داده های نیمه ایستا

الگوریتم درج در پشته

- عنصر x را در پشته S[Maxstk] درج کنید.

```
If (top==Maxstk-1)  write “Stack is full”
Else {
        top++
        S[top] = x
}
```

O(1)

داده های نیمه ایستا

الگوریتم حذف از پشته

- عنصر بالایی پشته S[Maxstk] را حذف کنید.

```
If (top==-1){
        write “Stack is empty”
}
Else {
        y = Q[top]
        top--
}
```

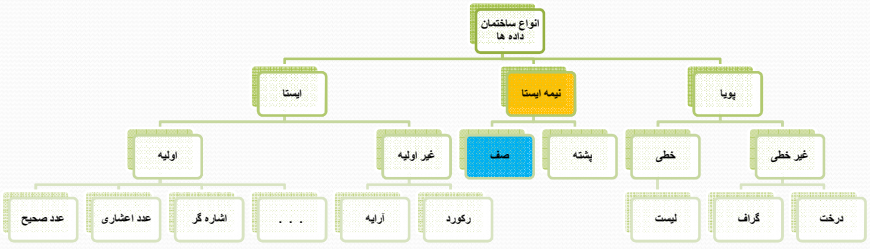
O(1)

تحلیل الگوریتم ها – داده های ایستا – **داده های نیمه ایستا** –
داده های پویای خطی – داده های پویای غیرخطی – الگوریتم های مرتب سازی

صف

دانشگاه جامع علمی-کاربردی

صف



داده های نیمه ایستا

تعریف صف

- صف (Queue) یک لیست خطی از عناصر است.
- عمل حذف تنها می‌تواند از یک انتهای آن موسوم به ابتدای صف (Front) انجام گیرد.
- عمل اضافه شدن تنها می‌تواند از انتهای دیگر آن به نام انتهای صف (Rear) انجام گیرد.
- اولین ورودی اولین خروجی (First In First Out) (FIFO)
- ترتیبی که عنصرها وارد صف می‌شوند همان ترتیبی است که از آن خارج می‌شوند.
- صفهای نانوایی و صف بانک مثالهای روزمره از صف هستند.
- صف پردازشهای سیستم عامل مثالی از کاربرد صف در رایانه است.

داده های نیمه ایستا

پیاده سازی صف

- برای نمایش صف معمولا از لیست یکطرفه یا آرایه خطی استفاده می‌شود.
- هر صف توسط عناصر زیر پیاده سازی می شود:
  - آرایه خطی Queue
  - متغیر اشاره‌گر Front: یک واحد کمتر از مکان عنصر ابتدای صف
  - متغیر اشاره‌گر Rear: مکان عنصر انتهای صف

داده های نیمه ایستا

پیاده سازی صف توسط آرایه

- Front: -1
- Rear: 3

A	B	C	D	...	
0	1	2	3	...	n-1

- Front: 0
- Rear: 3

	B	C	D	...	
0	1	2	3	...	n-1

- Front: 0
- Rear: 5

	B	C	D	E	F	...	
0	1	2	3	4	5	...	n-1

- Front: 1
- Rear: 5

		C	D	E	F	...	
0	1	2	3	4	5	...	n-1

داده های نیمه ایستا

شرایط اولیه و مرزی صف خطی

- انتخاب آرایه Q[n] برای پیاده سازی صف خطی دارای شرایط اولیه و مرزی زیر است:
  - ✓ وضعیت اولیه صف:
    - ✓ Front = Rear = -1
  - ✓ خالی بودن صف خطی:
    - ✓ Front = Rear
  - ✓ پر بودن صف خطی:
    - ✓ Rear = n-1

(هرچند ممکن است ابتدای صف خالی باشد اما قابل استفاده نیست.)

