المحاضرة الحادية عشرة

نظري ☑ عملي ☐

◄ دكتور المادة: الدكتور سعيد جبعز

◄ عنوان المحاضرة: برامج بلغة الجافا

أهلاً بكم في المحاضرة الحادية عشرة من مقررنا البرمجة والخوارزميات، والتي قام الدكتور فيها بحل برنامجين بلغة جافا :

البرنامج الأول :

اكتب صفاً لحساب !n بطريقتين تكرارية وعودية ثم استخدم هذا الصف في كتابة برنامج لحساب وطباعة الأعداد من 1 حتى n (حيث n عدد طبيعي مدخل) يحسب العاملي للأعداد الفردية بشكل عودي وللأعداد الزوجية بشكل تكراري.

```
class [1] Factorial { static [2] long fact (int m)
{ long r=1;
for (int i=2; i<=m; i++)
r*=i;
return r; }                                  [3]
static long [4] facR (int m) {
if (m==0) return 1;                          [5]
else return (m * FacR(m-1)); } }
class [6] Usefactorial { public static void main (String arg[])
{ int n;
do {n=stdin.readInt(); } while (n<=1);       [7]
for (int i=1; i<=n; i++)
if (i%2 != 0) System.out.println(Factorial.FacR(i));   [8]
else System.out.println(Factorial.Fact(i)); } }        [9]
```

1 صف اسميناه Factorial لننتبه جيداً لأسماء الصفوف والدوال عند استدعائها يجب أنه يكون نفس الاسم عند الاستدعاء والانتهاء لفظاً أي يجب مراعاة أحرف صغيرة والكبيرة)

2 دالة اسمها fact لحساب العاملي بشكل تكراري.

3 نصرح عن متحول r نحمل قيمة (1) حيادي الضرب كقيمة ابتدائية ثم حلقة for للمرور على جميع الأعداد من 2 حتى العدد المراد حساب عامله وبعد ثم ضرب هذه العناصر حتى يختل الشرط فيخرج من حلقة for ويعيد قيمة الناتج (r).

4 دالة اسمها fact لحساب العاملي بطريقة عودية.

5 حساب العاملي بالطريقة العودية: اذا كان العدد المدخل يساوي الصفر يرجع واحد، الا يرجع العدد × (ناتج استدعاء الدالة fact للعدد (n-1)) وهكذا حتى الوصول للقيمة صفر فيعيد الواحد وتنتهي الدالة من عملها

6 صف لاستدعاء الصف السابق اسمه Usefactorial

7 نصرح عن متحول n من نوع صحيح - حلقة do...while لإعادة إدخال العدد n طالما أصغر أو يساوي الواحد (أي غير محققة لشرط أنه طبيعي أكبر من الواحد).

8 حلقة for للمرور على جميع العناصر ابتداءً من 1 حتى n العدد المدخل
ثم شرط اذا كان العدد المدخل زوجي نحسب العاملي بالطريقة العودية لذلك استدعينا ~~...~~ دالة fact (عن طريق اسم الصف كونه وضعنا قبل الدالة fact الكلمة static) ضمن تعليمة الطباعة لتطبع قيمة الناتج من هذه الدالة.

9 اذا لم يتحقق شرط أن العدد المدخل زوجي (فردي) ومنه نطبع قيمة الدالة fact

البرنامج الثاني:

اكتب صف يمثل مجموعة أعداد صحيحة كعضو بياني خاص ويقوم بالبحث عن عنصر مدخل فيها ثم باستخدام الصف السابق اكتب برنامج يقوم بإدخال مجموعة أعداد صحيحة من البعد n حيث n عدد طبيعي مدخل أكبر أو يساوي الـ(2) والبحث عن عدد x فيها فيما اذا كان موجود أم لا.

```
class A {          [1]
private int B[];   [2]
```

```
public A (int n) {B = new int [n]; }                        3
public void read () {
  for (int i = 0; i < B.length; i++)                        4
      B [i] = Stdin.readInt (); }
public boolean find (int a) {
  for (int i = 0 ; i < B.length; i++)
  if (B [i] = = a) return (True);                           5
  return (false); }
                } → A قوس انتهاء الصف
class A2 {
public static void main (String args[ ]) {
int n;
do { n= Stdin.readInt (); } while (n < 2);                  6
A aa = new A(n);                                            7
  aa.read B ();                                             8
int x;
 x = Stdin.readInt ();                                      9
System.out.print (aa.find (x)); } }
```

