ALGORITHMS AND DATA STRUCTURE

محور الخوارزميات وبنى المعطيات

2020

By Eng. Joud Khattab

Content

- OOP. (البرمجة غرضية التوجه)
- Complexity. (التعقيد)
- Recursive Functions. (التوابع العودية)
- Data Structure: (بنى المعطيات)
  - Sequential: Linked List, Stacks, Queue.
  - Non Sequential: Tree, Graph.
- Sort & Search Algorithms. (خوارزميات البحث والترتيب)
- Advance Algorithms Methods (خوارزميات متقدمة)
  - Greedy, Backtracking, Dynamic, Divide & Conquer.
- Hashing Function. (توابع التقطيع)
- MCQ. (اختبارات)

Definitions (تعاريف)

- خوارزمية حل مسألة
  - هي توصيف صوري لطريقة الحل على شكل متتالية منتهية من العمليات البسيطة، تنفذ حسب تسلسل محدد.
- البرنامج

  هو توصيف لخوارزمية حل مسألة معينة بإحدى لغات البرمجة التي يقبلها الحاسوب.
- لغة البرمجة
  - هي مجموعة من المفردات والقواعد والدلالات المعرفة التي تسمح بكتابة برنامج يمكن تنفيذه على الحاسوب.
- المترجم
  - هو برنامج يفهم البرنامج المكتوب بلغة برمجة معينة، ويحوله إلى برنامج مكافئ مكتوب بلغة المجمع الخاصة بالمعالج الصغري للحاسوب.

OBJECT ORIENTED PROGRAMMING (OOP)

البرمجة غرضية التوجه

By Eng. Joud Khattab

Definitions (تعاريف)

- الصف أو الفئة (class)
  - الصف عبارة عن حاوية كبيرة تستطيع أن تحتوي كل الكود من متغيرات ودوال وكائنات إلخ...
- الكائن أو الغرض (object)
  - الكائن عبارة عن نسخة مطابقة لصف معين.
  - يمكننا القول إنه لا يمكن إنشاء كائن إذا لم يكن هناك صف.
  - في مفهوم برمجة الكائنات نقوم بإنشاء صف معين يسمونه (blue print) أي (النسخة الخام أو النسخة الأصلية)، وبعدها ننشئ نسخة أو أكثر من هذا الصف.

Definitions (تعاريف)

Definitions (تعاريف)

- الخصائص (instance variables)
  - كل كائن ننشئه من صف معين يملك نسخة خاصة به من المتغيرات الموجودة في هذا الصف.
  - المتغيرات التي يتم إعطاء نسخة منها لكل كائن من الصف هي ما تسمى بالخصائص.
- الدوال (methods)
  - الدالة عبارة عن كود جاهز يتنفذ فقط عندما تقوم باستدعائه.
- البيانات نوعين:
  - البيانات البدائية (primitive data types): كال byte – short – int – long – float – bool
  - البيانات المرجعية (reference/object data types): كالكائنات وأي نوع نضع له الكلمة new عند تعريفه.

Definitions (تعاريف)

- التابع المشيد أو البناء (constructor)
  - عبارة عن دالة لها نوع خاص، يتم استدعائها أثناء إنشاء كائن لتوليد قيم أولية للخصائص الموجودة فيه.
- نقاط مهمة حول ال constructor
  - كل صف يتم إنشاءه، يحتوي على constructor واحد على الأقل. وحتى إن لم تقم بتعريف أي constructor سيقوم المترجم بإنشاء واحد افتراضي عنك.
  - في كل مرة يتم إنشاء كائن جديد، يجب استدعاء constructor حتى يتم إنشاء هذا الكائن.
  - القاعدة الأساسية عند تعريف constructor هي أنه يجب أن يحمل نفس اسم الصف ويكون نوعه public
  - في حال قمت بتعريف constructor لن يقوم المترجم بإنشاء واحد افتراضي، أي لن يعود هناك default constructor
  - يمكنك تعريف أكثر من constructor. ويمكنك دائما إنشاء constructor فارغ، حتى تستخدمه إن كنت لا تريد إعطاء قيم أولية محددة للخصائص عند إنشاء كائن.

Definitions (تعاريف)

- المتغيرات التي يتم وضعها في الصف تقسم إلى ثلاث فئات أساسية:
  - Local variables: هي المتغيرات التي يتم تعريفها بداخل أي دالة.
  - Instance variables: هي المتغيرات التي يتم تعريفها بداخل الكلاس وخارج حدود أي دالة.
  - Class variables: هي المتغيرات التي يتم تعريفها ك static بداخل الكلاس. وخارج حدود أي دالة.

Definitions (تعاريف)

- مفهوم superclass و subclass
  - الصف الذي يرث من صف أخر يسمى subclass ويسمى أيضا (derived class او extended class او child class)
  - الصف الذي يورث محتوياته لصف أخر يسمى superclass ويسمى أيضا (base class او parent class)
- مفهوم super
  - تستخدم للتمييز بين المتغيرات والدوال الموجودة في ال superclass و subclass في حال كانت الأسماء مستخدمة في كلا الصفين.
  - تستخدم لاستدعاء المشيد الموجود في ال superclass

Question 1

- يمكن تطبيق مفهوم البرمجة غرضية التوجه في البرامج بدون استخدام الصفوف
  A. صح
  B. خطأ

Question 1

- يمكن تطبيق مفهوم البرمجة غرضية التوجه في البرامج بدون استخدام الصفوف
  A. صح
  B. خطأ

Question 2

- ما هي الخاصية الإضافية الموجودة في الصفوف (classes) وليست موجودة في البنية (struct)

A. Data members
B. Member functions
C. Static data allowed
D. Public access specifier

Question 2

- ما هي الخاصية الإضافية الموجودة في الصفوف (classes) وليست موجودة في البنية (struct)

A. Data members
B. Member functions
C. Static data allowed
D. Public access specifier

Question 3

- ما هو التعريف الأفضل للصفوف (class)

A. Parent of an object
B. Instance of an object
C. Blueprint of an object
D. Scope of an object

Question 3

- ما هو التعريف الأفضل للصفوف (class)

A. Parent of an object
B. Instance of an object
C. Blueprint of an object
D. Scope of an object

Note
What Is a Class? (ما هو الصف)

- In the real world, you'll often find many individual objects all of the same kind.
- There may be thousands of other bicycles in existence, all of the same make and model.
- Each bicycle was built from the same set of blueprints and therefore contains the same components.
- In object-oriented terms, we say that your bicycle is an instance of the class of objects known as bicycles.
- A class is the blueprint from which individual objects are created.

Question 4

- ما هي الخاصية الموجودة في البرمجة غرضيه التوجه التي توضح وتركز على مفهوم إعادة الاستخدام في البرمجة (reusability)

A. Polymorphism
B. Abstraction
C. Encapsulation
D. Inheritance

Question 4

- ما هي الخاصية الموجودة في البرمجة غرضيه التوجه التي توضح وتركز على مفهوم إعادة الاستخدام في البرمجة (reusability)

A. Polymorphism
B. Abstraction
C. Encapsulation
D. Inheritance

Question 6

- ما هو التعريف المناسب لشرح مفهوم التغليف (Encapsulation)

A. It is a way of combining various data members into a single unit
B. It is a way of combining various member functions into a single unit
C. It is a way of combining various data members and member functions into a single unit which can operate on any data
D. It is a way of combining various data members and member functions that operate on those data members into a single unit

Question 6

- ما هو التعريف المناسب لشرح مفهوم التغليف (Encapsulation)

A. It is a way of combining various data members into a single unit
B. It is a way of combining various member functions into a single unit
C. It is a way of combining various data members and member functions into a single unit which can operate on any data
D. It is a way of combining various data members and member functions that operate on those data members into a single unit

Question 7

- أي من الميزات التالية تتناسب مع بعضها البعض؟

A. Inheritance and Encapsulation
B. Encapsulation and Polymorphism
C. Encapsulation and Abstraction
D. Abstraction and Polymorphism

Question 7

- أي من الميزات التالية تتناسب مع بعضها البعض؟

A. Inheritance and Encapsulation
B. Encapsulation and Polymorphism
C. Encapsulation and Abstraction
D. Abstraction and Polymorphism

Note
OOP Features (خصائص البرمجة غرضية التوجه)

- Abstraction (التجريد)
  - Abstraction is a process where you show only “relevant” data and “hide” unnecessary details of an object from the user.
- Encapsulation (التغليف)
  - Encapsulation is binding the data with the code that manipulates it. It keeps the data and the code safe from external interference.
- Inheritance (الوراثة)
  - Inheritance is the mechanism by which an object acquires the some/all properties of another object. It supports the concept of hierarchical classification.
- Polymorphism (تعدد الأوجه)
  - Polymorphism means to process objects differently based on their data type.

Note
OOP Features (خصائص البرمجة غرضية التوجه)

- مفهوم التجريد (abstraction)
  - هو أسلوب يستخدم لتسهيل كتابة الأوامر على المبرمجين، فهو يجعلك قادرا على تنفيذ ما تريد دون الحاجة إلى معرفة كافة التفاصيل التي تم فيها تنفيذ ذلك.
  - اذا ال abstraction يجعلك تتعامل مع الأشياء بسطحية بدل أن تغوص في معرفة تفاصيل الكودات المعقدة.
- مفهوم التغليف (encapsulation)
  - هو أسلوب يمكن اتباعه لإخفاء البيانات الأساسية في الصف، أي لإخفاء الخصائص الموجودة فيه (global variables)، وجعل الصفوف الأخرى قادرة على التعامل مع هذه الخصائص فقط من خلال دوال يقوم بإنشائها المبرمج الأساسي للصف.
  - لتحقيق مبدأ التغليف، عليك تعريف الخصائص ك private وتعريف الدوال التي تستخدم للوصول اليهم ك public
  - مثال على التغليف: setter and getter

Note
OOP Features (خصائص البرمجة غرضية التوجه)

- مفهوم ال polymorphism
  - المقصود منه بناء دالة تنفذ أوامر مختلفة على حسب الكائن الذي يمرر لها ك argument
  - تعني قدرة الكائن على أخذ عدة اشكال، تحديدا قدرة الكائن على التحول إلى نوع مشتق منه.

Note
Polymorphism

- Overloading (Static Polymorphism)
  - Overloading in simple words means more than one method having the same method name that behaves differently based on the arguments passed while calling the method.
- Overriding (Dynamic Polymorphism)
  - Overriding means a derived class is implementing a method of its super class. The call to overridden method is resolved at runtime, thus called runtime polymorphism.

Note
Overriding

- Overriding:
  - تعني تعريف الدالة التي ورثها ال subclass من ال superclass من جديد.
  - هذه الدالة الجديدة تكون مشابهة للدالة الموروثة من حيث الشكل فقط. أي لها نفس الاسم والنوع وعدد الباراميترات لكن محتواها مختلف.
  - الهدف هو اتاحة الفرصة لل subclass ليعرف الدوال حسب حاجته

Note
Overriding

- شروط ال overriding:
  - يجب أن يكون ال modifier المستخدم للدالة الجديدة هو نفسه المستخدم للدالة القديمة، ويجب أن يكون نوعه public او protected.
  - الدالة المعرفة ك final لا يمكن أن نفعل لها override، لأن كلمة final تمنع تغير محتوى الدالة بعد تعريفها.

Question 8

- ماذا تسمي اللغة التي تدعم الصفوف (classes) ولا تدعم تعدد الاشكال (polymorphism)؟

A. Class based language
B. Procedure oriented language
C. Object based language
D. If classes are supported, polymorphism will always be supported

Question 8

- ماذا تسمي اللغة التي تدعم الصفوف (classes) ولا تدعم تعدد الاشكال (polymorphism)؟

A. Class based language
B. Procedure oriented language
C. Object based language
D. If classes are supported, polymorphism will always be supported

Note
Programming Paradigm

- The main programming paradigms:
  1. Imperative
     - Cobol, Fortran, C, Ada, Perl
  2. Object-oriented
     - Smalltalk, Java, C++, C#, Python
  3. Functional
     - Lisp, Scheme, ML, Haskell
  4. Logic
     - Prolog
  5. Declarative
     - SQL, HTML

Question 9

- If data members are private, what can we do to access them from the class object?
  A. Create public member functions to access those data members
  B. Create private member functions to access those data members
  C. Create protected member functions to access those data members
  D. Private data members can never be accessed from outside the class

Question 9

- If data members are private, what can we do to access them from the class object?
  A. Create public member functions to access those data members
  B. Create private member functions to access those data members
  C. Create protected member functions to access those data members
  D. Private data members can never be accessed from outside the class

Question 10

- أي من الخيارات التالية ينتهك مفهوم التغليف (encapsulation) دائما؟

A. Local variables
B. Global variables
C. Public variables
D. Array variables

Question 10

- أي من الخيارات التالية ينتهك مفهوم التغليف (encapsulation) دائما؟

A. Local variables
B. Global variables
C. Public variables
D. Array variables

Question 11

- أي مما يلي ليس شرطا ضروريا للبناة (constructor)؟

A. Its name must be same as that of class
B. It must not have any return type
C. It must contain a definition body
D. It can contains arguments

Question 11

- أي مما يلي ليس شرطا ضروريا للبناة (constructor)؟

A. Its name must be same as that of class
B. It must not have any return type
C. It must contain a definition body
D. It can contains arguments

Question 12

- How many types of constructors are available for use in general (with respect to parameters)?
  A. 2
  B. 3
  C. 4
  D. 5

Question 12

- How many types of constructors are available for use in general (with respect to parameters)?
  - A. 2
  - B. 3
  - C. 4
  - D. 5

Question 13

- Default constructor must be defined, if parameterized constructor is defined and the object is to be created without arguments.
  A. True
  B. False

Question 13

- Default constructor must be defined, if parameterized constructor is defined and the object is to be created without arguments.
  A. True
  B. False

Question 14

- If class C inherits class B. And B has inherited class A. Then while creating the object of class C, what will be the sequence of constructors getting called?
  A. Constructor of C then B, finally of A
  B. Constructor of A then C, finally of B
  C. Constructor of C then A, finally B
  D. Constructor of A then B, finally C

Question 14

- If class C inherits class B. And B has inherited class A. Then while creating the object of class C, what will be the sequence of constructors getting called?
  A. Constructor of C then B, finally of A
  B. Constructor of A then C, finally of B
  C. Constructor of C then A, finally B
  D. Constructor of A then B, finally C

COMPLEXITY

تعقيد الخوارزميات

By Eng. Joud Khattab

2020

Complexity (تعقيد الخوارزميات)

- يرتبط بعاملين اساسيين:
  - حجم الذاكرة:
    - اللازمة لتخزين البرنامج والمعطيات التي يعالجها.
  - زمن التنفيذ:
    - الزمن اللازم لتنتهي الخوارزمية من إنجاز تعليماتها كافة.
    - يقاس زمن التنفيذ بحساب عدد التعليمات والزمن اللازم لتنفيذ كل تعليمة.

Complexity (تعقيد الخوارزميات)

- يعتمد زمن تنفيذ البرنامج على حاسب على عوامل عديدة منها:
  - معطيات المسألة الخاصة بتجربة ما.
  - جودة الرماز الذي يولده المترجم من أجل بناء الملف التنفيذي.
  - طبيعة وسرعة التعليمات المتوفرة في الحاسوب (في المعالج الصغري).
  - فعالية الخوارزمية. (تعتمد على قدرة المبرمج)

Complexity (تعقيد الخوارزميات)

- لحساب زمن تنفيذ خوارزمية يجب تحديد بعد (او طول او حجم) معطيات المسألة من أجل كتابة درجة التعقيد بدلالة هذا البعد.
- أمثلة:
  - مسائل كثيرات الحدود:
    - يكون البعد هو درجة كثيرات الحدود.
  - مسائل المصفوفات:
    - يكون البعد بدلالة أبعاد المصفوفة.
  - مسائل البيانات:
    - يكون البعد بدلالة عدد العقد أو الأسهم أو مجموعها.
  - مسائل الفرز:
    - يكون البعد بدلالة عدد العناصر التي نريد ترتيبها.
  - مسائل التحليل القواعدي:
    - يكون البعد بدلالة طول الكلمة.

Complexity (تعقيد الخوارزميات)

- تذكرة ببعض القوانين الرياضية:

$$\sum_i cf(i) = c\sum_i f(i), \text{where c is constant}$$

$$\sum\big(f(i)+g(i)\big)=\sum f(i)+\sum g(i)$$

$$\sum_{i=a}^{b} c = c+c+\ldots+c = \big(|b-a|+1\big)c, \text{where c is constant}$$

$$\sum_{\mathrm{i}=1}^{\mathrm{m}} \mathrm{i} = 1+2+3+\ldots+m = \frac{m(m+1)}{2}$$

Complexity (تعقيد الخوارزميات)

- تذكرة ببعض القوانين الرياضية:

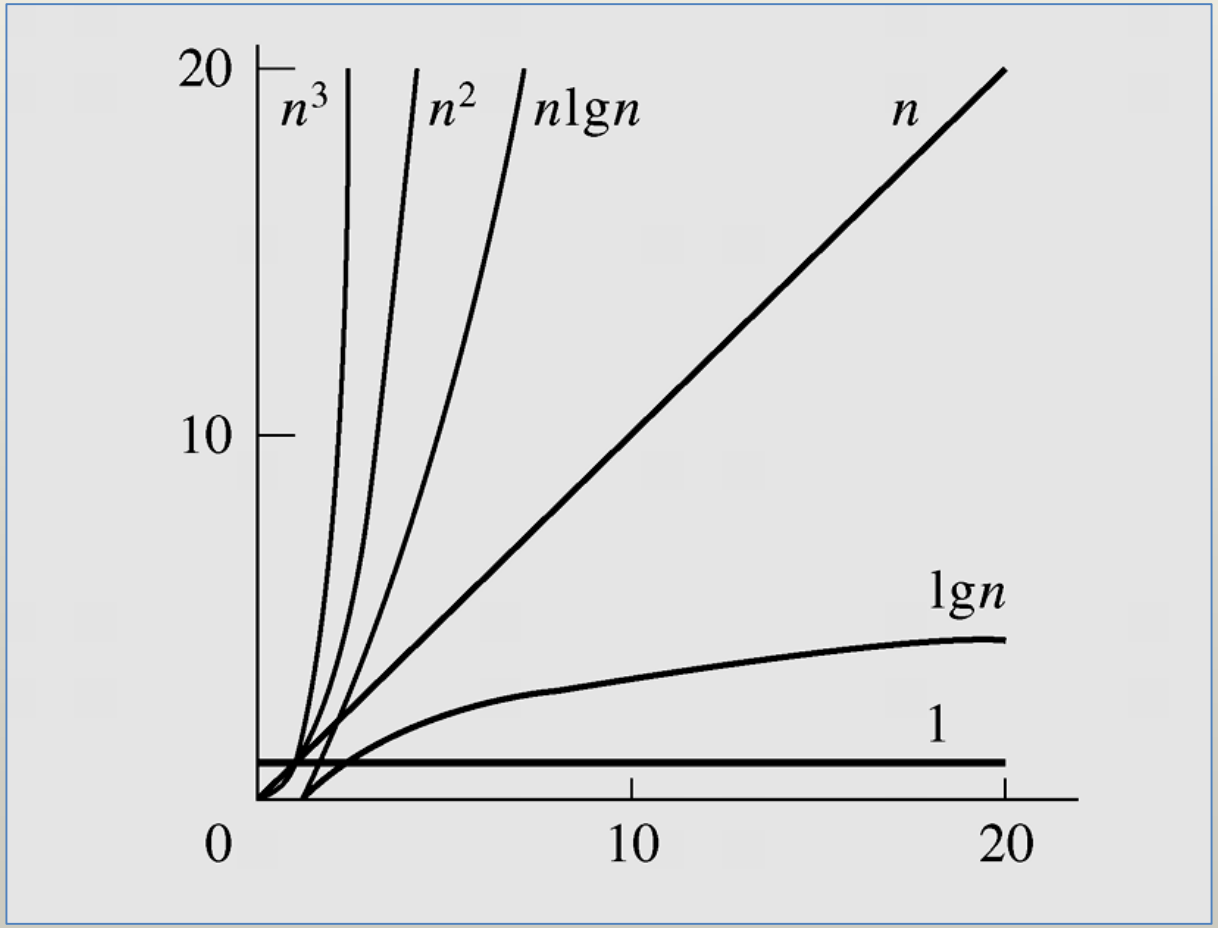
$$\sum_{i=1}^{N} i = \frac{N(N+1)}{2}$$

$$\sum_{i=1}^{N} i^2 = \frac{N(N+1)(2N+1)}{6}$$

$$\sum_{i=1}^{N} i^3 = \frac{N^2(N+1)^2}{4}$$

$$\sum_{i=1}^{N} \frac{1}{i} \approx \log n$$

Complexity (تعقيد الخوارزميات)



Complexity (تعقيد الخوارزميات)

	$T(n)$						
n	n	$n \log n$	n^2	n^3	n^4	n^{10}	2^n
10	.01µs	.03µs	.1µs	1µs	10µs	10s	1µs
20	.02µs	.09µs	.4µs	8µs	160µs	2.84h	1ms
30	.03µs	.15µs	.9µs	27µs	810µs	6.83d	1s
40	.04µs	.21µs	1.6µs	64µs	2.56ms	121d	18m
50	.05µs	.28µs	2.5µs	125µs	6.25ms	3.1y	13d
100	.1µs	.66µs	10µs	1ms	100ms	3171y	4×10^{13}y
10^3	1µs	9.96µs	1ms	1s	16.67m	3.17×10^{13}y	32×10^{283}y
10^4	10µs	130µs	100ms	16.67m	115.7d	3.17×10^{23}y	
10^5	100µs	1.66ms	10s	11.57d	3171y	3.17×10^{33}y	
10^6	1ms	19.92ms	16.67m	31.71y	3.17×10^{7}y	3.17×10^{43}y	

Complexity (تعقيد الخوارزميات)
Exercise

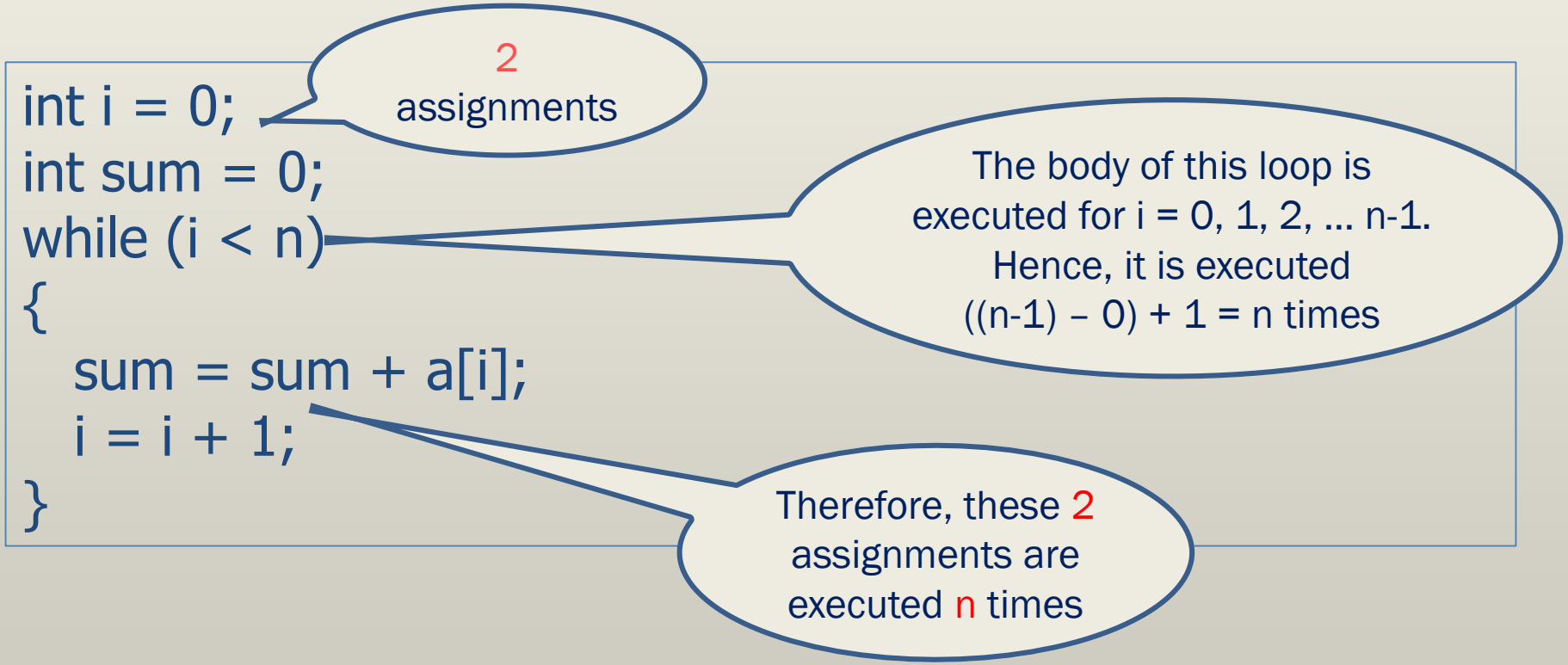
```
// a simple loop to calculate the sum of numbers in an array
int i , sum;
for (i = sum = 0; i<n ; i++)
   sum += a[i];
```

Complexity (تعقيد الخوارزميات)
Rule

If a loop iterates for i = **initial value** to **final value** in **step** of 1 then:

Number of iterations of the loop = final value – initial value + 1

Complexity (تعقيد الخوارزميات) Exercise



Answer: Number of Assignment Operations =
2 + n + n = 2n + 2

Complexity (تعقيد الخوارزميات)
Question 1

- السؤال: رتب التعقيدات التالية تصاعديا
  - n
  - n*n
  - n*logn
  - logn
- الإجابة:
  1. logn
  2. n
  3. n*logn
  4. n*n

Complexity (تعقيد الخوارزميات)
Question 2

- السؤال: أصغر تعقيد لخوارزمية الرفع لقوة
  - n
  - n*n
  - n*logn
  - logn
- الإجابة:
  - logn

Complexity (تعقيد الخوارزميات)
Question 2

```
int power (int x, int y) {
        int p = 1;
        for (int i = 1; i<=y; i++) {
                p = p*x; }
        return p; }
```

```
int power2 (int x, int y) {
        int res = 1;
        int factor = x;
        while (y>0) {
                if (y%2==1)
                        res = res*factor;
                factor = factor*factor;
                y = y / 2; }
        return res; }
```

Complexity (تعقيد الخوارزميات)

- تصنيف المسائل حسب زمن التنفيذ:
  - O(1): خوارزميات البحث في جداول التقطيع. (Hash Tables)
  - Log n: خوارزميات البحث الثنائي. (Binary Search)
  - n: خوارزميات البحث التسلسلي. (Search)
  - n * Log n: خوارزميات الفرز الجيدة. (Sort)
  - n*n: ضرب المصفوفات (تعالج معطيات متوسطة الحجم).
  - 2^n: أسي لا تستخدم إلا في حالة معطيات ذات حجم محدود.

Complexity (تعقيد الخوارزميات)

- يمكن ان نقول إن الخوارزميات الجيدة لحل مسألة معينة هي التي يكون زمن تنفيذها:
  - ثابتا مهما كان حجم المعطيات: مثل خوارزميات البحث في جداول التقطيع.
  - لوغاريتميا: مثل خوارزميات البحث الثنائي.
  - خطيا: مثل خوارزمية البحث التسلسلي.
  - من مرتبة n*log(n): مثل خوارزميات الفرز الجيدة.

Complexity (تعقيد الخوارزميات)
Question 3

```
void Main()
{
    for (int i=1; i<=n; i++)
    {
       m=m+1;
       p=p*2;
       l=l+3;
    }
}
```

- السؤال:
  - احسب تعقيد الخوارزميات التالية:

- الإجابة:

$$\sum_{i=1}^{n} 3 * 1 = 3 * \sum_{i=1}^{n} 1 = 3n = O(n)$$

Complexity (تعقيد الخوارزميات)
Question 4

```
void Main()
{
for (int i=1; i<=n; i++) {
    for (int j=1; j<=n; j++){
        m=m+1;
        p=p*2;
        l=l+3; } }
}
```

- السؤال:
  - احسب تعقيد الخوارزميات التالية:

- الإجابة:

$$\sum_{i=1}^{n}\sum_{j=1}^{n} 3 = 3 * \sum_{i=1}^{n}\sum_{j=1}^{n} 1 =$$

$$3\sum_{i=1}^{n} n = 3 * n\sum_{i=1}^{n} 1 = 3 * n * n = 3n^2 = O(n^2)$$

Complexity (تعقيد الخوارزميات)
Question 5

```
void Main()
{
for (int i=1; i<=n; i++) {
    for (int j=1; j<=i; j++){
        m=m+1;
        p=p*2;
        l=l+3; } }
}
```

- السؤال:
  - احسب تعقيد الخوارزميات التالية:

- الإجابة:

$$\sum_{i=1}^{n}\sum_{j=1}^{i} 3 = 3 * \sum_{i=1}^{n}\sum_{j=1}^{i} 1 =$$

$$3\sum_{i=1}^{n} i = 3\,\frac{n(n+1)}{2} = O(n^2)$$

Complexity (تعقيد الخوارزميات)
Question 6

```
void Main()
{
for (int cnt=5,i=1; i<=n; i++)
    for (int j=1; j<=n; j=j*2)
        cnt++;
}
```

- السؤال:
  - احسب تعقيد الخوارزميات التالية:
- الإجابة:
  - n * log n

RECURSIVE FUNCTIONS

الخوارزميات العودية

By Eng. Joud Khattab

Recursive Functions
(الخوارزميات العودية)

- هي خوارزمية تستدعي نفسها وينتهي الاستدعاء بتحقق شرط للتوقف.
- أمثلة:
  - حساب اعداد فيبوناتشي.
  - سلسلة أكريمان.
  - أبراج هانوي.
  - رسم المنحنيات.
  - مسألة الوزراء الثمانية.
  - منحنيات هلبرت.
- يؤول حساب تعقيد الخوارزمية العودية إلى حل معادلة تراجعية.

Recursive Functions
(الخوارزميات العودية)

- يمكن التمييز بين نوعين من الإجرائيات العودية:
  - الإجرائيات ذات العودية المباشرة:
    - نقول عن إجرائية P إنها عودية مباشرة إذا كانت تحوي استدعاء صريح لنفسها.
  - الإجرائيات ذات العودية غير المباشرة:
    - نقول عن إجرائية P إنها عودية غير مباشرة إذا كانت تستدعي إجرائية أخرى Q تستدعي P.

Recursive Functions
(الخوارزميات العودية)

- عملية الاستدعاء العودي مكلفة من حيث الزمن التنفيذ وحجم الذاكرة المطلوبة.
- يمكن تحويل خوارزمية عودية إلى تكرارية باستخدام مكدس
  - حيث يتم تخزين معاملات الدخل والمتحولات المحلية لكل استدعاء عودي فيه ثم يتم حذف عناصر المكدس لتنفيذ بقية التعليمات بعد الاستدعاء العودي على هذه العناصر حيث يتم التنفيذ من الاستدعاء الأخير إلى الأول.

Recursive Functions (الخوارزميات العودية)
Question 1

- السؤال: ما النتيجة إذا كان الاستدعاء f(33,33)

```
int f(int x, int y) {
        if (x==1)           return y;
        else if (y==1)      return x;
        else if (x%2==0)    return y+f(x/2,y);
        else                return x+f(x,y/2); }
```

- الإجابة:

198

Recursive Functions (الخوارزميات العودية)
Question 1

- السؤال: ما النتيجة إذا كان الاستدعاء f(22,22)

```
int f(int x, int y) {
        if (x==1)           return y;
        else if (y==1)      return x;
        else if (x%2==0)    return y+f(x/2,y);
        else                return x+f(x,y/2); }
```

- الإجابة:

77

Recursive Functions (الخوارزميات العودية)
Question 2

- السؤال: ما النتيجة إذا كان الاستدعاء f(1024*1024)

```
Public int f(int n) {
        int x=0;
        for (int i=2; i<=n ; i*=2) {
                x++; }
        return x; }
```

- الإجابة:

$f(2^{10}*2^{10})=20$

Recursive Functions (الخوارزميات العودية)
Question 3

- السؤال: ما النتيجة إذا كان الاستدعاء f(33)

```
Int f(int n) {
        for (int i=1;i<=n;i++) {
                i=i+1;
                n=n-1; }
        return n; }
```

- الإجابة:

22

Recursive Functions (الخوارزميات العودية)
Question 3

- السؤال: ما النتيجة إذا كان الاستدعاء f(10)

```
Int f(int n) {
        for (int i=1;i<=n;i++) {
                i=i+1;
                n=n-1; }
        return n; }
```

- الإجابة:

6

Recursive Functions (الخوارزميات العودية)
Question 4

```
int fun(int n) {
    if (n == 4)
       return n;
    else return 2*fun(n+1); }

int main() {
   printf("%d ", fun(2));
   return 0; }
```

1. 4
2. 8
3. 16
4. Runtime error.

Recursive Functions (الخوارزميات العودية)
Question 4

```
int fun(int n) {
    if (n == 4)
       return n;
    else return 2*fun(n+1); }

int main() {
   printf("%d ", fun(2));
   return 0; }
```

1. 4
2. 8
3. 16
4. Runtime error.

Recursive Functions (الخوارزميات العودية)
Question 4

```
int fun(int n) {
    if (n == 4)
       return n;
    else return 2*fun(n+1); }

int main() {
   printf("%d ", fun(6));
   return 0; }
```

1. 4
2. 8
3. 16
4. Runtime error.

Recursive Functions (الخوارزميات العودية)
Question 4

```
int fun(int n) {
    if (n == 4)
       return n;
    else return 2*fun(n+1); }

int main() {
   printf("%d ", fun(6));
   return 0; }
```

1. 4
2. 8
3. 16
4. Runtime error.

Recursive Functions (الخوارزميات العودية)
Question 5

```
int fun(int x, int y) {
  if (x == 0)
    return y;
  return fun(x - 1,  x + y); }


int main() {
   printf("%d ", fun(4,3));
   return 0; }
```

1. 13
2. 12
3. 9
4. 10

Recursive Functions (الخوارزميات العودية)
Question 5

```
int fun(int x, int y) {
  if (x == 0)
    return y;
  return fun(x - 1,  x + y); }


int main() {
   printf("%d ", fun(4,3));
   return 0; }
```

1. 13
2. 12
3. 9
4. 10

The rule: x(x+1)/2 + y

Recursive Functions (الخوارزميات العودية)
Question 6

```
n = 25


void fun(int n) {
  if (n == 0)
      return;
  printf("%d", n%2);
  fun(n/2); }
```

1. 11001
2. 10011
3. 11111
4. 00000

Recursive Functions (الخوارزميات العودية)
Question 6

```
n = 25


void fun(int n) {
  if (n == 0)
      return;
  printf("%d", n%2);
  fun(n/2); }
```

1. 11001
2. 10011
3. 11111
4. 00000

Recursive Functions (الخوارزميات العودية)
Question 7

```
n = 25


void fun(int n) {
  if (n == 0)
      return;
  fun(n/2);
  printf("%d", n%2); }
```

1. 11001
2. 10011
3. 11111
4. 00000

Recursive Functions (الخوارزميات العودية)
Question 7

```
n = 25


void fun(int n) {
  if (n == 0)
      return;
  fun(n/2);
  printf("%d", n%2); }
```

1. 11001
2. 10011
3. 11111
4. 00000

Recursive Functions (الخوارزميات العودية)
Question 8

What does the following function do?

```
int fun(int x, int y)
{
    if (y == 0)   return 0;
    return (x + fun(x, y-1));
}
```

1. x+y
2. x+x*y
3. x*y
4. x^y

Recursive Functions (الخوارزميات العودية)
Question 8

What does the following function do?

```
int fun(int x, int y)
{
    if (y == 0)   return 0;
    return (x + fun(x, y-1));
}
```

1. x+y
2. x+x*y
3. x*y
4. x^y

Recursive Functions (الخوارزميات العودية)
Question 9

```
int f(int n) {
    if(n <= 1)
        return 1;
    if(n%2 == 0)
        return f(n/2);
    return f(n/2) + f(n/2+1);  }

int main()  {
    printf("%d", f(11));
    return 0;  }
```

1. Stack overflow
2. 3
3. 4
4. 5

Recursive Functions (الخوارزميات العودية)
Question 9

```
int f(int n) {
    if(n <= 1)
        return 1;
    if(n%2 == 0)
        return f(n/2);
    return f(n/2) + f(n/2+1);  }

int main()  {
    printf("%d", f(11));
    return 0;  }
```

1. Stack overflow
2. 3
3. 4
4. 5

Recursive Functions (الخوارزميات العودية)
Question 10

Consider the following: f(5)

```
int fun (int n)
{
  int x=1, k;
  if (n==1) return x;
  for (k=1; k<n; ++k)
     x = x + fun(k) * fun(n – k);
  return x;
}
```

1. 0
2. 26
3. 51
4. 71

Recursive Functions (الخوارزميات العودية)
Question 10

Consider the following: f(5)

```
int fun (int n)
{
  int x=1, k;
  if (n==1) return x;
  for (k=1; k<n; ++k)
     x = x + fun(k) * fun(n – k);
  return x;
}
```

1. 0
2. 26
3. 51
4. 71

Recursive Functions (الخوارزميات العودية)
Question 11

If get(6) function is being called. how many times will the get() function be invoked?

```
void get (int n) {
   if (n < 1) return;
   get(n-1);
   get(n-3);
   printf("%d", n);  }
```

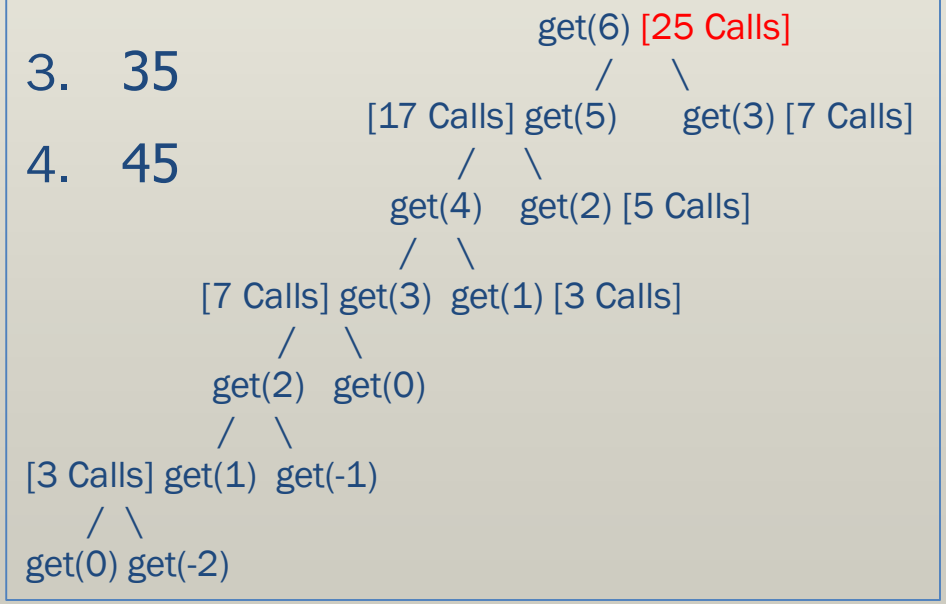
1. 15
2. 25
3. 35
4. 45

Recursive Functions (الخوارزميات العودية)
Question 11

If get(6) function is being called. how many times will the get() function be invoked?

```
void get (int n) {
   if (n < 1) return;
   get(n-1);
   get(n-3);
   printf("%d", n);  }
```

1. 15
2. 25
3. 35
4. 45



SEQUENTIAL DATA STRUCTURE

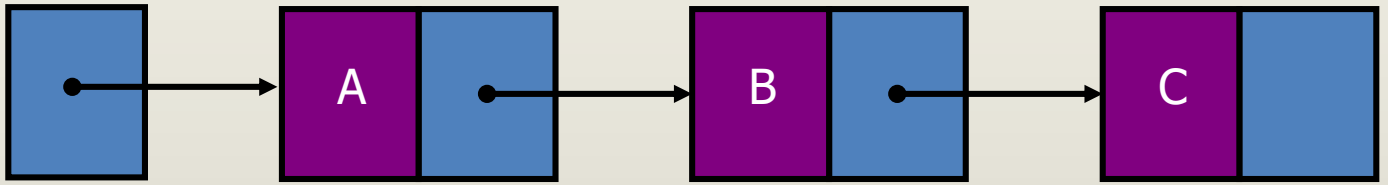
بنى المعطيات التسلسلية

By Eng. Joud Khattab

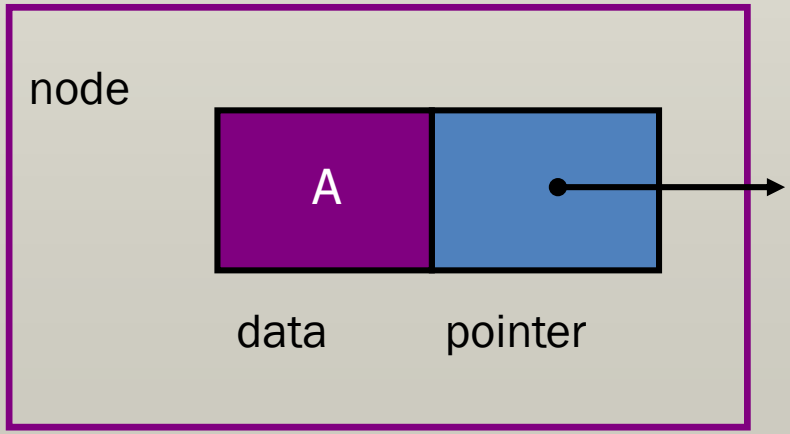
Linked List (السلاسل الخطية)

- هي بنية معطيات تمكن من تخزين عناصر من نفس النمط.
- ميزات السلاسل:
  - لا ضرورة لتحديد طولها مسبقا.
  - لا نحتاج إلى ازاحة عناصر السلسلة عند الحذف والاضافة.
- سيئاتها:
  - لا يمكن النفاذ المباشر إلى عنصر, وبالتالي تحتاج إلى n عملية مقارنة في أسوأ الاحوال للوصول إلى العنصر.
  - نحتاج إلى حجم تخزين إضافي لتخزين المؤشرات.
- يوجد أنواع أخرى للسلاسل مثل:
  - السلاسل الدائرية circular linked list.
  - السلاسل المضاعفة الارتباط doubly linked list.

Linked List (السلاسل الخطية)



Head



Linked List (السلاسل الخطية) V.S. Ordinary and Dynamic Arrays

Ordinary Arrays

- الميزات:
  - من السهل جدا إنشاء واستخدام.
  - الوصول المباشر إلى أي عنصر.
- السيئات:
  - يجب أن يكون حجمها معروفًا في وقت التحويل البرمجي، ولا يمكن تغييره في وقت التشغيل.
  - يتطلب إدراج عنصر أو حذفه نقل كل العناصر الأخرى.

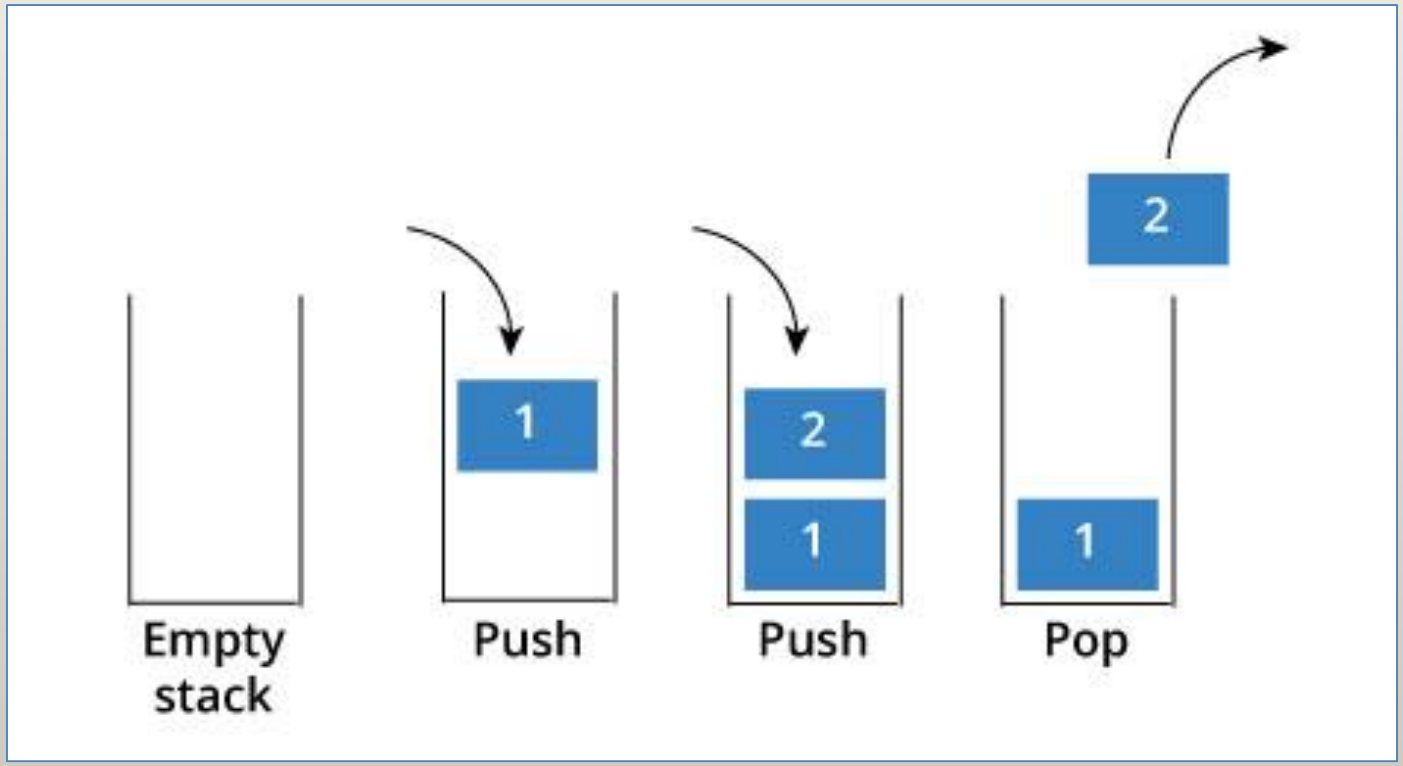
Dynamic Arrays

- الميزات:
  - من السهل جدا إنشائها.
  - الوصول المباشر إلى أي عنصر.
  - يمكن تحديد الحجم في وقت التشغيل.
- السيئات:
  - يتطلب إدراج عنصر أو حذفه نقل كل العناصر الأخرى.
  - إذا كان على المصفوفة أن تتخطى الحجم الأولي ، فيجب إعادة تشكيل المصفوفة بأكملها إلى موقع جديد.

Stack (المكدس)

- هي بنية خطية تشبه السلسلة، غير إن عمليات الإضافة والحذف تتم من جهة واحدة ندعوها قمة المكدس.
- هي بنية معطيات تمكن من تخزين عناصر من نفس النمط.
- تتبع مبدأ :
  - First In last Out (FILO)
  - Last In First Out (LIFO)
- العمليات الاساسية عليه:
  - يمكن الوصول إلى العناصر من خلال قمته (head)
  - إضافة عنصر إلى القمة (push)
  - حذف عنصر من القمة (pop)
- يمكن تمثيل المكدس باستخدام المصفوفات أو السلاسل الخطية.

Stack (المكدس)



Stack (المكدس)

- تطبيقات المكدسات:
  - سجل الصفحات المزارة في متصفح الانترنت.
  - عمليات التراجع في محررات النصوص.
  - ربط أزواج الأقواس في التعابير الرياضية.
  - التحويل من التعابير الرياضية النظامية إلى التعابير الملحقة وحساب قيمة هذه التعابير.
  - استدعاء تابع في جسم تابع.
  - التجول في الأشجار والبيانات بطريقة العمق أولا.
  - مكدس الأخطاء في المترجمات.

Stack (المكدس)

Infix (نظامي)	Prefix (مصدر)	Postfix (ملحق)
A+B	+AB	AB+
A+B*C	+A*BC	ABC*+
A*(B+C)	*A+BC	ABC+*
A*B+C	+*ABC	AB*C+
A+B*C+D-E*F	-++A*BCD*EF	ABC*+D+EF*-
(A+B)*(C+D-E)*F	**+AB-+CDEF	AB+CD+E-*F*

Stack (المكدس)

- Infix to Postfix:
  1. Scan the infix expression from left to right.
  2. If the scanned character is an operand, output it.
  3. Else,
     1. If the precedence of the scanned operator is greater or equal than the precedence of the operator in the stack (or the stack is empty), push it.
     2. Else, pop the operator from the stack until the precedence of the scanned operator is less-equal to the precedence of the operator residing on the top of the stack. Then push the scanned operator to the stack.
  4. If the scanned character is an '(', push it to the stack.
  5. If the scanned character is an ')', pop and output from the stack until an '(' is encountered.
  6. Repeat steps 2 to 6 until infix expression is scanned.
  7. Pop and output from the stack until it is not empty.

Stack (المكدس)

- Infix to Prefix:
  1. Reverse the infix expression i.e A+B*C will become C*B+A.
     - Note while reversing each ‘(‘ will become ‘)’ and each ‘)’ becomes ‘(‘.
  2. Obtain the postfix expression of the modified expression i.e CB*A+.
  3. Reverse the postfix expression. Hence in our example prefix is +A*BC.

Stack (المكدس)

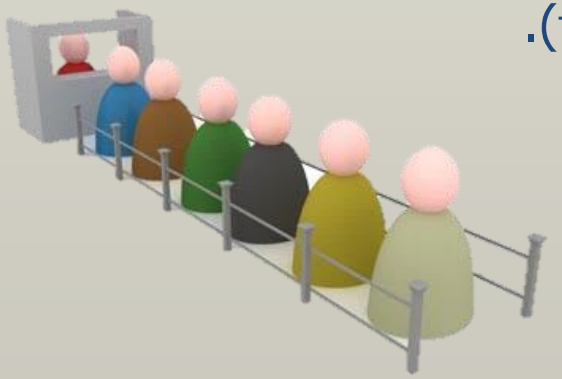
- Postfix Calculation:
  1. Create a stack to store operands (values).
  2. Scan the given expression and do following for every scanned element:
     1. If the element is a number, push it into the stack.
     2. If the element is a operator, pop operands for the operator from stack. Evaluate the operator and push the result back to the stack.
  3. When the expression is ended, the number in the stack is the final answer.

Stack (المكدس)

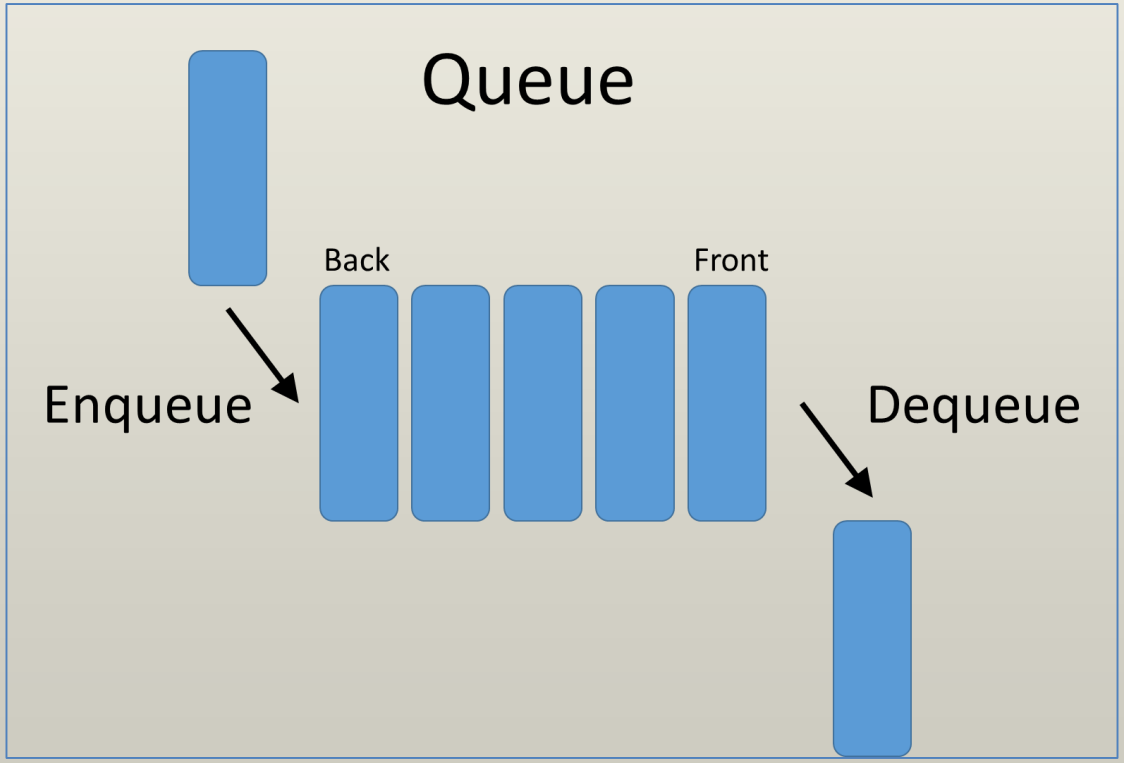
- Prefix Calculation:
  1. Put a pointer P at the end of the end.
  2. If character at P is a number push it to Stack.
  3. If the character at P is an operator pop two elements from the Stack. Operate on these elements according to the operator, and push the result back to the Stack.
  4. Decrement P by 1 and go to Step 2 as long as there are characters left to be scanned in the expression.
  5. The Result is stored at the top of the Stack.

Queue (الأرتال)

- هي بنية خطية تشبه السلسلة غير أن عمليات الإضافة تجري في جهة ندعوها ذيل الرتل. ويجري الحذف في الجهة المعاكسة التي نسميها بداية الرتل.
- هي بنية معطيات تمكن من تخزين عناصر من نفس النمط.
- تتبع مبدأ :
  - First In First Out (FIFO)
  - Last In Last Out (LILO)
- العمليات الاساسية عليه:
  - يمكن الوصول إلى العناصر من خلال بدايته (head) أو نهايته (tail).
  - إضافة عنصر إلى النهاية (enqueue).
  - حذف عنصر من البداية (dequeue).
- يمكن تمثيل الارتال باستخدام المصفوفات أو السلاسل الخطية.



Queue (الأرتال)

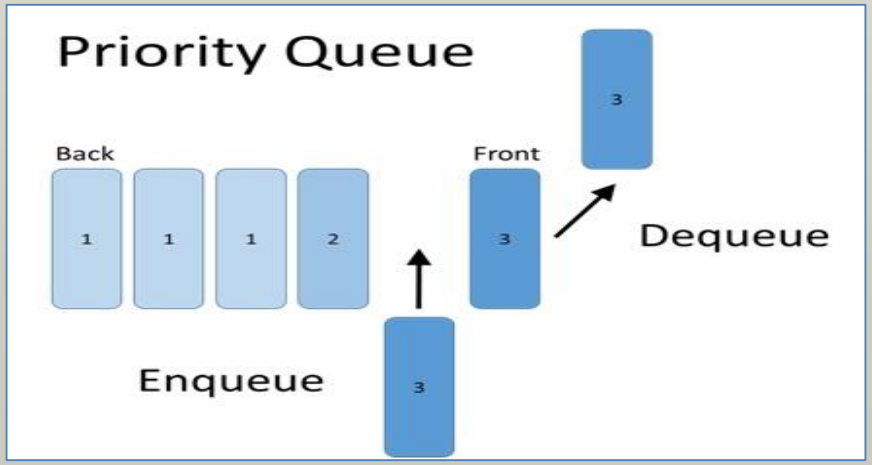


Queue (الأرتال)

- تطبيقات الأرتال:
  - مخازن المدخلات والمخرجات في الكمبيوتر (لوحة المفاتيح والطابعة).
  - بنية معطيات مساعدة في الخوارزميات مثل خوارزميات التجول بالعرض أولا في الأشجار والبيانات.
  - محاكاة أرتال الحياة اليومية.

Queue (الأرتال)

- رتل الاولويات:
  - يضاف عامل الأولوية إلى الرتل.
  - يتم تخديم العنصر ذو الأولوية الأعلى اولا.
  - مثال:
    - أرتال الانتظار في عيادة الطبيب حيث تكون الأولوية للحالات الإسعافية.



NON SEQUENTIAL DATA STRUCTURES

بنى المعطيات غير التسلسلية

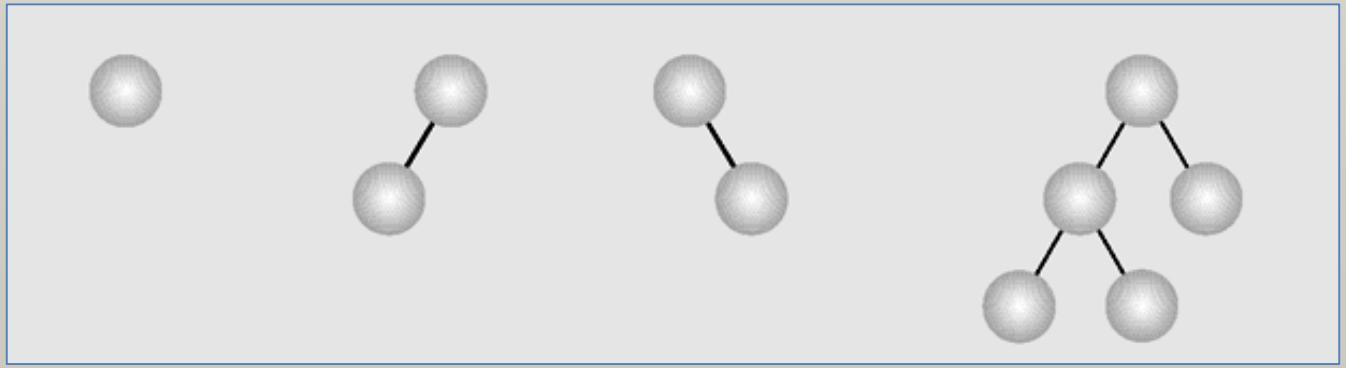
By Eng. Joud Khattab

Trees (الاشجار)

- نعرف الشجرة:
  - بأنها مجموعة من العناصر نسميها عقد (nodes)،
  - مرتبطة بين بعضها بروابط (Links)،
  - ومنظمة تنظيما هرميا (Hierarchical)،
  - لا يحتوي حلقات (Cycles) مغلقة،
  - أي إنه توجد عقدة مميزة ووحيدة نسميها جذر (Root).
- أمثلة:
  - تنظيم الملفات في مجلدات.
  - تمثيل عبارة حسابية.

Binary Trees (الأشجار الثنائية)

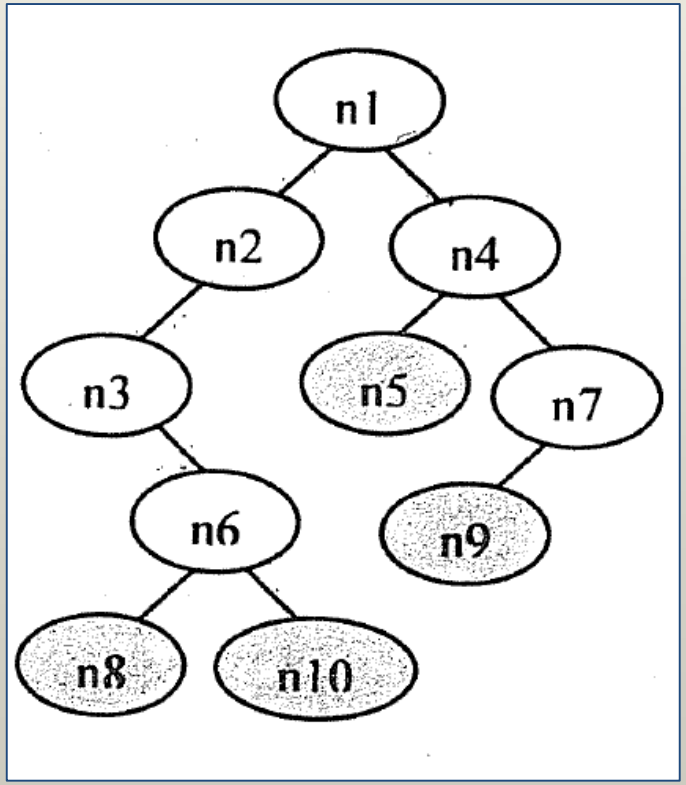
- حالة خاصة من الأشجار يكون فيها لكل عقدة ابنان علي الأكثر:
  - نسمي العقد التي لها ولد على الأقل عقد داخلية.
  - نسمي العقد التي ليس لها أي ولد عقد خارجية أو أوراق (Leaf).
  - نسمي طريقا (Path) كل متتالية من العقد.
  - نسمي فرعا كل طريق يصل بين الجذر وإحدى الأوراق.



Binary Trees (الأشجار الثنائية)

- تعاريف مهمة:
  - درجة الشجرة: العدد الأعظمي لأولاد عقدة. (درجة شجرة ثنائية هي 2)
  - حجم الشجرة: عدد عقدها.
  - ارتفاع عقدة: عدد الاتصالات في الطريق الواصل من العقدة الى الجذر.
  - ارتفاع الشجرة: أطول ارتفاع عقدة فيها.
  - عرض الشجرة: العدد الأعظم للعقد في مستوى ما.
  - مسافة التجول: مجموع ارتفاعات عقدها.
  - مسافة التجول الخارجي: مجموع ارتفاعات أوراقها.
  - مسافة التجول الداخلي: مجموع ارتفاعات عقدها الداخلية.

Binary Trees (الأشجار الثنائية)
Exercise



- جذر الشجرة: n1
- الابن الايسر: n2
- الابن الأيمن: n4
- الأوراق: n9 – n5 – n10 – n8
- العقد الداخلية: n7 – n6 – n4 – n3 – n2 – n1
- ارتفاع عقدة:
  n1=0 , n2=n4=1 , n3=n5=n7=2
- ارتفاع الشجرة: 4
- مسافة التجول: 22
- مسافة التجول الخارجي: 13
- مسافة التجول الداخلي: 9

Binary Trees (الأشجار الثنائية)

- بعض الأشجار الثنائية الخاصة:
  - الشجرة الثنائية الخطية: شجرة ثنائية يكون لكل عقدة منها ولد واحد على الأكثر.
  - الشجرة الثنائية التامة: كل شجرة تحوي عقدة واحدة في المستوى 0 وعقدتين في المستوى 1 وأربع عقد في المستوى 2 و 2^k عقدة في المستوى k.
    - (عدد العقد بشجرة تامة ارتفاعها h هو $2^{h+1} - 1$)
  - الشجرة الثنائية الكاملة: كل شجرة ثنائية تكون كل مستوياتها مليئة بالعقد ماعدا المستوى الأخير يمكن أن يحوي على فراغات.

Binary Trees (الأشجار الثنائية)
Question 1

- السؤال: ما هو ارتفاع شجرة خطية عدد عقدها n ؟
- الإجابة: n-1

- السؤال: كم عدد أوراق شجرة ثنائية تامة ؟
- الإجابة: 2^h حيث h هو ارتفاع الشجرة

- السؤال: كم عدد العقد لشجرة ثنائية تامة ؟
- الإجابة:

$$\sum_{i=0}^{i=h} 2^i = 2^{h+1} - 1$$

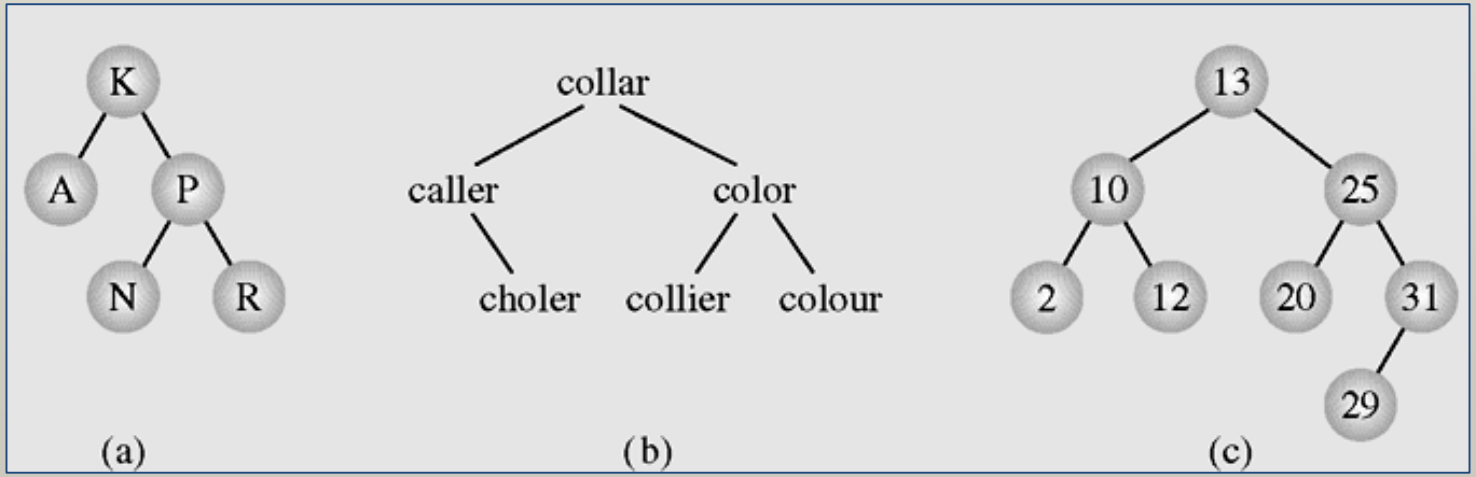
Binary Trees (الأشجار الثنائية)
Question 2

- السؤال: ماهي أسوأ حالة للبحث ضمن شجرة بحث ثنائية؟
- الإجابة: عندما تكون الشجرة خطية حيث تكون مسافة التجول أعظمية.

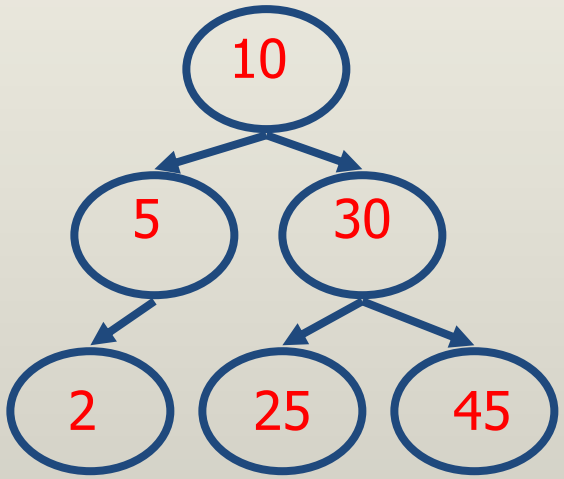
- السؤال: ماهي أفضل حالة للبحث ضمن شجرة بحث ثنائية؟
- الإجابة: عندما تكون الشجرة كاملة او متوازنة تكون مسافة التجول أصغرية.

Binary Search Trees
(أشجار البحث الثنائية)

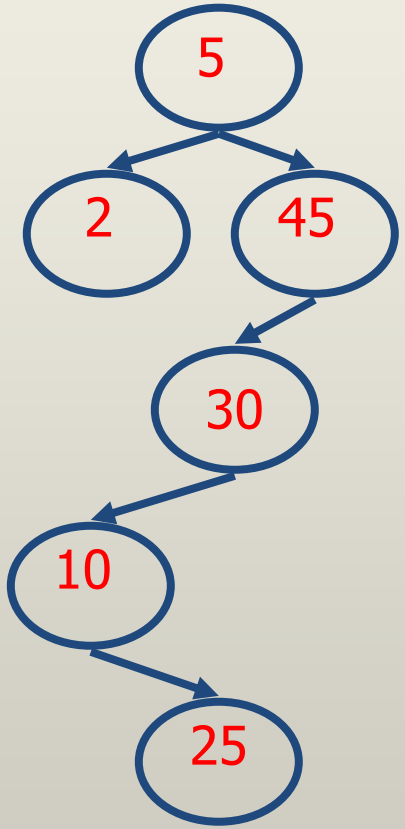
- هي شجرة ثنائية تحقق الخواص التالية:
  - من أجل عقدة محددة n كل القيم المخزنة في الشجرة الجزئية اليسارية هي أصغر من القيمة المخزنة في العقدة n.
  - من أجل عقدة محددة n كل القيم المخزنة في الشجرة الجزئية اليمينية هي أكبر من القيمة المخزنة في العقدة n.



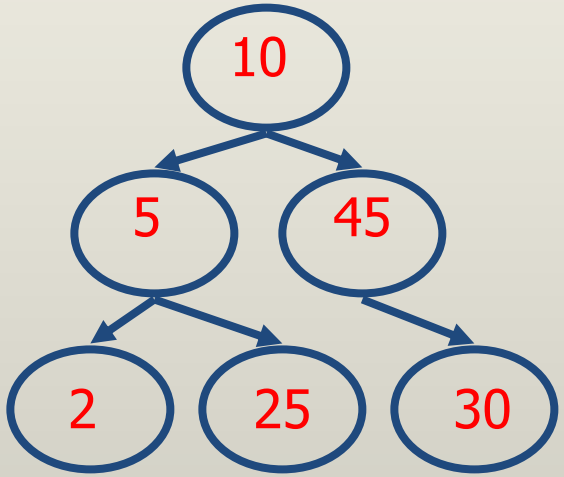
Binary Search Trees (الأشجار الثنائية)



Binary search tree

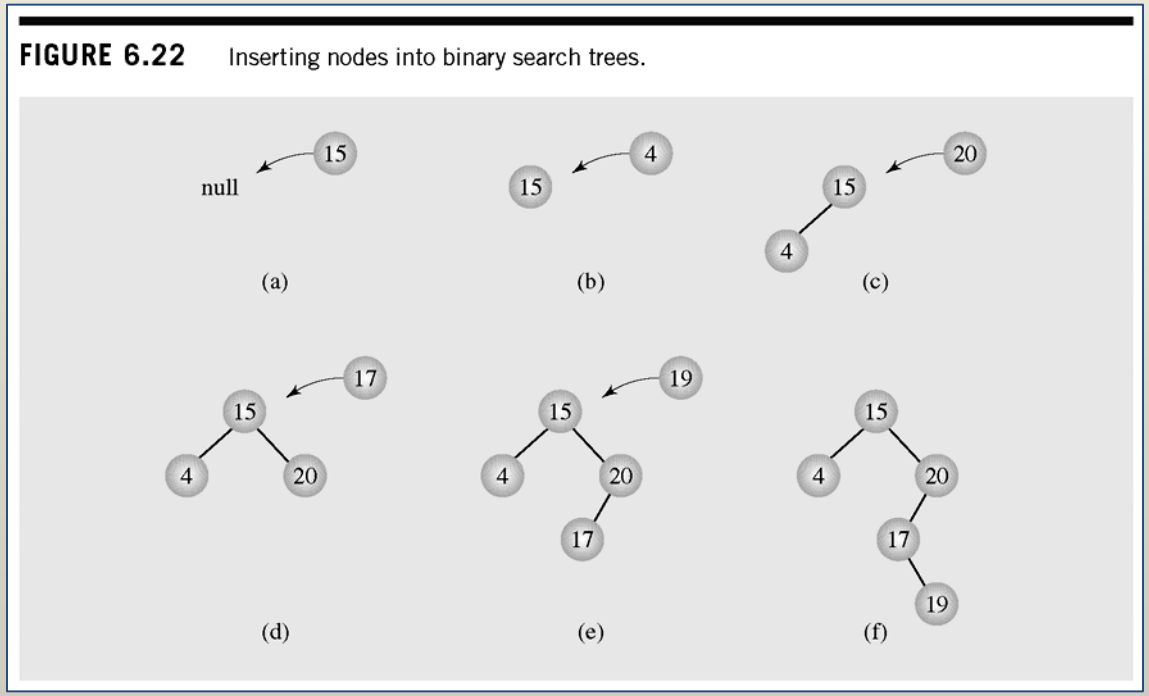


Binary search tree



Not a binary search tree

Binary Search Trees (أشجار البحث الثنائية)
Inserting Method

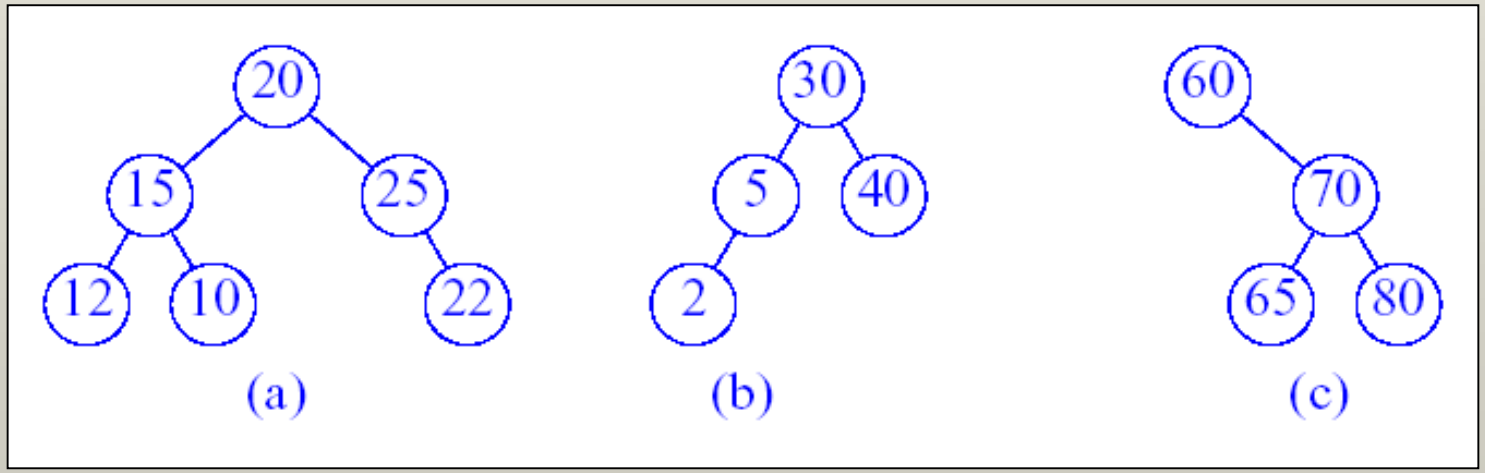


Binary Search Trees (أشجار البحث الثنائية) Representation Methods

- طرق تمثيل الأشجار الثنائية:
  - عن طريق المؤشرات.
  - عن طريق المصفوفات حيث يكون عدد أسطرها بعدد العقد وعدد أعمدتها 3 حيث يحوي العمود الأول رقم العقدة والثاني رقم الابن الايمن وفي الثالث رقم الابن الايسر.

Binary Search Trees (أشجار البحث الثنائية)
Question 3

- السؤال: أي من الشجر التالية تعتبر "شجرة بحث ثنائية"؟
- الإجابة: b و c

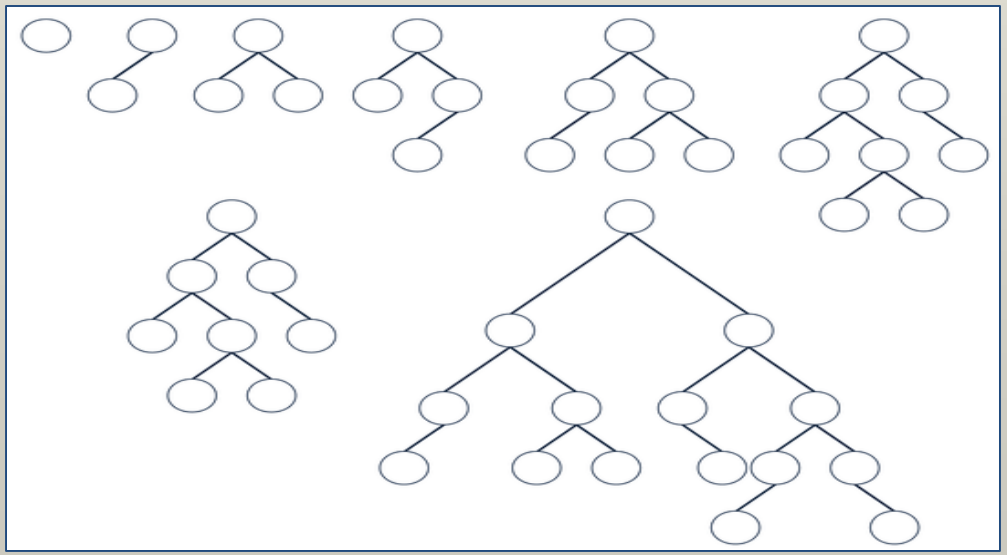


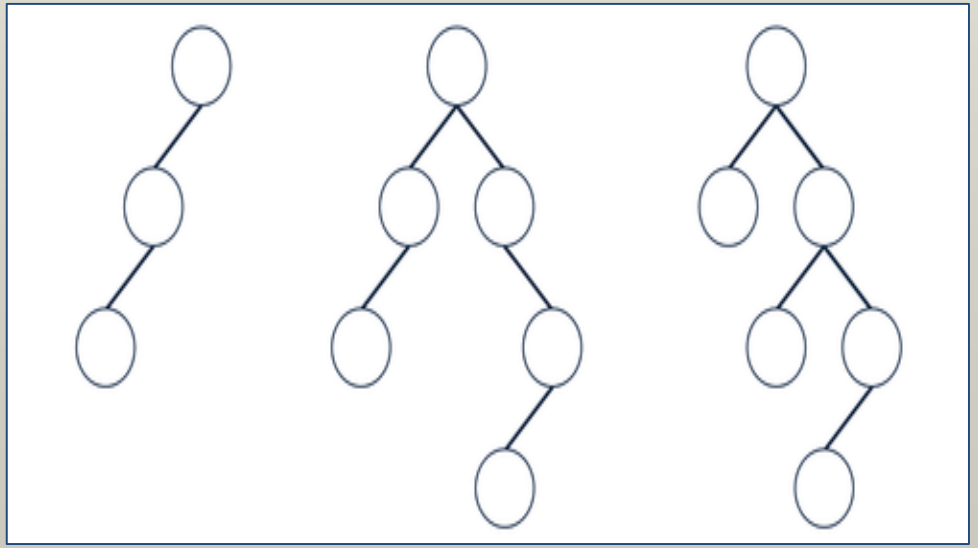
Binary Search Trees (أشجار البحث الثنائية)
Question 4

- السؤال: ماهي سيئة طريقة المصفوفات؟؟
- الإجابة: فيها هدر كبير للذاكرة في حالة الأشجار غير التامة أو الكاملة.

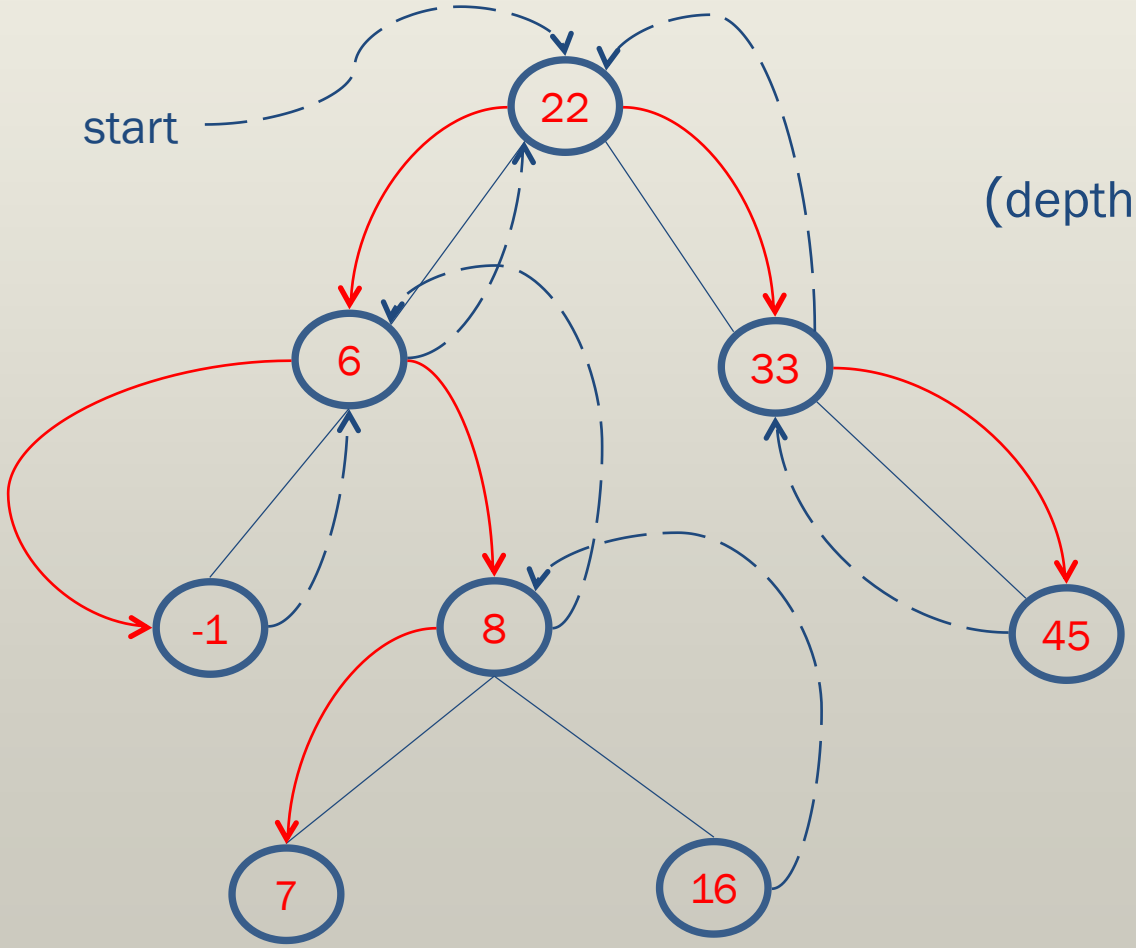
Binary Search Trees (أشجار البحث الثنائية)
AVL (Height-balanced Trees)

- An AVL tree (or height-balanced tree) is a binary search tree such that:
  - The height of the left and right subtrees of the root differ by at most 1
  - The left and right subtrees of the root are AVL trees





Binary Search Trees (أشجار البحث الثنائية)
DFS



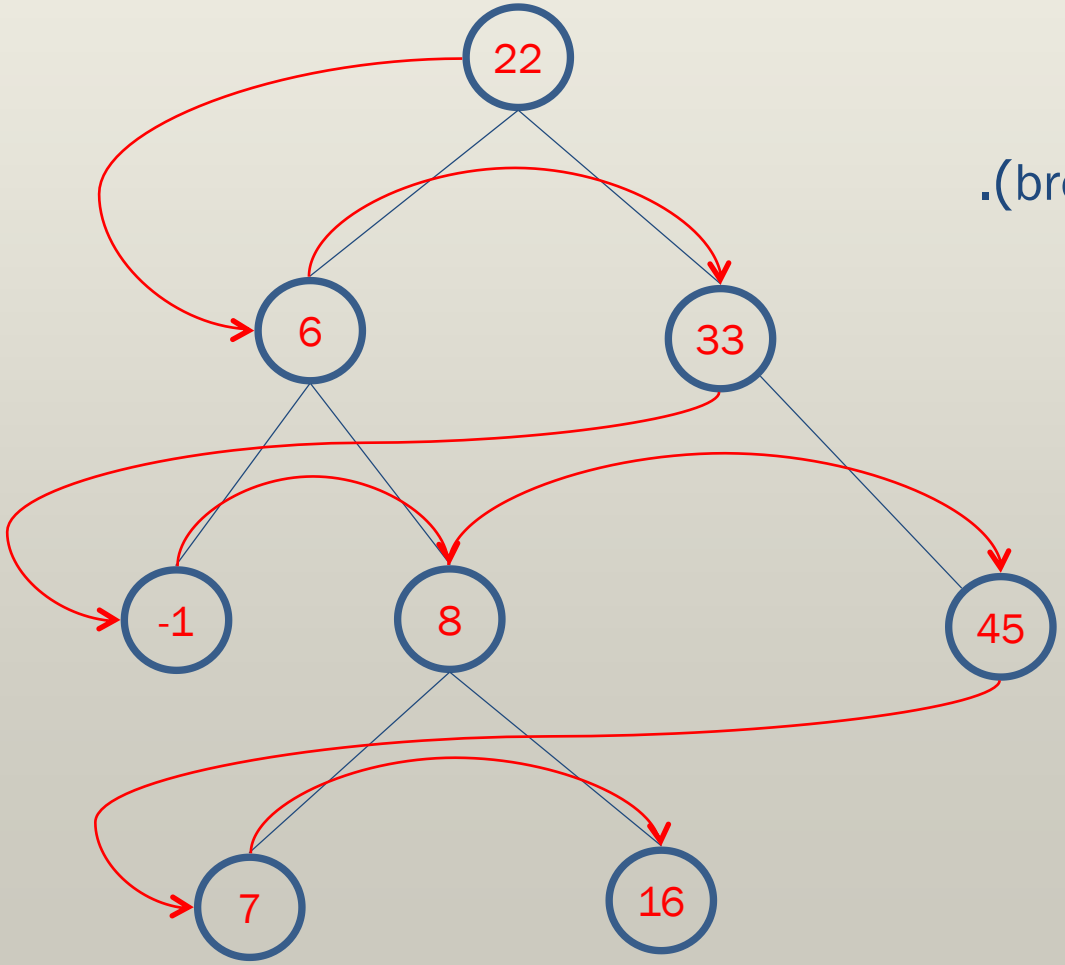
- طرق التجول في الشجرة:
  - التجول بالعمق أولا (depth first search)

ترتيب زيارة العقد في التجول VLR:
22,6,-1,8,7,16,33,45

Binary Search Trees (أشجار البحث الثنائية)
DFS

- طرق التجول في الشجرة: التجول بالعمق أولا (depth first search)
  - VLR يزور العقدة ثم الشجرة الجزئية اليسارية ثم اليمينية، ويدعى الترتيب المصدر (preorder).
  - VRL يزور العقدة ثم الشجرة الجزئية اليمينية ثم اليسارية.
  - LVR يزور الشجرة الجزئية اليسارية ثم العقدة ثم اليمينية ويدعى الترتيب المتناظر (inorder).
  - RVL
  - LRV ويدعى الترتيب الملحق (postorder).
  - RLV

Binary Search Trees (أشجار البحث الثنائية)
BFS



- طرق التجول في الشجرة:
  - التجول بالعرض أولا (breadth first search).

ترتيب زيارة العقد في التجول VLR:
22,6,33,-1,8,45,7,16

GRAPHS

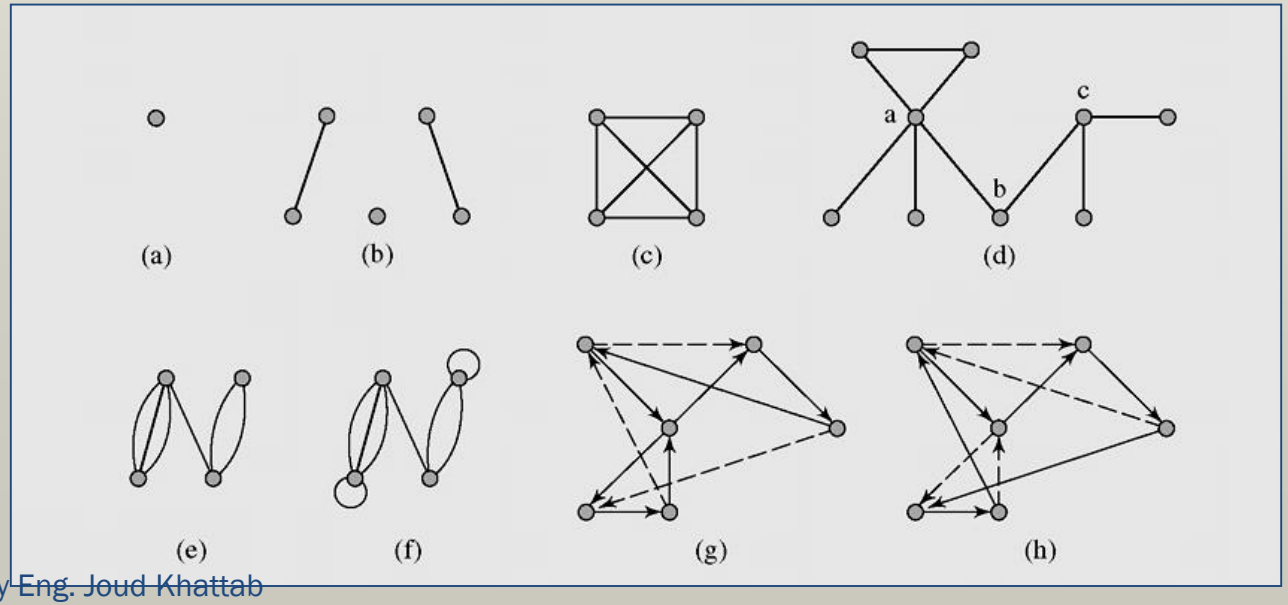
البيانات

2020

By Eng. Joud Khattab

Graphs (البيانات)

- هي بنية معطيات تخزن العناصر كعقد vertex ويوجد بينها وصلات تمثل العلاقات بين العقد edges.
- يمكن اعتبار البيانات تعميما للبنى الهرمية غير التسلسلية من الناحية النظرية، فالأشجار هي حالة خاصة من البيانات (بيانات لا تحوي حلقات).



Graphs (البيانات)

- تتواجد البيانات في كل مكان:
  - شبكات الطرق، الخطوط الجوية، شبكات الحواسيب.
- من أهم منهجيات حل المسائل في الذكاء الصنعي:
  - تحويل المسألة إلى مسألة بحث في بيان.

Graphs (البيانات)

- درجة العقدة Deg(v) هي عدد الأسهم الصادرة من العقدة والواردة إليها
  - حيث Deg-(v) هي الدرجة الواردة.
  - و Deg+(v) هي الدرجة الصادرة.

- يكون البيان متصل بشدة إذا كان في حالة كل عقدتين v,w يوجد طريق من v إلى w و من w إلى v

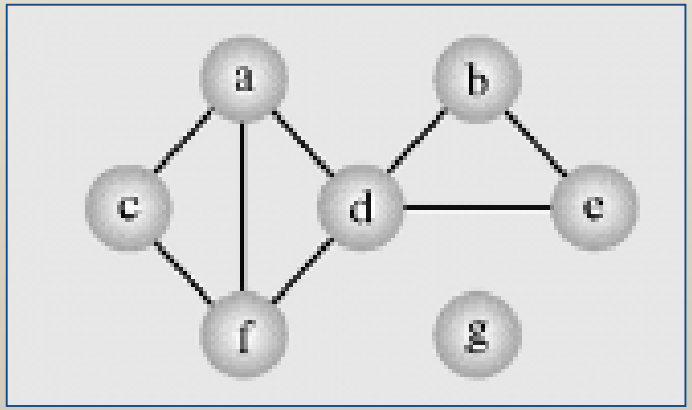
- عدد الروابط الأعظمي في بيان غير موجه يحوي n عقدة يساوي $\frac{n(n-1)}{2}$ وفي هذه الحالة نقول عن البيان إنه تام (Complete).

Graphs (البيانات)

- تمثيل البيان باستخدام مصفوفة التجاور (adjacency matrix).

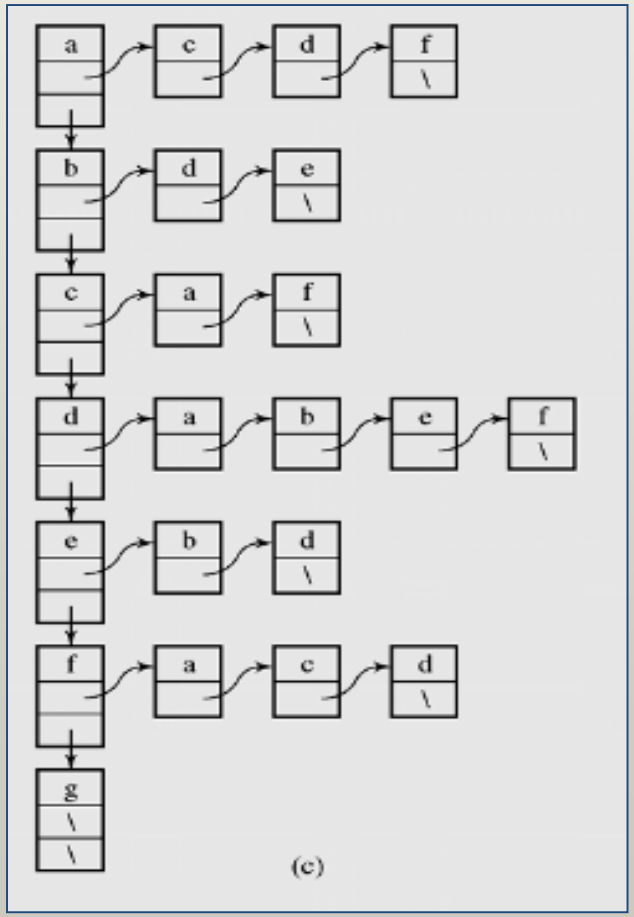
	a	b	c	d	e	f	g
a	0	0	1	1	0	1	0
b	0	0	0	1	1	0	0
c	1	0	0	0	0	1	0
d	1	1	0	0	1	1	0
e	0	1	0	1	0	0	0
f	1	0	1	1	0	0	0
g	0	0	0	0	0	0	0

The Adjacency Matrix

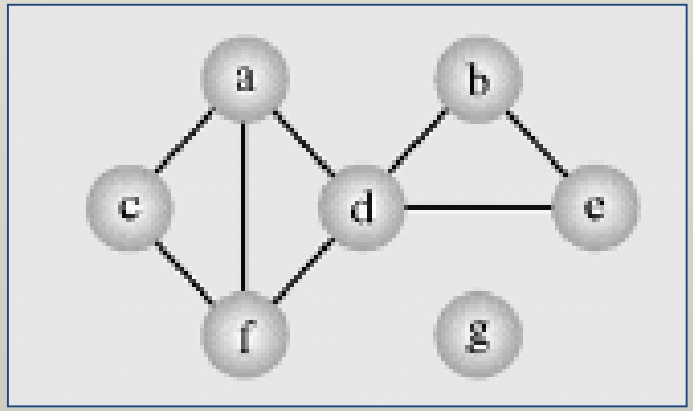


The Graph

Graphs (البيانات)



- تمثيل البيان باستخدام قائمة التجاور (adjacency list).



The Graph

The Adjacency List

Graphs (البيانات)
Question 1

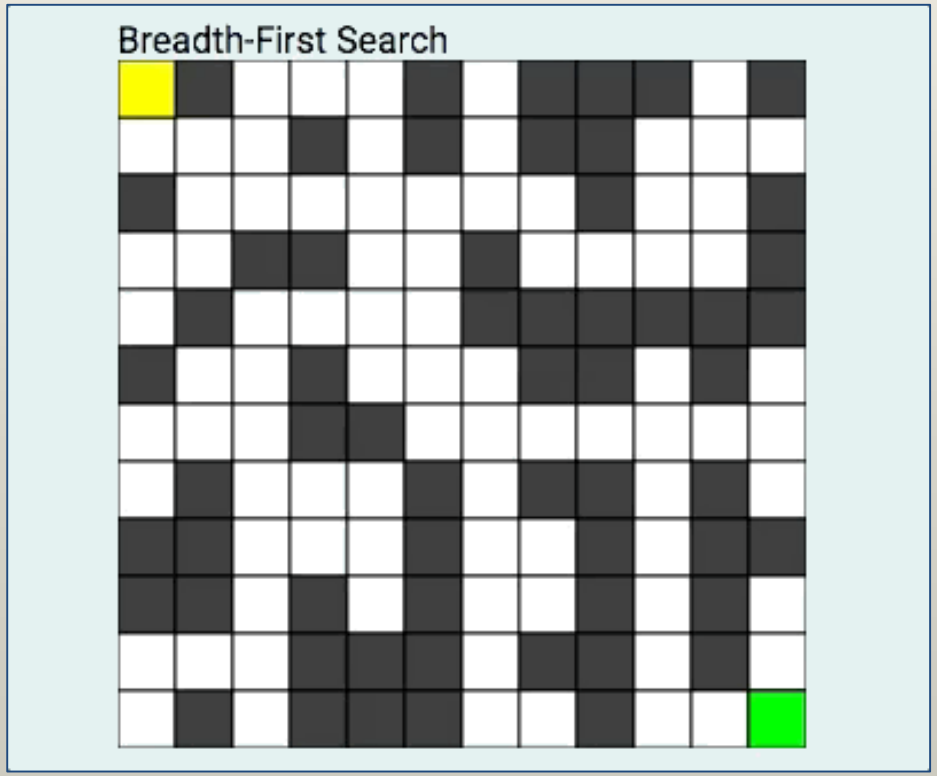
- السؤال: ماهي كلفة البحث عن عقد مجاورة لعقدة ممثلة بطريقة قائمة التجاور؟
- الجواب: O(deg(v))

- السؤال: ماهي كلفة مسح الروابط الواردة لعقدة في بيان ممثل بطريقة قائمة التجاور؟
- الجواب: O(V+E) مكلف جدا

Graphs (البيانات)

- طرق التجول في البيان:
  - العمق أولا (DFS).
  - العرض أولا (BFS): هي طريقة لإيجاد أقصر طريق في بيان غير موزون لأن الوصول للعقد يتم بمسح أقل عدد للروابط.

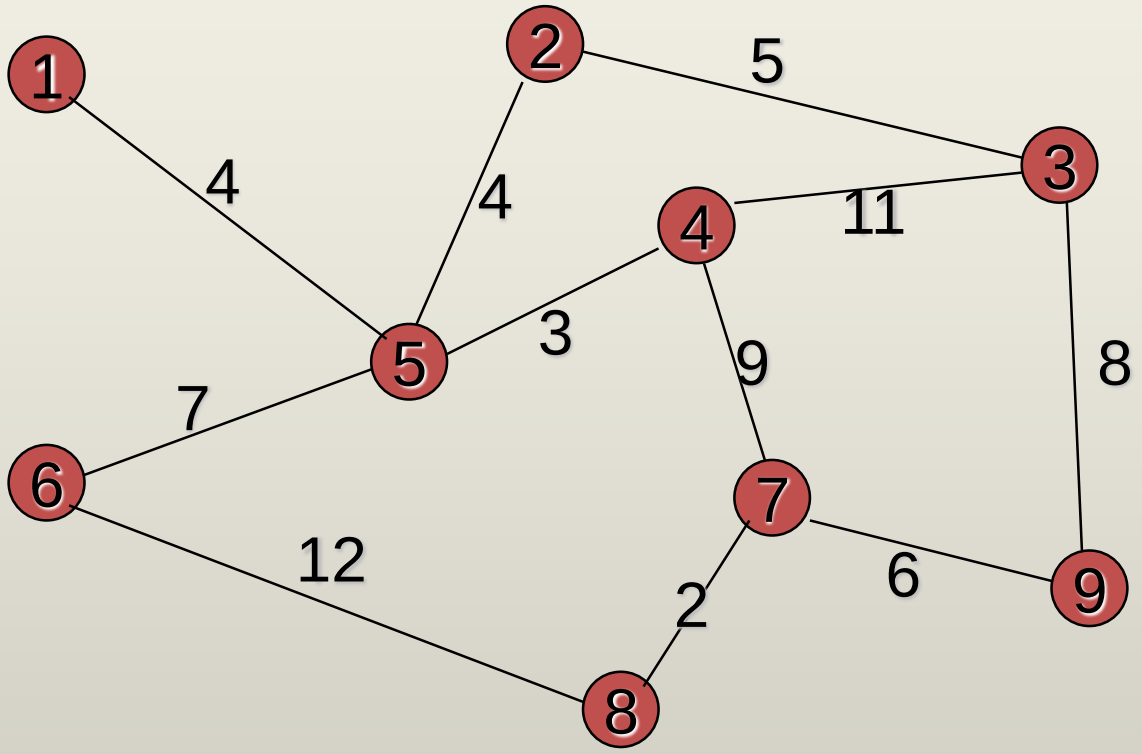
Graphs (البيانات)
DFS V.S BFS



Graphs (البيانات)
Dijkstra

- خوارزمية Dijkstra لإيجاد أقصر طريق في بيان موزون.

Graphs (البيانات)
Dijkstra

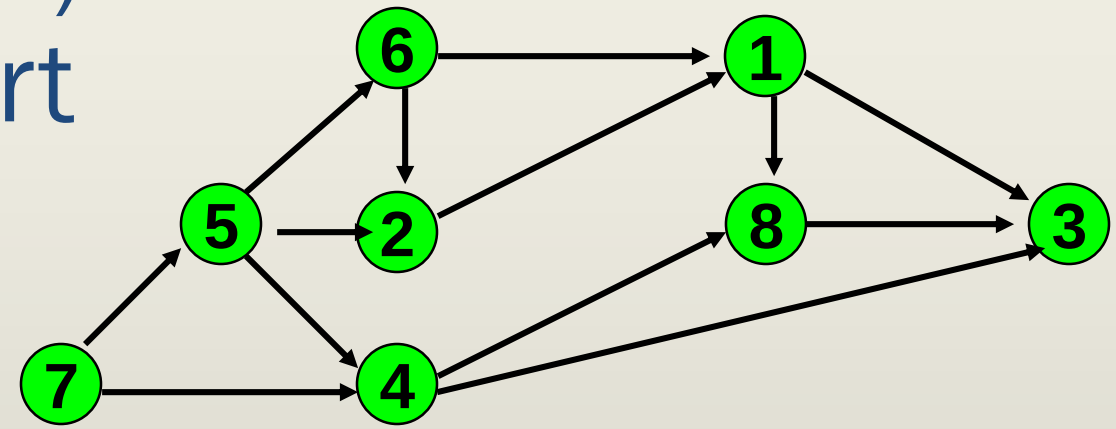


Processed	1	2	3	4	5	6	7	8	9
distance	0	?	?	?	?	?	?	?	?
predecessor	-1	-1	-1	-1	-1	-1	-1	-1	-1

Graphs (البيانات)
Topological Sort

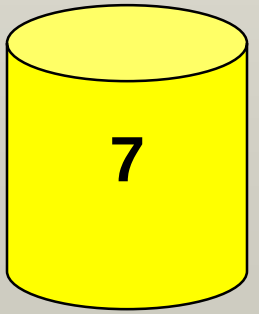
- الفرز الطبولوجي يعتمد على ترتيب العقد حسب درجات ورودها حيث تكون كل الأسهم او الوصلات تؤشر إلى جهة واحدة هي اليمين.
- لا يمكن تطبيقها على البيان الذي يحتوي على حلقات.
- مثال:
  - المنهاج السنوي لجامعة.

Graphs (البيانات)
Topological Sort



next 0

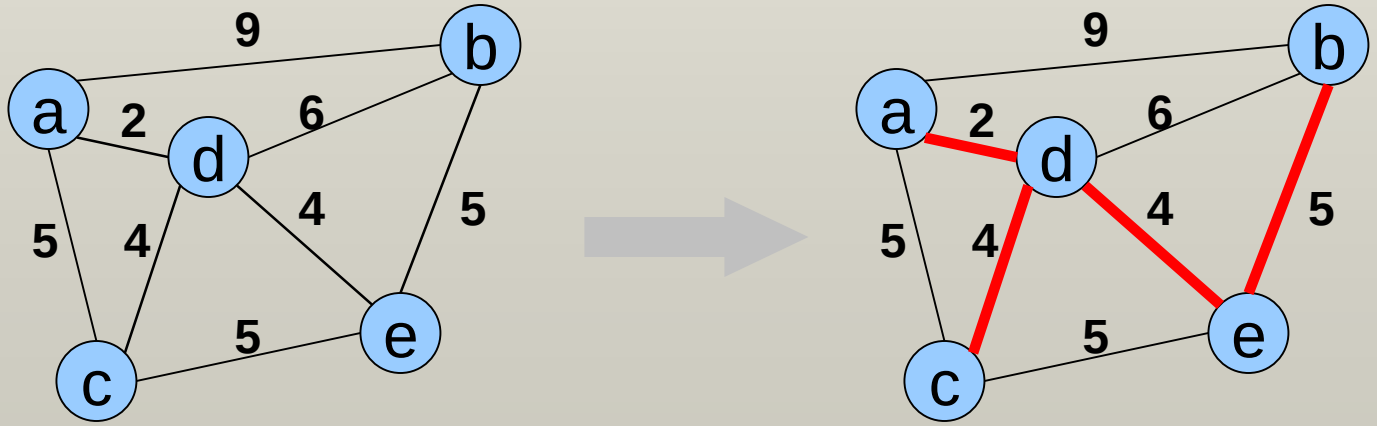
Node	1	2	3	4	5	6	7	8
Indegree	2	2	3	2	1	1	0	2



LIST

Graphs (البيانات)
Minimum Spanning Tree

- شجرة الارتباط الأصغرية: هي شجرة تربط البيان بأقل كلفة ممكنة من الروابط.
- تتواجد هذه الشجرة في حالة البيان المترابط فقط.
- Prim's algorithm



SORTING ALGORITHMS

خوارزميات الترتيب

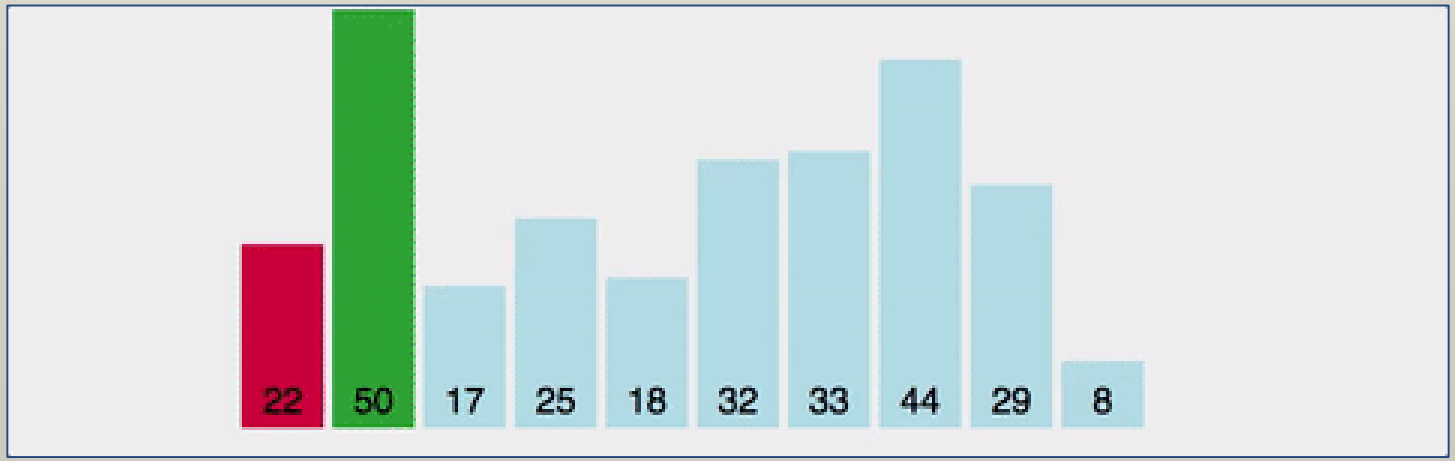
By Eng. Joud Khattab

Bubble Sort Algorithm
(خوارزمية الفرز الفقاعي)

- تعقيد خوارزمية الفرز الفقاعي:
  - أحسن الأحوال: عندما تكون المصفوفة مفروزة حيث لايوجد عمليات اسناد.
  - أسوأ الأحوال: عندما تكون المصفوفة مرتبة ترتيب عكسي يكون عدد عمليات الاسناد من مرتبة O(n*n).
  - الحالة الوسطية: عندما تكون المصفوفة مرتبة عشوائياً O(n*n).

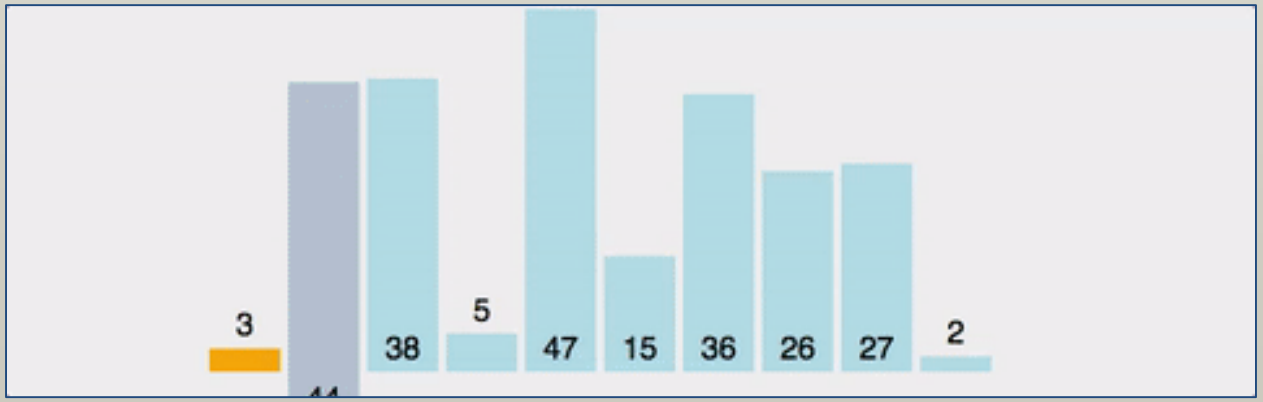
Selection Sort Algorithm
(خوارزمية الفرز بالاختيار)

- تعقيد الخوارزمية:
  - عدد عمليات المقارنة O(n*n)
  - عدد عمليات التبديل O(n)
- جيدة لحجم صغير وفي حالة كانت عملية التبديل مكلفة وعملية المقارنة غير مكلفة.

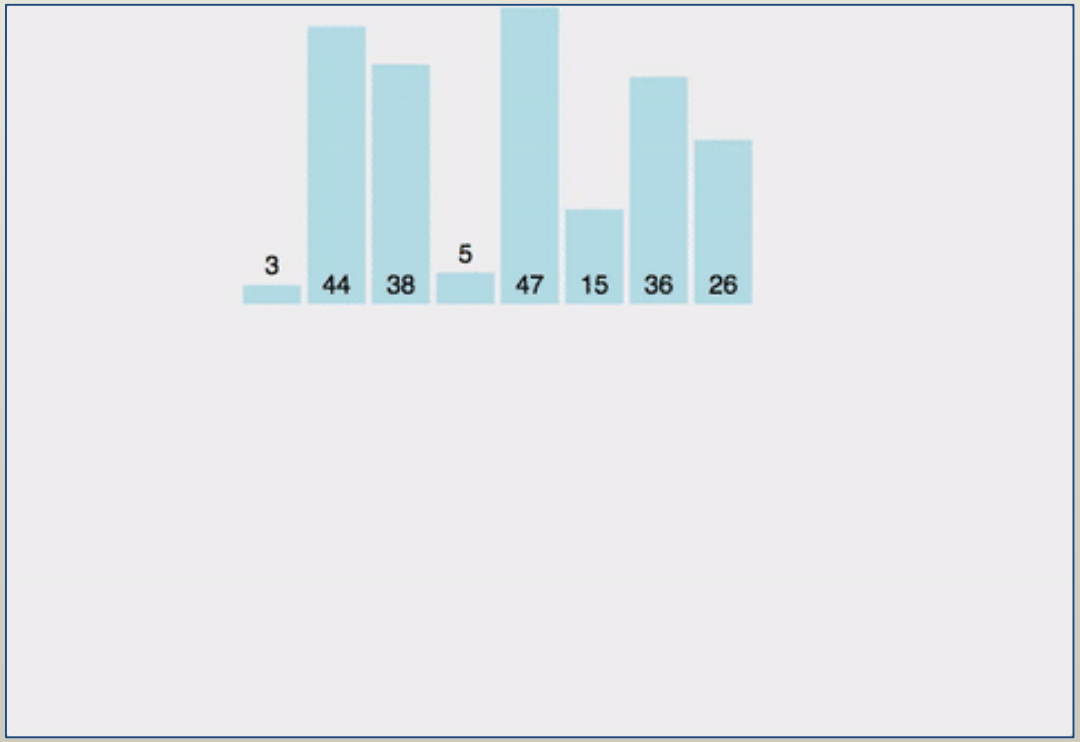


Insertion Sort Algorithm
(خوارزمية الفرز بالإضافة)

- تعقيد خوارزمية الفرز بالإضافة:
  - أحسن الأحوال: عندما تكون المصفوفة مفروزة حيث لا يوجد عمليات اسناد O(1) وعمليات المقارنة O(n).
  - أسوأ الأحوال: عندما تكون المصفوفة مرتبة ترتيب عكسي يكون عدد عمليات الاسناد من مرتبة O(n*n).
  - الحالة الوسطية: عندما تكون المصفوفة مرتبة عشوائياً O(n*n).



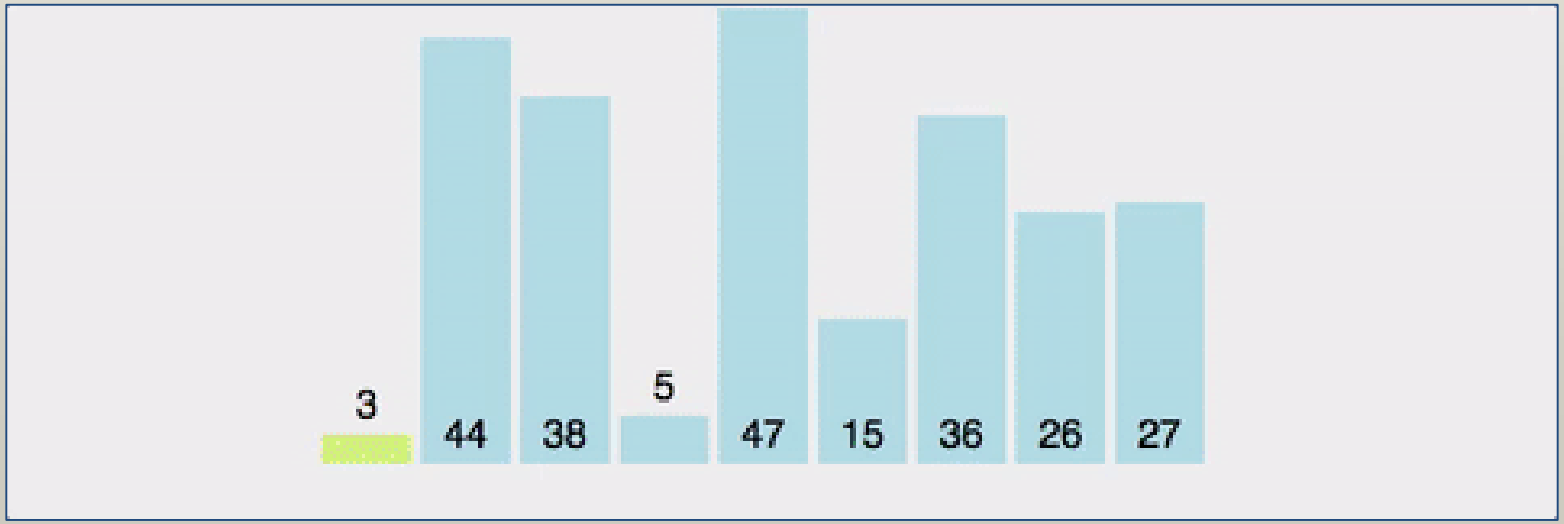
Merge Sort Algorithm
(خوارزمية الفرز بالدمج)



- تعتمد مبدأ فرق تسد.
- هي خوارزمية عوديه تقوم بما يلي:
  - تقسم المصفوفة إلى جزئين،
  - تفرز كل جزء على حدا،
  - تدمج الجزئيين المفروزين في مصفوفة واحدة مفروزة.
- تعقيد الخوارزمية :
  - O(nlog(n)) دوما

Quick Sort Algorithm
(خوارزمية الفرز السريع)

- تعقيد الخوارزمية:
  - O(nlog(n)) في أحسن الأحوال والحالة الوسطية.
  - O(n*n) في أسوأ الأحوال.



Sorting Algorithms

	Insertion	Selection	Bubble	Shell	Merge	Heap	Quick	Quick3
Random								
Nearly Sorted								
Reversed								
Few Unique								

SEARCH ALGORITHMS

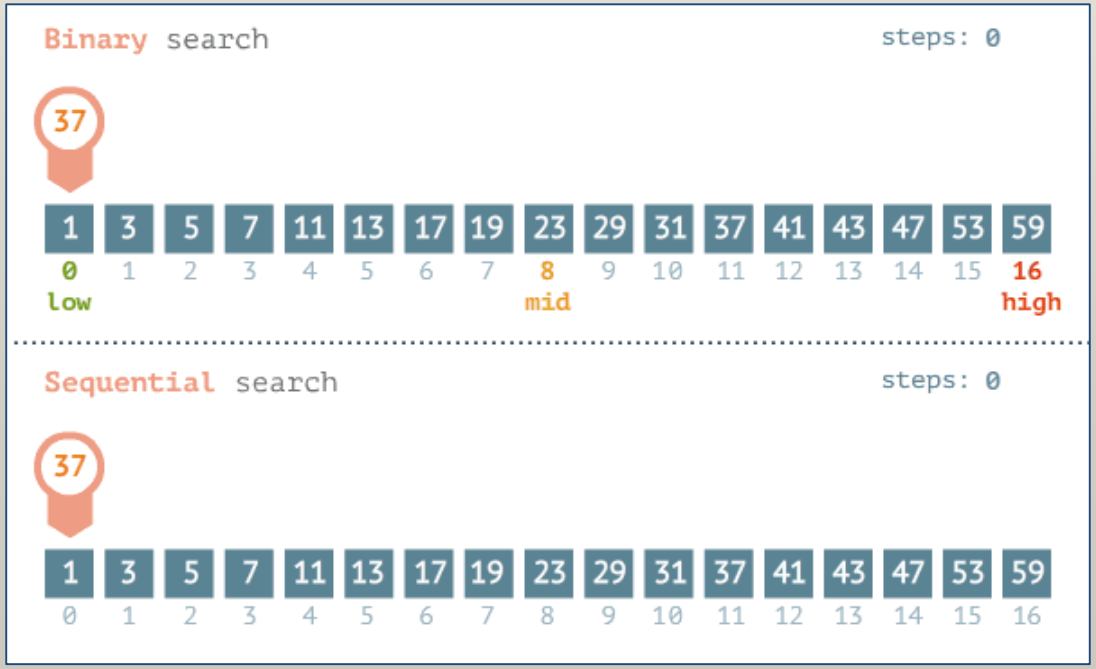
خوارزميات البحث في مصفوفة

By Eng. Joud Khattab

2020

Search Algorithms
Binary V.S. Sequential

- البحث التسلسلي : تعقيده O(n)
- البحث الثنائي : تعقيده O(log(n))



ADVANCE ALGORITHMS

خوارزميات متقدمة

By Eng. Joud Khattab

2020

Backtracking Algorithms
(الخوارزميات التراجعية)

- هي خوارزميات عودية تعتمد طريقة التجريب والخطأ في حل المشاكل.
- تتلخص هذه الطريقة ببناء الحل النهائي للمسألة عن طريق مجموعة من الخطوات:
  - في الخطوة نحدد الإمكانيات المتاحة للخطوة التالية،
  - ثم ندرس هذه الإمكانيات بأن ننتقي أحدها،
  - و نسجلها على أنها الخطوة التالية في الحل النهائي،
  - و نتابع الخوارزمية اعتماداً على هذه الخطوة.
  - عندما يظهر لنا أن اختيارنا لا يقود إلى الحل النهائي، أو يؤدي إلى طريق مسدود،
  - نعدل عن هذه الخطوة.
- تشبه هذه العملية بناء سجل أخطاء يفيد في تجنب الوقوع في الخطأ مرتين.

Backtracking Algorithms (الخوارزميات التراجعية)

- مثال :
  - مسألة جولة حصان الشطرنج:
    - إيجاد طريقة لتغطية رقعة الشطرنج من موقع معين يقف فيه الحصان وذلك باستخدام حركات الحصان المعروفة في الشطرنج وبشرط المرور بكل مربع مرة واحدة.
  - مسألة الوزراء الثمانية:
    - توزيع ثمانية وزراء على رقعة الشطرنج بحيث لا يكون فيها اي واحد مهدد للأخر.
  - مسألة الخيار الأمثل:
    - مقارنة الحلول التي يمكن ايجادها لمسألة معينة والاحتفاظ بالحل الأمثل وفق معايير محددة للمسألة.

Greedy Algorithms
(الخوارزميات الجشعة)

- هي خوارزمية التي تستند على الحدس المهني الذي يتم عن طريقه اختيار الإمكانية الأفضل المرئية في المرحلة الحالية (الحل الأمثل في المرحلة الحالية), من دون الأخذ بالحسبان تأثير هذه الخطوة على تكملة الحل.
- قد لا يقودنا الحل الأمثل المرحلي إلى الحل الأمثل للمسألة.
- خوارزمية الجشع لا يكن ان تتراجع عن خياراتها. هذا هو الفرق الرئيسي بينها وبين البرمجة الديناميكية.
- مثل خوارزمية برايم وأشجار هوفمان.

Divide and Conquer
(خوارزميات فرق تسد)

- عمل خوارزمية فرق تسد عن طريق تقسيم المسألة بشكل عودي إلى مسألتين جزئيتين أو أكثر من نفس النوع، حتى تصبح المسائل الجزئية بسيطة بما فيه الكفاية لتحل بشكل مباشر. ومن ثم تدمج حلول المسائل الجزئية لتعطي حلاً للمسألة الجزئية.
- امثلة:
  - خوارزميات البحث الثنائي.
  - خوارزميات الفرز بالدمج.
  - خوارزميات الفرز السريع.
  - خوارزمية ابراج هانوي.

Dynamic Programming (البرمجة الديناميكية)

- لحل مسألة ما، نحن بحاجة إلى حل أجزاء مختلفة من المسألة (مسائل فرعية)، ومن ثم جمع حلول المسائل الفرعية للحصول على حل شامل.
- نهج البرمجة الديناميكية:
  - البحث عن حل كل مسألة فرعية مرة واحدة فقط،
  - وبالتالي تقليل عدد الحسابات: حالما تم حساب حل مسألة فرعية ما، يتم حفظه، وفي المرة القادمة عند الحاجة للحل نفسه، يتم ببساطة استرجاعه.
- خوارزميات البرمجة الديناميكية ستدرس الحلول السابقة للمسائل الثانوية وتقوم بدمجها للحصول علي افضل حل للمسائلة المراد حله.

HASHING FUNCTION

(جداول التقطيع)

By Eng. Joud Khattab

Hashing Tables
(جداول التقطيع)

- هي طريقة للإيجاد عنصر مباشرة دون الحاجة إلى خوارزمية بحث من خلال النفاذ المباشر إلى موقعه.
- هي بنية فعالة في برمجة التطبيقات التالية:
  - التي تحتاج إلى عمليات معجم.
  - الامثلة التي لا تهتم بالترتيب.
- يخزن العنصر ذو المفتاح k في الخانة h(k) حيث h هو تابع التقطيع الذي يستخدم لحساب دليل الخانة من المفتاح k.
- قد يحدث تصادم في حال وجود مفتاحين لهما نفس قيمة تابع التقطيع.
- تخزن القيم على شكل ازواج من (key, value)
- جداول التقطيع ليست بنية جيدة للتجول حول سلسلة من القيم. (Iteration)

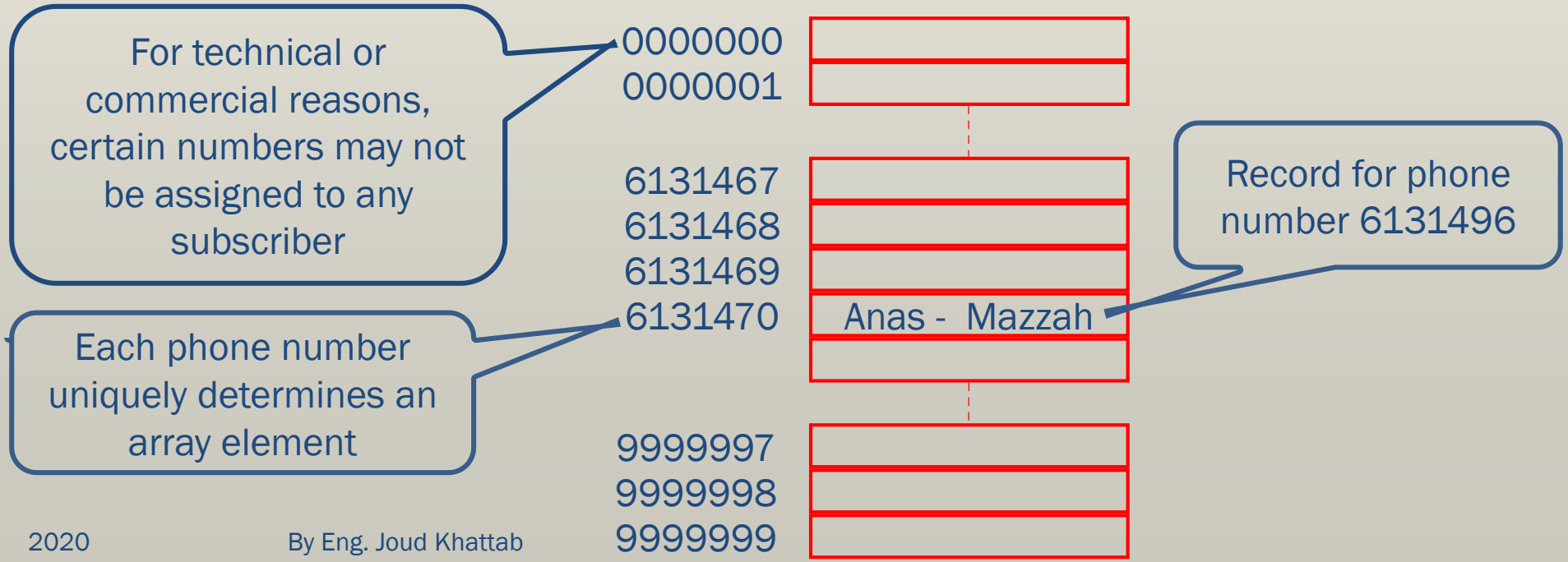
- الزمن المتوقع لإضافة والبحث عن عنصر في جدول تقطيع ضمن فرضيات مناسبة O(1). (الحالة الوسطية)

Hashing Tables V.S. Other Data Structures

- We want to implement the dictionary operations Insert(), Delete() and Search()/Find() efficiently.
- Arrays:
  - can accomplish in O(n) time
  - but are not space efficient (assumes we leave empty space for keys not currently in dictionary)
- Binary search trees
  - can accomplish in O(log n) time
  - are space efficient.
- Hash Tables:
  - A generalization of an array that under some reasonable assumptions is O(1) for Insert/Delete/Search of a key

Hashing Tables
(جداول التقطيع)

- مثال عن تخزين المعلومات باستخدام العنونة المباشرة:
- تستخدم عندما نستطيع منح موقع من الجدول لكل مفتاح (عدد العناصر المتوقع تخزينها قريب من العدد الكلي للعناصر.



Hashing Tables (جداول التقطيع)

- For example, if we hash keys 0…1000 into a hash table with 5 entries and use **h(key) = key mod 5** , we get the following sequence of events:

Insert 2

	key	data
0		
1		
2	2	…
3		
4		

Insert 21

	key	data
0		
1	21	…
2	2	…
3		
4		

Insert 34

	key	data
0		
1	21	…
2	2	…
3		
4	34	…

Insert 54

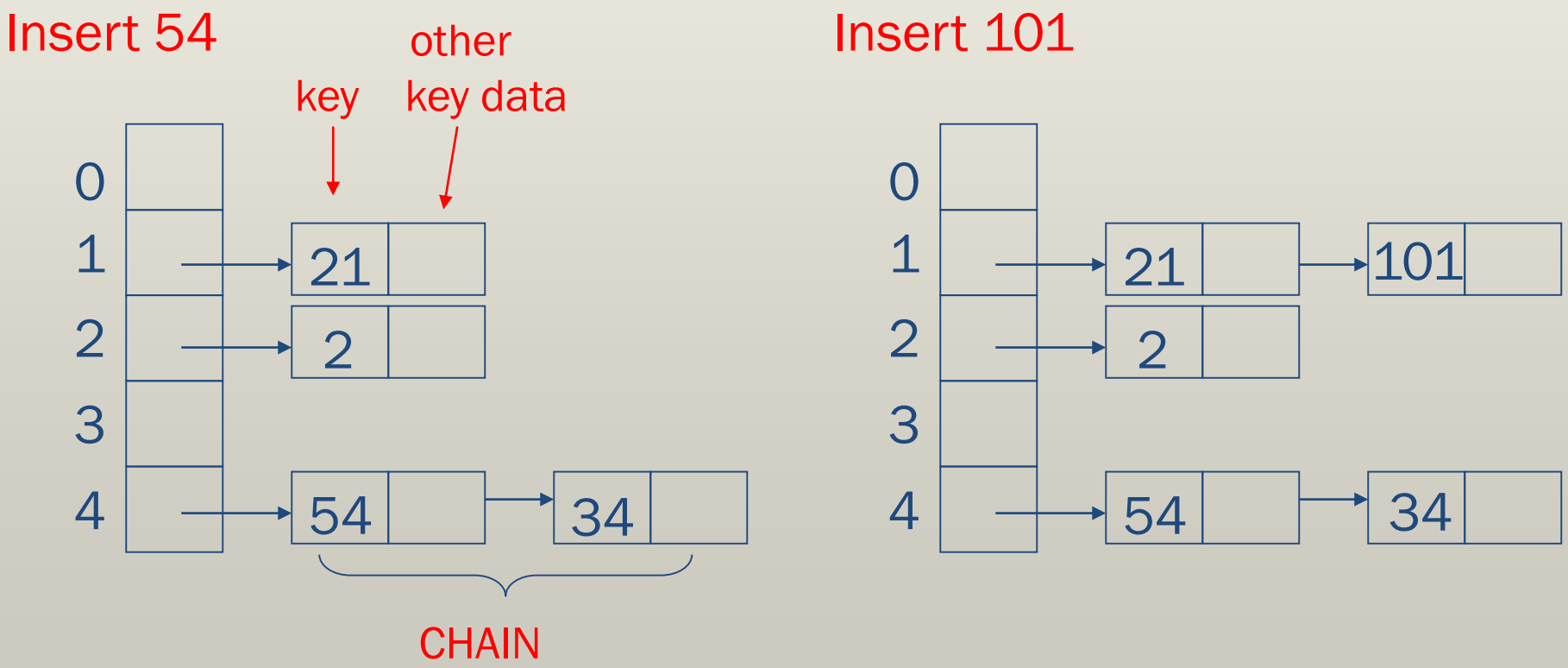
There is a ***collision*** at array entry #4

???

Hashing Tables
(جداول التقطيع)

- تقنية الربط لحل التصادم:
  - توضع جميع العناصر التي لها نفس قيمة تابع التقطيع في سلسلة خطية. (Hashing with Chaining)
  - تصبح خانة الجدول مؤشر إلى بداية سلسلة العناصر المتصادمة.

Hashing Tables (جداول التقطيع)
Hashing with Chaining



Hashing Tables (جداول التقطيع)
Hashing with Chaining

- What is the running time to insert/search/delete?
  - Insert: It takes O(1) time to compute the hash function and insert at head of linked list
  - Search: It is proportional to max linked list length
  - Delete: Same as search

Hashing Tables (جداول التقطيع)
Question 1

- السؤال: ماهي كلفة عملية إضافة عنصر؟
- الجواب: O(1) حيث تتم الإضافة للعناصر المتصادمة في رأس السلسلة.

- السؤال: ماهي كلفة عملية البحث والحذف في أسوأ الأحوال؟
- الجواب: تتعلق بطول السلسلة الخطية للعناصر المتصادمة وهي في أسوأ الاحوال O(n)عند اختيار تابع تقطيع سيء.

- السؤال: ماهي الكلفة الوسطية للحذف والاضافة؟
- الجواب: هو عدد العناصر الوسطي في السلسلة عند استخدام تابع تقطيع يحقق توزع منتظم وهو n/m حيث n عدد العناصر الكلي و m عدد الخانات في الجدول.

Hashing Function (توابع التقطيع)

- طريقة التقسيم:
  - بربط المفتاح k بالخانة التي تساوي باقي قسمة k على m.
  - h(k)=k%m
  - يجب تجنب قيم m من قوى 2 واختيار أعداد أولية بعيدة عنها.
- طريقة الضرب:
  - نضرب المفتاح k بثابت A حيث 0<A<1 ونأخذ الجزء الكسري ثم نضرب هذه القيمة ب m ونأخذ الجزء الطبيعي من العدد الناتج.
  - يمكن اختيار أي قيمة للعدد m.

MCQ

By Eng. Joud Khattab