



كلية الهندسة قسم المعلوماتية

بنى معطيات 1

Data Structure 1

ا.د. علي عمران سليمان

محاضرات الأسبوع الثامن

الأشجار 1

Tree 1

الفصل الثاني 2025-2026

Trees 1 +2	الأشجار 1 + 2
1- Introduction	1- مقدمة
2- General Trees.	2- الأشجار العامة.
3- Tree Traversal Algorithms	3- خوارزميات التجول عبر الشجرة.
4- Binary Trees	4- الأشجار الثنائية.
	5- تعريف شجرة البحث الثنائية كقاموس بيانات.
	6- تحقيق خوارزميات البحث.
	7- حساب أداء خوارزميات البحث.
	8 - تحقيق طرائق التبديل على شجرة بحث ثنائية.
	9 - تحقيق شجرة البحث الثنائية بلغة C++

References

- Deitel & Deitel, Java How to Program, Pearson; 10th Ed(2015)

- د.علي سليمان، بني معطيات بلغة JAVA، بني معطيات بلغة C++، بني معطيات بلغة Pascal جامعة تشرين 2014، 2007، 1998

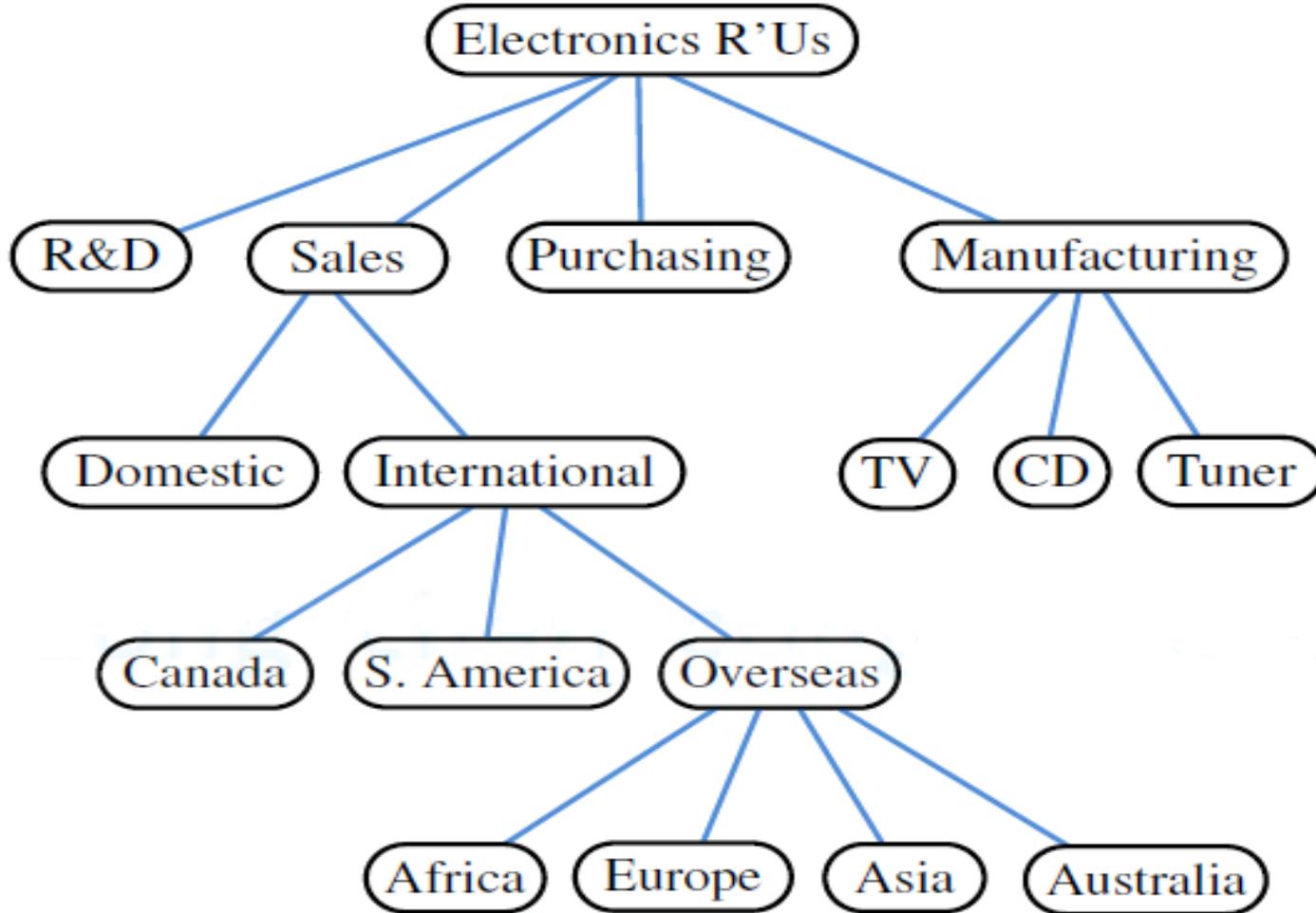
1- تعريف الشجرة وخصائصها

- يقول خبراء الإنتاجية إن التطور الحاسم في لغات البرمجة تأتي من خلال التفكير اللاخطي **nonlinearly**.
- تعتبر الأشجار **trees** من أهم بنى المعطيات اللاخطية في مجال الحوسبة،
- وهي أحد الفتوحات الكبيرة في مجال تنظيم البيانات، فهي تتيح لنا تحقيق عدة خوارزميات ذات أداء أسرع بكثير من حالة استخدام بنى معطيات خطية، مثل اللائحة.
- تقدم الأشجار أيضاً تنظيماً طبيعياً للبيانات، فأصبحت بنى موجودة في كل مكان في نظام الملفات، واجهة المستخدم الرسومية، قواعد البيانات، مواقع الويب، وباقي الأنظمة الحاسوبية.
- عندما نقول إن الأشجار لاخطية، فإننا نشير إلى علاقة تنظيمية أكثر غنى من علاقات " قبل before " و " بعد after " بين الأغراض في بنى المتتاليات.
- إن العلاقات في الأشجار مترتبة هرمياً **hierarchical**، حيث تكون بعض الأغراض " فوق above " وبعضها " تحت below ".
