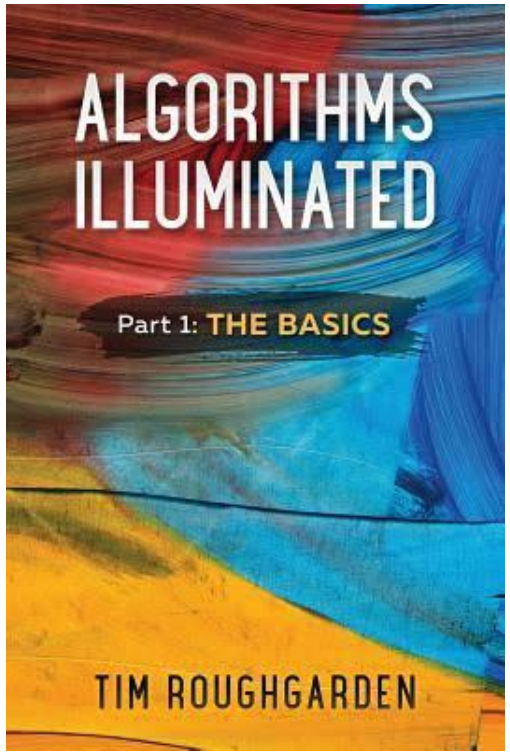
طراحی الگوریتم

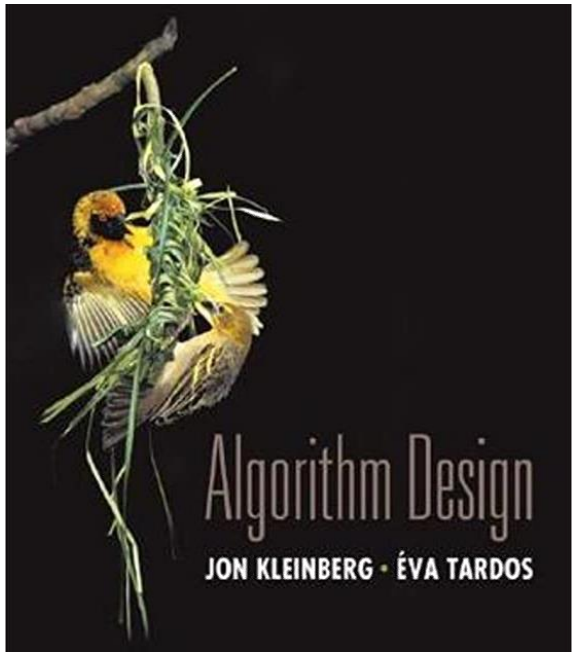
(تقسیم و غلبه–شمارش تعداد وارونگی‌ها)

دانشکده مهندسی برق و کامپیوتر، دانشگاه صنعتی اصفهان

بهار ۱۴۰۰

تقسیم و غلبه (یادآوری)





فصل سوم، صفحه ۶۱

تقسیم مساله به زیرمساله‌های کوچکتر

غلبه و حل زیرمساله‌ها

ادغام جواب زیرمساله‌ها

برای به دست آوردن جواب مساله اصلی

پیدا کردن تعداد وارونگی‌ها در یک دنباله از اعداد طبیعی

ورودی: یک دنباله از اعداد متمایز

هدف: تعداد وارونگی‌های دنباله ورودی

پیدا کردن تعداد وارونگی‌ها در یک دنباله از اعداد طبیعی

سوال: یک دنباله به طول n حداکثر چه تعداد وارونگی می‌تواند داشته باشد؟

کاربرد مساله (فیلترینگ مشارکتی)

Interstellar	Inception	Shawshank redemption	Shutter Island	Lord of rings	Prestige

حل مساله با روش جستجوی کامل (Brute-Force)

Brute-Force Search for Counting Inversions

Input: array A of n distinct integers.
Output: the number of inversions of A.

$numInv := 0$
for $i := 1$ to $n - 1$ **do**
  for $j := i + 1$ to n **do**
    if $A[i] > A[j]$ **then**
      $numInv := numInv + 1$
return $numInv$


شمارش
وارونگی‌های یک
دنباله

پیدا کردن تعداد وارونگی‌ها در یک دنباله از اعداد طبیعی

ورودی: یک دنباله از اعداد متمایز

هدف: تعداد وارونگی‌های دنباله ورودی

رویکرد تقسیم و غلبه

وارونگی‌های
یک دنباله

رویکرد تقسیم و غلبه

وارونگی‌های چپ: i و j هر دو سمت چپ قرار دارند.

وارونگی‌های یک دنباله

وارونگی راست: i و j هر دو سمت چپ قرار دارند.

وارونگی دوبخشی: i در نیمه چپ و j در نیمه راست قرار دارد.

رویکرد تقسیم و غلبه (تلاش اول)

رویکرد تقسیم و غلبه (تلاش اول)

CountInv

Input: array A of n distinct integers.
Output: the number of inversions of A.

---

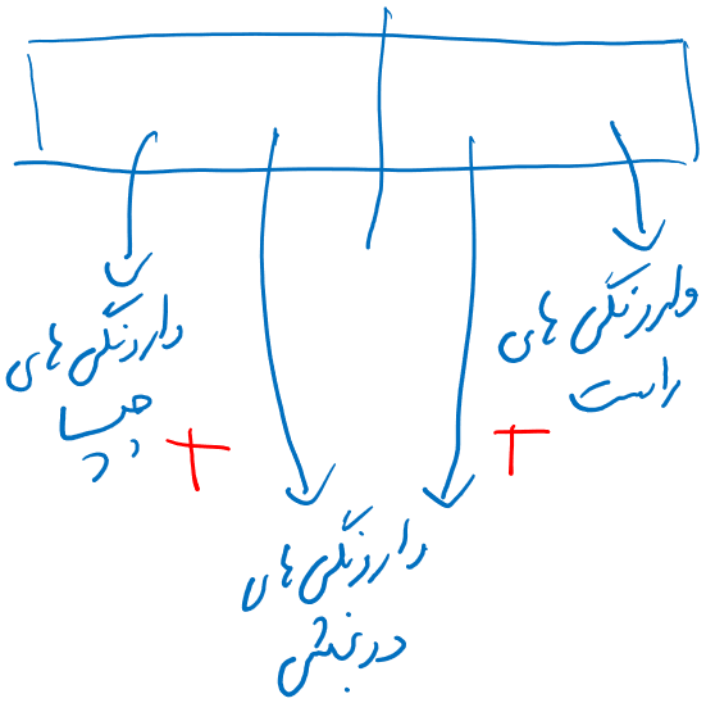
if $n = 0$ or $n = 1$ **then** // base cases
    return 0
else
    $leftInv :=$ `CountInv`(first half of A)
    $rightInv :=$ `CountInv`(second half of A)
    $splitInv :=$ `CountSplitInv`(A)
    return $leftInv + rightInv + splitInv$

رویکرد تقسیم و غلبه (تلاش اول)

CountInv

Input: array A of n distinct integers.
Output: the number of inversions of A.

```
if n = 0 or n = 1 then                    // base cases
    return 0
else
    leftInv := CountInv(first half of A)
    rightInv := CountInv(second half of A)
    splitInv := CountSplitInv(A)
    return leftInv + rightInv + splitInv
```

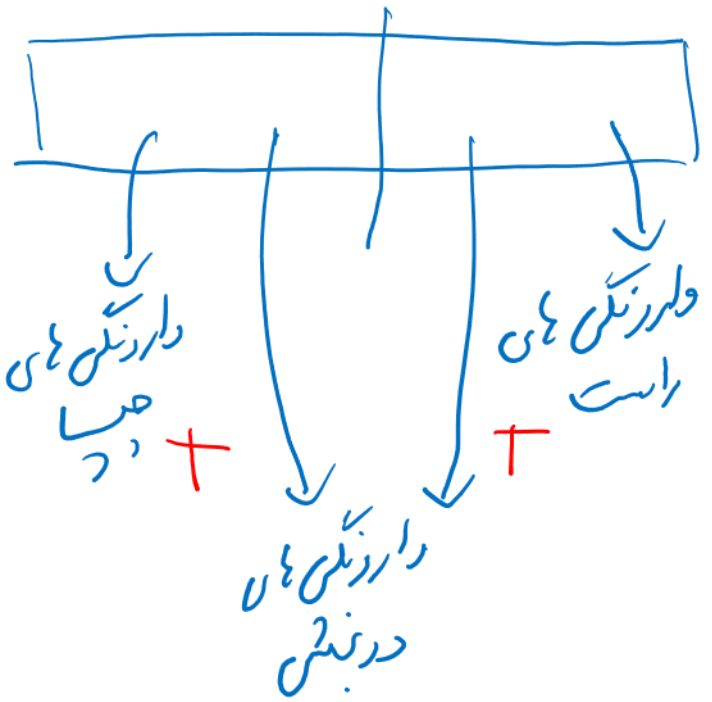


رویکرد تقسیم و غلبه (تلاش اول)

CountInv

Input: array A of n distinct integers.
Output: the number of inversions of A.

```
if n = 0 or n = 1 then                          // base cases
    return 0
else
    leftInv := CountInv(first half of A)
    rightInv := CountInv(second half of A)
    splitInv := CountSplitInv(A)
    return leftInv + rightInv + splitInv
```



شمارش وارونگی‌های دوبخشی (یادآوری مرتب‌سازی ادغامی)

Merge

Input: sorted arrays C and D (length $n/2$ each).
Output: sorted array B (length n).
Simplifying assumption: n is even.

$i := 1, j := 1$
for $k := 1$ to n **do**
  if $C[i] < D[j]$ **then**
    $B[k] := C[i], i := i + 1$
  else // $D[j] < C[i]$
    $B[k] := D[j], j := j + 1$

شمارش وارونگی‌های دوبخشی با ایده مرتب‌سازی ادغامی

شمارش وارونگی‌های دوبخشی با ایده مرتب‌سازی ادغامی

شمارش وارونگی‌های دوبخشی با ایده مرتب‌سازی ادغامی

Merge

Input: sorted arrays C and D (length $n/2$ each).
Output: sorted array B (length n).
Simplifying assumption: n is even.

$i := 1, j := 1$
for $k := 1$ to n **do**
  if $C[i] < D[j]$ **then**
    $B[k] := C[i], i := i + 1$
  else // $D[j] < C[i]$
    $B[k] := D[j], j := j + 1$

شمارش وارونگی‌های دوبخشی در زمان خطی

Merge-and-CountSplitInv

Input: sorted arrays C and D (length $n/2$ each).
Output: sorted array B (length n) and the number of split inversions.
Simplifying assumption: n is even.

$i := 1$, $j := 1$, $splitInv := 0$
for $k := 1$ to n **do**
  if $C[i] < D[j]$ **then**
    $B[k] := C[i]$, $i := i + 1$
  else // $D[j] < C[i]$
    $B[k] := D[j]$, $j := j + 1$
    $splitInv := splitInv + \underbrace{(\frac{n}{2} - i + 1)}_{\#\text{ left in } C}$
return $(B, splitInv)$

رویکرد تقسیم و غلبه (تلاش دوم)

Sort-and-CountInv

Input: array A of n distinct integers.
Output: sorted array B with the same integers, and the number of inversions of A.

---

if $n = 0$ or $n = 1$ **then** // base cases
    return $(A, 0)$
else
    $(C, leftInv) :=$ Sort-and-CountInv(first half of A)
    $(D, rightInv) :=$
     Sort-and-CountInv(second half of A)
    $(B, splitInv) :=$ Merge-and-CountSplitInv(C, D)
    return $(B, leftInv + rightInv + splitInv)$

رویکرد تقسیم و غلبه (تلاش دوم)

Sort-and-CountInv

Input: array A of n distinct integers.
Output: sorted array B with the same integers, and the number of inversions of A.

---

if $n = 0$ or $n = 1$ **then** `// base cases`
  return $(A, 0)$
else
  $(C, leftInv) :=$ `Sort-and-CountInv`(first half of A)
  $(D, rightInv) :=$
    `Sort-and-CountInv`(second half of A)
  $(B, splitInv) :=$ `Merge-and-CountSplitInv`(C, D)
  return $(B, leftInv + rightInv + splitInv)$

تحلیل زمانی پیدا کردن تعداد وارونگی‌ها

Sort-and-CountInv

Input: array A of n distinct integers.
Output: sorted array B with the same integers, and the number of inversions of A.

---

if $n = 0$ or $n = 1$ **then** // base cases
    return $(A, 0)$
else
    $(C, leftInv)$:= Sort-and-CountInv(first half of A)
    $(D, rightInv)$:=
     Sort-and-CountInv(second half of A)
    $(B, splitInv)$:= Merge-and-CountSplitInv(C, D)
    return $(B, leftInv + rightInv + splitInv)$