أو نستبدل هذه التعليمة بالتعليمة التالية :

```
if (aa.find (x) = = True) System.out.print (
   x + " is found");                                        10
else   System.out.print (x + " is not found"); } }.
```

11 صمم الصف A ليمثل متجهة.

2) تصريح عن متجهة أعداد صحيحة لعضو بياني خاص (باستخدام محدد وصول private)

3) باني لبناء (حجز) متجهة سيكون له وسيط الحجم للمتجهة

4) الدالة اسميناها read نوع إرجاعها void لأننا نريد منها قراءة (إدخال) عناصر نريد من هذه الدالة اسناد قيم للمتجهة حيث أن المتجهة من نوع خاص وهذا يعني أننا لن نستطيع الوصول لها من خارج الصف أي أننا ضمن الصف الثاني لن نستطيع أن نعطي المتجهة قيم بشكل مباشر لذلك نستخدم هذه الدالة العامة.

5) دالة عامة من نوع بولياني اسمها find للبحث عن عنصر فيما إذا كان موجود أو لا تأخذ وسيط واحد وهو العدد المراد البحث عنه وحلقة for للمرور على جميع عناصر المتجهة وإذا كان العدد موجود ترجع القيمة True وإذا لم يتحقق شرط نخرج من الحلقة لأول مرة تليها دون إعادة قيمة false.

6) تصريح عن متحول صحيح n وحلقة do - while لإعادة إدخاله طالما أنه أصغر من 2.

7) متحول aa من الصف A ليصرح (يحجز) عن متجهة من البعد n.

8) استدعاء الدالة read عن طريق المتحول aa لقراءة عناصر متجهة (إدخال).

9) التصريح عن متحول x وإدخاله (العدد المراد البحث عنه)

aa.find(x) : أي استدعاء الدالة من أجل المتحول x وسترجع قيمة.

System.out.print(aa.find(x)) طباعة قيمة راجعة من الدالة find.

10) طريقة ثانية بدل 9:

نختبر إذا كانت القيمة True يطبع أنه موجود - وإلا يطبع أنه غير موجود.

انتهت المحاضرة

مع التمنيات بالتوفيق

♡.....

المحاضرة الثانية عشرة

نظري ☑ عملي ☐

◄ دكتور المادة: سمير جعفر

◄ عنوان المحاضرة: كلفة الخوارزميات

- عند البحث عن خوارزمية لحل مسألة معينة من المحتمل أن نصادف عدداً كبيراً من الخوارزميات التي يمكنها حل هذه المسألة، وإذا وجدت أكثر من خوارزمية فيجب أن نكون قادرين على اختيار واحدة من هذه الخوارزميات لاستخدامها في حل المسألة، وهذا الاختيار مبني على معيار واضح، كأن نختار أسرع خوارزمية أو أدق خوارزمية في إعطاء النتائج أو أقلها حاجة لمساحة تخزينية.
- نسمي المعيار الأول ((الأسرع)) بالتعقيد الزمني للخوارزمية، والمعيار الأخير ((مساحة التخزين)) بالتعقيد التخزيني.
- حاسوبياً هناك محوران: القدرة التخزينية والزمن.

الحواسيب تملك قدرة تخزينية محدودة، فإن الخوارزمية التي تحتاج لعدد كبير من العمليات لحل مسألة معينة سوف تنفذ ببطء يتناسب مع عدد العمليات.

كما أنه هنالك خوارزميات لا يمكن تنفيذها حاسوبياً لأنها تحتاج لتنفيذ عدد كبير جداً من العمليات مما يجعل زمن تنفيذها كبيراً جداً لا يمكن للحاسوب تحمله.

