طراحی و تحلیل الگوریتم‌ها

مبحث سوم

روش‌های طراحی الگوریتم

روش استقرا

Methods of Algorithm Design: Induction

کاظم فولادی قلعه

دانشکده مهندسی، دانشکدگان فارابی

دانشگاه تهران

http://courses.fouladi.ir/algorithm

برای اثبات یک قضیه با روش استقرا:

اگر $P(n)$ یک گزاره روی اعداد صحیح n باشد:

- **(پایه‌ی استقرا)**
نشان می‌دهیم $P(n_0)$ درست است.
- **(فرض استقرا)**
فرض می‌کنیم $P(k)$ درست باشد.
- **(گام استقرا)**
با فرض درست بودن $P(k)$، نشان می‌دهیم که $P(k+1)$ درست است.

در این صورت، برای هر n، گزاره‌ی $P(n)$ درست است.

استقرای ریاضی

مثال

با استفاده از استقرا ثابت کنید که

$$P(n): \quad 1 + 3 + 5 + \cdots + (2n - 1) = n^2$$

- **(پایه‌ی استقرا)**
- **(فرض استقرا)**
- **(گام استقرا)**

اصل طراحی الگوریتم با استقرا

برای حل یک مسئله با روش استقرا:

- نشان می‌دهیم نمونه‌ی کوچکی از مسئله، قابل حل است **(حالت پایه)**.
- فرض می‌کنیم نمونه‌های کوچکتر مسئله حل شده باشد **(فرض استقرا)**.
- نشان می‌دهیم راه حل نمونه‌های بزرگ‌تر مسئله از روی راه حل نمونه‌های کوچکتر مسئله به دست می‌آید **(گام استقرا)**.

مثال: مسئله‌ی محاسبه‌ی مقدار یک چندجمله‌ای

طراحی الگوریتم با استقرا

- **روش اول**

هدف: محاسبه‌ی مقدار چندجمله‌ای درجه n زیر در نقطه‌ی x:

$$P_n(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_0$$

راه حل:

- پایه‌ی استقرا

$$P_0(x) = a_0$$

- فرض استقرا

$$P_{k-1}(x)$$

- گام استقرا

$$P_k(x) = P_{k-1}(x) + a_k x^k$$

مثال: مسئله‌ی محاسبه‌ی مقدار یک چندجمله‌ای

طراحی الگوریتم با استقرا

- **روش اول: شبه کد**

هدف: محاسبه‌ی مقدار چندجمله‌ای درجه n زیر در نقطه‌ی x:

$$P_n(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_0$$

```
POLYNOMIAL-VALUE(a[0..n], x)
    P ← a[0]
    for i ← 1 to n do
        P ← P + a[i] * POWER(x, i)
```

زمان اجرا: بر حسب تعداد اجرای عمل اصلی (ضرب): $\Theta(n^2)$

مثال: مسئله‌ی محاسبه‌ی مقدار یک چندجمله‌ای

طراحی الگوریتم با استقرا

- **روش دوم**

هدف: محاسبه‌ی مقدار چندجمله‌ای درجه n زیر در نقطه‌ی x:

$$P_n(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_0$$

راه حل:

- پایه‌ی استقرا

$$P_0(x) = a_0, \qquad S_0(x) = 1$$

- فرض استقرا

$$P_{k-1}(x), \qquad S_{k-1}(x)$$

- گام استقرا

$$S_k(x) = S_{k-1}(x) * x, \qquad P_k(x) = P_{k-1}(x) + a_k * S_k(x)$$

مثال: مسئله‌ی محاسبه‌ی مقدار یک چندجمله‌ای

طراحی الگوریتم با استقرا

- **روش دوم: شبه کد**

هدف: محاسبه‌ی مقدار چندجمله‌ای درجه n زیر در نقطه‌ی x:

$$P_n(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_0$$

```
POLYNOMIAL-VALUE(a[0..n], x)
    P ← a[0]
    S ← 1
    for i ← 1 to n do
        S ← S * x
        P ← P + a[i] * S
```

زمان اجرا: بر حسب تعداد اجرای عمل اصلی (ضرب): $\Theta(2n)$

مثال: مسئله‌ی محاسبه‌ی مقدار یک چندجمله‌ای

طراحی الگوریتم با استقرا

- **روش سوم**

هدف: محاسبه‌ی مقدار چندجمله‌ای درجه n زیر در نقطه‌ی x:

$$P_n(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_0$$

راه حل:

- پایه‌ی استقرا

$$P_0(x) = a_n$$

- فرض استقرا

$$P_{k-1}(x)$$

- گام استقرا

$$P_k(x) = P_{k-1}(x) * x + a_{n-k}$$

قاعده‌ی هورنر

مثال: مسئله‌ی محاسبه‌ی مقدار یک چندجمله‌ای

طراحی الگوریتم با استقرا

- **روش سوم: شبه کد**

هدف: محاسبه‌ی مقدار چندجمله‌ای درجه n زیر در نقطه‌ی x:

$$P_n(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_0$$

```
POLYNOMIAL-VALUE(a[0..n], x)
    P ← a[n]
    for i ← 1 to n do
        P ← P * x + a[n - i]
```

قاعده‌ی هورنر

زمان اجرا: بر حسب تعداد اجرای عمل اصلی (ضرب): $\Theta(n)$

مثال: مسئله‌ی یافتن بزرگ‌ترین مقدار یک آرایه

طراحی الگوریتم با استقرا

هدف: یافتن بزرگ‌ترین مقدار آرایه‌ی $A[1..n]$

- *پایه‌ی استقرا* $(n = 1)$

  بزرگ‌ترین عنصر آرایه‌ای به طول یک، همان تک عنصر است.
- *فرض استقرا* $(n = k - 1)$

  روش یافتن بزرگ‌ترین عنصر را برای آرایه‌ی $A[1..k-1]$ می‌دانیم.
- *گام استقرا*

  روش یافتن بزرگ‌ترین عنصر $A[1..k]$ بر اساس نتیجه‌ی بزرگ‌ترین عنصر $A[1..k-1]$:
  - $A[k]$ را با بزرگ‌ترین عنصر $A[1..k-1]$ مقایسه می‌کنیم.

    هر کدام بزرگ‌تر بود، پاسخ مسئله است.

مثال: مسئله‌ی یافتن بزرگ‌ترین مقدار یک آرایه

طراحی الگوریتم با استقرا

هدف: یافتن بزرگ‌ترین مقدار آرایه‌ی $A[1..n]$

```
MAXIMUM(A[1..n])
    if n = 1 then
        return A[1]
    else
        if n > 1 then
            X ← MAXIMUM(A[1..n - 1])
            return MAX(X, A[n])
```

$$\text{MAX}(a,b) = \begin{cases} b & ,a \leq b \\ a & ,a > b \end{cases}$$

زمان اجرا: بر حسب تعداد اجرای عمل اصلی (مقایسه): $\Theta(n)$

مثال: مسئله‌ی یافتن همزمان بزرگ‌ترین و کوچک‌ترین مقدار یک آرایه

طراحی الگوریتم با استقرا

هدف: یافتن همزمان بزرگ‌ترین و کوچک‌ترین مقدار آرایه‌ی $A[1..n]$

- *پایه‌ی استقرا* $(n = 1, n = 2)$

  بزرگ‌ترین و کوچک‌ترین عنصر آرایه‌ای به طول یک، همان تک عنصر است.

  بزرگ‌ترین و کوچک‌ترین عنصر آرایه‌ای به طول دو با یک مقایسه مشخص می‌شود.

- *فرض استقرا* $(n = k - 2)$

روش یافتن همزمان بزرگ‌ترین و کوچک‌ترین عنصر را برای آرایه‌ی $A[1..k-2]$ می‌دانیم.

- *گام استقرا*

  روش یافتن همزمان بزرگ‌ترین و کوچک‌ترین عنصر $A[1..k]$ بر اساس نتیجه‌ی یافتن همزمان بزرگ‌ترین و کوچک‌ترین عنصر $A[1..k-2]$:

  - نتیجه‌ی مربوط به بزرگ‌ترین و کوچک‌ترین عناصر $A[k-1..k]$ را با نتیجه‌ی مربوط به $A[1..k-2]$ مقایسه می‌کنیم تا پاسخ مشخص شود.

