خوارزميات وبنى معطيات 2

للدكتور غيث بلال

المحاضرة الثالثة

جَـامعة
المَـنارة
MANARA UNIVERSITY

الخوارزميات العودية

Recursive Algorithms

1. إن الدالة العودية : هي الدالة التي تستدعي نفسها بقيم مختلفة للوسيط الذي يمثل بعد او عمق الخوارزمية (عدد مرات التكرار).

- حل المسائل بالطريقة العودية يمكننا من إيجاد حلول طبيعية وبسيطة لهذه المشاكل مقارنة مع الحلول بطرق أخرى.

2. الخواص التي تتمتع بها المسائل التي تحل بمفهوم العودية

a– أن يكون للمسألة حل واحد أو أكثر غير عودي يدعى حالة التوقف.

b– يمكن تخفيض حجم الحالات الأخرى للمسألة باستخدام التعاود إلى مسائل مشابهة وأقرب إلى حالات التوقف.

c– أو تخفيض المسألة في النهاية إلى حالات التوقف فقط والتي تتمتع بحل سهل نسبياً.

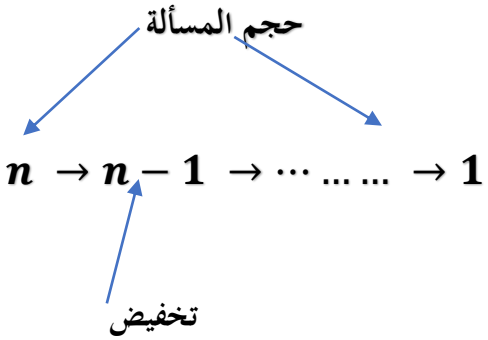
3. الشكل العام للخوارزمية العودية

أن الخوارزمية العودية تتألف بشكل عام من عبارة *if* الشرطية على الشكل التالي:

1– إذا تم التوصل إلى حالة التوقف عندئذٍ

حد المسألة عند هذه الحالة

2– وإلا خفض المسألة باستخدام التعاود.



4. امثلة عن تعاريف ومسائل عودية:

1- تعريف الأعداد الطبيعية

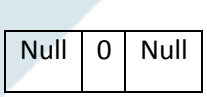
a- الصفر هو عدد طبيعي

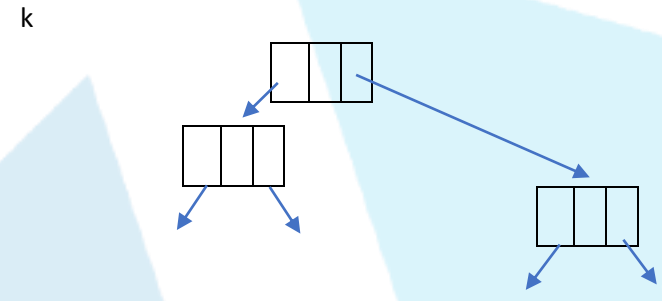
b- إذا كان n عدداً طبيعياً فإن $n+1$ هو عدد طبيعي.

2- تعريف بنية الشجرة الثنائية:

a- الشجرة 0 لا تحوي أية عقدة نسميها الشجرة الفارغة.

b- إذا كانت A و B شجرتين ثنائيتين فإن (k,A,B) هو شجرة ثنائية جذرها k وشجرتها الجزئية اليسارية A وشجرتها الجزئية اليمينية B.





