المحاضرة الثامنة

طرق تصميم الخوارزميات

أ- استراتيجية فرق- تسد (Divide-and-Conquer)

الفكرة لحل مسألة كبيرة هو تقسيمها الى مجموعة مسائل جزئية وحل كل منها، ثم توحيد هذه الحلول للحصول على حل المسألة الاصلية. عادة تكون للمسائل الجزئية المولدة امثلة اصغر للمسألة الاصلية ويمكن حاها تداخليا باستعمال استراتيجية فرق- تسد.

س) متى تكون هذه الطريقة مناسبة؟
ج) عندما تكون للمسائل الجزئية نفس نوع(طبيعة) المسألة الاصلية.

وفيما يلي تجريد ضبطي (control abstraction) لهذه الاستراتيجية.

```
Type  DAndC(P){
   if  small (P)  return  S(P);
   else{
     Divide  P  into smaller instances P1, P2,…, Pk, k>1;
     Apply  DAndC  to each of these subproblems;
     Return  Combine ( DAndC(P1), DAndC(P2),…, DAndC(Pk));
   }
}
```

تعقيدات وقت استراتيجية فرق- تسد

اذا كان حجم المسألة P هو n واحجام المسائل الجزئية التي عددها k هو $n_1, n_2, \ldots, n_k$ على الترتيب، فان وقت احتساب DAndC يوصف بالعلاقة التالية:-

$$t(n) = \begin{cases} g(n) & , \text{ if n small} \\ t(n_1) + t(n_2) + \ldots + t(n_k) + f(n) & , \text{ otherwise} \end{cases}$$

حيث t(n) هو زمن احتساب DAndC لاي مدخلات ذات حجم n.
g(n) هو زمن احتساب الاجابة للمسألة الصغيرة.
f(n) هو زمن تقسيم المسألة P الى مسائلها الجزئية و/ أو توحيدها.

ملاحظة/ يجب تجنب استخدام طريقة فرق- تسد في الحالتين التاليتين:
١) مسألة ذات حجم n يتم تقسيمها الى اثنين او اكثر من المسائل الجزئية كل ذات حجم n تقريبا(مثلا مسألة فيبوناتشي).
٢) مسالة ذات حجم n يتم تقسيمها الى n من المسائل الجزئية تقريبا كل ذات حجم (n/c) (حيث c هو ثابت).

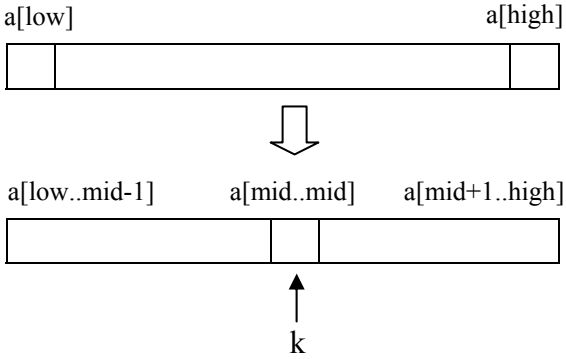
الاولى تقود الى خوارزميات اسية، اما الثانية فتقود الى خوارزميات ذات تعقيدات $n^{\Theta(\log n)}$.

واجب/ باستخدام طريقة فرق- تسد، اكتب خوارزمية تداخلية لحل مسألة ايجاد العنصرالاكبر في مصفوفة احادية البعد عدد عناصرها n.

امثلة على استراتيجية فرق- تسد

١) البحث الثنائي (Binary search)

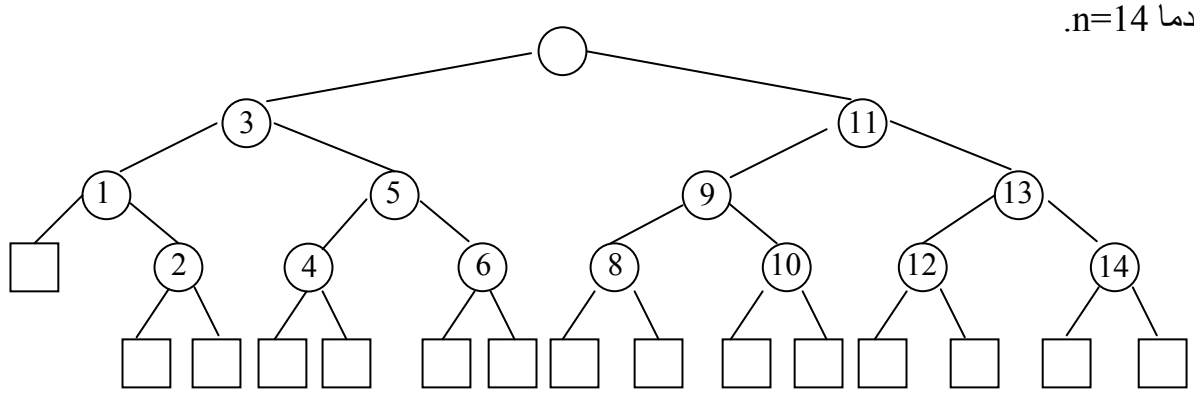
للبحث عن عنصر ما k في قائمة مرتبة a[1..n].



الفكرة/ للبحث عن k في القائمة a[low.high] فاننا نبحث عن k في ثلاثة قوائم جزئية a[low..mid-1] و a[mid..mid] و a[mid+1..high] وبمقارنة k مع a[mid] يمكن حذف اثنين من هذه القوائم الجزئية.

```
Algorithm  BinarySearch(A, x, low, high):
Input:An array A[1..n] of n elements sorted in nondecreasing
      order, an element x, and integers low and high.
Output: An index j if x=A[j], 1 ≤ j ≤ n , and 0 otherwise.
   1. if  low=high then
   2.    if x=A[low] then return low
   3.    else return 0
   4.    endif
   5. else
   6.    mid ← ⌊(low + high)/2⌋
   7.    if x=A[mid] then return mid
   8.    else if x<A[mid] then
   9.      return BinarySearch(A, x, low, mid-1)
   10.   else return BinarySearch(A, x, mid+1, high)
   11. endif
```

يمكن وصف سلوكية (اداء) خوارزمية البحث الثنائي من خلال شجرة القرارات(decision tree) والشكل التالي يبين ذلك عندما n=14.



تعقيدات الخزن: كل تنشيط تداخلي يحتاج $\Theta(n)$ من الخزن وبما ان عمق التداخل الاقصى(حالة النجاح) هو $O(\log n)$ لذلك فان:

$$S_{BinarySearch}(n)=O(\log n)$$

تعقيدات الوقت:

في الحالة الاسوأ للبحث الناجح:

$$t(n)=\begin{cases} t(1) & , \quad n=1 \\ t(n/2)+c & , \quad otherwise \end{cases}$$

$$t(n)=t(n/2)+c$$
$$=t(n/2^2)+2c$$
$$=t(n/2^3)+3c$$
.
.
.
$$=t(n/2^m)+mc$$

وبوضع $n=2^m$ وباخذ log للطرفين نحصل على $m=\log n$

$$=t(1)+c \log n$$
$$t(n)= \Theta(\log n)$$

الخلاصة/

للبحوث الناجحة

الافضل	المتوسطة	الاسوأ
$\Theta(1)$	$\Theta(\log n)$	$\Theta(\log n)$

للبحوث الفاشلة

الافضل	المتوسطة	الاسوأ
$\Theta(\log n)$		