مثال: مسئله‌ی یافتن همزمان بزرگ‌ترین و کوچک‌ترین مقدار یک آرایه

طراحی الگوریتم با استقرا

هدف: یافتن همزمان بزرگ‌ترین و کوچک‌ترین مقدار آرایه‌ی $A[1..n]$

```
MAXIMUM-MINIMUM(A[1..n])
    if n = 1 then
        return A[1]
    else
        if n = 2 then
            if A[1] > A[2] then
                max ← A[1], min = A[2]
            else
                max ← A[2], min = A[1]
            return (max, min)
        if n > 2 then
            (max', min') ← MAXIMUM-MINIMUM(A[1..n -2])
            (max",min") ← MAXIMUM-MINIMUM(A[n -1..n])
            max ← MAX(max', max")
            min ← MIN(min', min")
            return (max, min)
```

زمان اجرا: بر حسب تعداد اجرای عمل اصلی (مقایسه): $\Theta(1.5n)$

مثال: مسئله‌ی مرتب‌سازی درجی

طراحی الگوریتم با استقرا

INSERTION SORT

هدف: مرتب‌سازی آرایه‌ی $A[1..n]$ به صورت صعودی

- پایه‌ی استقرا $(n = 1)$
  آرایه‌ای به طول یک مرتب است.
- فرض استقرا $(n = k - 1)$
  روش مرتب‌سازی برای آرایه‌ی $A[1..k-1]$ را می‌دانیم.
- گام استقرا
  روش مرتب‌سازی $A[1..k]$ بر اساس نتیجه‌ی مرتب شده‌ی $A[1..k-1]$:
  - $A[k]$ را در محل مناسب از $A[1..k-1]$ مرتب شده درج می‌کنیم.

مثال: مسئله‌ی مرتب‌سازی درجی

طراحی الگوریتم با استقرا

INSERTION SORT

هدف: مرتب‌سازی آرایه‌ی $A[1..n]$ به صورت صعودی

```
INSERTION-SORT(A[1..n])
    if n = 1 then
        return A[1]
    else
        if n > 1 then
            X ← INSERTION-SORT(A[1..n - 1])
            return INSERT(X, A[n])
```

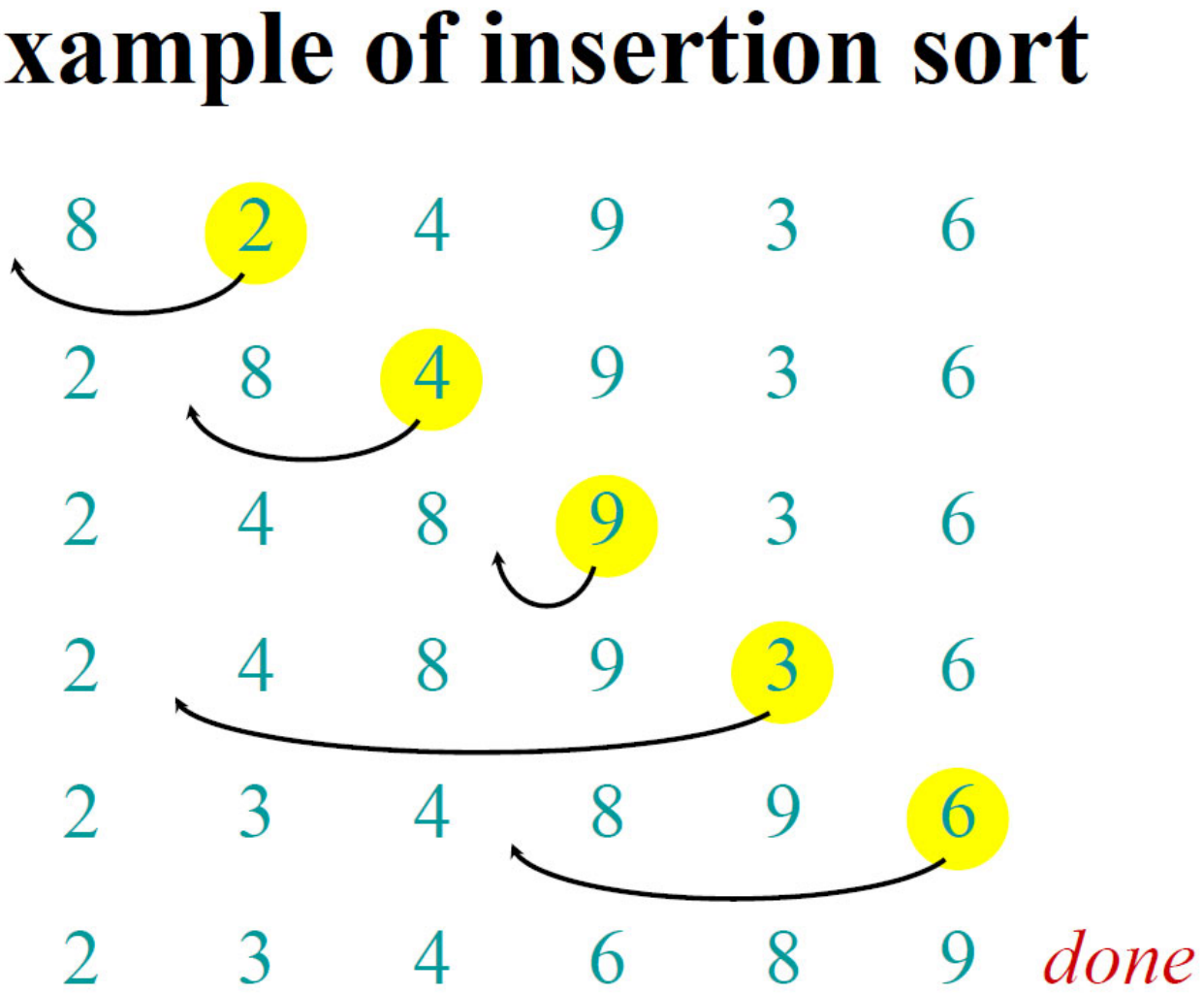
محل عنصر جدید در X را با جستجو پیدا می‌کند و با شیفت دادن عناصر پس از آن به سمت جلو، جا را برای آن عنصر جدید باز می‌کند.

