بسم الله الرحمن الرحیم

طراحی و تحلیل الگوریتم‌ها

مبحث اول

مقدمه‌ای بر الگوریتم‌ها

An Introduction to Algorithms

کاظم فولادی

دانشکده مهندسی برق و کامپیوتر

دانشگاه تهران

http://courses.fouladi.ir/algorithm

۱

مقدمه

ALGORITHM AND COMPUTER SCIENCE

علم کامپیوتر، علم مطالعه‌ی الگوریتم‌هاست

برای افرادی که با علوم کامپیوتر سر و کار دارند، کلمه‌ی «الگوریتم» نامی آشناست. در واقع می‌توان در یک عبارت گفت که علم کامپیوتر، علم مطالعه‌ی الگوریتم‌هاست و بسیاری از مباحث مورد بررسی در علوم کامپیوتر، با الگوریتم‌ها مرتبط هستند.

الگوریتم یک علم ایرانی

کلمه‌ی الگوریتم، از نام دانشمند و ریاضیدان بزرگ ایرانی، خوارزمی، گرفته شده است.

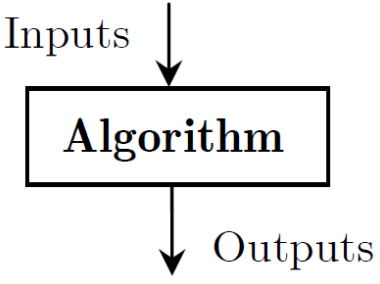
الخوارزمی - الخوریزم - الخوریسم - الگوریتم

کلمه‌ی الگوریتم، از نام دانشمند و ریاضیدان بزرگ ایرانی، «خوارزمی»، گرفته شده است. چرا که او نخستین کسی بود که مفهومی را به عنوان محاسبات قدم به قدم مطرح کرد و از این رو اکنون کمتر کتابی در زمینه‌ی الگوریتم‌ها وجود دارد که نام او را به طور صریح ذکر نکرده باشد.

تعریف ابتدایی الگوریتم

یک الگوریتم، دنباله‌ای از قدم‌های محاسباتی است که ورودی را به خروجی تبدیل می‌کند.

یک الگوریتم، رویه‌ی محاسباتی خوش‌تعریفی است که یک مقدار یا مجموعه‌ای از مقادیر را به عنوان ورودی دریافت می‌کند و یک مقدار یا مجموعه‌ای از مقادیر را به عنوان خروجی تولید می‌کند.



مثال

مسئله‌ی مرتب‌سازی

SORTING PROBLEM

مسئله‌ی مرتب‌سازی

ورودی: دنباله‌ای از n عدد $\langle a_1, a_2, \ldots, a_n \rangle$

خروجی: یک جایگشت از دنباله‌ی ورودی به صورت $\langle a'_1, a'_2, \ldots, a'_n \rangle$ به طوری که $a'_1 \leq a'_2 \leq \cdots \leq a'_n$.

برای مثال، دنباله‌ی ورودی $\langle 31,41,59,26,41,58 \rangle$ داده شده است. الگوریتم مرتب‌سازی، خروجی را به صورت دنباله‌ی

$\langle 26,31,41,41,58,59 \rangle$

بر می‌گرداند. یک چنین دنباله‌ی ورودی، نمونه‌ای از مسئله‌ی مرتب‌سازی خوانده می‌شود.

تعریف الگوریتم

الگوریتم را به عنوان روالی که مسئله را حل می‌کند تعریف می‌کنیم.

یک الگوریتم، مجموعه‌ی محدودی از دستورالعمل‌هاست که اگر دنبال و اجرا شود، هدف خاصی را برآورده می‌کند. به عبارتی، یک الگوریتم شامل موارد زیر است:	
ورودی	***Input***
• هیچ یا چند کمیت به عنوان ورودی الگوریتم	
خروجی	***Output***
• حداقل یک کمیت به عنوان خروجی	
قطعیت	***Definiteness***
• دستورات باید کاملاً واضح و فاقد ابهام باشند.	
متناهی بودن	***Finiteness***
• الگوریتم باید پس از طی مراحل محدودی خاتمه یابد.	
کارآیی	***Effectiveness***
• هر دستور باید انجام‌پذیر باشد.	

اصطلاحات پایه

مفهوم	تعریف
مسئله	پرسشی است که به دنبال پاسخ آن هستیم.
پارامترهای مسئله	متغیرهایی که در صورت مسئله مقدار مشخصی به آن نسبت داده نشده است.
نمونه‌ی مسئله	مسئله‌ای که با نسبت دادن یک مقدار به پارامتر آن مسئله بدست می‌آید.
حل مسئله	پاسخ به پرسشی که در مسئله خواسته شده است.
الگوریتم	روندی که مسئله را حل می‌کند.

