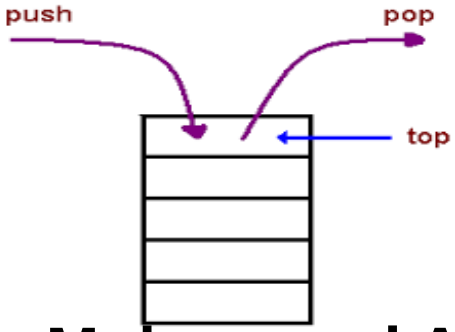
# جامعة حماة
# الكلية التطبيقية
# قسم تقنيات الحاسوب
# الفصل الثاني


- المقرر: الخوارزميات وبنى المعطيات
- المحاضرة :الثالثة

**Dr. Mohammed AL-Mohammed**

# Chapter 3

## خوارزميات الفرز والترتيب

## Sorting Algorithms

## الجزء -1 -

## المقدمة

- خوارزميات الترتيب هي خوارزميات تمكن من تنظيم مجموعة عناصر حسب ترتيب محدد(تصاعدي - تنازلي) ، العناصر المراد ترتيبها توجد في مجموعة مزودة بعلاقة ترتيب معينة.
- تصنيف خوارزميات الترتيب مهم جدا، لانه يمكن من اختيار نوع الخوارزمية الأكثر مناسبة للشكل المعالج، مع الأخذ بعين الاعتبار السلبيات الموجودة في الخوارزمية
- أي أن الترتيب هو عبارة عن عملية ترتيب مجموعة من العناصر البيانية وفق قيمة معينة تسمى حقل أو وفق حقول تسمى المفتاح إما بصورة تصاعدية أو تنازلية .
- وبالتالي فالغرض من الترتيب هو:

-زيادة كفاءة الخوارزمية " البحث عن عناصرها"

- تبسيط معالجة الملفات
- حل مشكلة تشابه القيود

## أنواع الترتيب Types of Sorting

1 –الترتيب الداخلي (Internal Sort)

2 –الترتيب الخارجي (External Sort)

- الترتيب الداخلي: يحدث داخل الذاكرة بحيث يكون حجم البيانات مناسب وليس كبير ويشمل:

1. ترتيب الاختيار (Selection Sort)
2. الترتيب الفقاعي (Bubble Sort)
3. ترتيب الحشر أو الاضافة (Insertion Sort)
4. ترتيب شيل (Shell sort)

5. الترتيب السريع (Quick Sort)
6. ترتيب الاساس (Radix Sort)
7. ترتيب المؤشرات (pointers Sort)
8. الترتيب الشجري(Tree Sort)

- الترتيب الخارجي : وهو الترتيب الذي يحدث خارج الذاكرة في أواسط الخزن الثانوي عندما يكون حجم البيانات كبير بحيث يتعذراستيعابها في الذاكرة اثناء عملية الترتيب ويشمل:

1. الترتيب بالدمج (Merge Sort)
2. الترتيب بالدمج المتوازن ذو المسارين (Balanced two Merge Sort)
3. الترتيب بالدمج باستخدام طريقة فرق تسد(Divided & conquer Merge Sort)

## المقدمة:

من خلال دراستك لخوارزميات الترتيب المختلفة ستجد أن مفهوم **Swap** أو تبادل مُحتوى متغيرين يتكرر باستمرار .. فأغلب خوارزميات الترتيب (إن لم تكن كلها !) تعتمد في مرحلة ما على تبديل محتوى خانتين لوضعهما في الترتيب الصحيح, لذا سنوضح كيفية تبديل محتوى متغيرين قبل أن نبدأ بكتابة الخوارزمية بلغة السي++.

سنتناول فيما يلي 4 طرق لعمل Swap, لكل منها إيجابيات كما لها سلبيات أيضا.