- إن المصطلحات الرئيسية لبنية المعطيات الشجرة، مشتقة من شجرة العائلة، مع استخدام مصطلحات مثل " أب parent " و " ابن child "، " سلف ancestor " و " سليل descendent " كأكثر المصطلحات شيوعاً لوصف العلاقات.

- إن الشجرة tree هي نمط بيانات مجرد يخزن العناصر هرمياً، فإن كل عنصر له عنصر أب parent وقد يكون له عنصر ابن child أو أكثر، مع استثناء العنصر الأعلى (root) لا أب له.
- يتم تمثيل الشجرة عادة من خلال وضع العناصر في أشكال مستطيلة أو بيضوية، ورسم وصلات أو حدود بين الأبناء والأبناءهم بواسطة خطوط مستقيمة (كما يبين الشكل 2-5)، نسمي العنصر الأعلى في الشجرة باسم جذر الشجرة root، ولكنه يرسم كأعلى عنصر، وتكون باقي العناصر متصلة إلى الأسفل.
- يبين الشكل 2-5 شجرة ذات 17 عقدة تمثل شركة وهمية، حيث إن الجذر هو Electronics R'Us وأبناء الجذر هم R&D، Manufacturing، Purchasing، sales.... إلخ.

Tree Definition and Properties

1- تعريف الشجرة وخصائصها 3



الشكل 5-2- شجرة تمثل شركة افتراضية. العمق الارتفاع
NLR

رسمياً، نعرف الشجرة T بأنها مجموعة من العقد تخزن عناصر بحيث إن جميع العقد تملك العلاقة "أب-إبن parent-child" والتي تحقق الصفات التالية:

- إذا كانت T غير فارغة، فإنها تحوي عقدة خاصة تدعى جذر الشجرة T أو $root$ على الأقل، وهذه العقدة ليس لها أب.
 - كل عقدة v في T (ماعدا الجذر)، لها عقدة أب $parent$ واحدة لتكن w ، وكل عقدة يكون أبوها w ، تدعى "إبن child" w .
- وفقاً لهذا التعريف، يمكن للشجرة أن تكون فارغة، وهذا يعني أنها لا تحوي أي عقدة.
- يتيح هذا الاصطلاح أن تعرف الشجرة عودياً بحيث إن الشجرة T هي إما فارغة أو تحوي عقدة r تسمى جذر T و مجموعة من الأشجار التي تكون جذورها هي أبناء r (وقد لا تملك أبناء).