الخوارزمية: هي مجموعة من الخطوات ((التعليمات)) المتسلسلة المنطقية البسيطة والمرتبة والمنتهية التي تؤدي لحل مسألة ما.

كلفة الخوارزمية:

بالنسبة للذاكرة: نحسب الكلفة من عدد المتحولات (المتغيرات) المستخدمة × حجم كل متحول.

بالنسبة للزمن: نحسب الكلفة من عدد العمليات (التعليمات) الموجودة بدلالة حجم المسألة، وعادةً ما نرمز له بـ n، فتكون كلفة الخوارزمية عبارة عن تابع بدلالة n $\boxed{T(n)}$

((دراستنا للكلفة تكون مستقلة عن الحاسب وطبيعة مكوناته)).

الزمن في أحسن الأحوال Best case :

نفرض لدينا خوارزمية A وحجم المسألة n ، ونفرض D هي مجموعة جميع المعطيات الممكنة :

$$min_A = \min\{ cost_A(d_i) \quad ; d_i \in D \}$$

حيث $cost_A(d)$ هي كلفة الخوارزمية A من أجل المدخل d .

الزمن في أسوأ الأحوال Worst case :

$$max_A = \max\{ cost_A(d_i) \quad ; d_i \in D \}$$

الزمن الوسطي :

نفرض $p(d_i)$ هو احتمال اختيار المعطاة d_i ، فإن :

$$Average_A = \sum_{d_i \in D} p(d_i) \, . \, cost_A(d_i)$$

في حال تساوي احتمالات اختيار المعطيات :

$$Average_A = \frac{1}{Card(D)} \sum_{d_i \in D} cost_A(d_i)$$

حيث $Card(D)$ قدرة المجموعة D ، ونفرض قدرة المجموعة $= n$:

$$Average_A = \frac{1}{n} \sum_{i=1}^{n} Cost_A(d_i)$$

- لحساب كلفة خوارزمية نقوم بعدّ العمليات الموجودة في تعليماتها .
- نفرض أنّ $p(x)$ هو عدد العمليات في التعليمة x ، عندها :

$$p(x_1 ; x_2) = p(x_1) + p(x_2)$$

$$p\left(\sum_i x_i\right) = \sum_i p(x_i)$$

$$p(\text{if}(A) \text{ then } B \text{ else } C) \leq p(A) + \max(p(B), p(C))$$

((عدد العمليات الأكبر بين B, C)) ((عدد العمليات المكونة للشرط))

حيث A شرط منطقي ، B, C مجموعتان من التعليمات .

- في حال وجود دالة (غير عودية):

الكلفة = عدد الاستدعاءات للدالة x كلفة الاستدعاء الواحد.

- لتكن لدينا الخوارزمية:

```
int i = 1;
while ((i<=n) and (L[i] != x))
    i = i + 1;
if (i>n) i = 0;
```

→ A

هذه الخوارزمية تقوم بالبحث عن دليل أول ظهور للعنصر x في المتجهة L ، حيث L تحوي n عنصر.

- زمن التنفيذ في أحسن الأحوال: بالنسبة لعمليات المقارنة من الشكل $L[i] != x$:

$min_A = 1$ ((أفضل الأحوال هو ظهور x كعنصر أول في المتجهة))

- زمن التنفيذ في أسوأ الأحوال:

$max_A = n$ ((ظهور x كعنصر أخير، أو عدم وجوده في المتجهة))

- الزمن الوسطي:

لتكن D مجموعة كل المتجهات من البعد n:

نقسم D إلى مجموعتين: $D_{n,0}$: مجموعة المتجهات التي لا يظهر فيها x.

$D_{n,i}$: مجموعة المتجهات التي يظهر فيها x في الخانة i

نفرض احتمالات اختيار المعطيات ((المتجهات)) من المجموعة D متساوية

ونفرض أن (q) هو احتمال ظهور x في متجهة ما ((بغض النظر عن موقعه))

وبالتالي (1-q) هو احتمال عدم ظهور x في أي متجهة.