زمینه‌های مورد مطالعه در حیطه‌ی الگوریتم‌ها

چگونگی ایجاد الگوریتم‌ها

- شامل روش‌های طراحی الگوریتم‌ها

بررسی اعتبار الگوریتم‌ها

- شامل اثبات درستی الگوریتم‌ها

تحلیل کارآیی الگوریتم‌ها

- از لحاظ فضای مورد نیاز، زمان اجرای مورد نیاز

۲

الگوریتم‌های ساخت‌یافته و ساختارهای کنترل برنامه

الگوریتم‌های ساخت‌یافته

STRUCTURED ALGORITHM

منظور از یک الگوریتم ساخت‌یافته، الگوریتمی است که در آن از ساختارهای کنترلی استاندارد استفاده شده است.

الگوریتم ساخت‌یافته، بر اساس موارد زیر تعریف می‌شود:	
۱	طراحی سلسله مراتبی ساختارهای الگوریتم با استفاده از شکل‌های کنترلی ساده مثل ترتیب، انتخاب و تکرار
۲	نمایش طراحی سلسله‌مراتبی الگوریتم با استفاده از ساختارهای کنترلی
۳	اجرای دستورها به صورت ترتیب فیزیکی آنها
۴	استفاده از گروه‌هایی از دستورها با یک هدف
۵	استفاده از متغیرهای محلی و عدم استفاده از متغیرهای سراسری

ترتیب، تصمیم، تکرار

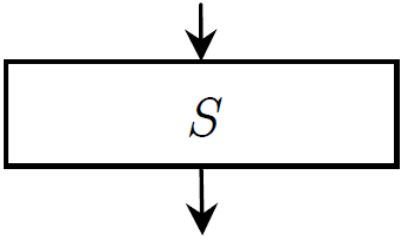
ساختارهای کنترلی	
ترتیب	***Order***
• دستور • دستور مرکب	
تصمیم	***Decision***
• دستور شرطی if-then • دستور شرطی if-then-else	
تکرار	***Repeat***
• دستور تکرار for • دستور تکرار while-do • دستور تکرار do-while • دستور تکرار repeat-until	

ثابت می‌شود که هر الگوریتم، با استفاده از سه ساختار ترتیب، تصمیم و تکرار قابل بیان می‌باشد.

دستور

STATEMENT

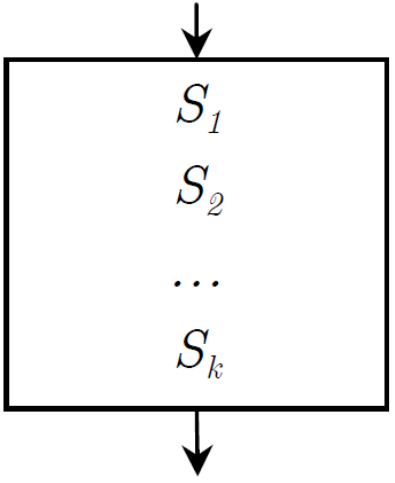
دستور را با S نمایش می‌دهیم. یک دستور، کوچکترین واحد الگوریتمی است.



دستور مرکب

COMPOUND STATEMENT

دنباله‌ای از دو یا چند دستور، دستور مرکب نام دارد.
به عبارت دیگر دنباله‌ی چند دستور نیز یک دستور است که دستور مرکب نام دارد.

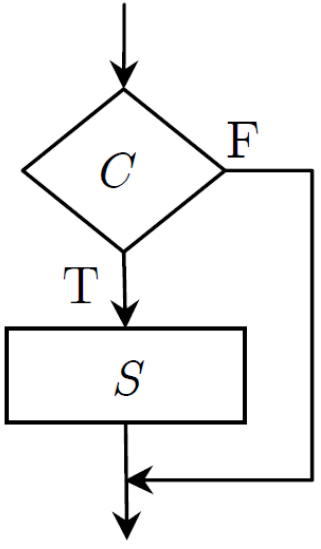


$$S_1; S_2; \ldots; S_k$$

دستور شرطی

IF-THEN STATEMENT

اگر C یک شرط و S یک دستور باشد، ساختار زیر نیز یک دستور است،
به طوری که اگر شرط درست باشد، دستور اجرا می‌شود:

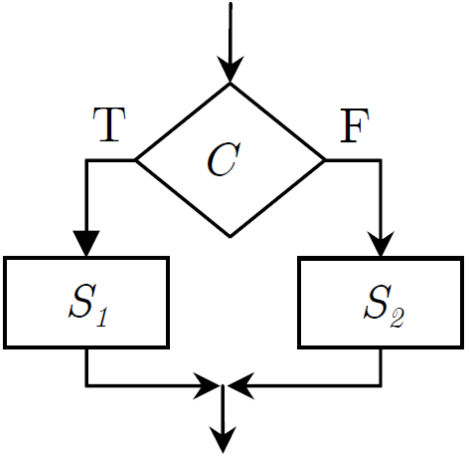


$$\textbf{if } C \textbf{ then } S$$

دستور شرطی

IF-THEN-ELSE STATEMENT

اگر C یک شرط و S_1 و S_2 دستور باشد، ساختار زیر نیز یک دستور است، به طوری که اگر شرط درست باشد، دستور S_1 و اگر شرط نادرست باشد، دستور S_2 اجرا می‌شود.

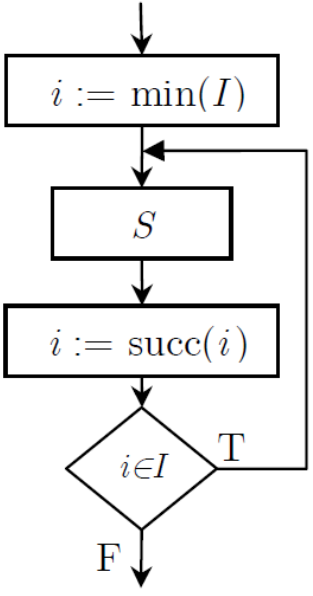


$$\textbf{if } C \textbf{ then } S_1 \textbf{ else } S_2$$

دستور تکرار ساده

FOR STATEMENT

اگر یک مجموعه‌ی اندیس‌گذار (مجموعه‌ی مرتب و گسسته) باشد،
ساختار زیر یک دستور است که دستور S را به ازای هر $i \in I$ اجرا می‌کند.

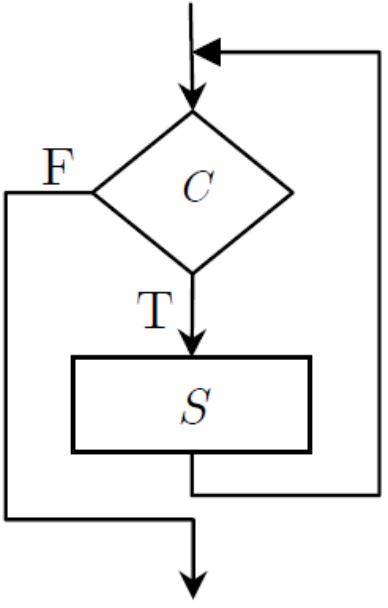


$$\textbf{for } i \in I \textbf{ do } S$$

دستور تکرار شرطی

WHILE-DO STATEMENT

اگر C یک شرط و S یک دستور باشد، ساختار زیر نیز یک دستور است،
به طوری که تا وقتی که شرط درست باشد، دستور اجرا می‌شود.
(ابتدا شرط بررسی می‌شود)

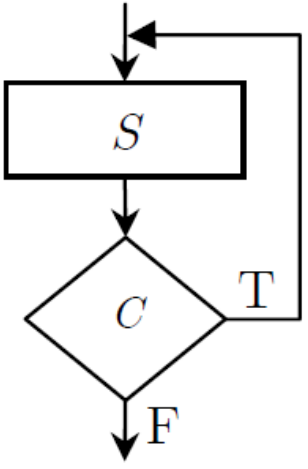


while C **do** S

دستور تکرار شرطی

DO-WHILE STATEMENT

اگر C یک شرط و S یک دستور باشد، ساختار زیر نیز یک دستور است،
به طوری که تا وقتی که شرط درست باشد، دستور اجرا می‌شود.
(انتها شرط بررسی می‌شود)

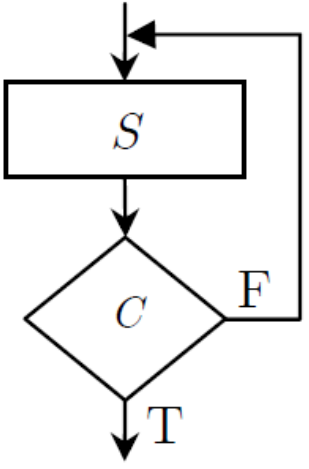


$$\textbf{do } S \textbf{ while } C$$

دستور تکرار شرطی

REPEAT-UNTIL STATEMENT

اگر C یک شرط و S یک دستور باشد، ساختار زیر نیز یک دستور است،
به طوری که تا وقتی که شرط درست باشد، دستور اجرا می‌شود
(انتها شرط بررسی می‌شود)



$$\textbf{repeat } S \textbf{ until } C$$

$$\textbf{repeat } S \textbf{ until } C \quad \equiv \quad \textbf{do } S \textbf{ while } \neg C$$

چرا الگوریتم‌های ساخت‌یافته؟

علت تاکید بر الگوریتم‌های ساخت‌یافته این است که تحلیل آنها آسان‌تر است،
در حالی که تحلیل یک الگوریتم غیرساخت‌یافته، بسیار دشوار است و عملاً امکان‌پذیر نیست.

با توجه به تعریف الگوریتم ساخت‌یافته، نتایج زیر حاصل می‌شود:

- استفاده از دستورات پرش، مانند **goto**، مجاز نیست.
- حتی الامکان نباید در حلقه‌ها از دستوراتی مانند **break**، **continue** و **exit** استفاده کرد.

۳

نمایش الگوریتم‌ها

نمایش الگوریتم‌ها

۱

شبه کد

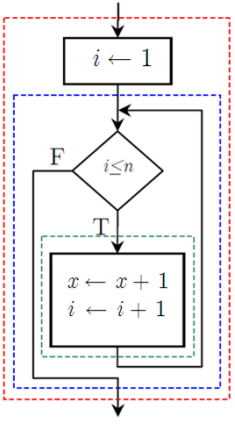
Pseudo-code

```
TEST(n)
  i ← 1
  while i ≤ n do
      x ← x + 1
      i ← i + 1
```

۲

فلوچارت

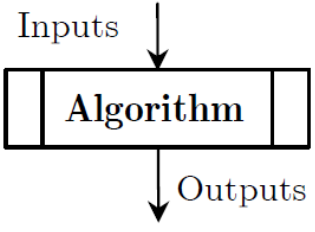
Flowchart



۳

نمودار بلوکی

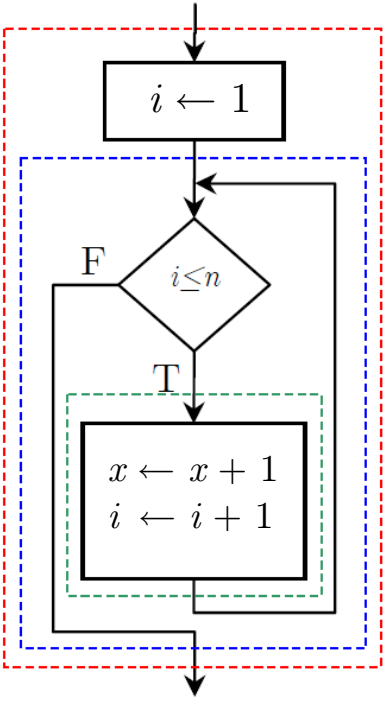
Block diagram



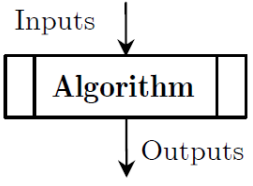
نمایش الگوریتم‌ها

مثال

فلوچارت
Flowchart



نمودار بلوکی
Block diagram




```
TEST(n)
    i ← 1
    while i ≤ n do
        x ← x + 1
        i ← i + 1
```

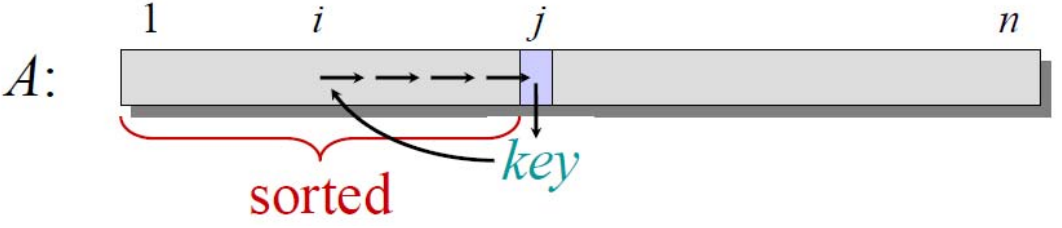
شبه کد
Pseudo-code

مثال: مرتب‌سازی درجی

Insertion sort

"pseudocode"

```
INSERTION-SORT (A, n)      ▷ A[1 . . n]
  for j ← 2 to n
      do key ← A[j]
         i ← j – 1
         while i > 0 and A[i] > key
             do A[i+1] ← A[i]
                i ← i – 1
         A[i+1] = key
```



مثال: مرتب‌سازی درجی

نمونه‌ی اجرا

Example of insertion sort