داده های نیمه ایستا

درج در صف و حذف از آن

- درج در صف:
  - ✓ برای درج، در صورت پر نبودن صف، چون Rear به خانه آخر صف اشاره میکند، ابتدا Rear یک واحد اضافه شده (به جلو حرکت کرده) سپس داده مورد نظر در خانه انتهای صف درج می شود
- حذف از صف:
  - ✓ برای حذف، در صورت خالی نبودن صف، چون Front همیشه به خانه قبل از اول صف اشاره می کند، ابتدا Front یک واحد اضافه شده (به جلو حرکت کرده) سپس داده ابتدای صف خارج شده و در متغیر مورد نظر قرار می گیرد.

داده های نیمه ایستا

تابع درج در صف

- عنصر

```
void add (int x){
    If (rear==n-1)  write "Queue is full"
    Else {
        rear++
        Q[rear] = x
    }
}
```

O(1)

داده های نیمه ایستا

تابع حذف از صف

- عنصر اول صف Q[n] را حذف کنید.

```
int delete(){
    If (front==rear){
        write “Queue is empty”
        return 0
    }
    Else {
        front++
        y = Q[front]
        return y
    }
}
```

O(1)

داده های نیمه ایستا

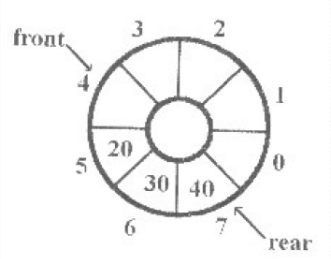
مشکل صفهای خطی

- مشکل:
  - ✓ حرکت تدریجی عناصر به سمت انتهای صف و عدم امکان استفاده از خانه های ابتدای صف که در اثر حذف خالی شده اند. مشکل صف های خطی این است.
- راه حل:
  1. شیفت خانه های انتهای صف به سمت ابتدای صف که این عمل در صورتی که تعداد خانه های صف زیاد باشد هزینه بالایی خواهد داشت.
  2. استفاده از صف حلقوی به این مفهوم که زمانی که به انتهای صف رسیدیم، عمل درج را دوباره از ابتدای صف انجام می دهیم و این حرکت چرخشی می باشد.
- روش دوم در عمل مقرون به صرفه تر بوده و استفاده می شود.

داده های نیمه ایستا

صف حلقوی

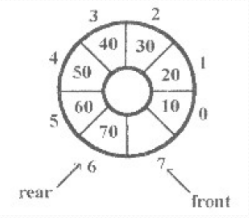
- آرایه n تایی Q[n] را می توان به صورت یک صف حلقوی در نظر گرفت به طوری که
  - در این صف زمانی که Rear = n-1 شد، عنصر بعدی در خانه 0 قرار می گیرد.
  - حرکت در خلاف جهت حرکت عقربه های ساعت است.
  - نیز هم Front به یک مکان قبل از عنصر اول اشاره دارد.
- مثال: درج عنصر ۵۰ در صف حلقوی مقابل؟



داده های نیمه ایستا

شرایط اولیه و مرزی صف حلقوی

- انتخاب آرایه Q[n] برای پیاده سازی صف حلقوی دارای شرایط اولیه و مرزی زیر است:
  - ✓ وضعیت اولیه صف: Front = Rear = -1
  - ✓ خالی بودن صف حلقوی: Front = Rear
  - ✓ پر بودن صف حلقوی: (Rear + 1) mod n = Front
- در هر صف حلقوی n عضوی همیشه یک خانه خالی وجود دارد و حداکثر از n-1 خانه صف می توانیم استفاده کنیم. این به آن علت است که بتوانیم وضعیت پر یا خالی بودن صف حلقوی را تشخیص بدهیم.



داده های نیمه ایستا

الگوریتم درج در صف حلقوی

- عنصر x را در صف Q[n] درج کنید.

```
k = (rear+1) % n
If (front == k)  write "Queue is full"
Else {
        rear=k
        Q[rear] = x
}
```

O(1)

داده های نیمه ایستا

الگوریتم حذف از صف حلقوی

- عنصر اول صف Q[n] را حذف کنید.