3- تعريف العاملي $y = n!$

a- $0! = 1$

b- إذا كان $n > 0$ فإن $y = n! = n(n-1)!$

4- تعريف متتالية فيبوناشي Fibonacei

a- $Fib(1) = 1 , Fib(0) = 0$

b- إذا كان $n > 1$ فإن $Fib(n) = Fib(n-1) + Fib(n-2)$

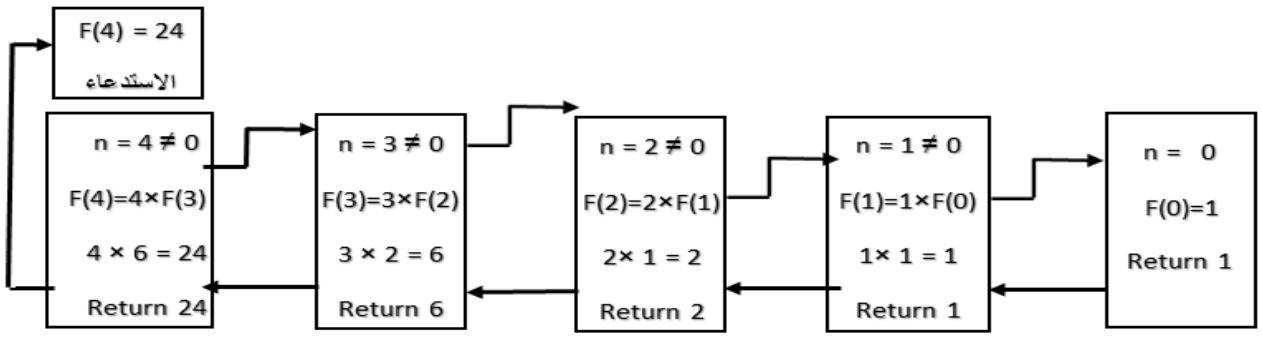
ملاحظة: عندما يتم تنفيذ الخوارزميات العودية على الحاسب يجب أن يكون عمر العودية صغير ومنتهي (عدد مرات التكرار ...) وذلك لضمان عدم طفح مكدس التدفيق وتوقف البرنامج فوراً قبل الانتهاء من التنفيذ لأن كل استدعاء عودي يتطلب تخزين وسائط التابع ومتحولاتها الموضعية وحالتها من استدعاء لآخر.

امثلة : حساب $y = n!$

```
1. Start.
2. function factor( int : n ).
   2.1. if( n = 0 ).
      2.1.1. return 1.
   2.2. else.
      2.2.1. return n * factor(n – 1 ).
3. End.
```

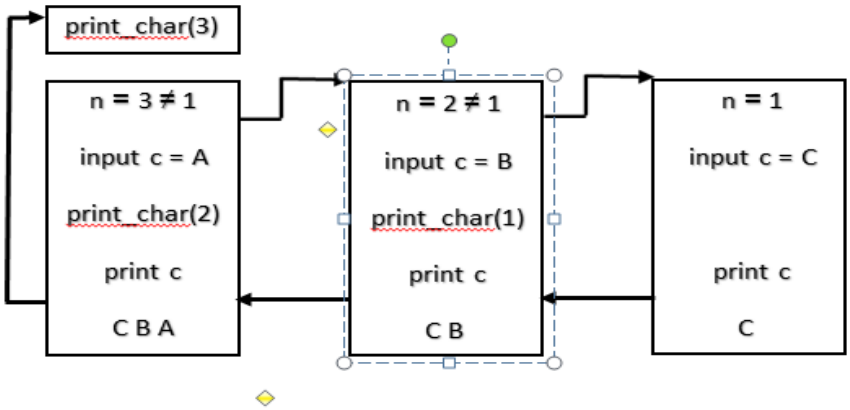
مثال 2 : فرض انه طلب من المستخدم ادخال مجموعة من المحارف من لوحة المفاتيح والمطلوب تصميم تابع عودي يقوم يطباعة هذه المحارف على الشاشة بالترتيب العكسي لإدخالها.

1. **Start.**
2. **function print_char(int : n).**
   - 2.1. **if(n = 1).**
     - 2.1.1. **input c.**
     - 2.1.2. **print c.**
   - 2.2. **else.**
     - 2.2.1. **input c.**
     - 2.2.2. **print_char(n – 1).**
     - 2.2.3. **print c.**
3. **End.**



كما نلاحظ أننا نقوم بعدد مرات يساوي عمق التعاودية إلى أن نصل إلى حالة معلومة هي $0! = 1$ ثم نعود بالاتجاه المعاكس حتى نصل إلى المطلوب

تنفيذ المثال الثاني:



نلاحظ بأن عملية الطباعة الخطوة (2.2.3) لا تنفذ إلا بالعودة من التكرارات أو التعاود وبالتالي يتم تنفيذ عملية الطباعة من آخر عنصر إلى أول عنصر لان خطوة الطباعة تلي الاستدعاء التعاودي أما الإدخال فيتم قبل الاستدعاء التعاودي .

امثلة:

- اكتب خوارزمية عودية لحساب جداء عددين $n \times m$ عن طريقة الجمع التكراري.

- اكتب خوارزمية عودية لحساب وطباعة قيمة $y = a^n = a \times a \times \dots\dots\dots \times a$

حيث أن n عدد صحيح موجب فقط.

خوارزميات الترتيب العودية

1. خوارزمية الترتيب أو الفرز بالدمج (merge sort).

