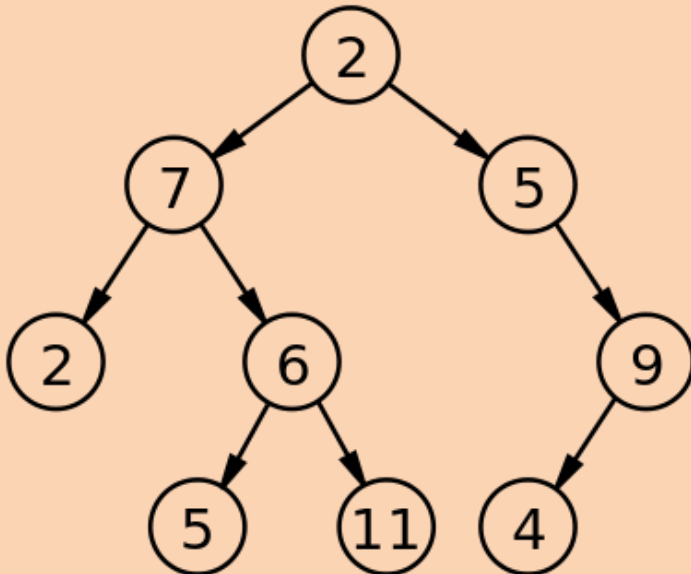




ساختارهای داده

مرتب‌سازهای جا‌به‌جا، درج‌ر و انتخابی

*Bubble Sort, Insertion Sort, and
Selection Sort*



مدرس:

سید کمال الدین غیاثی شیرازی

مساله مرتب سازی

پایداری الگوریتم‌های مرتب سازی

- یک الگوریتم مرتب سازی پایدار (stable) است هرگاه ترتیب عناصر مساوی را به هم نزند.

sorted by time	sorted by location (not stable)	sorted by location (stable)
Chicago 09:00:00	Chicago 09:25:52	Chicago 09:00:00
Phoenix 09:00:03	Chicago 09:03:13	Chicago 09:00:59
Houston 09:00:13	Chicago 09:21:05	Chicago 09:03:13
Chicago 09:00:59	Chicago 09:19:46	Chicago 09:19:32
Houston 09:01:10	Chicago 09:19:32	Chicago 09:19:46
Chicago 09:03:13	Chicago 09:00:00	Chicago 09:21:05
Seattle 09:10:11	Chicago 09:35:21	Chicago 09:25:52
Seattle 09:10:25	Chicago 09:00:59	Chicago 09:35:21
Phoenix 09:14:25	Houston 09:01:10	Houston 09:00:13
Chicago 09:19:32	Houston 09:00:13	Houston 09:01:10
Chicago 09:19:46	Phoenix 09:37:44	Phoenix 09:00:03
Chicago 09:21:05	Phoenix 09:00:03	Phoenix 09:14:25
Seattle 09:22:43	Phoenix 09:14:25	Phoenix 09:37:44
Seattle 09:22:54	Seattle 09:10:25	Seattle 09:10:11
Chicago 09:25:52	Seattle 09:36:14	Seattle 09:10:25
Chicago 09:35:21	Seattle 09:22:43	Seattle 09:22:43
Seattle 09:36:14	Seattle 09:10:11	Seattle 09:22:54
Phoenix 09:37:44	Seattle 09:22:54	Seattle 09:36:14

no longer sorted by time

still sorted by time

Stability when sorting on a second key

مرتب‌سازی حبابی (Bubble Sort)

- الگوریتم مرتب‌سازی حبابی دائماً دو عنصر مجاور را با یکدیگر مقایسه می‌کند و اگر ترتیب آنها صحیح نیست، جای آنها را عوض می‌کند.

مرتب‌سازی حبابی (Bubble Sort)

```
n = length(A)
for i = n-1 down to 1 do
  for j = 1 to i do
    if A[j-1] < A[j] then
      swap(A[j-1], A[j])
```

نسخه‌ی بهینه‌شده مرتب‌سازی حبابی

```
n = length(A)
repeat
  newn = 0
  for i = 1 to n-1 do
    if A[i-1] < A[i] then
      swap(A[i-1], A[i])
      newn = i
  n = newn
until n = 0
```

تحليل الكوريتم Bubble sort

- زمان اجرا در بدترین حالت از مرتبه $\Theta(n^2)$
- پایدار؟
- تعداد دفعات جابجایی عناصر از مرتبه $\Theta(n^2)$ است.

تحليل الكوريتم Bubble sort

- زمان اجرا در بدترین حالت از مرتبه $\Theta(n^2)$
- پایدار است.
- تعداد دفعات جابجایی عناصر از مرتبه $\Theta(n^2)$ است.
- بنابراین اگر اندازه عناصر لیست بر حسب بایت بالا باشد، بسیار کند عمل خواهد کرد.

مرتب‌سازی درجی (Insertion Sort)

- الگوریتم Insertion sort همانند اثبات استقرایی عمل می‌کند. $P(1) \rightarrow P(k) \rightarrow P(k+1)$
- آرایه ای به طول یک مرتب است.

- فرض کنیم آرایه به طول $1 - z$ مرتب باشد، در این صورت عنصر z ام را در محل صحیح در آرایه $1 - z$ تایی درج می‌کنیم تا یک آرایه z تایی مرتب بوجود آید.