الطريقة الأولى (و هي الأكثر شهرة) تكمن في استخدام وسيط مساعد يُساعدنا على تبديل محتوى المتغيرين، فنضع قيمة المتغير الأول في الوسيط ثم نضع قيمة المتغير الثاني في المتغير الأول ثم نضع قيمة الوسيط في المتغير الثاني، و بهذا نكون قد بادلنا بين قيمة المتغير الأول و الثاني. لنفرض أن x=5 و y=-1 :

```
t = x //   t = 5    x = y   //   x = -1   y = t   //   y = 5
```

في النهاية تكون x=-1 و y=5 و بالتالي تم تبديل قيمتي x و y

هذه الطريقة بسيطة و تقليدية أيضا، إلا أنها تأخذ مساحة أكثر من الذاكرة بالإعلان عن الوسيط الثالث.

يمكننا عمل Swapبدون حاجة إلى وسيط مساعد عن طريق إجراء بعض العمليات الحسابية البسيطة على المتغيرين.

**الطريقة الثانية** (تصلح للمتغيرات العددية فقط) :

```
x=x+y ;   y=x-y ;   x=x-y ;
```

**الطريقة الثالثة** (تصلح للمتغيرات العددية فقط و يجب أن تختلف قيمة yعن الصفر) :

```
x=x*y;   y=x/y;   x=x/y;
```

**ملاحظة** : الطريقتان السابقتان يمكنهما أن يتسببا في حدوث Overflow إذا كانت قيمة x+y أو x*y أكبر من القيمة العظمى لنوع المتغيرين x , y

**الطريقة رابعة** لا تحتاج إلى وسيط مساعد و لا يمكن أيضا أن تكون سببا في حدوث **Overflow**

تعتمد هذه الطريقة على أحد الـ Bitwise (مؤثرات الـ Bit) وهو المؤثر XOR اختصار لـ (Exclusive OR) و الذي يُرمز له بـ ^ ، هذا المؤثر يعطينا True إذا و فقط إذا كان المدخلين مختلفين، أما إذا كانا متشابهين فالنتيجة ستكون False و يعمل هذا المؤثر كالتالي :

```
/*
  True = 1
  False = 0
*/
   15 ^ 8 = 7
   00001111 // 15 in Binary
^
   00001000//8 in Binary
=  00000111// 7 in Binary
```

كمثال على ذلك

```
/*
  True = 1
  False = 0
*/
1 ^ 1 = 0
0 ^ 0 = 0
1 ^ 0 = 1
0 ^ 1 = 1
```

سنقوم بتجربة هذه الطريقة على المتغيرين x و y حيث x = 15 و y=7 ، لذلك سنقوم بالخطوات الثلاثة التالية :

x = x^y // x = 15 XOR 7 = 8 y = x^y

// y = 8 XOR 15 = 7 x = x^y

// x = 7 XOR 8 = 15

و بهذا تم تبديل محتوى المتغيرين x و y

يمكننا الآن كتابة دالة تبادل بين مُحتوى مُتغيرين باستخدام إحدى الطرق الأربعة الموضحة أعلاه، مع الأخذ في الاعتبار سلبيات كل طريقة.

إذا أدرت التوسع أكثر في هذه الجزئية، فيمكنك البحث عن أفضل طريقة لتبديل محتوى متغيرين بحيث تحقق النقاط الآتية في آن واحد :

- لا تحتاج إلى وسيط مساعد (المستوى : **بسيط**)
- لا تسبب Overflow (المستوى : **صعب**)
- لا تضع أي شرط على قيم أو نوع المتغيرين (المستوى:**أكثر صعوبة**، إن لم يكن مستحيلا !)