نوجد الزمن الوسطي بالشكل:

$$Average_A = \sum_{d_i \in D} p(d_i) . cost_A(d_i)$$

$$= \sum_{d_i \in D_{n,0}} p(d_i) . cost_A(d_i) + \sum_{d_i \in D_{n,i}} p(d_i) . cost_A(d_i)$$

((بالاستفادة من التقسيم السابق للمجموعة D))

$$Average_A = \sum_{i=1}^{n} \underbrace{\left(\frac{1-q}{n}\right).n}_{\text{مقدار ثابت}} + \sum_{i=1}^{n} \underbrace{\left(\frac{q}{n}\right)}_{\text{مقدار ثابت}}.i$$

((مقدار ثابت))

((بجمعه مع نفسه n مرة يصبح لدينا:))

((مقدار ثابت))

$$= n.\left(\frac{1-q}{n}\right).n + \frac{q}{n}\underbrace{\sum_{i=1}^{n} i}_{\text{متسلسلة شهيرة}}$$

((متسلسلة شهيرة))

$$= (1-q)\,n + \frac{q}{n}.\frac{n(n+1)}{2}$$

$$= (1-q)\,n + \frac{q}{2}(n+1)$$

انتهت المحاضرة

المحاضرة الثالثة عشرة

◄ دكتور المادة: سمير جعفر

◄ عنوان المحاضرة: متابعة كلفة الخوارزميات + كلفة الخوارزمية العودية

نظري ☒
عملي ☐

- بالنسبة للدوال غير العودية الكلفة = عدد الاستدعاءات × كلفة الاستدعاء الواحد.
- عند حساب كلفة خوارزمية ← إما أن نحسبها بشكل عام بالنسبة لجميع العمليات.
  ← أو بالنسبة لعملية أساسية (مختارة).

مثال: احسب كلفة الخوارزمية التالية بالنسبة لعدد عمليات الجمع:

```
S = 0;
for(i = 0 ; i <= n ; i++)
    S = S + i ;
```

A

الحل:

- نريد الكلفة بالنسبة لعدد عمليات الجمع من الشكل S+i فقط:

عملية الجمع تتم فقط ضمن الحلقة for، ومنه عدد عمليات الجمع هو عدد مرات الدخول إلى الحلقة for، وهناك (n+1) دخول للحلقة for ((من) i=0 إلى i=n))

ومنه عدد عمليات الجمع من الشكل S+i هو $(1) \times (n+1)$

((كلفة الاستدعاء الواحد))

فيكون تابع الكلفة: $T(n) = n+1$

- أما بالنسبة لعمليات الجمع بشكل عام:

① عدد عمليات الجمع من الشكل $(S+i)$ هو $n+1$ كما شرحنا سابقاً.

② عدد عمليات الجمع من الشكل $(i++)$ هو $n+1$

ومنه تابع الكلفة:

$$T(n) = 2(n+1) = 2n+2$$

مثال: احسب كلفة الخوارزمية التالية بالنسبة لعدد عمليات الضرب، وماهي قيمة المتحول S بعد انتهاء تنفيذ الخوارزمية.

```
A ← [ S = 1;
      while (n > 1) do [ S = S * n;
                         n = n / 2;
```

الحل:

$\circledast$ عدد عمليات الضرب = 1 × عدد مرات الدخول إلى الحلقة while

((كلفة الدخول الواحد))

قيم المتحول n	n	$\frac{n}{2}$	$\frac{n}{2^2}$	$\frac{n}{2^3}$	—	$\frac{n}{2^{i-1}}$
رقم الدخول (مرات الدخول)	1	2	3	4	—	i

وذلك بفرض أنّ i عدد مرات الدخول ⟸ $\frac{n}{2^i}$ لا يحقق شرط الحلقة

أي $\frac{n}{2^i} \leq 1$ ، وهي أول قيمة لا تحقق شرط الحلقة، إذاً:

$$\frac{n}{2^i} = 1 \longrightarrow n = 2^i$$

$$\log_2 n = i \log_2 2$$

$$\Longrightarrow \boxed{i = \log_2 n}$$

إذاً عدد مرات الدخول إلى الحلقة هو $\log_2 n$ ، وكلفة الخوارزمية بالتعويض في $\circledast$:

$$T(n) = 1 \times \log_2 n = \log_2 n$$

كلفة الخوارزمية العودية:

خوارزمية حساب $(n!)$ بالطريقة العودية (دالة لحساب العاملي):

```
int f(int n){
  if (n==0) return 1;
  else return(n * f(n-1)); }
```

كلفة هذه الخوارزمية بالنسبة لعدد عمليات الضرب هي:

الكلفة = عدد عمليات الضرب في الاستدعاء الواحد × عدد مرات الاستدعاء

$$= 1 \times n$$

$$\Longrightarrow \boxed{T(n) = n}$$

- نرمز لعدد مرات استدعاء الدالة لنفسها بالعمق العودي.
- نفرض أنّ عدد عمليات الضرب المنجزة في التابع $f(n)$ هو $T(n)$ فالكلفة:

$$T(n) = 1 + T(n-1)$$

((لأنه لدينا عملية ضرب وحيدة منجزة من أجل القيمة n، بالإضافة إلى عدد عمليات الضرب المنجزة من أجل القيمة $n-1$)) بالمتابعة بنفس الطريقة:

$$T(n) = 1 + (1 + T(n-2))$$

$$= 2 + T(n-2)$$

$$= 2 + (1 + T(n-3)) = 3 + T(n-3)$$

وهكذا حتى نصل إلى i وهو عدد مرات الاستدعاء:

$$T(n) = i + T(n-i)$$

- وفي الحالة الموجودة لدينا عدد مرات الاستدعاء هو n (أي $n = i$) إذاً:

$$T(n) = n + T(0) = n + 0 = n$$

- في حالة الخوارزمية التكرارية لحساب العاملي:

```
int f1 (int n){
  int r=1;
  for(int i=0 ; i<=n ; i++)
     r=r*i;
  return r; }
```

① كلفة الخوارزمية بالنسبة للزمن لن تختلف عن العودية:

$$T_{f_1}(n) = n$$

② كلفة الخوارزمية بالنسبة للذاكرة:

في حال الخوارزمية العودية:

$$S_f(n) = n \times \text{size of (int)}$$

$$= 4n \text{ Bytes}$$

أي أنها تحتاج $4n$ بايت في الذاكرة.

أما حالة التكرارية:

$$S_{f_1}(n) = 2 \times \text{size of (int)}$$
$$= \underline{2} \times 4 = 8 \text{ Bytes}$$

((لأننا في الخوارزمية التكرارية احتجنا لحجز متغيرين في الذاكرة فقط))

إذاً تحتاج إلى 8 بايت في الذاكرة، أي أنّ كلفة الخوارزمية العودية أكبر من التكرارية، وهذا يعني أنّ الخوارزمية التكرارية أفضل بكثير من العودية من حيث استهلاك الذاكرة.

مثال: حساب x^n حيث $n \in N^*$ بطريقة عودية:

```
int p(int x, int n) {
  if (n==0) return 1;

  else return (x * p(x, n-1)); }
```

لإيجاد الكلفة (بالنسبة لعمليات الضرب):

$$T(n) = 1 + T(n-1)$$
$$= 1 + [1 + T(n-2)]$$
$$\vdots$$
$$= n + T(0) = n + 0 \Rightarrow \boxed{T(n) = n}$$

وبالنسبة لاستهلاك الذاكرة:

$$S(n) = 4n \text{ Byte}$$

أي تحتاج $4n$ بايت في الذاكرة، ونرى دوماً أنّ الخوارزمية التكرارية لحساب x^n أفضل.

- سنتابع في المحاضرة القادمة بكتابة الخوارزمية السابقة بطريقة تقلّل من كلفتها.

انتهت المحاضرة