جَامعة
المَنارة
MANARA UNIVERSITY

كلية الهندسة - قسم المعلوماتية

بنى معطيات 1

Data Structure 1

ا. د. علي عمران سليمان

محاضرات الأسبوع الأول

فعالية الخوارزمية

Algorithm Efficiency

الفصل الأول 2025-2026

محتوى المقرر

1- Introduction.	1- مقدمة.
2- Analysis of algorithms and standard algorithms.	2- تحليل الخوارزميات والخوارزميات القياسية.
3- Data structures and abstract data type.	3- بنى معطيات وأنماط البيانات المجردة.
4 – Stacks.	4 - المكدسات.
5- queues.	5- الأرتال .
6- lists More linked lists .	6- البنى المترابطة اللوائح .
7 - Binary trees & Hash tables.	7 - الأشجار الثنائية .
8-Binary search tree.	8-شجرة البحث الثنائية .
9- AVL Tree.	9- AVL tree.
10- ptree	10- ptree

References:

- **ADTs, Data Structures, and Problem Solving with C++: International Edition, by Larry R. Nyhoff, Publisher:** Pearson , 2004
- Data Structures and Algorithms in Java, 6 th, Roberto Tamassia, Michael T. Goodrich, Michael H. Goldwasser, Pub. Wiley 2014
- د.علي سليمان، بنى معطيات بلغة JAVA، بنى معطيات بلغة C++، بنى معطيات بلغة Pascal جامعة تشرين 2014، 2007، 1998

محتوى المحاضرة

6-1- Introduction: 6-1-1- Classification methods 6-1-2-algorithm properties 6-1-3-algorithm types 6-2- Algorithm effectiveness: 6-3- Search algorithms: 6-3-1- search concept: 6-3-2- Linear (sequential) search algorithm: 6-3-3- Binary search algorithm: 6-4- Sort algorithms sort concept, bubble sorting, swap sorting, selection sorting (selective sorting), insetion sorting, merging sorting, merging concept, merge sorting, quick sorting, Bucket sorting.	6-1- مقدمة : Why Data Structures 6-1-1- طرق التصنيف. 6-1-2-خصائص الخوارزمية 6-1-3-أنواع الخوارزميات1 6-2- فعالية الخوارزمية : 6-3- خوارزميات البحث: 6-3-1- مفهوم البحث: 6-3-2- خوارزمية البحث الخطي (التتابعي): 6-3-3- خوارزمية البحث الثنائي : 6-4- خوارزميات الترتيب. مفهوم الترتيب، الترتيب الفقاعي، الترتيب بالتبديل، الترتيب بالاختيار (الترتيب الانتقائي)، الترتيب بالحشر،الترتيب بالدمج، مفهوم الدمج، الترتيب الدمج، الترتيب السريع، ترتيب Bucket Sort.

- **ADTs, Data Structures, and Problem Solving with C++: International Edition, by Larry R. Nyhoff, Publisher:** Pearson , 2005
- Data Structures and Algorithms in Java, 6 th, Roberto Tamassia, Michael T. Goodrich, Michael H. Goldwasser, Pub. Wiley 2014
- **د.علي سليمان، بنى معطيات بلغة JAVA، بنى معطيات بلغة C++، بنى معطيات بلغة Pascal جامعة تشرين 2014، 2007، 1998**

6-1- Introduction

1-6 -مقدمة:

1-6 -مقدمة: Why Data Structures in C++

- البرنامج مؤلف من Data و instructions تطبق الأوامر على البيانات للوصول للنتيجة.
- قبل تنفيذ البرنامج سيتم تحميل البرنامج والبيانات إلى الذاكرة من وحدة التخزين الثانوية.
- تنفيذ البرنامج يعني تنفيذه تعليمة تعليمة وقد تكون غير متماسة وفي كل تعليمة قد يحتاج بيانات مما يتطلب قراءتها وفق الطلب.
- هنا تظهر أهمية طريقة تخزين البيانات ضمن الذاكرة والتي تؤثر على سرعة عمل البرنامج.
- البيانات الأولية مشكلتها محلولة لإن كل منها يملك اسماً وعنوان وهى متماسة, البنى المركبة : تلعب طريقة التخزين دوراً هاماً في سرعة التنفيذ وهو ماتحدده ببنى المعطيات.
- بنى المعطيات: هي طريقة لتخزين وترتيب البيانات بحيث تؤمن عند استخدامها والوصول إليها فعالية كبيرة (أسرع مايمكن) .
- سيتم تخزن نص البرنامج في جزء المخصص لل code ، إنما المعطيات الأولية سيتم حجز أماكن لها ضمن الجزء static من الذاكرة ويتم خلال الترجمة compiler Time أما في run Time فيتم الحجز الديناميكي في جزء heap , كل تابع يملك Activation Record يتضمن المعطيات وعندما ينادي تابع لتابع أخر يتم تخزين سجل التابع الحالي ضمن المكدس وينتقل التحكم للتابع المنادى وتعريف سجل يتضمن أماكن المتغيرات للتابع الجديد وهكذا... وبعد إنتهاء العمل مع التابع المنادى سيتم حذف سجله وسحب سجل التابع الذي ناداه وهكذا... حتى العوده للتابع الرئيس الذي سينتهي البرنامج بإنتهائة (كل ماذكر هو ضمن Static memory allocation).

6-1- Introduction

1-6- مقدمة:

بالنظر للبرنامج المجاور نجد:

```
Void main() {      int i; int *A; cout<<"Enter array size  "; cin>>i;
        A=new int[i];                    delete []A ; }
```

- تم تعريف متغير معلوم الحجم i وبالتالي سيتم الحجز له ضمن منطقة Stack
- متغير أخر من نوع مؤشر A يتضمن عنوان مصفوفة سيتم تحديد حجمها عند Run Time سيتم الحجز لها ضمن منطقة heap
- التابع main لايستطيع الوصول لـ heap إلا من خلال المؤشر المتضمن العنوان للمصفوفة والموجود في منطقة Stack.
- هذا مايعرف بالـ Dynamic allocation.
- أية مساحة محجوزة ضمن heap عند الانتهاء منها يجب أن يتم حذفها delete []A كي لايحصل تسرب للذاكرة.
- المساحات ضمن stack يتم حذفها بشكل تلقائي عند الخروج من مجال الرؤيا لها.

سنهتم بحساب درجة تعقيد الخوارزميات والتعبير عن هذه الدرجة وفق حالة أو أكثر من الحالات الثلاث:

1-الحالة السيئة (Worst Case) Big Oh Notation(O) كأن يكون العنصر المبحوث عنه آخر عنصر أو غير موجود.

2-الحالة المثلي(Best Case) Big Omega Notation (Ω) والتي يكون العنصر المبحوث عنه هو الأول.

3-الحالة الوسطي (Average Case)(θ)Big Theta Notation أن يكون في منتصف اللائحة.

خصائص الخوارزمية algorithm properties

1. **تعريف الخوارزمية:** هي عدد محدد من الخطوات المتسلسلة الضرورية والواضحة من اجل أداء مهمة أو حل مسألة ما وفق قيم الدخل، وتهدف على الحصول على نتائج محددة اعتباراً من معطيات ابتدائية محدده، وتكتب بلغة معينة أو ترسم باستخدام المخططات لتمثيلها.
2. خصائص الخوارزمية :
   1. المدخلات Inputs: تستقبل الخوارزمية مجموعة محددة من معطيات مدخلة.
   2. المخرجات Outputs : تنتج الخوارزمية مخرجات محددة لكل معطيات مدخلة وتعرف بحل المسألة.
   3. الوضوح definiteness : يجب ان تكون كل خطوة مكتوبة بطريقة واضحة ومعبرة وخالية من الغموض.
   4. المحدودية Limited : يجب على الخوارزمية أن تنتج قيم الخرج المرجوة بعد عدد محدود من الخطوات وذلك من أجل كل مجموعة محددة من قيم الدخل .
   5. التفرد Uniqueness : يتم تحديد نتائج كل خطوة بشكل فريد وتعتمد فقط على المدخلات ونتائج الخطوات السابقة.
   6. العمومية generality : تطبق الخوارزمية على مجموعة معممة من المدخلات.
   7. الفعالية effectiveness : يجب ان تكون فعالة عند استخدامها لحل المشكلة البرمجية.

6-1- Classification methods

1-6- طرق التصنيف :

- يمكن تصنيف بنى المعطيات وفق (Existence, Based on Memory Allocation, Based on Representation)

1. التصنيف وفق Existence:

✓ Physical DS as Arrays, linked list (Array fixed list (Sequential) ; linked list can distributed in memory).

✓ Logical DS Can't be created independently as Stack, Queues, Trees, Graphs.

- بنى المعطيات المنطقية يجب أن توضع لها الخوارزميات لتوضيح ألية عملها في الذاكرة.

2. التصنيف وفق Based on Memory Allocation:

✓ Static (or fixed sized) DS Such as Static Arrays.

✓ Dynamic DS (change size as needed) Such as Linked List, Dynamic Arrays.

3. التصنيف وفق Based on Representation :

✓ Linear DS such as Arrays, Linked list, Stack, Queues.

✓ Non Linear DS such as Trees, Graphs.

- العمليات الأساسية على بنى المعطيات Operations on DS:

✓ Traversing, Searching, Insertion, Deletion, Sorting and Merging,.

أنواع الخوارزميات1

algorithm types 1

3. أنواع الخوارزميات algorithm types

- يمكن تصنيف الخوارزميات بحسب الوظيفة التي تقوم بها، والمسألة الرئيسية التي تحلها، على سبيل المثال:

1. خوارزميات البحث search algorithms هدفها البحث عن عنصر معطيات محدد.
2. خوارزميات الفرز sort algorithms هدفها ترتيب مجموعة من عناصر المعطيات ترتيباً متتالياً اعتماداً على أحد بنود العناصر أو على اجتماع عدة بنود محددة.

- بحسب التقنية المستخدمة في تشكيل التعليمات وتسلسلها، مثل:

1. الخوارزميات التتابعية sequential algorithms يجري تنفيذها وفق تتابع التعليمات وبترتيب معين.
2. الخوارزميات المتوازية parallel algorithms يجري تنفيذ أكثر من جزء . في أن واحد معاً، ويجري ذلك عادة على عدة معالجات.
3. الخوارزميات العودية recursive algorithms وهي خوارزميات تستخدم ضمن تعليماتها استدعاة للخوارزمية نفسها. مثالها خوارزمية حساب !n.
4. الخوارزميات التراجعية backtracking algorithms وهي خوارزميات تستخدم لإيجاد حل ضمن مجموعة محاولات ممكنة، حيث تمثل المحاولات على شكل فروع في شجرة. يجري تجريب أحد الفروع، فإن لم نجد الحل نعود إلى الوراء لنختار مساراً آخر نجربه وهكذا، حتى تعثر على المسار المناسب. من أمثالتها تلوين خارطة بما لا يزيد على أربعة ألوان.، تمارين المتاهات

أنواع الخوارزميات2

algorithm types 2

هذا ويمكن تجميع (تصنيف) الخوارزميات التي تستخدم طرق مشابهة في حل المسائل مع بعضها البعض أخذين بعين الاعتبار بأن التصنيف ليس شمولياً وليس منفصلاً أي يُمكن لتصنيفين ما أن يتقاطعا:

5. خوارزميات فرق تسد divide and conquer algorithms يتم في هذا النوع من الخوارزميات أولاً تقسيم المسألة المراد حلها إلى مسائل جزئية أصغر من نفس النمط ومن ثم حل تلك المسائل الجزئية بطريقة عودية. ثانياً تجميع حلول المسائل الجزئية التي تم الحصول عليها ضمن حل واحد للمسألة الأصلية. مثال على ذلك خوارزمية الفرز السريع quicksort وكذلك خوارزمية الترتيب بالدمج merge sort .
6. خوارزميات البرمجة الديناميكية dynamic programming algorithms: عبارة عن خوارزميات تتذكر النتائج السابقة وتستخدمها لإيجاد نتائج جديدة تستخدم الخوارزميات الديناميكية عادة لإيجاد الحلول المثلى optimization problems مثال على ذلك خوارزمية Dijkstra في إيجاد أقصر مسار shortest path بين عقدتين (مدينتين) لتحليل K-Means Clustering البيانات.
7. خوارزميات الطموحة Greedy Algorithms تستخدم لإيجاد الحل الأمثلي للمسائل المطروحة، وهي تعمل على مراحل في كل مرحلة أولاً نأخذ في لحظة معينة الحل الأمثل بدون النظر إلى النتائج (الطريقة) Activity selection, Job sequencing، Knapsack algorithm، Huffman coding .

2-6- فعالية الخوارزمية 1

Algorithm Efficiency 1

2-6- فعالية الخوارزمية :

Algorithm Efficiency

- إن تغليف المعطيات ممكن وبعدة طرق ويمكن استخدام هذا الشكل من المعطيات بأكثر من نوع دون التأثر بطريقة التغليف هذه . وتقيّم طريقة التغليف بدرجة تعقيد الخوارزمية والتي تعتمد على أساسين :

1. حجم الذاكرة اللازم لتخزين هذه المعطيات وإعطاء إمكانية استخدامها ويعرف بالتعقيد الحجمي Space complexity.
2. الوقت اللازم لإدخال المعطيات إلى الذاكرة والوقت المطلوب لتنفيذ الخوارزمية ويعرف بالتعقيد الزمني Time complexity ويهمل العامل الأول بالقياس لأهمية العامل الثاني وإمكانية التفاعل معة.

التعقيد الزمني يتعلق بعوامل منها: أ-حجم الدخل، ب- نوع وعدد التعليمات، ت- نوعية الشيفرة ، ث- سرعة الآلة .

ونظراً لإن العاملين الآخيرين تتطلب تبديل الجهاز أو اللغة ولا يمكن تحديدهما بشكل دقيق في وحدات الزمن الحقيقي كااجزاء للثو اني مثلاً سوف نهتم بالعاملين أ و ب.

- حجم الدخل، يتطلب زمن من أجل التعبير عنها وزمن مستغرق لتصنيف لائحة يتعلق بالتأكيد بعدد عناصر هذه اللائحة وبالتالي فإن زمن التنفيذ تابع لعدد العناصر (n) أي أن زمن التنفيذ هو T(n).

ب- نوع وعدد التعليمات: زمن تنفيذ التعليات (= ,+، -، *، /، %) يختلف فيما بينها، أما عددها يمكن العمل عليه كثيراً.

ومن أجل التعرف على طريقة حساب هذا الزمن ندرس الخوارزمية (6-1) الخاصة بحساب المتوسطة الحسابي لـ nعدداً .

Algorithm Efficiency 2

2-6- فعالية الخوارزمية 2

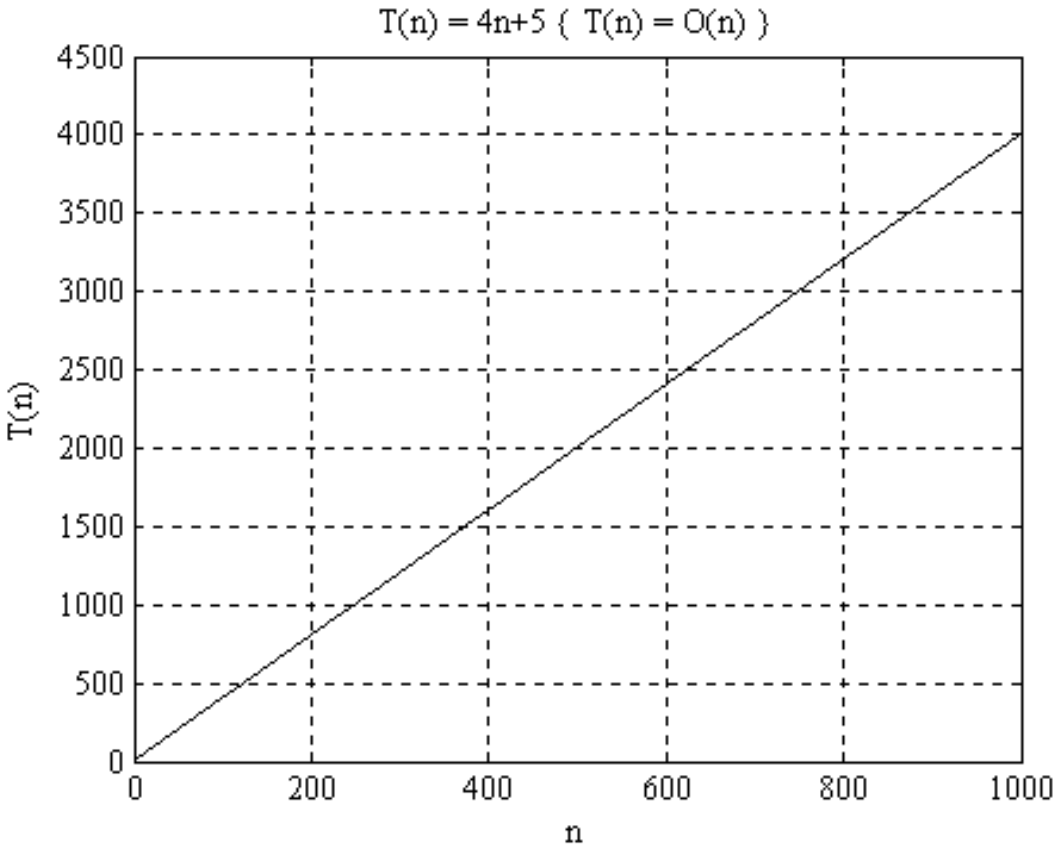
ALGORITHM TO CALCULATE MEAN : (*Algorithm to read a set of n numbers and calculate their mean*) 1. Read n. 2. Initialize Sum To 0. 3. Initialize I To 1. 4. While I<=N Do The Following A. Read Number . B. Add Number To Sum. C. Increment I By 1. 5. Calculate Mean = Sum/N إن كلاً من الخطوات 1, 2 , 3 ,5 تنفذ مرة واحدة بينما A,B,C ينفذ كل منها n مرة بينما 4 تنفذ n+1 مرة :	Statement	# of times executed
	1	1
	2	1
	3	1
	4	n+1
	A	n
	B	n
	C	n
	5	1
	Total:	4n+5

جدول (1-6) يمثل درجة تعقيد خوارزمية حساب المتوسط:ملاحظة: الخطوه 5قد تحسب عمليتين قسمه ونسب.

2-6- فعالية الخوارزمية 3

Algorithm Efficiency 3

وعليه يعطي الزمن اللازم لهذه الخوارزمية بالعلاقة $T(n)=4n+5$ حيث n عدد طبيعي يمثل قيم الدخل ومن العلاقة نجد أن T(n) يتزايد بشكل خطي مع n ويعبر الخط البياني المبين في الشكل (6-1)عن هذه العلاقة:



الشكل (6-1)علاقة زمن التنفيذ بعدد العناصر.

2-6- فعالية الخوارزمية 4

Algorithm Efficiency 4

نقول أن T(n) متزايدة خطياً مع n أو من مرتبة n (Order of magnitude n) ويعبر عن ذلك عادة باستخدام ما يعرف بـ تدوين (O) الكبيرة BONويرمز له عادة : T(n)=O(n) ونقرؤه كالتالي: زمن تنفيذ الخوارزمية T(n) تابع من مرتبة n .

وفي الحالة العامة نقول أن الزمن اللازم لتنفيذ خوارزمية ما يملك مرتبة تابع ما في n مثل f(n) ونعبر عن ذلك كما يلي:

T(n) = O(f(n)) ويعرف asymptotic upper bound الحد العلوي المقارب (المسيطر) أي أن التابع f(n) هو أكبر من T(n) اعتباراً من حد معين n_0، أي يسيطر عليه.

إذا وفقط إذاوجد ∃ثابتان طبيعيان C, n_0 يحققان العلاقة :

$$T(n) \le C.f(n) \ ; \text{for all } n \ge n_0$$

أي أن T(n) محدود من الأعلى (أصغر أو يساوي) بقيمة هي ثابت ما مضروباً في قيمة التابع f(n) لكل قيم n اعتباراً من نقطة معينة n_0 ولو عدنا لمثالنا لوجدنا:

وعليه فإن الثوابت :

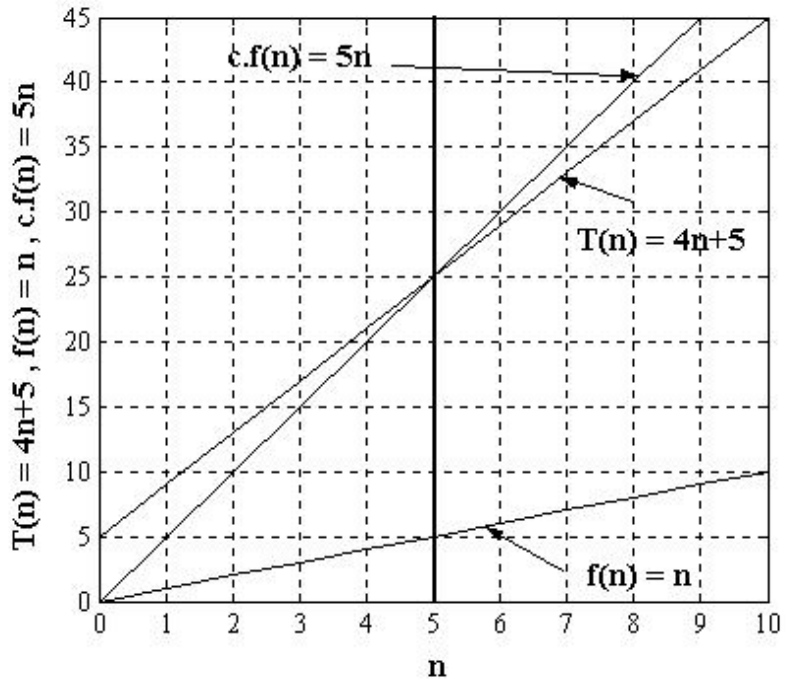
$$T(n)= 4n +5 \Rightarrow 4n +5 \le 4n +n \ \text{for } n \ge 5 \Rightarrow T(n) \le 5n \text{ for all } n_0 \ge 5$$

$$f(n) = n, n_0 = 5, and \quad C = 5 \Rightarrow T(n) = O(n)$$

ويبين الشكل (2-6) التوابع f(n)=n و 5.f(n)=5n والتابع T(n)=4n+5

Algorithm Efficiency 5

2-6- فعالية الخوارزمية 5



الشكل (2-6) التوابع f(n)=n و 5.f(n)=5n والتابع T(n)=4n+5

لاحظ أن T(n) =< 5.f(n) عندما n>= 5

ويعرف c.f(n)على أنه asymptotic upper bound للتابع(T(n من أجل الثابتين c=5 and n_0 >=5

النص(6-1) يبين برنامجاً بلغة C++ لتنفيذ هذه الخوارزمية البسيطة (في الشريحة 19):

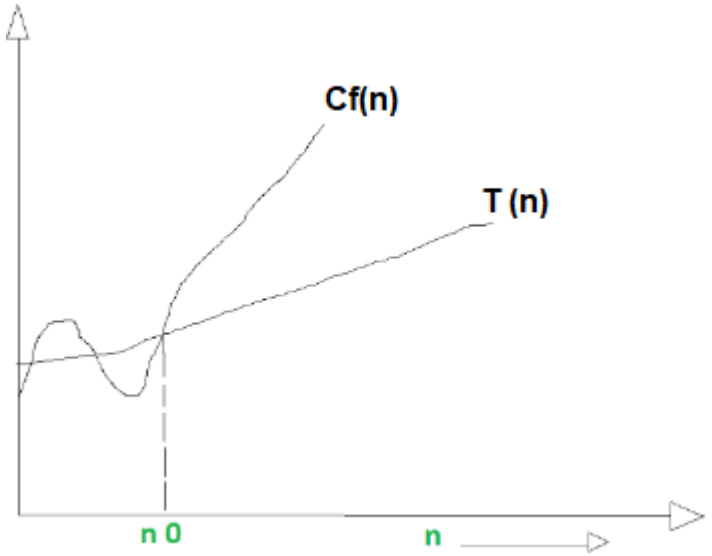
مثال (1): Is $f(n)= n^4$ upper bound of $T(n) = 4 n^2 + 4n +6$ as $n>= n_0$ هل التابع $f(n)= n^4$ هو تابع asymptotic upper bound للتابع $T(n) = 4 n^2 + 4n +6$، بالنظر نجد أن $T(n)<= C f(n)$ بعد حد معين نجد ذلك صحيح لإن الأس 4 سيكون أكبر من الأس 2

حيث سنثبت ذلك ولمرةً واحدةً فقط من خلال إعطاء قيم تجريبيه، ولن يكون ضروريًا لاحقًا. ∃ C, n_0 if and only if exists تحقق المعادلة $T(n)<=Cf(n)$ عندها يكون f(n) تابع asymptotic upper bound للتابع T(n).

Algorithm Efficiency 6

6-2- فعالية الخوارزمية 6

n	T(n)	F(n)	result
0	6	0	no
1	14	1	no
2	30	16	no
3	54	81	yes



بفرض C=1 ونعطي n قيم صحيحة موجبة (طبيعية).

نجد عندما C=1 و n_0 >=3 وما فوق تتحقق المتراجحة T(n)<=O(Cf(n))

أي أن (f(n) upper bound of T(n)) وهو المطلوب.

$T(n) = 4n^2 + 4n + 6$ و $f(n)= n^4$ ونبحث عن

مثال(2): هل التابع $f(n)= n^2$ هو تابع asymptotic upper bound للتابع $T(n) = an+b$، من أجل كل قيم a, b ؟ بالنظر نجد أن $an+b<=C\ n^2$ بعد حد معين لإن الأس 2 سيكون أكبر من الأس 1 .

مثال(3): هل التابع $f(n)= n^4$ هو asymptotic upper bound للتابع $T(n)=3n^3$؟

نعم لإن الاس الأكبر سيكون عند حد معين محقق للمتراجحة(n=3).

مثال(4):هل$C\ f(n^2)$ هو مسيطر على$3\ n^2 +15$ ؟T{n}= 3 n^2 +15 وتكتب $3\ n^2 +15=O(n^2)$

نحن نبحث عن قيم طبيعية لكل من C و n_0 تحقق $3\ n^2 +15< = C\ n^2$ إن القيم C>3 وn_0 > 3 تحقق ذلك.

2-6- فعالية الخوارزمية 7

Algorithm Efficiency 7

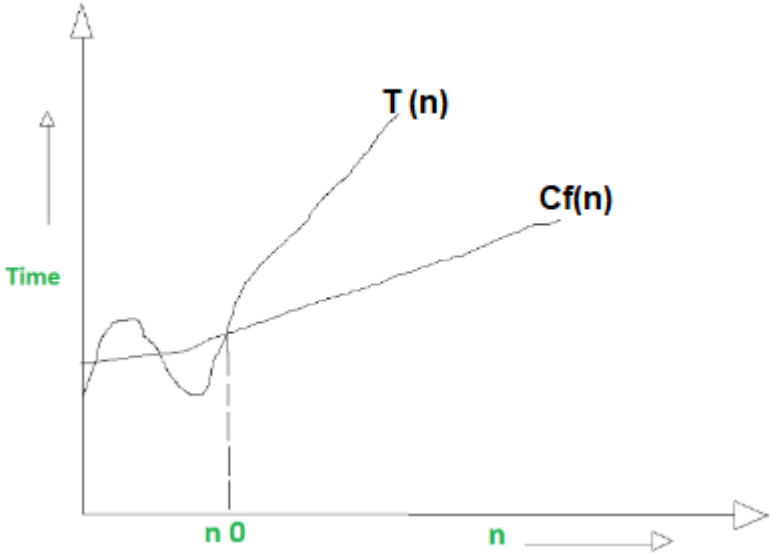
مثال(5): هل $C(2^n)$ مسيطر على 2^{n-1} ؟تكتب ($2^{n-1}=O(2^n)$) سنبسط المعادلة هل $2^{n-1} <= C2^n$ ويمكن أن تكتب $2^n * 2^{-1} <= C2^n$ وباختصار 2^nمن الطرفين نجد $2^{-1} <= C$ أي من أجل $C>=1/2$ أي $C>1$ وبالتالي نعم .

مثال(6): هل $C(2^n)$ مسيطر على 2^{2n} ؟ ($2^{2n}=O(2^n)$)نحن نبحث عن قيم لكل من C و n_0 تحقق $2^{2n} <= C\ 2^n$ هنا لايمكن لإن الأساس هو ذاته 2 والاس مختلف (نظراً لإن n قيم كبيرة جداً).

2-الحالة المثلي(Best Case) Big Omega Notation (Ω) حيث تستغرق الخوارزمية أقل وقت لها (أسرع وقت للاكتمال)

هل ($T(n) = \Omega(f(n))$: نقول أن التابع f(n) هو تابع asymptotic lower bound للتابع T(n). إذا وفقط إذا تم إيجاد عددين طبيعين C و n_0 يتحقق من إجلهما $T(n) >= Cf(n)>=0$. حيث $n>= n_0$

كما في الشكل المجاور.



مثال1: بفرض $T(n) = 4\ n^2 + 4n + 6$ و $f(n)= n^2$ هل ? $T(n) = \Omega\ f(n)$ نقول نعم

إذا وفقط إذا تم إيجاد عددين طبيعين C و n_0 يتحقق من إجلهما $T(n) >= Cf(n)>=0$.

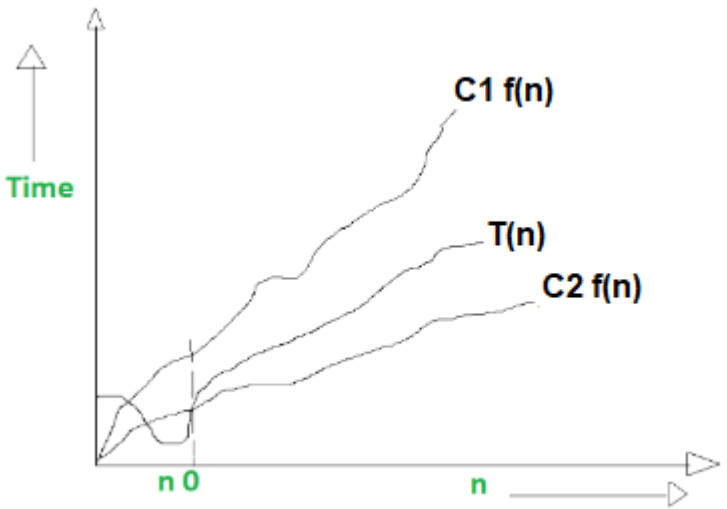
حيث $n>= n_0$ $4\ n^2 + 4n + 6 >= C\ n^2$ نختار $C<=4$ فتصبح محققة من أجل أية قيمة لـ n

مثال2: بفرض $T(n) = 4\ n^2 + 4n + 6$ و $f(n)= n^3$ هل ? $T(n) = \Omega\ f(n)$ نقول لا

السبب $T(n) >= Cf(n)>=0$ المكعبة أكبر من التربيعية.هى upper.

Algorithm Efficiency 8

2-6- فعالية الخوارزمية 8



3-الحالة الوسطي (Average Case)Big Theta Notation (θ)أن يكون في منتصف اللائحة ويعرف asymptotic tight bound أحياناً أكبر وأحياناً أصغر.

هل $T(n) = \theta(f(n))$: نقول أن التابع f(n) هو تابع asymptotic tight bound للتابع T(n). إذا وفقط إذا تم إيجاد ثلاث أعداد طبيعية C1, C2 و n_0 يتحقق من إجلهما

$0 <= C2f(n) <= T(n) <= C1f(n)$ من أجل $n >= n_0$.

(يقع بين الحالة O والحالة Ω) كما في الشكل المجاور. والثابت C1, C2 هو من يحدد lower, upper أي $T(n) = \Omega f(n)$ and $T(n) = O f(n)$ عندها $T(n) = \theta f(n)$.

مثال(1): هل $T(n) = 4n^2 + 4n$ هي $T(n) = \theta f(n^2)$ نعم إذا حققت الحالتين:

1- $T(n) = Of(n^2)$ نعم عندما نجدد $4n^2 + 4n <= C1\, n^2$ هي محققة من أجل كل قيم $C1 > 4$ و $n >= 4$

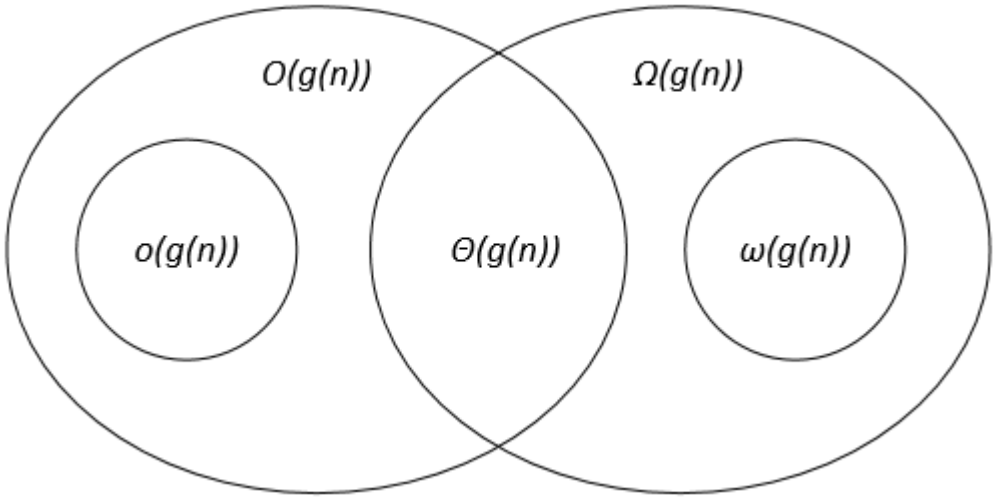
2- هل هي $T(n) = \Omega f(n^2)$ ؟ نعم عندما نجدد $4n^2 + 4n >= C2\, n^2$ هي محققة من أجل كل قيم C2<4 .

بالتالي $T(n) = \theta(n^2)$ لإنها حققت الحالتين.

وهذا طبيعي لإن الاس هو ذاته ومن يحدد هو C (n>=4, C1>4, C2<4)

Algorithm Efficiency 9

2-6- فعالية الخوارزمية 9



مثال(2): إذا كان $T(n) = 4n^2 + 4n$ هل هي $T(n) = \theta f(n^3)$:

هي محققه الشرط الاول وغير محققة الثاني أي:

1- $T(n) = Of(n^3)$ نعم هي Upper bound لإن الاس أكبر.

2-لكن ليست Ω (lower) نظراً لإن الاس أكبر .

إذاً هي ليست θ لعدم تحقق الشرط الثاني. وبالنهاية إذا كانت من نفس الاس ممكن.

ملاحظة: إذا كان الاس للتابع g(n) أكبر فهى Oh أي Upper ولو كان أصغر فهى Ω أي Lower ولو كانت متساويه ستكون θ.

ويوجد small O و small Ω وتتم دراستهما لاحقاً ، إنما تقع خارج حدود هذا المقرر.

Algorithm Efficiency 10

2-6- فعالية الخوارزمية 10

```
#include "stdafx.h"
#include<iostream>
using namespace std;
void main(void)
    {
        int n,i;    float Sum;          float Number;
        cin>>n;
        Sum=0.0;            i =1 ;
        while(i<=n)
            {cin>>Number;           Sum += Number;i++;}
        float mean= Sum / n;cout<<mean<<endl;
        system("pause");
    }
```

Examples of algorithm effectiveness 1

أمثلة على فعالية الخوارزمية 1

```
int   sumOfList( int A[ ], int n)
{
    int   sum = 0, i;
    for(i = 0; i < n; i++)
        sum = sum + A[i];
    return   sum;
}
```

Cost Time require for line (Units)	Repeatation No. of Times Executed	Total Total Time required in worst case
1	1	1
1 + 1 + 1	1 + (n+1) + n	2n + 2
2	n	2n
1	1	1
		4n + 4 Total Time required

Examples of algorithm effectiveness 2

أمثلة على فعالية الخوارزمية 2

```
for (int i = n; i > 1; i /= 2) {
  /* constant time expressions O(1) */
}
```

- بتحليل الخوارزمية سيتم عمل n/2 عدد من المرات (ليكن k) وذلك حتى يصبح العدد i المعبر عن n أقل أو يساوي 1 ونحصل على $n/2^k \leq 1$ بضرب طرفي العلاقة 2^k نحصل على $n \leq 2^k$ ونأخذ لوغاريتم الطرفين نجد $k \geq \log_2 n$ حيث أن k هى عدد العمليات (خطوات k) في الخوارزمية الموضحة أعلاه, وبالتالي فإن درجة تعقيد الخوارزمية هو $\log_2 n$ أي من المرتبة اللوغاريتمية.

- ملاحظة 1 : الخطوة الضرب بـ 2 أو التقسيم عليها, يكون $\log_2$ وإذا كان الضرب بـ x أو التقسيم عليها يكون $\log_x$
- ملاحظة 2 : بشكل عام سوف نهتم للدرجة الأعلى بشكل أساسي ونهمل الدرجات الأقل نظراً لتأثيرها القليل عندما تصبح n كبيرة جداً.

Examples of algorithm effectiveness 3

أمثلة على فعالية الخوارزمية 3

```
function(int n)
{   if (n==1)       return;
   for (int i=1; i<=n; i++)
   {
      for (int j=1; j<=n; j++)
      {        printf("*");        break;
      }
   }
}
```

Examples of algorithm effectiveness 3

أمثلة على فعالية الخوارزمية 3

```
function(int n)
{   if (n==1)       return;
   for (int i=1; i<=n; i++)
   {      // Inner loop executes only one  time due to break statement.
      for (int j=1; j<=n; j++)
      {          printf("*");          break;
      }
   }
}
```

Time Complexity: O(n), Even though the inner loop is bounded by n, but due to the break statement, it is executing only once.

Examples of algorithm effectiveness 4

أمثلة على فعالية الخوارزمية 4

```
void function(int n)
{   int count = 0;
   for (int i=n/2; i<=n; i++)

      for (int j=1; j<=n; j = 2 * j)

         for (int k=1; k<=n; k = k * 2)

            count++;
}
```

Examples of algorithm effectiveness 4

أمثلة على فعالية الخوارزمية 4

```
void function(int n)
{   int count = 0;
   for (int i=n/2; i<=n; i++)
       // Executes n/2  times
      for (int j=1; j<=n; j = 2 * j)
          // Executes O(Log n) times
         for (int k=1; k<=n; k = k * 2)
           // Executes O(Log n) times
             count++;
}
```

Time Complexity: $O(n \log^2 n)$.

Examples of algorithm effectiveness 5

أمثلة على فعالية الخوارزمية 5

```
void function(int n)
{   int count = 0;

   for (int i=n/2; i<=n; i++)

      for (int j=1; j+n/2<=n; j = j++)

         for (int k=1; k<=n; k = k * 2)
            count++;
}
```

Examples of algorithm effectiveness 5

أمثلة على فعالية الخوارزمية 5

```
void function(int n)
{   int count = 0;
    // outer loop executes n/2 times
   for (int i=n/2; i<=n; i++)
       // middle loop executes  n/2 times
      for (int j=1; j+n/2<=n; j = j++)
          // inner loop executes logn times
         for (int k=1; k<=n; k = k * 2)
            count++;
}
```

Time Complexity: $O(n^2 \log n)$.

انتهت محاضرات الأسبوع الأول

Examples of algorithm effectiveness 6

أمثلة على فعالية الخوارزمية 6

مزايا وسلبيات دراسة الحالات الثلاث:

المزايا:

1. تسمح هذه التقنية للمطورين بفهم أداء الخوارزميات في ظل سيناريوهات مختلفة، مما يمكن أن يساعد في اتخاذ قرارات مستنيرة بشأن الخوارزمية التي سيتم استخدامها لمهمة معينة.
2. يوفر تحليل الحالة الأسوأ ضمانًا على الحد الأعلى لوقت تشغيل الخوارزمية، مما يمكن أن يساعد في تصميم خوارزميات موثوقة وفعالة.
3. يوفر متوسط تحليل الحالة تقديرًا أكثر واقعية لوقت تشغيل الخوارزمية، وهو ما يمكن أن يكون مفيدًا في سيناريوهات العالم الحقيقي.

السلبيات:

1. يمكن أن تستغرق هذه التقنية وقتًا طويلاً لأنها تتطلب فهمًا جيدًا للخوارزمية التي يتم تحليلها.
2. لا يوفر تحليل الحالة الأسوأ أي معلومات حول وقت التشغيل النموذجي للخوارزمية، وهو ما يمكن أن يكون عيبًا في سيناريوهات العالم الحقيقي.
3. ويتطلب تحليل الحالة المتوسطة معرفة التوزيع الاحتمالي لبيانات المدخلات، والتي قد لا تكون متاحة دائمًا.

Examples of algorithm effectiveness 7

أمثلة على فعالية الخوارزمية 7

TABLE 1-3 Time for $f(n)$ instructions on a computer that executes 1 billion instructions per second (1 GHz)

n	$f(n) = n$	$f(n) = \log_2 n$	$f(n) = n\log_2 n$	$f(n) = n^2$	$f(n) = 2^n$
10	0.01μs	0.003μs	0.033μs	0.1μs	1μs
20	0.02μs	0.004μs	0.086μs	0.4μs	1ms
30	0.03μs	0.005μs	0.147μs	0.9μs	1s
40	0.04μs	0.005μs	0.213μs	1.6μs	18.3min
50	0.05μs	0.006μs	0.282μs	2.5μs	13 days
100	0.10μs	0.007μs	0.664μs	10μs	4×10^{13} years
1000	1.00μs	0.010μs	9.966μs	1ms	
10,000	10μs	0.013μs	130μs	100ms	
100,000	0.10ms	0.017μs	1.67ms	10s	
1,000,000	1 ms	0.020μs	19.93ms	16.7m	
10,000,000	0.01s	0.023μs	0.23s	1.16 days	
100,000,000	0.10s	0.027μs	2.66s	115.7 days	

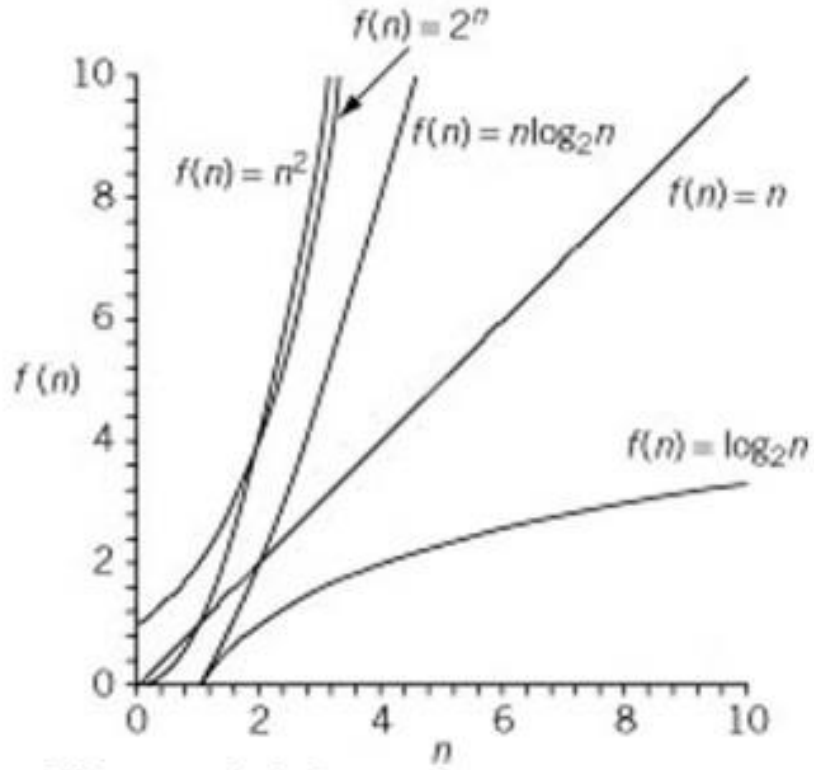


Figure 1-4 Growth rate of functions in Table 1-3

Data Structures Using C++ :

Big-O Examples

Example-1 Find upper bound for $f(n) = 3n + 8$

Solution: $3n + 8 \leq 4n$, for all $n \geq 8$ $\therefore 3n + 8 = O(n)$ with $c = 4$ and $n_0 = 8$ // $\therefore$(therefore)

Example-2 Find upper bound for $f(n) = n^2 + 1$

Solution: $n^2 + 1 \leq 2\,n^2$, for all $n \geq 1$ $\therefore n^2 + 1 = O(n^2)$ with $c = 2$ and $n_0 = 1$

Example-3 Find upper bound for $f(n) = n^4 + 100n^2 + 50$

Solution: $n^4 + 100\,n^2 + 50 \leq 2\,n^4$, for all $n \geq 11$

$\therefore n^4 + 100\,n^2 + 50 = O(n^4)$ with $c = 2$ and $n_0 = 11$

Example-4 Find upper bound for $f(n) = 2\,n^3 - 2n^2$

Solution: $2n^3 - 2n^2 \leq 2n^3$, for all $n > 1$ $\therefore 2n^3 - 2n^2 = O(2n^3)$ with $c = 2$ and $n_0 = 1$

Example-5 Find upper bound for $f(n) = n$

Solution: $n \leq n$, for all $n \geq 1$ $\therefore n = O(n)$ with $c = 1$ and $n_0 = 1$

Example-6 Find upper bound for $f(n) = 410$

Solution: $410 \leq 410$, for all $n > 1$ $\therefore 410 = O(1)$ with $c = 1$ and $n_0 = 1$

Example-1 Find upper bound for $f(n) = 3n + 8$

Solution: $3n + 8 \leq 4n$, for all $n \geq 8$

$\therefore 3n + 8 = O(n)$ with c = 4 and $n_0 = 8$

Example-2 Find upper bound for $f(n) = n^2 + 1$

Solution: $n^2 + 1 \leq 2n^2$, for all $n \geq 1$

$\therefore n^2 + 1 = O(n^2)$ with $c = 2$ and $n_0 = 1$

Example-3 Find upper bound for $f(n) = n^4 + 100n^2 + 50$

Solution: $n^4 + 100n^2 + 50 \leq 2n^4$, for all $n \geq 11$

$\therefore n^4 + 100n^2 + 50 = O(n^4)$ with $c = 2$ and $n_0 = 11$

Example-4 Find upper bound for $f(n) = 2n^3 - 2n^2$

Solution: $2n^3 - 2n^2 \leq 2n^3$, for all $n > 1$

$\therefore 2n^3 - 2n^2 = O(n^3)$ with $c = 2$ and $n_0 = 1$

Example-5 Find upper bound for $f(n) = n$

Solution: $n \leq n$, for all $n \geq 1$

$\therefore n = O(n)$ with $c = 1$ and $n_0 = 1$

Example-6 Find upper bound for $f(n) = 410$

Solution: $410 \leq 410$, for all $n > 1$

$\therefore 410 = O(1)$ with $c = 1$ and $n_0 = 1$

Ω Examples

Example -1 Find lower bound for $f(n) = 5n^2$.

Solution: ∃ C, n_0 Such that: $0 \leq cn^2 \leq 5n^2 \Rightarrow cn^2 \leq 5n^2 \Rightarrow c = 1$ and $n_0 = 1$

$\therefore 5n^2 = \Omega(n^2)$ with $c = 1$ and $n_0 = 1$

Example-2 Prove $f(n) = 100n + 5 \neq \Omega(n^2)$.

Solution: ∃ C, n_0 Such that: $0 \leq cn^2 \leq 100n + 5$

$100n + 5 \leq 100n + 5n \ (\forall n \geq 1) = 105n$

$cn^2 \leq 105n \Rightarrow n(cn - 105) \leq 0$

Since n is positive $\Rightarrow cn - 105 \leq 0 \Rightarrow n \leq 105/c$

⇨ Contradiction: n cannot be smaller then a constant

Example-3 $2n = \Omega(n)$, $n^3 = \Omega(n^3)$, $\log n = \Omega(\log n)$.

Θ Examples

Example 1 Find Θ bound for $f(n) = \frac{n^2}{2} - \frac{n}{2}$

Solution: $\frac{n^2}{5} \leq \frac{n^2}{2} - \frac{n}{2} \leq n^2$ for all, $n \geq 2$

$\therefore \frac{n^2}{2} - \frac{n}{2} = \Theta(n^2)$ with $c_1 = 1/5, c_2 = 1$ and $n_0 = 2$

Example 2 Prove $n \neq \Theta(n^2)$

Solution: $c_1 n^2 \leq n \leq c_2 n^2 \Rightarrow$ only holds for: $n \leq 1/c_1$

$\therefore n \neq \Theta(n^2)$

Example 3 Prove $6n^3 \neq \Theta(n^2)$

Solution: $c_1 n^2 \leq 6n^3 \leq c_2 n^2 \Rightarrow$ only holds for: $n \leq c_2/6$

$\therefore 6n^3 \neq \Theta(n^2)$

Example 4 Prove $n \neq \Theta(logn)$

Solution: $c_1 logn \leq n \leq c_2 logn \Rightarrow c_2 \geq \frac{n}{\log n}, \forall n \geq n_0$ – Impossible

P33 Data Structures And Algorithms Made Easy


Portugal
Spain
Italy
Bul.
Armenia
Turkey
Cyprus
Greece
Syr.
Lebanon
Iraq
Iran
Uzbek.
Turkm.
Kyrgyzsta
Tajikistan
Afghan.
Pakistan
Nepal
Bhuta
Kuwait
Qatar
Jordan
Israel
Saudi
Arabia
U.A.E.
Oman
Yemen
Djibouti
India
Bang
Sri
Morocco
Tunisia
Algeria
Libya
Egypt
Western
Sahara
Maurit.
Mali
Niger
Chad
Eritrea
Sudan
Senegal
e Gambia
B.Faso
ea-Bissau
Guinea
Nigeria
C.A.R.
South
Ethiopia

Job	J1	J2	J3	J4	J5
Deatline	2	1	3	2	1
profit	60	100	20	40	20

Job	J2	J1	J4	J3	J5
Deatline	1	2	2	3	1
profit	100	60	40	20	20

1	2	3
J2	J1	J3

Greedy algorithm, Job sequencing

The Knapsack Problem

Greedy algorithm,

Knapsack algorithm

1-Fractional Knapsack

2- Knapsack

Activity (ai)	1	2	3	4	5
si	4	0	1	1	3
fi	6	2	3	6	4

Here, s_i and f_i are the starting and the finishing time of the activity a_i.

Activity (ai)	2	3	5	1	4
si	0	1	3	4	1
fi	2	3	4	6	6

Sorted according to finish time