## كتابة دالة تقوم بعملية التبديل :

بعد أن تطرقنا إلى الطرق الأربعة التي يمكننا من خلالها تبديل محتوى متغيرين. سنقوم الآن بكتابة دالة تُبادل بين محتوى خانتين من المصفوفة X باستخدام الطريقة الأولى (لأنها الأكثر شعبية .. رغم احتوائها على انتهاك صارخ لحقوق الـ : ! Overflow)

```
1  void Swap(int X[], int i, int j)
2  {
3      int temp;
4      temp = X[i];
5      X[i] = X[j];
6      X[j] = temp;
7  }
```

```
1  void Swap(int *X, int i, int j)
2  {
3      int temp;
4      temp = *(X + i);
5      *(X + i) = *(X + j);
6      *(X + j) = temp;
7  }
```

# ترتيب الاختيار *selection sort*

**الفكرة العامة:** مقارنة كل عنصر من عناصر المصفوفة مع باقي العناصر واستخدام تابع *Swap* للتبديل بين العناصر

سوف نعتمد في هذا البحث على بنية المعطيات "المصفوفة".

حيث يبحث عن اصغر(أكبر) عنصر في المصفوفة و يقارنه مع اول عنصر في المصفوفة.

**مثال :**

| | |
|---|---|
| المصفوفة | 8 , 4 , 3 , 5 , 1 |
| البحث عن العنصر الاول | 1 , 4 , 3 , 5 , 8 |
| البحث عن العنصر الثاني | 1 , 3 , 4 , 5 , 8 |

الكود البرمجي:

```
1.int maxSelect (int a[],int n)
2. {
3. int maxPos(0),currentPos(1);
4. while (currentPos<n){
5. if(a[currentPos]>a[maxPos])
6. maxPos=currentPos;
7. currentPos++;
8. }
9. return maxPos;
10. }
11. void swapElements(int a[],int maxPos,int last);
12. void selectionSort(int a[ ],int n)
13. {
14. int last(n-1);
15. int maxPos;
16. while(last>0){
17. maxPos=maxSelect(a,last+1);
18. swapElements(a,maxPos,last);
19. last --;
20. }
21. }
```

هذه الخوارزمية تبين ترتيب عناصر مصفوفة تصاعدياً

أولاً عرفنا عن التابع $maxSelect$ للحصول على أعلى قيمة في المصفوفة $a$ التي تحوي $n$ عنصر كالتالي:

عرفنا متغير $maxPos$ وهو دليل أو $index$ عنصر المصفوفة الأكبر قيمة وأعطيناه قيمة ابتدائية هي $0$ أي افترضنا أن $a[0]$ هي أكبر حد وعرفنا متغير $currentPos$ هو $index$ العنصر الحالي الذي نستخدمه للمقارنة مع $a[maxPos]$ فإذا كان $a[maxPos] < a[currentPos]$ نغير $index$ العنصر الأكبر وهكذا حتى نحصل على أكبر عنصر في المصفوفة فنعيد قيمة ترتيبه في المصفوفة.

في السطر $12$ عرفنا تابع الترتيب $selectionSort$ وعرفنا متغير $last$ وأعطيناه قيمة ابتدائية هي $n-1$ أي أن $a[last]$ هو آخر عنصر في المصفوفة وبما أننا في نهاية حلقة $while$ نقوم بإنقاص قيمة الـ $last$ فوضعنا شرط أن تستمر حلقة $while$ طالما $last > 0$ ثم عرفنا قيمة $maxPos$ وأسندنا لها القيمة التي يعيدها التابع $maxSelect$ أي دليل أكبر عنصر في المصفوفة.

ثم استدعينا التابع $swapElements$ للتبديل بين $index$ العنصر الأكبر في المصفوفة و$index$ العنصر الأخير أي أننا وضعنا العنصر الأكبر في المصفوفة آخر عنصر وباستمرار العمليات تترتب المصفوفة تصاعدياً.

# التعقيد Big-O لهذه الخوارزمية

لمعرفة التعقيد سنقوم بعد العمليات في التابع الأساسي للخوارزمية $Selection\ Sort$ فنلاحظ أن الـ $while$ تتكرر $n-1$ مرة حيث أن القيمة الابتدائية لـ $last = n-1$ وتتكرر $while$ طالما $last > 0$ حيث أن $last$ تتناقص وبالتالي فإن $while$ تتكرر $n-1$ مرة.

