الخوارزميات وبنى المعطيات 2

للدكتور غيث بلال

المحاضرة الرابعة

خوارزمية الفرز (الترتيب) بالتجزئة (الفرز السريع)

QuickSort

تعتبر هذه الخوارزمية كخوارزمية الترتيب بالدمج والتي تقسم سلسلة العناصر المطلوب فرزها إلى سلسلتين جزئيتين بحيث يكون كل عناصر السلسلة الجزئية الأولى اليسارية أصغر من كل عناصر السلسلة الجزئية الثانية اليمينية.

نستمر بهذه الفكرة (فكرة التقسيم) للسلسلتين السابقتين حتى نحصل على سلسلة مؤلفة من عنصر واحد.

وبالتالي تقسيم السلسلة الأصلية إلى جزئين يحققان الشرط المطلوب.

أما آلية عمل هذه الخوارزمية فتعتمد على اختيار أحد عناصر المصفوفة ونسميه عنصراً موجهاً(**pivot**) نختار هنا العنصر الأول وبالتالي ننشئ سلسلة جزئية كل عناصرها أصغر أو يساوي هذا العنصر الموجه وسلسلة جزئية أخرى كل عناصرها أكبر تماماً من هذا العنصر الموجه.

- إن إنشاء السلسلتين الجزئيتين على السلسلة نفسها (دون استعمال مصفوفة مساعدة) يؤدي إلى وضع العنصر الموجه في المكان النهائي الصحيح الذي يجب ان يبقى فيه حتى نهاية الترتيب وبالتالي نحتاج إلى خوارزمية أخرى تسمى خوارزمية التقسيم (**partition**) شكلها كما يلي:

Partition(t, i, j, k)

حيث

t مصفوفة

i دليل العنصر الأول **first**

j دليل العنصر الأخير **last**

k الدليل الموجه **Pivot index**

مثال: ليكن لدينا المصفوفة التالية:

طبق خوارزمية الترتيب السريع

n = 6

110	120	130	62	40	20

العنصر الموجه L=2 K=6

نبدأ بالمقارنة بحيث يتحقق:

L : نقف عندما نحصل على عنصر أكبر من العنصر الموجه.

K : نقف عندما نحصل على عنصر أصغر من العنصر الموجه.

110	120	130	62	40	20

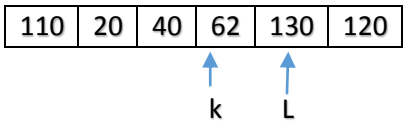
أذا كانت الشرط L< K محقق نبادل بين t[k] و t[L] نحصل على

110	20	130	62	40	120

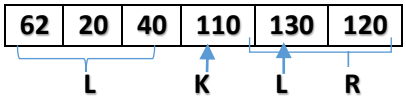
L محقق الشرط نبادل k

110	20	40	62	130	120

L k



نلاحظ ان L تجاوز K لذلك نقف ونبادل بين العنصر الموجه و t[K]



وهكذا حصلنا على سلستين جزئيتين يكون موقع العنصر الموجه صحيحاً بحيث تحتوي السلسلة الجزئية الأولى جميع العناصر التي هي أصغر من العنصر الموجه وتحتوي الثانية جميع العناصر التي هي أكبر منه
نكرر نفس العمليات السابقة على السلسلتين **L** و **R**

خوارزمية التقسيم **pseudo-code**

1. Start.
2. Function Partition(t : array, i, j, K : int).
   2.1. L ← i + 1.
   2.2. k ← j.
   2.3. while(L ≤ K).
      2.3.1. while(t[K] > t[i]&& k>0).
         2.3.1.1. k ← k – 1.
      2.3.2. while(t[L] ≤ t[i]&& L< j).
         2.3.2.1. L ← L + 1.
      2.3.3. if(L < K).
         2.3.3.1. w ← t[L].
         2.3.3.2. t[L] ← t[K].
         2.3.3.3. t[K] ← w.
         2.3.3.4. k ← k – 1.
         2.3.3.5. L ← L + 1.
   2.4. w ← t[i].
   2.5. t[i] ← t[K].
   2.6. t[K] ← w.
3. End.

أي عندما نقرأ قيمة عنصر أصغر من الموجه نقف (K).

وعندما نقرأ قيمة عنصر أكبر من الموجه نقف (L).

هنا نبدل بين t[L] و t[K] بشرط (L<K).