تعتمد هذه الخوارزمية على استراتيجية التقسيم والتجميع وتتمثل استراتيجية الترتيب (الفرز) المطبقة فيها بتقسيم الشعاع (المصفوفة) A نصفين R,L حيث L يرمز للنصف اليساري و R للنصف اليميني وبعدئذٍ ترتيب كل من R,L على حدا ثم إعادة دمجهما مع بعض للحصول على الشعاع A المرتب وهكذا يمكن التعبير عن هذه الخوارزمية كما يلي.

1. قسم الشعاع (المصفوفة) A[1,n] إلى نصفين R,L .
2. رتب النصف R اليميني مع تكرار التقسيم.
3. رتب النصف L اليساري مع تكرار التقسيم.
4. ادمج R,L مع بعضهما لترتيب عناصر المصفوفة A .

مثلاً: المصفوفة A=[5,3,8,6,7,2,4,1] ترتب وفق الخوارزمية كما يلي:

1. نقسم المصفوفة إلى شعاعين

L=[5,3,8,6] , R=[7,2,4,1]

2. ترتيب كل من R و L

L=[3,5,6,8]

R=[1,2,4,7]

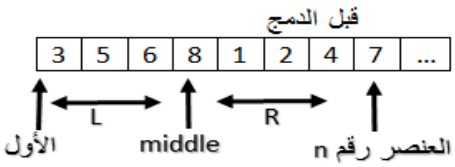
3. دمجهما في مصفوفة مرتبة

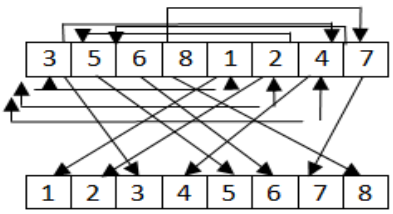
merge(L,R)=[1,2,3,4,5,6,7,8].

ولكن كيف تتم عملية الدمج؟

إن هذه العملية تتم بالبدء بأخذ شعاع (مصفوفة) فارغة حجمها يساوي حجم المصفوفة الأصلية ثم البدء بمقارنة كل عنصر من L مع العنصر المقابل له في R وننقل العنصر الأصغر بينهما إلى المصفوفة الجديدة الفارغة وهكذا.....

حتى ننقل جميع العناصر إلى هذه المصفوفة





إذن ال middle يعين دليل آخر عنصر بالقسم اليساري وبالتالي باستخدام فكرة التعاودية يمكن حل المسألة بسهولة وبالتالي نطبق الترتيب بالدمج على كل جزء اليساري واليميني حتى نصل إلى الحالة الأخيرة وهي الجزء اليساري مرتب والجزء اليميني مرتب ثم ندمجهما في مصفوفة واحدة. حيث تعطى الخوارزمية بطريقة شبه الترميز كما يلي حيث أن first دليل أول عنصر و last دليل آخر عنصر

1. Start.
2. Function mergesort(A,first,last)
3. if(first < last).
   3.1. middle ← (first + last) / 2.
   3.2. mergesort(A,first,middle).
   3.3. mergesort(A,middle + 1, last).
   3.4. merge(A, first, middle, last)
4. Print A.
5. End.

كما نلاحظ أن هذه الخوارزمية هي خوارزمية تعاودية ولكن هنالك أكثر من نداء تعاودي.

أما التابع الذي يصف علمية الدمج فيعطى بالخوارزمية التالية
حيث أن A المصفوفة الأساسية و temp مصفوفة مساعدة لإنجاز الترتيب.

1. Start.
2. function merge(A : array , first, middle, last).
   2.1. L ← first.
   2.2. R ← middle + 1.
   2.3. t ← first.
   2.4. while(L ≤ middle and R ≤ last).
      2.4.1. if(A[L] < A[R]).
         2.4.1.1. temp[t] ← A[L].
         2.4.1.2. L ← L + 1.
      2.4.2. else.
         2.4.2.1. temp[t] ← A[R].
         2.4.2.2. R ← R + 1.
      2.4.3. t ← t + 1.
   2.5. while(L ≤ middle).
      2.5.1. temp[t] ← A[L].
      2.5.2. L ← L + 1.
      2.5.3. t ← t +1.
   2.6. while(R ≤ last).
      2.6.1. temp[t] ← A[R].
      2.6.2. R ← R + 1.
      2.6.3. t ← t + 1.
   2.7. for(t ← first ; first ≤ last ; t ← t + 1).
      2.7.1. A[t] ← temp[t].
3. End.