**ولكن:** من خلال عمليات $while$ السابقة قمنا باستدعاء التابع $maxSelect$ الذي يحتوي أيضاً على حلقة $while$ وبارامترات التابع هي $a$ و$last + 1$ أي $n$ وبالتالي تتكرر $while$ الداخلية $n-1$ مرة.

وفي التكرار الثاني لحلقة $while$ الخارجية "للتابع $Selection\ Sort$" تصبح $last = n-2$ أي بارامترات التابع $maxSelect$ هي $a$ و$n-1$ فتتكرر حلقة $while$ الداخلية $n-2$ مرة ... إلخ.

بمعنى آخر لدينا $n-1$ تكرار وكل تكرار سيقوم بمقارنات محددة:

أول تكرار يقوم بـ $n-1$ مقارنة .. ثاني تكرار يقوم بـ $n-2$ مقارنة .... آخر تكرار يقوم بمقارنة واحدة.

عدد المقارنات ← زمن تنفيذ الخوارزمية:

$$\sum (n-1)+(n-2)+\cdots+1=\sum_{i=1}^{n-1} i \Rightarrow \sum_{i=1}^{n-1} i=\frac{n*(n-1)}{2}$$

$$\frac{(n-1)n}{2}=\frac{n^2}{2}-\frac{n}{2}\Rightarrow O(n^2)$$

ولدينا $n$ عملية $swap$ أي تبديل بين العناصر.

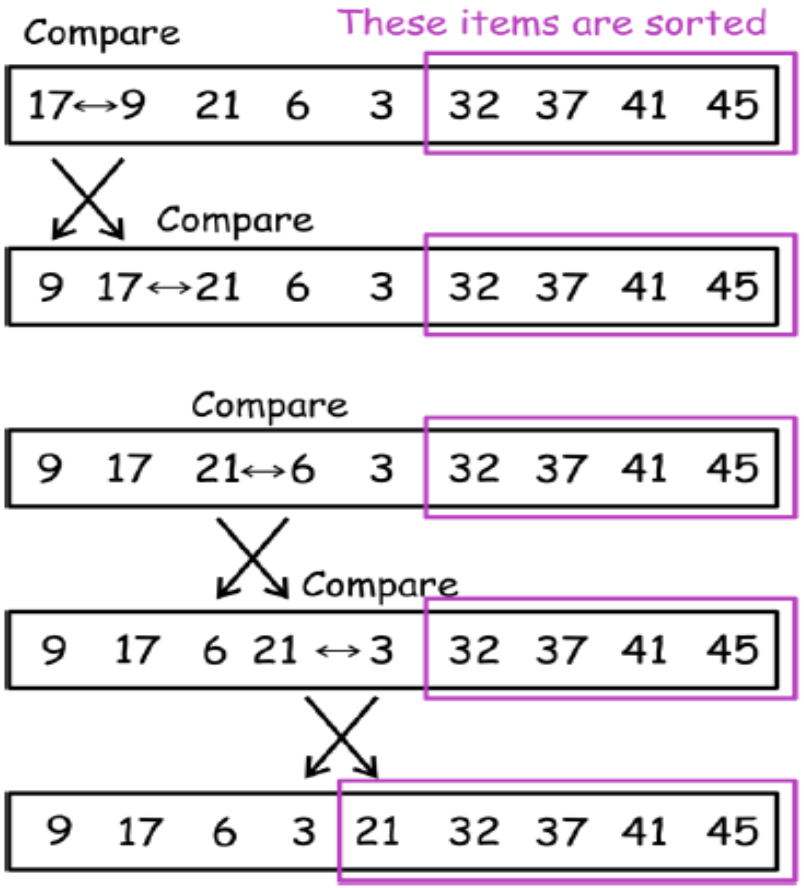
**ملاحظة:** نعتمد على عدد المقارنات في حساب التعقيد لأنها أكثر العمليات التي تستهلك زمناً .