1 1 5

الگوریتم مرتب سازی درجی

```
n = length(A)
for i=1 to n-1 do
```

```
  v = A[i]
```

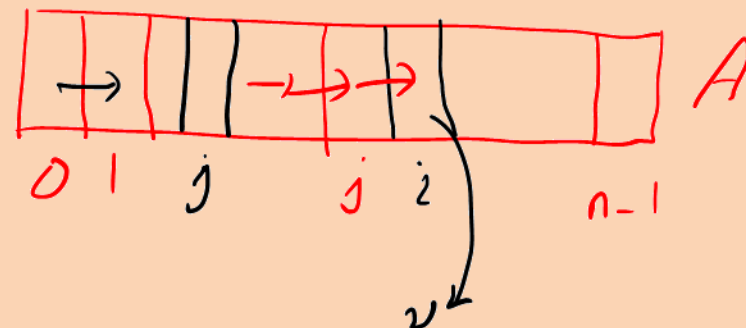
```
  j = i-1
```

```
  while j >= 0 and v < A[j] do
```

```
    A[j+1] = A[j]
```

```
    j = j - 1
```

```
  A[j+1] = v
```



در این مثال $A[i]$ به $A[0 \dots i-1]$ منتقل می شود و به جای $A[i]$ قرار می گیرد و به ترتیب $A[0 \dots i]$ به درستی می آید.

$$\sum_{i=1}^{n-1} i = 1 + 2 + \dots + n-1 = \frac{(n-1)n}{2} \in \Theta(n^2)$$

$n = \text{len}(A)$

for $i=1$ to $n-1$ do

$j = i-1$

while $j \geq 0$ and $A[j+1] < A[j]$ do

$A[j+1] \leftrightarrow A[j]$

$j = j - 1$

$A[j+1] = v$

$n = \text{length}(A)$
for $i=1$ to $n-1$ do

$v = A[i]$

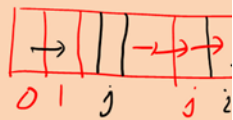
$j = i-1$

while $j \geq 0$ and $v < A[j]$ do

$A[j+1] = A[j]$

$j = j - 1$

$A[j+1] = v$



7 4 1 5 3 2 3

1 4 7

1 4 5 7

1 3 4 5 7



تحليل الكوريتم Insertion sort



- زمان اجرا در بدترین حالت از مرتبه $\Theta(n^2)$

- پایدار؟

- تعداد دفعات جابجایی عناصر از مرتبه $\Theta(n^2)$ است.

$n, n-1, n-2, \dots, 1$

تحليل الكوريتم Insertion sort

- زمان اجرا در بدترین حالت از مرتبه $\Theta(n^2)$
- پایدار است.
- تعداد دفعات جابجایی عناصر از مرتبه $\Theta(n^2)$ است.
- بنابراین اگر اندازه عناصر لیست بر حسب بایت بالا باشد، بسیار کند عمل خواهد کرد.

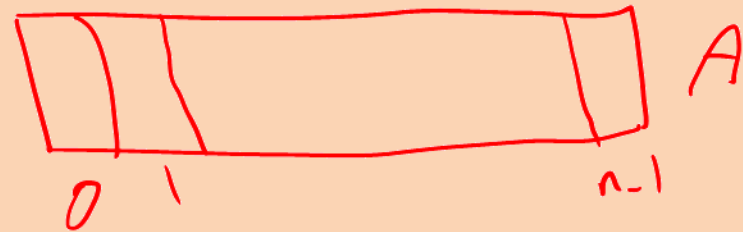
مرتب‌سازی انتخابی (Selection Sort)

- در Selection Sort محل بزرگ ترین عنصر در لیست عناصر مشخص می شود و جای آن با عنصر آخر لیست عوض می شود.
- با تکرار همین کار، لیست مرتب می شود.

9, 1, 12, 13, 7, 4, 15, 6
 9, 1, 12, 13, 7, 4, 6, 15
 9, 1, 12, 6, 7, 4, 13, 15

الگوریتم مرتب‌سازی انتخابی

```
n = length(A)
for i=1 to n-1 do
    maxVal = A[0]
    maxIdx = 0
    for j=1 to n-i do
        ✓ if maxVal < A[j]
            maxVal = A[j]
            maxIdx = j
    swap (A[n-i], A[maxIdx])
```



$$\sum_{i=1}^{n-1} (n-i) = n-1 + \dots + 1$$
$$= \frac{(n-1)n}{2} \in \Theta(n^2)$$

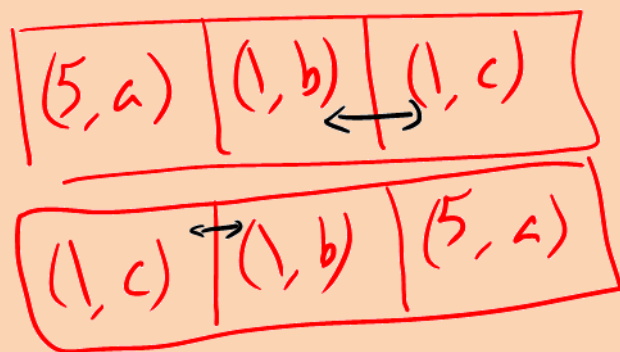
ستقل از ترتیب داده‌ها
جابجایی = $n-1$

تحلیل الگوریتم مرتب‌سازی انتخابی

• زمان اجرا در بدترین حالت از مرتبه $\Theta(n^2)$

• پایدار؟ \times

• تعداد دفعات جابجایی عناصر از مرتبه $\Theta(n)$ است.



تحلیل الگوریتم مرتب سازی انتخابی

- زمان اجرا در بدترین حالت از مرتبه $\Theta(n^2)$

- پایدار نیست. مثال زیر ناپایداری را نشان می دهد.

$(5, a), (1, b), (1, c)$

- تعداد دفعات جابجایی عناصر از مرتبه $\Theta(n)$ است. بنابراین اگر اندازه عناصر لیست بر

حسب بایت بالا باشد، این الگوریتم نسبت به دو الگوریتم قبل رجحان دارد.