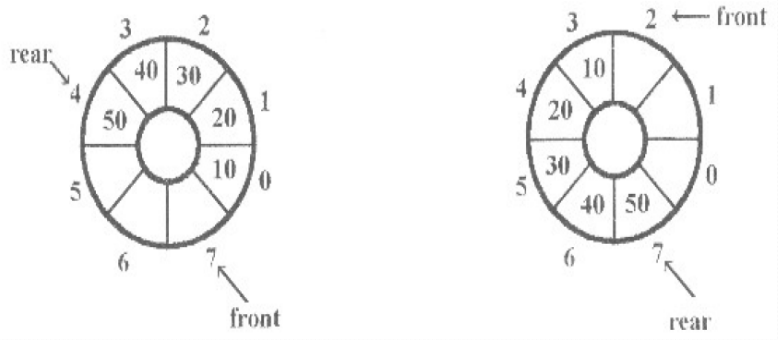
```
If (front==rear){
        write "Queue is empty"
}
Else {
        front = (front+1) % n
        y = Q[front]
}
```

O(1)

داده های نیمه ایستا

مثال

- درج و حذف از صف های حلقوی مقابل؟



داده های نیمه ایستا

ارزیابی

1- با استفاده از دستورات زیر در آخرین مرحله مقدار پشته(STACK) چیست؟

- STACK.PUSH('A');
- STACK.PUSH('B');
- STACK.POP( );
- STACK.PUSH('F');
- STACK.POP( );
- STACK.POP( );

(a) A ,B ,F
(b) F
(c) A
(d) پشته خالی می‌شود.

داده های نیمه ایستا

ارزیابی

2- در یک صف حلقوی که طول صف 7 است اگر front=3 و rear=6 باشد، محل عنصر جدید در کجا می‌باشد؟

(a 1
(b 4
(c 6
(d 7

داده های نیمه ایستا

ارزیابی

3- شرط پر بودن و خالی بودن یک صف حلقوی کدام گزینه است؟

(a rear=n و front=0
(b (Rear + 1) mod n = Front و front=0
(c rear=front و (Rear + 1) mod n = Front
(d rear=n و front=n

ارزیابی

4- الگوریتم زیر چه کاری انجام می دهد؟

```
If (top==m-1)  write "is full"
Else {
        top++
        A[top] = x
}
```

a) درج در صف حلقوی
b) درج در لیست پیوندی
c) درج در پشته
d) حذف از صف

ارزیابی

6- کدام گزینه خروجی تابع زیر را به ازائ مقادیر داده شده n=45 و m=147 نمایش می‌دهد؟

a) 3
b) 9
c) 1
d) 36

```
Int gcd(int m, int n)
{
  If (m%n==0)
     return n;
  Else
     return gcd(n,m%n);
}
```

داده های نیمه ایستا

ارزیابی

5- ماتریس خلوت مناسب آرایه دو بعدی زیر را بنویسید.

داده های نیمه ایستا

تمرین

1. **کدام یک از گزینه های زیر جزء خصوصیات الگوریتم نیست؟**
   a) خروجی داشتن
   b) قطعی بودن
   c) نامحدود بودن
   d) کارا بودن

2. **پیچیدگی زمانی کدام یک از توابع زیر بیشتر است؟**
   a) Log n
   b) n!
   c) 2^n
   d) n^n

3. **آرایه جزء کدام یک از انواع ساختمان داده هاست؟**
   a) ایستا - اولیه
   b) ایستا - غیر اولیه
   c) نیمه ایستا
   d) پویا – خطی

4. **کدام یک از عبارات زیر درست نیست؟**
   a) اگر عناصر زیر قطر اصلی ماتریس مربعی صفر باشد، ماتریس همانی است.
   b) اگر عناصر بالای قطر اصلی ماتریس مربعی صفر باشد، ماتریس پایین مثلثی است.
   c) اگر تعداد عناصر صفر ماتریس بیشتر از عناصر غیر صفر آن باشد. ماتریس خلوت است.
   d) ماتریسی که تعداد سطر و ستون برابری داشته باشد. ماتریس مربعی است.