زمان اجرا: بر حسب تعداد اجرای عمل اصلی (مقایسه): $\Theta(n(n-1)/2)$

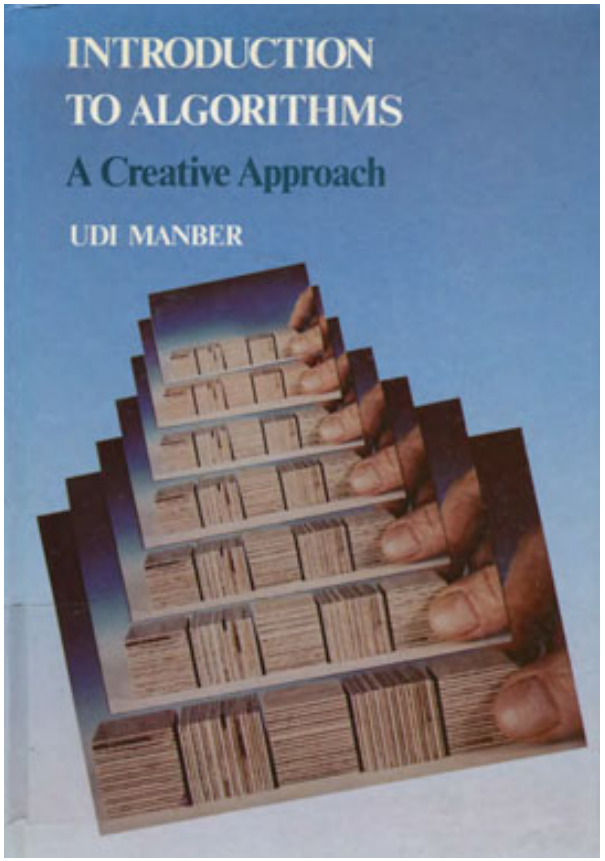
مثال: مسئله‌ی مرتب‌سازی درجی

طراحی الگوریتم با استقرا

INSERTION SORT



مرجع



U. Manber,
Introduction to Algorithms: A Creative Approach,
Addison-Wesley, 1989.

Chapter 5

CHAPTER 5

DESIGN OF ALGORITHMS BY INDUCTION

Nothing is more important than to see the sources of invention, which are, in my opinion, more interesting than the inventions themselves.

G. W. Leibniz (1646–1716)

Invention breeds invention.

R. W. Emerson (1803–1882)

5.1 Introduction

In this chapter, we introduce our approach to algorithm design using the analogy to mathematical induction. We include relatively simple examples, and present the basic principles and techniques on which the method is based. The analogous induction techniques have been described in Chapter 2. When appropriate, we repeat the discussion here to make this chapter self contained.

Mathematical induction is based on a **domino principle**. Imagine that we have a line of upended dominoes, and that we wish to knock down all of them by knocking down only the first. To make sure that all dominoes will fall down, we need only to verify that we have pushed the first one and that each domino will topple the next one as it falls. We need not collapse the whole arrangement every time we add a new domino to verify that the new arrangement will work. The same principle can be applied to algorithm design.