طراحی و تحلیل الگوریتم ها

مدرس: سعدون عزیزی

s.azizi@uok.ac.ir

گروه مهندسی کامپیوتر

نیم سال دوم ۹۷-۹۶

الگوریتم چیست؟

تعریف الگوریتم

یک الگوریتم مجموعه‌ای محدود از دستورالعمل‌های دقیق است که برای انجام محاسبات یا حل یک مسئله طراحی می‌شود.

- ویژگی‌های مهم یک الگوریتم
  - ورودی
  - خروجی
  - صحیح
  - محدود
  - بدون ابهام

اهداف

- اهداف این درس عبارتند از:
  - چگونه الگوریتم‌ها را ابداع کنیم؟
  - چگونه الگوریتم‌ها را بیان کنیم؟
  - چگونه الگوریتم‌ها را اعتبارسنجی کنیم؟
  - چگونه الگوریتم‌ها را تحلیل کنیم؟
  - چگونه الگوریتم‌ها را تست کنیم؟

اهداف

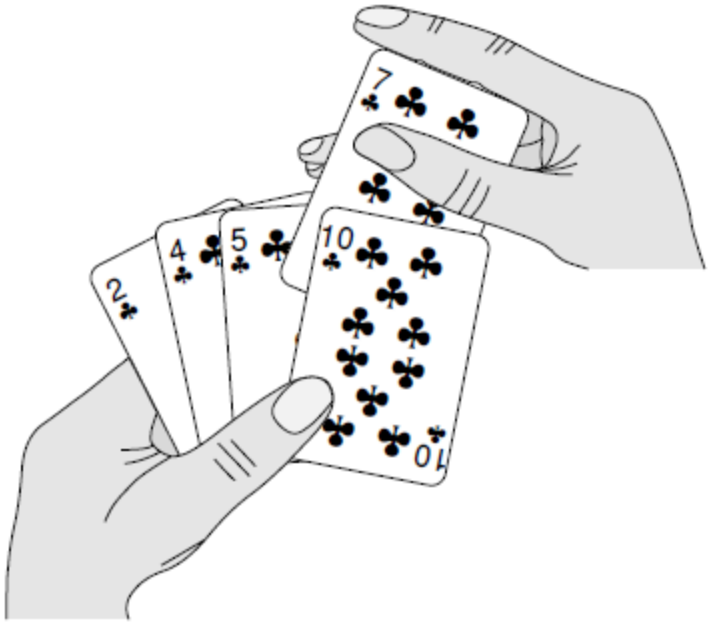
- پنج تکنیکی که در این درس بحث می‌شوند:
  - تقسیم و حل
  - برنامه‌ریزی پویا
  - الگوریتم‌های حریصانه
  - الگوریتم‌های عقبگرد
  - الگوریتم‌های شاخه و حد

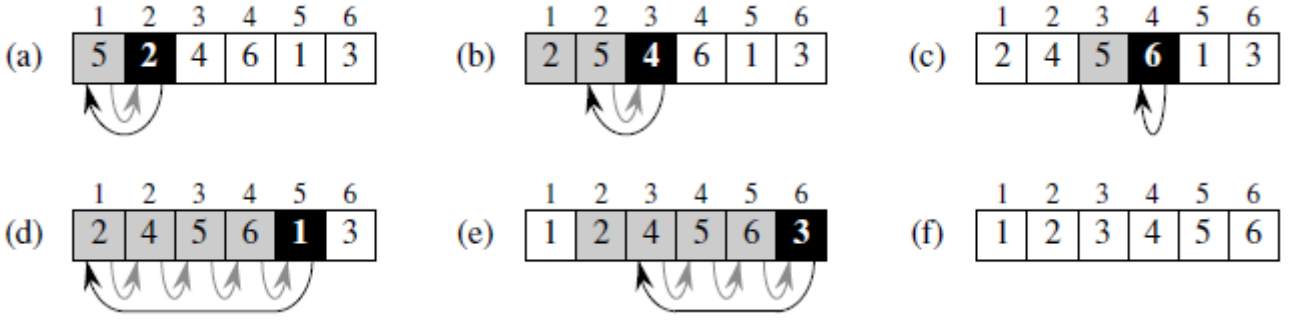
مرتب‌سازی درجی

- **ورودی:** دنباله‌ای از n عدد $(a_1, a_2, \ldots, a_n)$
- **خروجی:** مرتب‌سازی دنباله ورودی به صورت صعودی
- **مثال:**