# الفرز الفقاعي Bubble Sort

**الفكرة العامة:** الفقاعات الكبيرة تصعد والفقاعات الصغيرة ترسو الفقاعات الكبيرة لا تقبل أن تكون تحت الصغيرة فتصعد إلى فوقها.

عملياً: نقارن كل عنصرين مع بعضهما إذا كان العنصر الثاني أصغر من الأول نبدل بينهما ونستمر بمقارنة العنصر الأكبر مع باقي العناصر حتى نجد الأكبر ...

حيث يقارن اول عنصرين ثم يقارن العنصرين التاليين و يبدل بينهم و هكذا....



تقارن هذه الخوارزمية بين قيم الخانات المتجاورة، تبدأ بالمقارنة بين أول خانتين من المصفوفة، إذا كان محتوى الخانة الأولى أكبر من محتوى الخانة الثانية سيتم تبادل محتوى الخانتين و هكذا مع بقية الخانات. عند انتهاء الدورة الأولى ستكون الخوارزمية قد أنجزت n-1 مقارنة و بهذا ستتم إزاحة أكبر عناصر المصفوفة إلى الخانة الأخيرة.

بقي الآن ترتيب الـ n-1 عنصر. و هكذا الأمر مع بقية الدورات ...

النتيجة : عملية الترتيب تختلف عن عملية البحث في هذه النقطة، فالترتيب لا يقبل وجود عدة احتمالات في النتيجة، فعند تطبيق الخوارزمية سنحصل على مصفوفة مرتبة تصاعديا و بالتالي لا توجد احتمالات للمناقشة.

الإيجابيات : الخوارزمية سهلة التصور و بسيطة المفهوم.

السلبيات : بطيئة شيئا ما وغير عملية، خصوصا عند معالجة المصفوفات الضخمة.

## الكود البرمجي:

```
1. void swapElements(int a[],int b,int c);
2. void bubbleSortPhase(int a[],int last)
3. {
4. int pos;
5. for(pos=0;pos<last-1;pos++)
6. if(a[pos]>a[pos+1]){
7. swapElements(a,pos,pos+1);
8. }
9. }
10. void bubbleSort(int a[],int n)
11. {
12. int i;
13. for(i=n-1;i>0;i--)
14. bubbleSortPhase(a,i);
15. }
```

تقوم فكرة الفرز الفقاعي على مقارنة أول عنصر من المصفوفة مع العنصر الذي يليه فإذا كان أكبر منه يتم التبديل ومقارنة العنصر الثاني مع الثالث وإذا كان أكبر يتم التبديل وهكذا ..

نلاحظ التابع الأساسي $bubbleSort$ يحتوي على حلقة $for$ تتكرر $n-1$ مرة يستدعى بداخلها التابع $bubbleSortPhase$ التي يتم من خلاله عملية المقارنة.

يتم من خلال هذه المقارنة الحصول على أكبر رقم ووضعه في آخر خانة من المصفوفة وبدوران حلقة $for$ نقوم بترتيب الأعداد تصاعدياً وذلك بوضع الأرقام الكبيرة في آخر عنصر يتم الوصول إليه من المصفوفة.

## التعقيد Big-O لهذه الخوارزمية

نلاحظ أن عدد العمليات في الفرز الفقاعي لا يفرق عن الفرز الانتقائي:

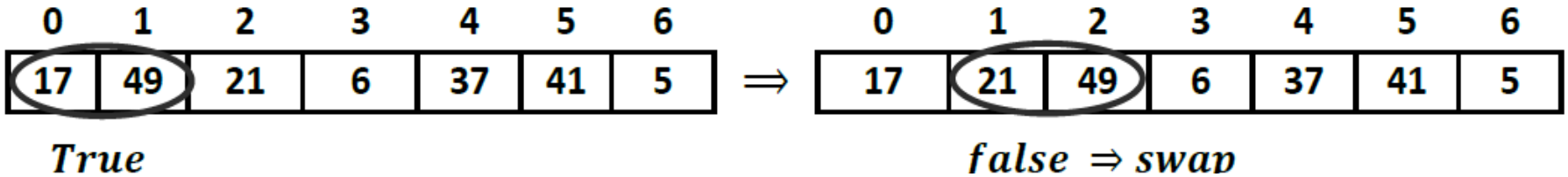
فيمثل عدد المقارنات كالتالي:

$$\sum (n-1)+(n-2)+\cdots+1=\sum_{i=1}^{n-1} i \Rightarrow \sum_{i=1}^{n-1} i=\frac{n*(n-1)}{2}$$

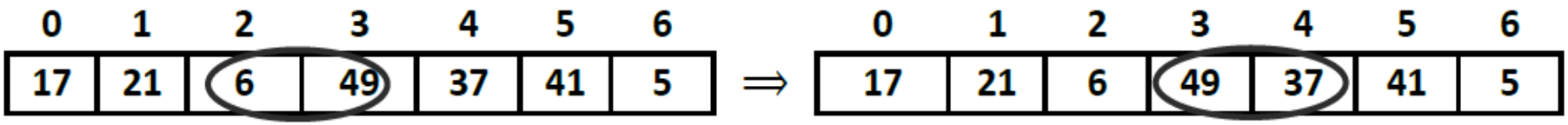
$$\frac{(n-1)n}{2}=\frac{n^2}{2}-\frac{n}{2}\Rightarrow O(n^2)$$

# مثال:

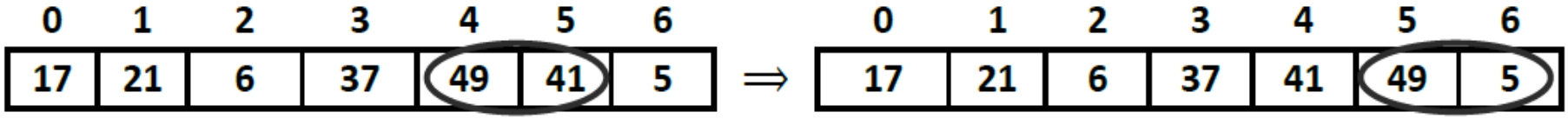
ليكن لدين المصفوفة التالية ونريد ترتيبها ترتيب تصاعدي باستخدام الفرز الفقاعي:




$True$ $false \Rightarrow swap$

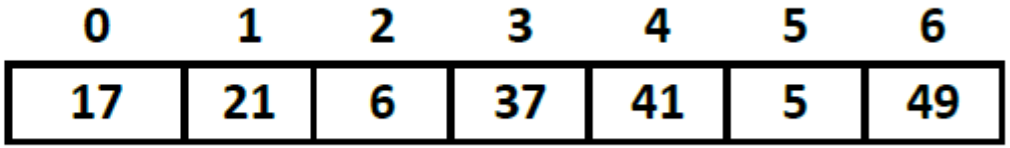



$false \Rightarrow swap$ $false \Rightarrow swap$

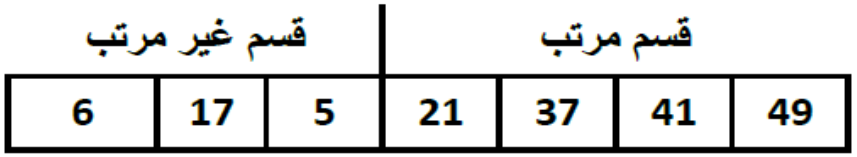



$false \Rightarrow swap$ $false \Rightarrow swap$

وهكذا تكون المصفوفة بعد المرور الأول.




ويكون شكل المصفوفة بعد المرور الرابع:




وبعد سبعة مرورات تترتب بالكامل: