

# Chapter 1

## Time Complexity

مذكرات شرح وتمارين محلولة،  
امتحانات سابقة للعديد من المواد أدناه  
متاحة مجاناً على الموقعين المذكورين

لاستلام نسخ إلكترونية من نوات الموقع مجاناً على إيميلك قم بزيارة [eng-hs.net](http://eng-hs.net)

النوتات متوفرة مجاناً في كل من تصوير الفرع أمام الهندسة أسفل صالون رنيم 24814916 أو تصوير الجمعية الرئيسية بالسرداب أسفل بيانو 24926388

النوتات متوفرة مجاناً بالموقعين [eng-hs.net](http://eng-hs.net), [eng-hs.com](http://eng-hs.com)

(1)

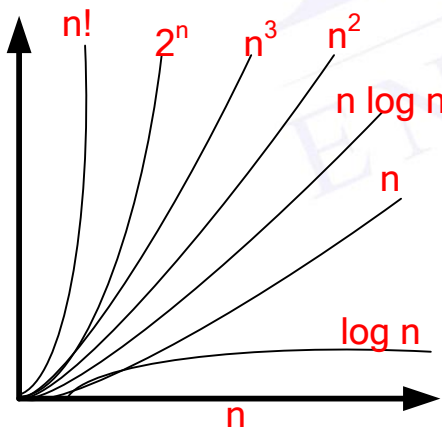
م. حمادة شعبان 9 4444 260 [info@eng-hs.com](mailto:info@eng-hs.com)

6. Reorder the following efficiencies from smallest to largest:

- $2^n$
- $n!$
- $n^5$
- 10,000
- $n \log_2(n)$

عند ترتيب وقت الدوال يجب الأخذ في الاعتبار ما يلي:

- أي رقم ثابت مهما كان كبيراً فهو  $O(1)$ .
- نقوم بإهمال **lower terms** عند المقارنة فمثلاً  $(7n^2 + 100n)$  نتعامل معها كأنها  $(7n^2)$  ونهمل  $(100n)$  لأنه في حال قيم  $n$  كبيرة تكون  $(7n^2 \gg 100n)$
- نقوم بإهمال **leading factors** فمثلاً  $7n^2 = O(n^2)$  وكذلك  $100n^2 = O(n^2)$
- فكأن كلاهما لهما نفس **efficiency**.
- عند ذكر  $\log n$  فإنها تعني  $\log_2 n$  ما لم يذكر خلاف ذلك.
- يمكن الاسترشاد بالرسم البياني الموضح.



answer:

- 10,000
- $n \log n$
- $n^5$
- $2^n$
- $n!$

المنافسة الحقيقية دائماً ما تكون بين ما قمت بفعله وما أنت قادر على فعله

لاستلام نسخ إلكترونية من نوات الموقع مجاناً على إيميلك قم بزيارة [eng-hs.net](http://eng-hs.net)

النوتات متوفرة مجاناً في كل من تصوير الفرع أمام الهندسة أسفل صالون رنيم 24814916 أو تصوير الجمعية الرئيسية بالسرداب أسفل بيانو 24926388

النوتات متوفرة مجاناً بالموقعين [eng-hs.net](http://eng-hs.net), [eng-hs.com](http://eng-hs.com)

(2)

م. حمادة شعبان 260 4444 9 [info@eng-hs.com](mailto:info@eng-hs.com)

7. Reorder the following efficiencies from smallest to largest:

- $n \log_2(n)$
- $n + n^2 + n^3$
- $2^4$
- $n^{0.5}$

عند ترتيب وقت الدوال يجب الأخذ في الاعتبار ما يلي:

• أي رقم ثابت مهما كان كبيراً فهو  $O(1)$ .

• نقوم بإهمال **lower terms** عند المقارنة فمثلاً  $(7n^2 + 100n)$  نتعامل معها كأنها  $7n^2$

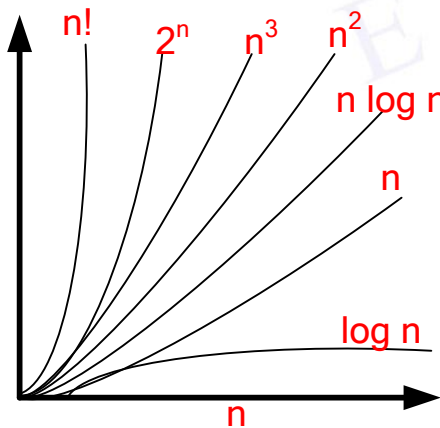
$(n^2)$  ونهمل  $(100n)$  لأنه في حال قيم  $n$  كبيرة تكون  $(7n^2 \gg 100n)$

• نقوم بإهمال **leading factors** فمثلاً  $7n^2 = O(n^2)$  وكذلك  $100n^2 = O(n^2)$  فكان

كلاهما لهما نفس **efficiency**.

• عند ذكر **logn** فإنها تعني  $\log_2 n$  ما لم يذكر خلاف ذلك.

• يمكن الاسترشاد بالرسم البياني الموضح.



**answer:**

- $2^4 = O(1)$
- $n^{0.5}$
- $n \log n$
- $n + n^2 + n^3$

وضع الللمسة الأخيرة على عملك  
بمثابة التاج الذي يكمل نجاحك

لاستلام نسخ إلكترونية من نوات الموقع مجاناً على إيميلك قم بزيارة [eng-hs.net](http://eng-hs.net)

النوتات متوفرة مجاناً في كل من تصوير الفرع أمام الهندسة أسفل صالون رنيم 24814916 أو تصوير الجمعية الرئيسية بالسرداب أسفل بيانو 24926388

النوتات متوفرة مجاناً بالموقعين [eng-hs.com](http://eng-hs.com), [eng-hs.net](http://eng-hs.net)

(3)

م. حمادة شعبان 260 4444 9 [info@eng-hs.com](mailto:info@eng-hs.com)

8. Determine the big-O notation for the following:

- $5n^{5/2} + n^{2/5}$
- $6\log_2(n) + 9n$
- $3n^4 + n\log_2(n)$
- $5n^2 + n^{3/2}$

عند ترتيب وقت الدوال يجب الأخذ فى الاعتبار ما يلي:

• أي رقم ثابت مهما كان كبيراً فهو  $O(1)$ .

• نقوم بإهمال **lower terms** عند المقارنة فمثلاً  $(7n^2 + 100n)$  نتعامل معها كأنها

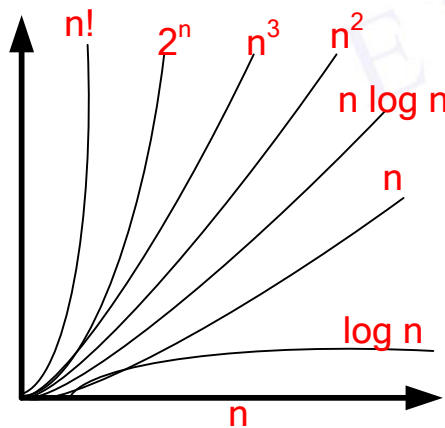
$(7n^2)$  و نهمل  $(100n)$  لأنه فى حال قيم  $n$  كبيرة تكون  $(7n^2 \gg 100n)$ .

• نقوم بإهمال **leading factors** فمثلاً  $7n^2 = O(n^2)$  وكذلك  $100n^2 = O(n^2)$  فكأن

كلاهما لهما نفس **efficiency**.

• عند ذكر **logn** فإنها تعنى  $\log_2 n$  ما لم يذكر خلاف ذلك.

• يمكن الاسترشاد بالرسم البياني الموضح:



**Answer:**

- $6 \log n + 9n = O(n)$
- $5n^2 + n^{3/2} = O(n^2)$
- $5n^{5/2} + n^{2/5} = O(n^{2.5})$
- $3n^4 + n \log n = O(n^4)$

إذا أردت أن تنجز عملاً بشكل جيد، فستجد وسيلة لإنجازه مهما كانت الظروف

لاستلام نسخ إلكترونية من نوات الموقع مجاناً على إيميلك قم بزيارة [eng-hs.net](http://eng-hs.net)

النوتات متوفرة مجاناً في كل من تصوير الفرع أمام الهندسة أسفل صالون رنيم 24814916 أو تصوير الجمعية الرئيسية بالسرداب أسفل بيانو 24926388

النوتات متوفرة مجاناً بالموقعين [eng-hs.net](http://eng-hs.net), [eng-hs.com](http://eng-hs.com)

(4)

م. حمادة شعبان 260 4444 9 [info@eng-hs.com](mailto:info@eng-hs.com)

9. Calculate the run-time efficiency of the following program segment:

```
1 i = 1
2 loop (i <= n)
  1 print (i)
  2 i = i + 1
3 end loop
```

عند طلب حساب الوقت المطلوب لتنفيذ أليجوريزم يجب مراعاة التالي:

- في حال عدم وجود loop أو recursion بالأليجوريزم فإن الوقت المطلوب هو  $O(1)$ .
- في حال وجود recursion فإنه يقوم مقام loop.
- الحد الأقصى لوقت algorithm يحتوى على loop واحدة هو  $O(n)$  ويمكن أن يكون أقل من ذلك كأن يكون  $O(\log n)$  أو  $O(1)$ ، لكن لا يمكن أن يكون  $O(n^2)$  مثلاً أو  $O(n \log n)$ .
- الحد الأقصى لوقت algorithm يحتوى على two nested loops هو  $O(n^2)$  ويمكن أن يكون أقل من ذلك كأن يكون  $O(n \log n)$  أو  $O(n)$  أو  $O(\log^2 n)$  أو  $O(\log n)$  أو  $O(1)$  والعبرة في ذلك هو شروط إيقاف loops.
- في حال loop counter يبدأ من 1 ويتضاعف كل مرة بضربه  $\times 2$  أو يبدأ من  $n$  ويتم قسمته في كل مرة على 2 فإن الوقت المطلوب لهذه loop هو  $O(\log n)$ .
- عدد الخطوات داخل loop لا يؤثر على big oh notation لأننا لا نهتم بالـ leading factor عند الحكم على أليجوريزم، وطالما أن هذه الخطوات لا تحتوى على loops أو تستدعى أليجوريزمات أخرى لها وقت تنفيذ خلاف  $O(1)$ .

answer  $\rightarrow O(n)$

الذين يسعون دائماً نحو التميز والأفضل  
يصبحوا تلقائياً مثلاً للآخرين

لاستلام نسخ إلكترونية من نوات الموقع مجاناً على إيميلك قم بزيارة [eng-hs.net](http://eng-hs.net)

النوتات متوفرة مجاناً في كل من تصوير الفرع أمام الهندسة أسفل صالون رنيم 24814916 أو تصوير الجمعية الرئيسية بالسرداب أسفل بيانو 24926388

م. حمادة شعبان 260 4444 9 [info@eng-hs.com](mailto:info@eng-hs.com) النوتات متوفرة مجاناً بالموقعين [eng-hs.net](http://eng-hs.net), [eng-hs.com](http://eng-hs.com)

(5)

10. Calculate the run-time efficiency of the following program segment:

```

1  i = 1
2  loop (i <= n)
    1  j = 1
    2  loop (j <= n)
        1  k = 1
        2  loop (k <= n)
            1  print (i, j, k)
            2  k = k + 1
        3  end loop
    3  end loop
    4  j = j + 1
3  end loop

```

عند طلب حساب الوقت المطلوب لتنفيذ الجوريزم يجب مراعاة التالي:

- في حال عدم وجود loop أو recursion بالجوريزم فإن الوقت المطلوب هو  $O(1)$ .
- في حال وجود recursion فإنه يقوم مقام loop.
- الحد الأقصى لوقت algorithm يحتوى على loop واحدة هو  $O(n)$  ويمكن أن يكون أقل من ذلك كأن يكون  $O(\log n)$  أو  $O(1)$ ، لكن لا يمكن أن يكون  $O(n^2)$  مثلاً أو  $O(n \log n)$ .
- الحد الأقصى لوقت algorithm يحتوى على two nested loops هو  $O(n^2)$  ويمكن أن يكون أقل من ذلك كأن يكون  $O(n \log n)$  أو  $O(n)$  أو  $O(\log^2 n)$  أو  $O(\log n)$  أو  $O(1)$  والعبرة في ذلك هو شروط إيقاف loops.
- في حال loop counter يبدأ من 1 ويتضاعف كل مرة بضربه  $\times 2$  أو يبدأ من  $n$  ويتم قسمته في كل مرة على 2 فإن الوقت المطلوب لهذه loop هو  $O(\log n)$ .

answer  $\rightarrow O(n^3)$

ابدل كل ما تستطيع لكل شيء أنت مسئول عنه

لاستلام نسخ إلكترونية من نوات الموقع مجاناً على إيميلك قم بزيارة [eng-hs.net](http://eng-hs.net)

النوتات متوفرة مجاناً في كل من تصوير الفرع أمام الهندسة أسفل صالون رنيم 24814916 أو تصوير الجمعية الرئيسية بالسرداب أسفل بيانو 24926388

النوتات متوفرة مجاناً بالموقعين [eng-hs.net](http://eng-hs.net), [eng-hs.com](http://eng-hs.com)

(6)

م. حمادة شعبان 260 4444 9 [info@eng-hs.com](mailto:info@eng-hs.com)



11. If the algorithm doIt has an efficiency factor of  $5n$ , calculate the run-time efficiency of the following program segment:

```

1  i = 1
2  loop i <= n
    1  doIt (...)
    2  i = i + 1
3  end loop

```

عند طلب حساب الوقت المطلوب لتنفيذ الجوريزم يجب مراعاة التالي:

- في حال عدم وجود loop أو recursion بالجوريزم فإن الوقت المطلوب هو  $O(1)$ .
- في حال وجود recursion فإنه يقوم مقام loop.
- الحد الأقصى لوقت algorithm يحتوى على loop واحدة هو  $O(n)$  ويمكن أن يكون أقل من ذلك كأن يكون  $O(\log n)$  أو  $O(1)$ ، لكن لا يمكن أن يكون  $O(n^2)$  مثلاً أو  $O(n \log n)$ .
- الحد الأقصى لوقت algorithm يحتوى على two nested loops هو  $O(n^2)$  ويمكن أن يكون أقل من ذلك كأن يكون  $O(n \log n)$  أو  $O(n)$  أو  $O(\log^2 n)$  أو  $O(\log n)$  أو  $O(1)$  والعبرة في ذلك هو شروط إيقاف loops.
- في حال loop counter يبدأ من 1 ويتضاعف كل مرة بضربه  $\times 2$  أو يبدأ من  $n$  ويتم قسمته في كل مرة على 2 فإن الوقت المطلوب لهذه loop هو  $O(\log n)$ .
- عدد الخطوات داخل loop لا يؤثر على big oh notation لأننا لا نهتم بالـ leading factor عند الحكم على الجوريزم، وطالما أن هذه الخطوات لا تحتوى على loops أو تستدعى الجوريزمات أخرى لها وقت تنفيذ خلاف  $O(1)$ .

$$\begin{aligned}
 \text{answer} &\rightarrow O(n * O(\text{doIt})) \\
 &= O(n * O(n)) \\
 &= O(n^2)
 \end{aligned}$$

لا ترض بأقل من  
التميز المطلق

12. If the efficiency of the algorithm doIt can be expressed as  $O(n) = n^2$ , calculate the efficiency of the following program segment:

```

1  i = 1
2  loop (i <= n)
    1  j = 1
    2  loop (j < n)
        1  doIt (...)
        2  j = j + 1
    3  end loop
    4  i = i + 1
3  end loop

```

عند طلب حساب الوقت المطلوب لتنفيذ الجوريزم يجب مراعاة التالي:

- في حال عدم وجود loop أو recursion بالجوريزم فإن الوقت المطلوب هو  $O(1)$ .
- في حال وجود recursion فإنه يقوم مقام loop.
- الحد الأقصى لوقت algorithm يحتوى على loop واحدة هو  $O(n)$  ويمكن أن يكون أقل من ذلك كأن يكون  $O(\log n)$  أو  $O(1)$ ، لكن لا يمكن أن يكون  $O(n^2)$  مثلاً أو  $O(n \log n)$ .
- الحد الأقصى لوقت algorithm يحتوى على two nested loops هو  $O(n^2)$  ويمكن أن يكون أقل من ذلك كأن يكون  $O(n \log n)$  أو  $O(n)$  أو  $O(\log^2 n)$  أو  $O(\log n)$  أو  $O(1)$  والعبرة في ذلك هو شروط إيقاف loops.
- في حال loop counter يبدأ من 1 ويتضاعف كل مرة بضربه  $2 \times$  أو يبدأ من  $n$  ويتم قسمته في كل مرة على 2 فإن الوقت المطلوب لهذه loop هو  $O(\log n)$ .
- عدد الخطوات داخل loop لا يؤثر على big oh notation لأننا لا نهتم بالـ leading factor عند الحكم على الجوريزم، وطالما أن هذه الخطوات لا تحتوى على loops أو تستدعى الجوريزمات أخرى لها وقت تنفيذ خلاف  $O(1)$ .

$$\text{answer} \rightarrow O(n * n * O(n^2)) = O(n^4)$$

اجعل التميز هو  
علامتك المسجلة

لاستلام نسخ إلكترونية من نوات الموقع مجاناً على إيميلك قم بزيارة [eng-hs.net](http://eng-hs.net)

النوتات متوفرة مجاناً في كل من تصوير الفرع أمام الهندسة أسفل صالون رنيم 24814916 أو تصوير الجمعية الرئيسية بالسرداب أسفل بيانو 24926388

النوتات متوفرة مجاناً بالموقعين [eng-hs.net](http://eng-hs.net), [eng-hs.com](http://eng-hs.com)

(8)

م. حمادة شعبان 260 4444 9 [info@eng-hs.com](mailto:info@eng-hs.com)



**13.** If the efficiency of the algorithm doIt can be expressed as  $O(n) = n^2$ , calculate the efficiency of the following program segment:

```

1  i = 1
2  loop (i < n)
    1  doIt (...)
    2  i = i * 2
3  end loop

```

عند طلب حساب الوقت المطلوب لتنفيذ أليوريزم يجب مراعاة التالي:

- في حال عدم وجود loop أو recursion بالأليوريزم فإن الوقت المطلوب هو  $O(1)$ .
- في حال وجود recursion فإنه يقوم مقام loop.
- الحد الأقصى لوقت algorithm يحتوى على loop واحدة هو  $O(n)$  ويمكن أن يكون أقل من ذلك كأن يكون  $O(\log n)$  أو  $O(1)$ ، لكن لا يمكن أن يكون  $O(n^2)$  مثلاً أو  $O(n \log n)$ .
- الحد الأقصى لوقت algorithm يحتوى على two nested loops هو  $O(n^2)$  ويمكن أن يكون أقل من ذلك كأن يكون  $O(n \log n)$  أو  $O(n)$  أو  $O(\log^2 n)$  أو  $O(\log n)$  أو  $O(1)$  والعبرة في ذلك هو شروط إيقاف loops.
- في حال loop counter يبدأ من 1 ويتضاعف كل مرة بضربه  $\times 2$  أو يبدأ من  $n$  ويتم قسمته في كل مرة على 2 فإن الوقت المطلوب لهذه loop هو  $O(\log n)$ .
- عدد الخطوات داخل loop لا يؤثر على big oh notation لأننا لا نهتم بالـ leading factor عند الحكم على أليوريزم، وطالما أن هذه الخطوات لا تحتوى على loops أو تستدعى أليوريزمات أخرى لها وقت تنفيذ خلاف  $O(1)$ .

$$\text{answer} \rightarrow O(\log n * O(n^2)) = O(n^2 \log n)$$

الكثير من العمل الجيد يضيع بسبب  
العجز عن بذل ما هو أكثر قليلاً

لاستلام نسخ إلكترونية من نوات الموقع مجاناً على إيميلك قم بزيارة [eng-hs.net](http://eng-hs.net)

النوتات متوفرة مجاناً في كل من تصوير الفرع أمام الهندسة أسفل صالون رنيم 24814916 أو تصوير الجمعية الرئيسية بالسرداب أسفل بيانو 24926388

النوتات متوفرة مجاناً بالموقعين [eng-hs.net](http://eng-hs.net), [eng-hs.com](http://eng-hs.com)

(9)

م. حمادة شعبان 260 4444 9 [info@eng-hs.com](mailto:info@eng-hs.com)

- 14.** Given that the efficiency of an algorithm is  $5n^2$ , if a step in this algorithm takes 1 nanosecond ( $10^{-9}$ ), how long does it take the algorithm to process an input of size 1000?

Time required = (number of steps) \* (time of each step)

$$= (5n^2) * (t)$$

$$= (5 * (1000)^2) * (10^{-9})$$

$$= 5 * 10^{-3} \text{ seconds}$$

$$= 0.005 \text{ second}$$

ازرع بذرة الرغبة في عقلك وسوف  
تشكل نواة ذات قوة تجذب إليه كل  
شيء تحتاجه لتحقيق نجاحه

لاستلام نسخ إلكترونية من نوات الموقع مجاناً على إيميلك قم بزيارة [eng-hs.net](http://eng-hs.net)

النوتات متوفرة مجاناً في كل من تصوير الفرع أمام الهندسة أسفل صالون رنيم 24814916 أو تصوير الجمعية الرئيسية بالسرداب أسفل بيانو 24926388

النوتات متوفرة مجاناً بالموقعين [eng-hs.com](http://eng-hs.com), [eng-hs.net](http://eng-hs.net)

(10)

م. حمادة شعبان 260 4444 9 [info@eng-hs.com](mailto:info@eng-hs.com)

16. Given that the efficiency of an algorithm is  $5n\log_2(n)$ , if a step in this algorithm takes 1 nanosecond ( $10^{-9}$ ), how long does it take the algorithm to process an input of size 1000?

Time required = (number of steps) \* (time of each step)

$$= (5n \log n) * (t)$$

$$= (5 * 1000 * \log(1000)) * (1 * 10^{-9})$$

$$\simeq 50 * 10^{-6} \text{ seconds}$$

الرغبات الضعيفة تحقق نتائج ضعيفة،  
تماماً كما أن قدراً صغيراً من النيران لا  
يصنع إلا قدراً صغيراً من الحرارة

لاستلام نسخ إلكترونية من نوات الموقع مجاناً على إيميلك قم بزيارة [eng-hs.net](http://eng-hs.net)

النوتات متوفرة مجاناً في كل من تصوير الفرع أمام الهندسة أسفل صالون رنيم 24814916 أو تصوير الجمعية الرئيسية بالسرداب أسفل بيانو 24926388

النوتات متوفرة مجاناً بالموقعين [eng-hs.com](http://eng-hs.com), [eng-hs.net](http://eng-hs.net)

(11)

م. حمادة شعبان 260 4444 9 [info@eng-hs.com](mailto:info@eng-hs.com)

8. Find the complexity of the function used to find the  $k$ th smallest integer in an unordered array of integers

```
int selectkth(int a[], int k, int n) {
    int i, j, mini, tmp;
    for (i = 0; i < k; i++) {
        mini = i;
        for (j = i+1; j < n; j++)
            if (a[j] < a[mini])
                mini = j;
        tmp = a[i];
        a[i] = a[mini];
        a[mini] = tmp;
    }
    return a[k-1];
}
```

عند طلب حساب الوقت المطلوب لتنفيذ أليجوريزم يجب مراعاة التالي:

- في حال عدم وجود **loop** أو **recursion** بالأليجوريزم فإن الوقت المطلوب هو  $O(1)$ .
- في حال وجود **recursion** فإنه يقوم مقام **loop**.
- الحد الأقصى لوقت **algorithm** يحتوى على **loop** واحدة هو  $O(n)$  ويمكن أن يكون أقل من ذلك كأن يكون  $O(\log n)$  أو  $O(1)$ ، لكن لا يمكن أن يكون  $O(n^2)$  مثلاً أو  $O(n \log n)$ .
- الحد الأقصى لوقت **algorithm** يحتوى على **two nested loops** هو  $O(n^2)$  ويمكن أن يكون أقل من ذلك كأن يكون  $O(n \log n)$  أو  $O(n)$  أو  $O(\log^2 n)$  أو  $O(\log n)$  أو  $O(1)$  والعبرة في ذلك هو شروط إيقاف **loops**.
- في حال **loop counter** يبدأ من 1 ويتضاعف كل مرة بضربه  $\times 2$  أو يبدأ من  $n$  ويتم قسمته في كل مرة على 2 فإن الوقت المطلوب لهذه **loop** هو  $O(\log n)$ .

Answer  $\rightarrow O(k * O(n)) = O(kn)$

يمكن أن يكون  $k$  أي قيمة من 1 إلى  $n$ ، أي أن  $(k = O(n))$  وبالتالي فإن أقصى وقت لهذه الدالة هو  $O(n^2)$ .

نقطة الانطلاق لكل الإنجازات هي  
الرغبة، تذكر هذا دائماً وباستمرار

لاستلام نسخ إلكترونية من نوات الموقع مجاناً على إيميلك قم بزيارة [eng-hs.net](http://eng-hs.net)

النوتات متوفرة مجاناً في كل من تصوير الفرع أمام الهندسة أسفل صالون رنيم 24814916 أو تصوير الجمعية الرئيسية بالسرداب أسفل بيانو 24926388

النوتات متوفرة مجاناً بالموقعين [eng-hs.net](http://eng-hs.net), [eng-hs.com](http://eng-hs.com)

(12)

م. حمادة شعبان 260 4444 9 [info@eng-hs.com](mailto:info@eng-hs.com)

9. Determine the complexity of the following implementations of the algorithms for adding, multiplying, and transposing  $n \times n$  matrices:

```
for (i = 0; i < n; i++)
    for (j = 0; j < n; j++)
        a[i][j] = b[i][j] + c[i][j];
```

$$O(n^2)$$

```
for (i = 0; i < n; i++)
    for (j = 0; j < n; j++)
        for (k = 0; k < n; k++)
            a[i][j] += b[i][k] * c[k][j];
```

$$O(n^3)$$

```
for (i = 0; i < n - 1; i++)
    for (j = i + 1; j < n; j++) {
        tmp = a[i][j];
        a[i][j] = a[j][i];
        a[j][i] = tmp;
    }
```

$$O\left(\frac{n^2}{2}\right) = O(n^2)$$

عند طلب حساب الوقت المطلوب لتنفيذ الجوريزم يجب مراعاة التالي:

- في حال عدم وجود **loop** أو **recursion** بالآلجوريزم فإن الوقت المطلوب هو  $O(1)$ .
- في حال وجود **recursion** فإنه يقوم مقام **loop**.
- الحد الأقصى لوقت **algorithm** يحتوى على **loop** واحدة هو  $O(n)$  ويمكن أن يكون أقل من ذلك كأن يكون  $O(\log n)$  أو  $O(1)$ ، لكن لا يمكن أن يكون  $O(n^2)$  مثلاً أو  $O(n \log n)$ .
- الحد الأقصى لوقت **algorithm** يحتوى على **two nested loops** هو  $O(n^2)$  ويمكن أن يكون أقل من ذلك كأن يكون  $O(n \log n)$  أو  $O(n)$  أو  $O(\log^2 n)$  أو  $O(\log n)$  أو  $O(1)$  والعبرة في ذلك هو شروط إيقاف **loops**.
- في حال **loop counter** يبدأ من 1 ويتضاعف كل مرة بضربه  $\times 2$  أو يبدأ من  $n$  ويتم قسمته في كل مرة على 2 فإن الوقت المطلوب لهذه **loop** هو  $O(\log n)$ .
- عدد الخطوات داخل **loop** لا يؤثر على **big oh notation** لأننا لا نهتم بالـ **leading factor** عند الحكم على آلجوريزم، وطالما أن هذه الخطوات لا تحتوى على **loops** أو تستدعى آلجوريزمات أخرى لها وقت تنفيذ خلاف  $O(1)$ .

مواهبنا هي فرص لإنجاز أعمال طيبة

لاستلام نسخ إلكترونية من نوات الموقع مجاناً على إيميلك قم بزيارة [eng-hs.net](http://eng-hs.net)

النوتات متوفرة مجاناً في كل من تصوير الفرع أمام الهندسة أسفل صالون رنيم 24814916 أو تصوير الجمعية الرئيسية بالسرداب أسفل بيانو 24926388

م. حمادة شعبان 260 4444 9 [info@eng-hs.com](mailto:info@eng-hs.com) النوتات متوفرة مجاناً بالموقعين [eng-hs.net](http://eng-hs.net), [eng-hs.com](http://eng-hs.com)

(13)



10. Find the computational complexity for the following four loops:

a. `for (cnt1 = 0, i = 1; i <= n; i++)  
    for (j = 1; j <= n; j++)  
        cnt1++;`

answer →  $cnt1 = n + n + n \dots + n + n = n^2$

b. `for (cnt2 = 0, i = 1; i <= n; i++)  
    for (j = 1; j <= i; j++)  
        cnt2++;`

$$cnt2 = 1 + 2 + 3 + 4 + \dots + (n-1) + (n)$$

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

$$= \frac{n^2}{2} + \frac{n}{2}$$

c. `for (cnt3 = 0, i = 1; i <= n; i *= 2)  
    for (j = 1; j <= n; j++)  
        cnt3++;`

$$cnt3 = n + n + n \dots (\log n \text{ times})$$

$$= n * \log n$$

d. `for (cnt4 = 0, i = 1; i <= n; i *= 2)  
    for (j = 1; j <= i; j++)  
        cnt4++;`

$$cnt4 = (2^0 + 2^1 + 2^2 + 2^3 + \dots \log n)$$

$$= \frac{2^{\log n + 1} - 1}{2 - 1}$$

$$= 2^{\log n + 1} - 1$$

الخيال يعدو بأقصى سرعة،  
والمنطق يسير بالكاد