اما اذا تجاوز L الـK هنا نبدل بين t[K] والعنصر الموجه ويصبح دليل الموجه K

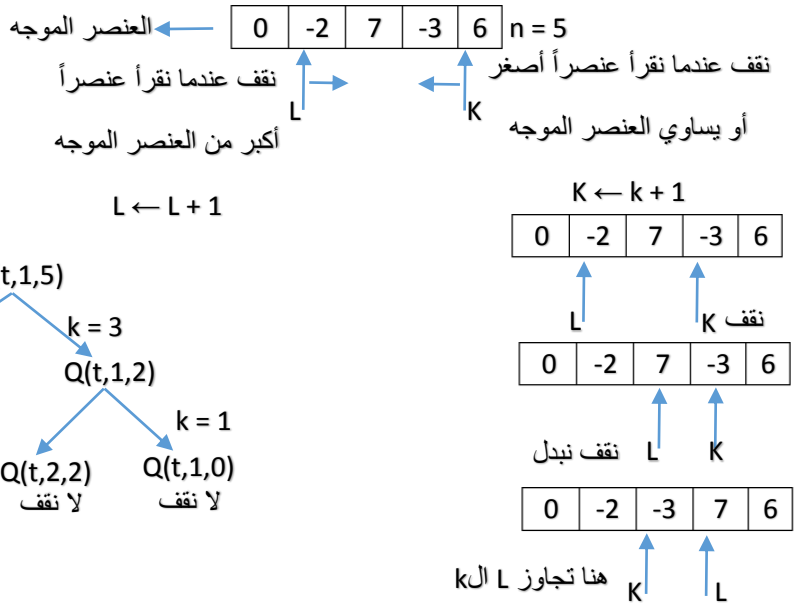
اما الخوارزمية الأساسية فتعطى كما يلي:

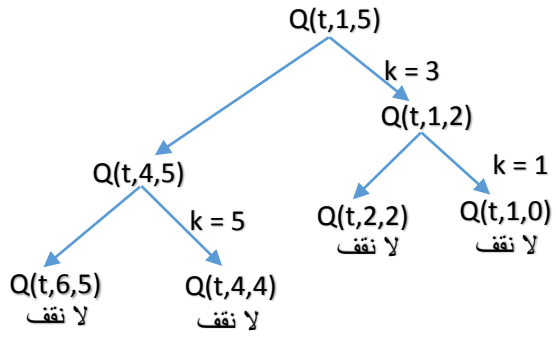
1. Start.
2. function Quicksort(t : array , i,j : int).
   2.1. if(i < j).
      2.1.1. partition(t,i,j,k).
      2.1.2. Quicksort(t,i,k-1).
      2.1.3. Quicksort(t,k+1,j).
3. Print t.
4. End.

حيث أن:

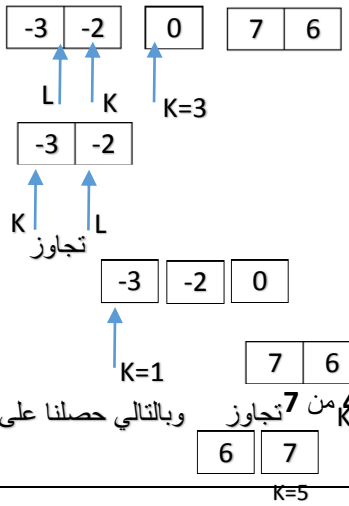
i : دليل أول عنصر وهو العنصر الموجه.
j : دليل آخر عنصر ويساوي حجم المصفوفة t.
t : المصفوفة ذات الحجم n المراد ترتيبها.
k : دليل العنصر المراد تبديله مع العنصر الموجه بحيث يحدث التقسيم عند التبديل.

مثال: طبق خوارزمية الترتيب السريع لترتيب هذه المصفوفة





نبدل بين الموجه و t[k]



وبالتالي حصلنا على المصفوفة المرتبة

-3	-2	0	6	7

امثلة: طبق خوارزمية الترتيب السريع على المصفوفات التالية

1. n = 4 | 3 | -2 | 0 | 1 |

2. n = 5 | 5 | 7 | 2 | 0 | -2 |

نفذ خطوة خطوة

خوارزميات البحث

تعتبر عملية البحث ضمن المصفوفات (أو ضمن الملفات) من المسائل الهامة في الخوارزميات وذلك لكثرة تطبيقاتها.

1. خوارزمية البحث التسلسلي:

إن ابسط طريقة للبحث عن عنصر $\boxed{X}$ ضمن سلسلة (أو مصفوفة) هو مقارنة العنصر X بجميع عناصر السلسلة

ما أن وجد العنصر ضمن السلسلة نعود بدليل (index) ذلك العنصر وإلا نعود بالدليل يساوي 1-

وبالتالي نستطيع التعبير عن هذه الخوارزمية بالشكل التالي:

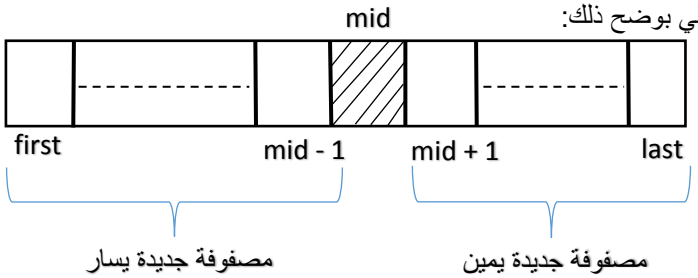
```
1. Start.
2. function Search_list(t : array , X : anytype , n : int).
    2.1.    for(i ← 1 ; i ≤ n ; i ← i + 1).
        2.1.1.  if(X = t[ i ])
            2.1.1.1.    return i.
    2.2.    i ← -1.
    2.3.    return i.
3. End.
```

2. خوارزمية البحث الثنائي (Binary search).

تعتبر هذه الخوارزمية من الخوارزميات التي تعتمد على استراتيجية التقسيم والتجميع (فرق - تسد)
تقوم هذه الخوارزمية بالبحث في مصفوفة مرتبة عن عنصر ما X فيما إذا كان موجوداً ام لا.
حيث تعود بموقع العنصر إذا كان موجوداً وإلا فإنها ترجع بالقيمة 1- لتدل على عدم وجود هذا العنصر.

الشكل التكراري لخوارزمية البحث الثنائي:
ليكن mid دليل العنصر الموجود في منتصف الشعاع أو المصفوفة و first دليل أول عنصر و last دليل آخر عنصر ولتكن المصفوفة مرتبة تصاعدياً عندئذٍ يمكن أن نعبر عن الخوارزمية بالشكل التالي:
نحسب دليل العنصر الموجود بالمنتصف mid ونقارن قيمته مع X (العنصر المراد البحث عن وجوده في المصفوفة) فإذا كان يساويه نكون بالتالي قد وجدنا العنصر X نرجع دليله وهو دليل عنصر المنتصف.
وإلا : إذا كان X < A[mid] بالتالي نبحث عنه في الجهة اليسارية من المصفوفة ونتابع بنفس الأسلوب.
وإلا : إذا كان X ≥ A[mid] عندئذٍ نبحث عنه في الجهة اليمينية من المصفوفة ونأخذ شعاع جديد للبحث وهكذا.....
حتى نجد العنصر أو لا نجده ونعود بالدليل 1-

والشكل التالي بوضح ذلك:



1. Start.
2. function Bsearch(A : array , x : anytype , n : int)
   2.1. first ← 1.
   2.2. last ← n.
   2.3. while(first < last).
      2.3.1. mid ← (first + last) div 2.
      2.3.2. if(x = A[mid]).
         2.3.2.1. return index_element ← mid.
      2.3.3. else if (x < A[mid]).
         2.3.3.1. last ← mid – 1.
      2.3.4. else.
         2.3.4.1. first ← mid + 1.
   2.4. index_element ← -1.
   2.5. return index_element.
3. End.

الشكل العودي لخوارزمية البحث الثنائي:

كما هو ملاحظ أن هذه المسألة أو الخوارزمية هو تعريف تعاودي لذلك نكتب هذه الخوارزمية بالشكل التعاودي

1. Start.
2. function Bsearch(A : array , first , last : int , x : anytype).
   2.1. mid ← (first + last) div 2.
   2.2. if(first > last).
      2.2.1. return -1.
   2.3. else if(x = A[mid]).
      2.3.1. return index_element ← mid.
   2.4. else if(x < A[mid]).
      2.4.1. Bsearch(A, first, mid-1, x).
   2.5. else if(x > A[mid]).
      2.5.1. Bsearch(A, mid+1, last, x).
3. End.

مثال: طبق خوارزمية البحث الثنائي على المصفوفة المرتبة التالية مع رسم شجرة البحث