جَامعة
المَنارة
MANARA UNIVERSITY

# البرمجة التفرعية

# Parallel Programming

Dr.-Eng. Samer Sulaiman

2023-2024

# مفردات المنهاج

- أساسيات البرمجة التفرعية
  - مقدمة
  - معامل التسريع
  - أنواع الأنظمة المتعددة المعالجات والبرمجيات الداعمة لها
  - موازنة الأعباء وتحمل الخلل
  - تطبيقات البرمجة التفرعية
  - أشكال معالجة المعطيات على التوازي
- الحواسيب التفرعية
  - تصنيف فلاين Flynn's Classification Scheme
  - شبكات الربط الداخلية Interconnection Networks
- مبادئ تصميم الخوارزميات المتوازية
  - مفاهيم أساسية
  - الإجرائيات والمقابلة
  - تقنيات التقسيم
- البرمجيات الداعمة للبرمجة التفرعية
  - المعتمدة على الذاكرة المشتركة
  - المعتمدة على تمرير الرسائل
- تحليل الأداء Performance Analysis

# مبادئ تصميم الخوارزميات المتوازية

- أمثلة للخوارزميات المتوازية:
  - خوارزمية الفرز الفقاعي وتوابعها (Bubble Sort)
    - تعتمد على مقارنة واستبدال العناصر المتجاورة في السلسلة التي سترتب.
    - ليكن لدينا السلسلة $(a_1, a_2, \ldots\ldots a_n)$
      - سيتم اجراء عملية المقارنة والاستبدال n-1 مرة على الشكل التالي:
        - $(a_1, a_2), (a_2, a_3), \ldots, (a_{n-1}, a_n)$
        - هنا سيتم إزاحة العنصر الأكبر إلى نهاية السلسلة
        - بعد ذلك سيتم تجاهل العنصر الأخير لأنه أخذ الترتيب الصحيح له
      - سيتم تكرار عملية المقارنة-والاستبدال على السلسلة الناتجة وفي كل تكرار يتم إزاحة العنصر الأكبر إلى آخر موضع في السلسلة لم يتم تجاهله
      - وستكون السلسلة مرتبة بعد عدد n-1 من التكرارات
        - إن التكرار ضمن الحلقة الداخلية في خوارزمية الفرز الفقاعي تأخذ من الوقت O(n)
        - بالإضافة إلى ذلك يتم أداء ما مجموعه O(n) تكرار بسبب الحلقة الخارجية
        - بالتالي ستكون درجة التعقيد للفرز الفقاعي مساوية إلى $O(n^2)$
    - تقوم خوارزمية الفرز الفقاعي بمقارنة جميع الأزواج المتجاورة بالترتيب ولهذا السبب فهي خوارزمية تسلسلية

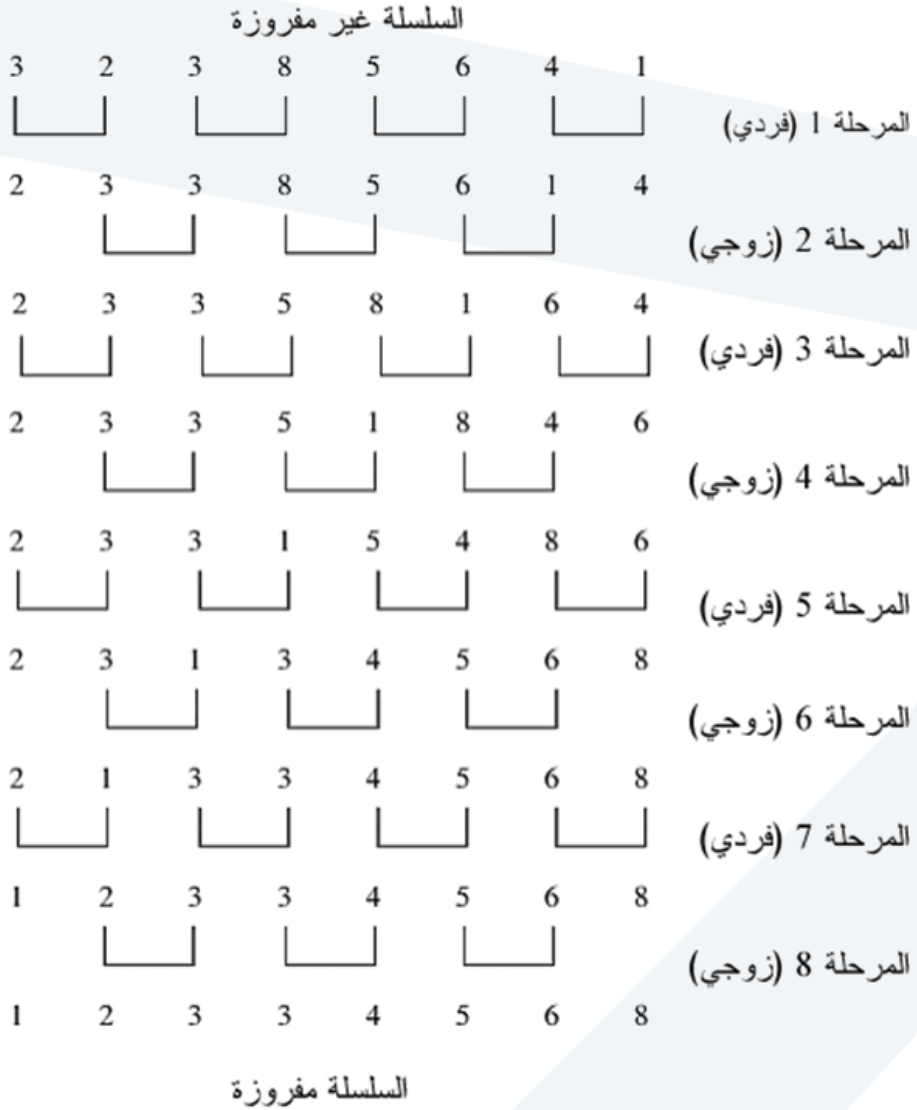
# مبادئ تصميم الخوارزميات المتوازية

- أمثلة للخوارزميات المتوازية:
  - خوارزمية الفرز الفقاعي وتوابعها (Bubble Sort)

- procedure BUBBLE_SORT(n)
  begin
  for i := n - 1 downto 1 do
  for j := 1 to i do
  compare-exchange(aj, aj + 1);
  end BUBBLE_SORT

- الإبدال الزوجي-الفردي (Odd-Even Transposition)
  - تعتمد على فرز n عنصر بـ n مرحلة (بحيث يكون n عدد زوجي)
  - كل مرحلة تتطلب n/2 من عمليات المقارنة-والاستبدال
  - تتناوب بين مرحلتين وهما مرحلة الفردي ومرحلة الزوجي
  - ليكن لدينا السلسلة $(a_1, a_2, \ldots\ldots a_n)$
    - خلال مرحلة الفردي: سيتم مقارنة العناصر ذوات الدليل الفردي مع ما يجاورها إلى اليمين واستبدالها في حال الضرورة
      - $(a_1,a_2), (a_3,a_4), \ldots\ldots\ldots, (a_{n-1},a_n)$
    - خلال المرحلة الزوجية: سيتم مقارنة العناصر التي لها دليل زوجي مع ما يجاورها ناحية اليمين واستبدالها في حال الضرورة
      - $(a_2,a_3), (a_4,a_5), \ldots\ldots\ldots, (a_{n-2},a_{n-1})$

# مبادئ تصميم الخوارزميات المتوازية




- أمثلة للخوارزميات المتوازية:
  - الإبدال الزوجي-الفردي (Odd-Even Transposition)
    - بعد n مرحلة ستكون السلسلة قد رتبت بالفعل
    - كل مرحلة من الخوارزمية (فردية أو زوجية) تتطلب O(n) عملية مقارنة واستبدال
      - بالتالي ستكون درجة التعقيد للخوارزمية مساوية إلى $O(n^2)$
    - مثال: ليكن لدينا السلسة الموضحة بالشكل
    - المطلوب ترتيب وفرز هذه السلسة باستخدام خوارزمية الابدال الزوجي-الفردي

# مبادئ تصميم الخوارزميات المتوازية

- أمثلة للخوارزميات المتوازية:
  - الإبدال الزوجي-الفردي (Odd-Even Transposition)
  - الصيغة التسلسلية:

```
procedure ODD-EVEN(n)
begin
        for i := 1 to n do
        begin
                if i is odd then
                        for j := 0 to n/2 - 1 do
                                compare-exchange(a2j + 1, a2j + 2);
                if i is even then
                        for j := 1 to n/2 - 1 do
                                compare-exchange(a2j, a2j + 1);
        end for
end ODD-EVEN
```

# مبادئ تصميم الخوارزميات المتوازية

- أمثلة للخوارزميات المتوازية:
  - الإبدال الزوجي-الفردي (Odd-Even Transposition)
    - الصيغة التفرعية (المتوازية):
      - بفرض لدينا سلسلة من n عدد ولدينا نظام من n معالج (اجرائية) مرتبة في مصفوفة احادية
      - في البداية سيستقر العنصر $a_i$ في الإجرائية $P_i$ حيث i=1,2,3....n
      - خلال المرحلة الفردية ستقوم كل إجرائية لها دليل فردي بإجراء عملية مقارنة-واستبدال لعناصرها مع العناصر المستقرة في جارتها اليمنى.
      - بشكل مشابه، خلال المرحلة الزوجية ستقوم كل إجرائية دليلها زوجي بإجراء عملية مقارنة-واستبدال لعناصرها مع العناصر المستقرة في جارتها اليمنى
      - خلال كل مرحلة من الخوارزمية، تؤدي العمليات الزوجية أو الفردية عملية مقارنة-واستبدال مع الجار الأيمن.
        - يتطلب ذلك من الوقت O(1) وإجمالا سيتم أداء ى مرحلة مماثلة
        - لذلك، سيكون وقت التشغيل للصيغة المتوازية هو O(n)

# مبادئ تصميم الخوارزميات المتوازية

- أمثلة للخوارزميات المتوازية:
  - الإبدال الزوجي-الفردي (Odd-Even Transposition)
  - الصيغة التفرعية (المتوازية):

```
procedure ODD-EVEN_PAR (n)
begin
    id := process's label
    for i := 1 to n do
    begin
                if i is odd then
                        if id is odd then
                                compare-exchange_min(id + 1);
                        else
                                compare-exchange_max(id - 1);
                if i is even then
                        if id is even then
                                compare-exchange_min(id + 1);
                        else
                                compare-exchange_max(id - 1);
    end for
end ODD-EVEN_PAR
```

# مبادئ تصميم الخوارزميات المتوازية

- أمثلة للخوارزميات المتوازية:
  - خوارزمية الفرز السريع (Quicksort)

```
void quickSort(int A[], int p, int r) {
   if (p < r) {
      int q = partition(A, p,r);
      quickSort(A, p, q - 1);
      quickSort(A, q + 1, r);    } }
int main() {
   int a[] = {2,6,5,1,3,4};
   int n = sizeof(a)/sizeof(a[0]);
   quickSort(a,0,n-1);
   for(int i=0;i<n;i++)
   cout<<a[i]<<" ";
   return 0; }
```

```
#include <iostream>
using namespace std;
void swap(int *a,int *b) {
   int temp = *a;    *a=*b;    *b = temp; }
int partition (int A[], int p, int r) {
   int x = A[r];    int i = p - 1;
for (int j = p; j <= r- 1; j++) {
      if (A[j] <= x) {
         i++;
         swap (&A[i], &A[j]);      }   }
   swap (&A[i + 1], &A[r]);
   return (i + 1); }
```

# مبادئ تصميم الخوارزميات المتوازية

- أمثلة للخوارزميات المتوازية:
  - خوارزمية الفرز السريع (Quicksort)

```
void quickSort(int A[], int p, int r) {
   if (p < r) {
      int q = partition(A, p,r);
      quickSort(A, p, q - 1);
      quickSort(A, q + 1, r);    } }
int main() {
   int a[] = {2,6,5,1,3,4};
   int n = sizeof(a)/sizeof(a[0]);
   quickSort(a,0,n-1);
   for(int i=0;i<n;i++)
   cout<<a[i]<<" ";
   return 0; }
```