A=(5,2,4,6,1,3)

ایده




(a)
1 2 3 4 5 6
5 2 4 6 1 3
(b)
1 2 3 4 5 6
2 5 4 6 1 3
(c)
1 2 3 4 5 6
2 4 5 6 1 3
(d)
1 2 3 4 5 6
2 4 5 6 1 3
(e)
1 2 3 4 5 6
1 2 4 5 6 3
(f)
1 2 3 4 5 6
1 2 3 4 5 6

بیان

```
INSERTION-SORT(A)
1   for j ← 2 to length[A]
2       do key ← A[j]
3          ▷ Insert A[j] into the sorted sequence A[1 . . j − 1].
4          i ← j − 1
5          while i > 0 and A[i] > key
6              do A[i + 1] ← A[i]
7                 i ← i − 1
8          A[i + 1] ← key
```

درستی

استفاده از تکنیک loop invariants برای اثبات درستی الگوریتم

- **شروع:** j=2 پس داریم A[1..1]=A[1]؛ که مرتب شده است.
- **ادامه:** A[j] در مکان صحیح وارد می‌شود، پس A[1..j] مرتب شده است.
- **پایان:** وقتی اتفاق می‌افتد که j=n+1 باشد. در این حالت A[1..j-1]=A[1..n] که یک آرایه مرتب شده است.

تحلیل

- محاسبه مقدار منابعی که اجرای الگوریتم به آنها نیاز دارد؛ منابعی مانند زمان، حافظه، پهنای باند و غیره

INSERTION-SORT(A)	*cost*	*times*
1 **for** $j \leftarrow 2$ **to** *length*$[A]$	c_1	n
2 **do** $key \leftarrow A[j]$	c_2	$n-1$
3 ▷ Insert $A[j]$ into the sorted sequence $A[1..j-1]$.	0	$n-1$
4 $i \leftarrow j-1$	c_4	$n-1$
5 **while** $i > 0$ and $A[i] > key$	c_5	$\sum_{j=2}^{n} t_j$
6 **do** $A[i+1] \leftarrow A[i]$	c_6	$\sum_{j=2}^{n}(t_j-1)$
7 $i \leftarrow i-1$	c_7	$\sum_{j=2}^{n}(t_j-1)$
8 $A[i+1] \leftarrow key$	c_8	$n-1$

- جایی که t_j بیانگر تعداد دفعاتی است که حلقه while در خط ۵ با توجه به مقدار j اجرا می‌شود.

پس:

$$T(n) = c_1 n + c_2(n-1) + c_4(n-1) + c_5 \sum_{j=2}^{n} t_j + c_6 \sum_{j=2}^{n}(t_j - 1) + c_7 \sum_{j=2}^{n}(t_j - 1) + c_8(n-1) .$$

تحلیل (ادامه)

- **بهترین حالت:** زمانی که آرایه از اول به صورت مرتب شده باشد؛ پس $t_j = 1$ و داریم:

$$\begin{aligned} T(n) &= c_1 n + c_2(n-1) + c_4(n-1) + c_5(n-1) + c_8(n-1) \\ &= (c_1 + c_2 + c_4 + c_5 + c_8)n - (c_2 + c_4 + c_5 + c_8) . \end{aligned}$$

- **بدترین حالت:** زمانی که آرایه به صورت برعکس مرتب شده باشد- یعنی به صورت نزولی؛ پس $t_j = j$ و داریم:

$$\begin{aligned} T(n) &= c_1 n + c_2(n-1) + c_4(n-1) + c_5\left(\frac{n(n+1)}{2} - 1\right) \\ &\quad + c_6\left(\frac{n(n-1)}{2}\right) + c_7\left(\frac{n(n-1)}{2}\right) + c_8(n-1) \\ &= \left(\frac{c_5}{2} + \frac{c_6}{2} + \frac{c_7}{2}\right)n^2 + \left(c_1 + c_2 + c_4 + \frac{c_5}{2} - \frac{c_6}{2} - \frac{c_7}{2} + c_8\right)n \\ &\quad - (c_2 + c_4 + c_5 + c_8) . \end{aligned}$$

تست

- برای تست الگوریتم، کافی است که شبه‌کد را با یکی از زبان‌های برنامه‌نویسی پیاده‌سازی کنید و سپس با استفاده از نمونه‌های مختلف آن را اجرا کنید.