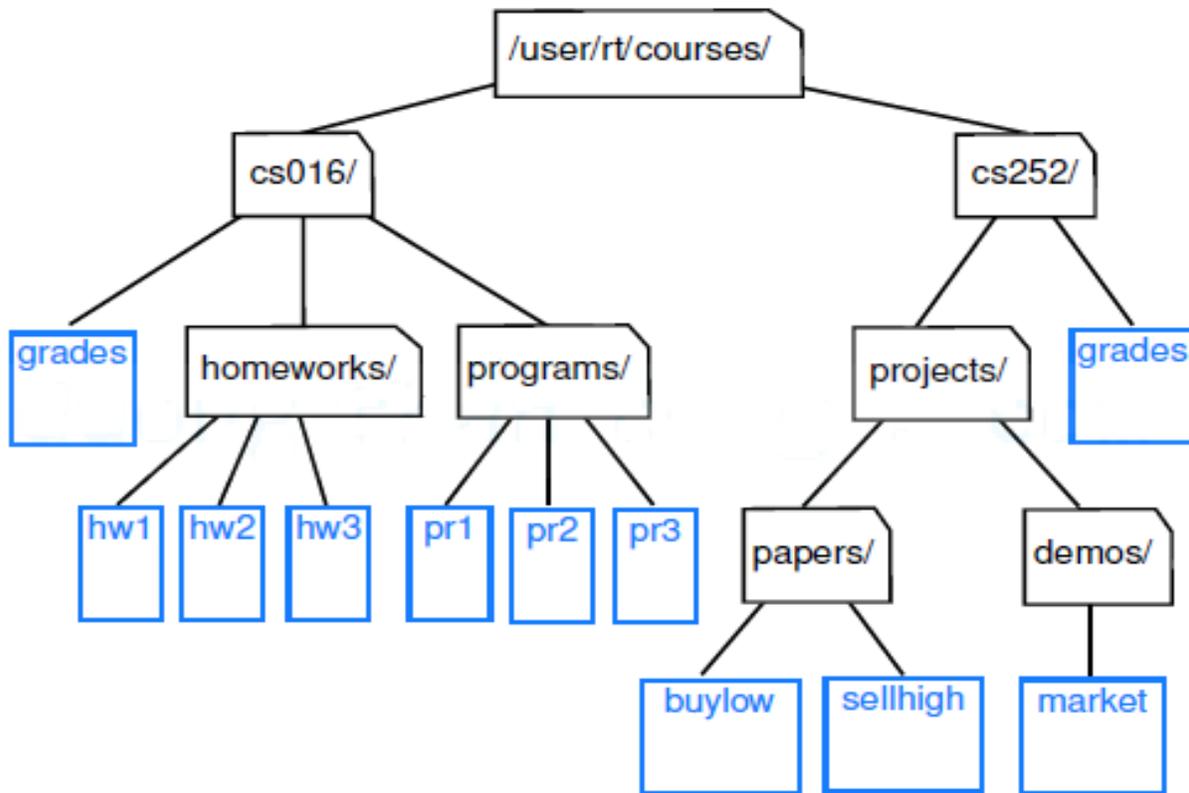
عقدة الأب يكون لها أبناء، ندعو عقدتين أبناء لنفس الأب والمستوى بأنهن أخوة siblings. ونقول عن عقدة v أنها خارجية external إذا لم يكن لها أبناء وتعرف باسم الأوراق leaves، وندعو عقدة بأنها داخلية internal or Interior إذا كان لها أب وإبن أو أكثر.

مثال 1-تنظم الملفات في أغلب نظم التشغيل هرمياً في مجلدات متداخلة، حيث يتم عرضها للمستخدم على شكل شجرة (الشكل 5-3). وبتخصيص أكبر، تربط العقد الداخلية للشجرة مع مجلدات، وتربط العقد الخارجية مع ملفات. يدعى جذر الشجرة في نظم التشغيل Linux أو Unix أو Windows أو macOS بالاسم المجلد الجذر root directory ويشار إليه بالرمز /.

Formal Tree Definition

2- التعريف الرسمي للشجرة

تكون عقدة u سلف v ancestor لعقدة v ، إذا كانت u أب v أو كانت u سلف لأب العقدة v ، وباتجاه معاكس، نقول أن عقدة v هي سليفة descendant لعقدة u ، إذا كانت u سلف v .
cs252/ is an ancestor of papers/, and pr3 is a descendant of cs016/.



على سبيل المثال، في الشكل 3-5، المجاور

- إن cs252/ هي سلف papers/،

- pr3 هي سليفة cs016/.

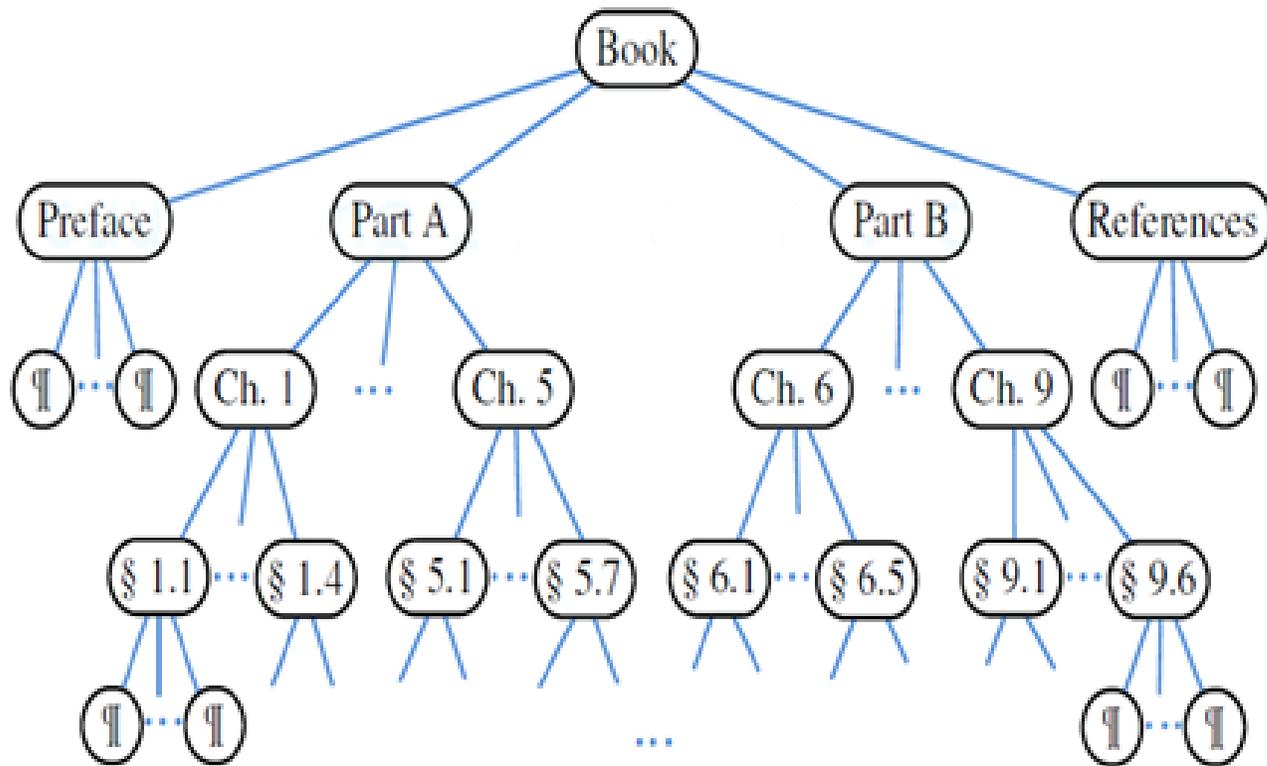
- الشجرة الجزئية subtree من الشجرة T ذات الجذر v هي

شجرة تحوي v وجميع نسل v في T.

- على سبيل المثال، الشجرة الجزئية ذات الجذر cs016/

تحتوي العقد cs016/، grades/، homeworks/،

programs/، hw1، hw2، hw3، pr1، pr2 و pr3.



الشكل 4-5- شجرة مرتبة تعبر عن كتاب.

- **حد edge الشجرة T أو Branch** هو الوصلة ما بين زوج من العقد (u,v) بحيث إن u هي أب لـ v ، أو العكس.

- أما **مسار path الشجرة T** هو تتالي من العقد مع حدودها على سبيل المثال، تحوي الشجرة في الشكل 3-5 المسار: $(cs252/, projects/, demos/, market)$

- مثال 3- إن مكونات مستند هيكلية، مثل كتاب على سبيل المثال (الشكل 4-5)، منظمة هرمياً على شكل شجرة تكون عقدها الداخلية هي الأجزاء، الفصول، الفقرات. وتكون عقدها الخارجية هي المقاطع، الجداول، الأشكال ... وهكذا.

تخزن الشجرة كنمط بيانات مجرد عناصرها في مواقع `positions`.
المواقع `positions` في شجرة هي عقدها `nodes`، حيث تحقق المواقع المجاورة هرمياً العلاقات أب-إبن `parent-child` والتي تعرف شجرة.

وبالتالي، فإن المصطلحان موقع `position` وعقدة `node` مترادفان بالنسبة للأشجار.
وكما في موقع اللائحة، فإن الغرض `position` بالنسبة للشجرة يدعم الطريقة التالية:

- `element()`: يعيد الغرض المخزن في ذلك الموقع.
إن قوة مواقع العقد في الشجرة تتأتى من طرائق الوصول `accessory methods` للشجرة والتي تعيد وتقبل المواقع، كما في الطرائق التالية:

- تعيد جذر الشجرة، يحدث خطأ إذا كانت الشجرة فارغة.

- `parent(v)`: تعيد أب `v`، يحدث خطأ إذا كانت `v` هي الجذر.

- `children(v)`: تعيد مجموعة تتضمن أبناء العقدة يحدث خطأ إذا كانت `v` هي ورقة.

إذا كانت الشجرة `T` مرتبة، فإن المجموعة `children(v)` تخزن أبناء `v` بشكل مرتب. وإذا كانت `v` عقدة خارجية، عندئذ `children(v)` مجموعة فارغة.

The Tree Abstract Data Type and Paths in Trees



الشجرة كنمط بيانات مجرد

بالإضافة إلى طرائق الوصول الأساسية السابقة، نقوم أيضاً بتضمين طرائق الاستعلام query methods التالية:

- `isInternal(v)`: تختبر فيما إذا كانت العقدة `v` داخلية.

- `isExternal(v)`: تختبر فيما إذا كانت العقدة `v` خارجية.

- `isRoot(v)`: تختبر فيما إذا كانت العقدة `v` جذراً.

إن هذه الطرائق تجعل البرمجة باستخدام الأشجار أسهل وأكثر قابلية للقراءة، وذلك لأننا نستخدمها في الأوامر الشرطية `if` والحلقات مثل `while`.

هناك أيضاً عدد من الطرائق العامة generic methods يجب أن تدعمها الأشجار رغم أنها ليست بالضرورة مرتبطة بالبنية الشجرية، من بينها:

- `size()`: تعيد عدد العقد في الشجرة.

- `isEmpty()`: تختبر فيما إذا كانت الشجرة فارغة.

- `iterator()`: تعيد المكرر iterator لجميع العناصر المخزنة في عقد الشجرة.

- `positions()`: تعيد مجموعة قابلة للتجول التكراري من جميع العقد في الشجرة.

- `replace(v,e)`: تعيد العنصر المخزن في العقدة `v` وتستبدله بـ `e`.

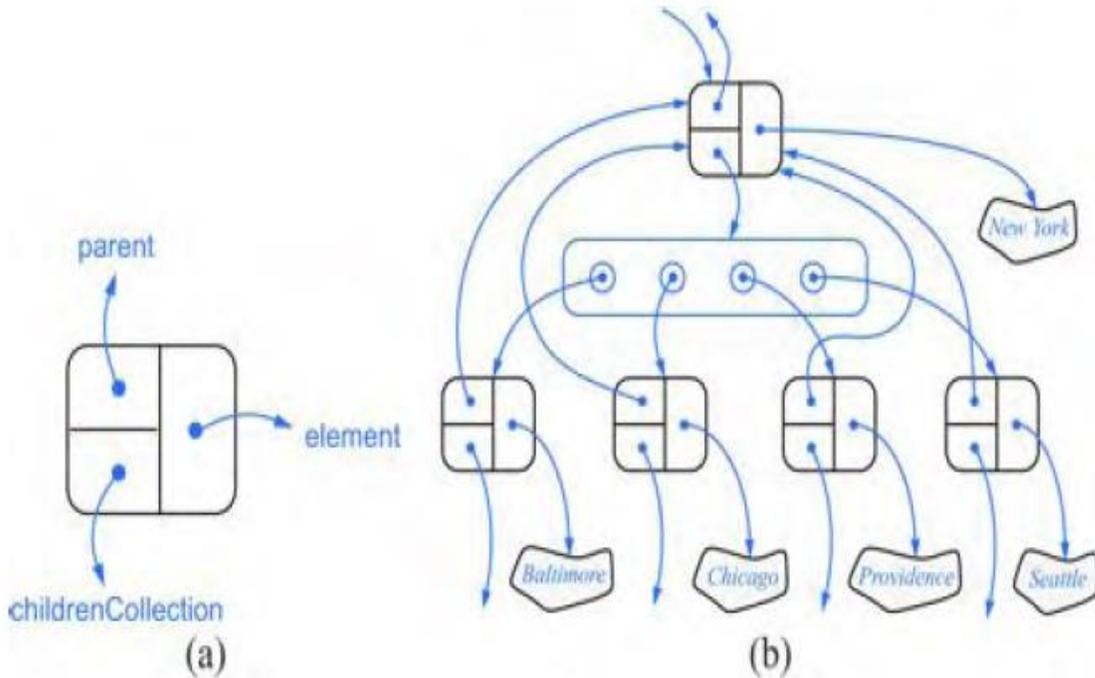
ALinked Structure for General Trees

البنية المترابطة للأشجار العامة

من الطرق الطبيعية لتحقيق شجرة T هي استخدام البنية المترابطة، حيث تمثل كل عقدة v من T من خلال الغرض position (الشكل 5-5-a) بحيث يحوي الحقول التالية:

1- مرجع إلى العنصر المخزن في v. 2- ارتباط link إلى أب العقدة v. 3- مجموعة من نوع ما (لائحة أو مصفوفة) لتخزين روابط إلى أبناء

v. إذا كانت v هي جذر T، عندئذ فإن حقل الأب لـ v هو null. كما أننا نخزن مرجعاً إلى جذر T وأرقام العقد في T في متحولات داخلية. يمكن تمثيل هذه البنية تخطيطياً في الشكل 5-5-b.



الشكل 5-5- البنية المترابطة لشجرة عامة.

Operation	Time
size , isEmpty	O(1)
positions	O(n)
replace	O(1)
root, parent	O(1)
isInternal, isExternal, isRoot	O(1)

جدول يمثل درجة التعييد للعمليات المدرجة.

توجد طريقتان للتجول:

- التجول بالعمق: Depth-first traversal (DFT)
- التجول بالعرض: Breadth-first traversal (BFT)

خوارزميات إنجاز عمليات التجول بالعمق على شجرة من خلال الوصول إليها من خلال طرائق الشجرة كنمط بيانات مجرد.

نعرف العمق والارتفاع Depth and Height: لتكن v عقدة في الشجرة T ، عمق depth العقدة v هو عدد الآباء لـ v ، على سبيل المثال، في الشجرة المبينة في الشكل 2-5، عمق العقدة التي تخزن International هو 2، لاحظ أن هذا التعريف يعني أن عمق جذر الشجرة T هو 0 وعمق الشجرة الفارغة هو 1.

إن زمن التنفيذ للخوارزمية $depth(T, v)$ هو $O(d_v)$ حيث تمثل d_v عمق العقدة v في الشجرة T . وذلك لأن الخوارزمية تنفذ خطوة عودية ذات زمن ثابت من أجل كل أب لـ v . وبالتالي، فإن الخوارزمية $depth(T, v)$ تنفذ في زمن $O(n)$ في الحالة الأسوأ، حيث n هو العدد الكلي للعقد في T ، وذلك بما أن عقدة ما في الشجرة T يمكن أن تكون ذات عمق $n-1$ في الحالة الأسوأ.

نعرف ارتفاع height عقدة v في شجرة T هو عدد الحدود الأعظمي للوصول إلى العقدة الخارجية، وبشكل عودي على النحو:

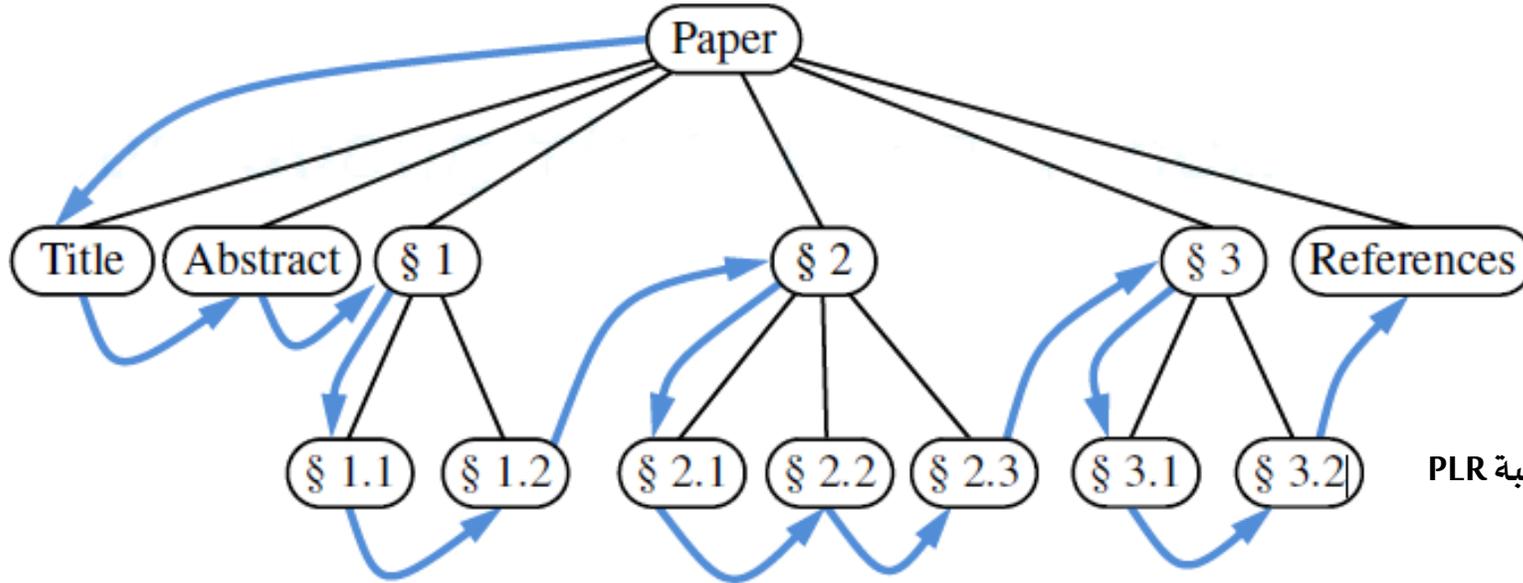
- إذا كانت v عقدة خارجية، عندئذ فإن ارتفاعها هو 0.
 - وإلا، إرتفاع v هو الارتفاع الأعظمي لابن العقدة v مضافاً إليه واحداً.
- إن ارتفاع شجرة غير فارغة T هو ارتفاع جذر الشجرة، على سبيل المثال، الشجرة في الشكل 2-5 ارتفاعها هو 4.

توجد لدينا ثلاث طرق قياسية للتجول في العمق:

- التجول السابق للترتيب (Node, Left, Right NLR) . Pre-order Traversal
- التجول اللاحق للترتيب (Left, Right, Node LRN) . Post-order Traversal
- التجول بالترتيب (Left, Node, Right LNR) . In-order Traversal

نبدأ بالتجول السابق للترتيب preorder traversal لشجرة T، يتم زيارة جذر الشجرة T أولاً ومن ثم يتم التجول عودياً عبر الأشجار الفرعية والتي جذورها هي أبناء جزر الشجرة T. إذا كانت الشجرة مرتبة ordered عندئذ فإن الأشجار الفرعية يتم التجول عليها وفقاً لترتيب الأبناء. إن الفعل المحدد المرافق لعملية زيارة عقدة ما v يتوقف على التطبيق الخاص بالتجول، وقد يتضمن أي شيء بدءاً من مزايدة عداد وانتهاء بالقيام ببعض العمليات الحسابية المعقدة على v .

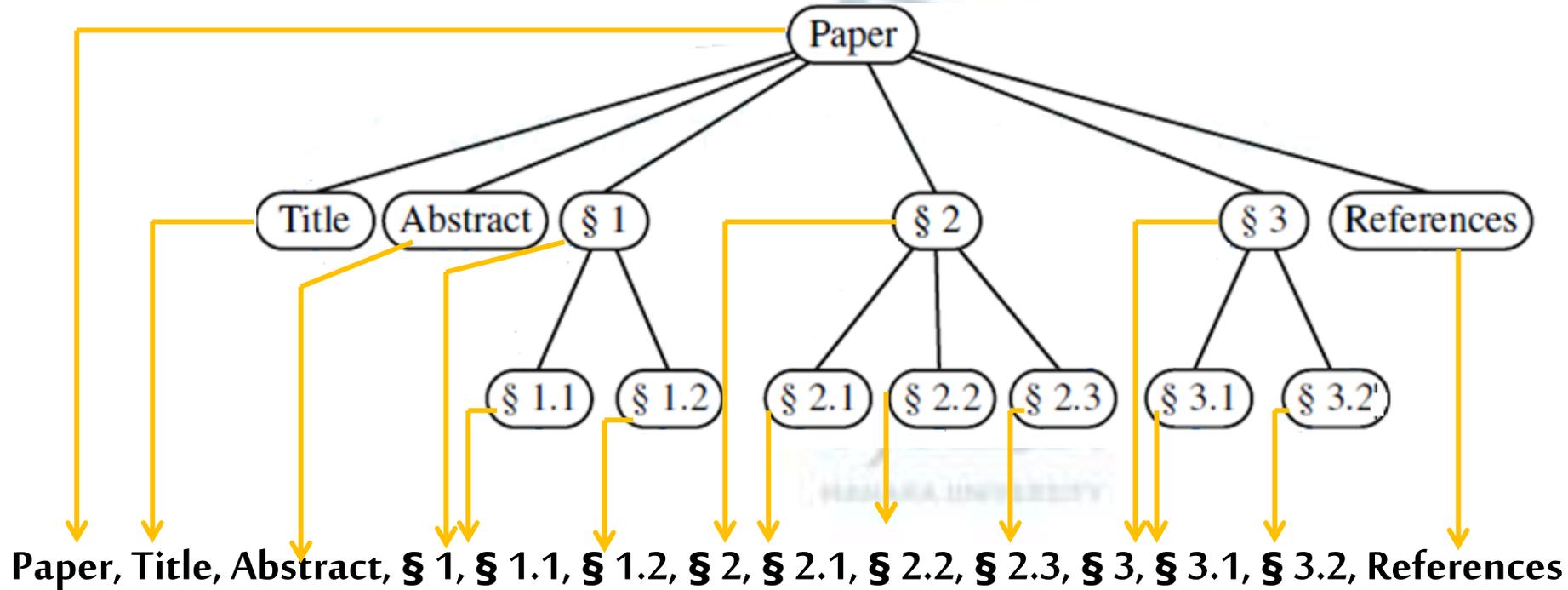
إن خوارزمية التجول السابق للترتيب مفيدة من أجل إنتاج ترتيب خطي لعقد شجرة بحيث إن الآباء يجب أن يأتوا قبل أبنائهم في الترتيب. إن مثل هذا الترتيب له العديد من التطبيقات المختلفة، نبين في المثال التالي أحدها. إن التجول السابق للترتيب للشجرة المرفقة بمسند (كما في مثال سابق) تختبر كامل المسند بشكل متتالي، منذ البداية وحتى النهاية. إذا تمت إزالة العقد الخارجية قبل التجول، عندئذ فإن التجول يختبر جدول المحتويات للمسند (كما يبين الشكل التالي).



الشكل 5-6- التجول السابق للترتيب لشجرة مرتبة PLR

Paper, Title, Abstract, § 1, § 1.1, § 1.2, § 2, § 2.1, § 2.2, § 2.3, § 3, § 3.1, § 3.2, References

إن خوارزمية التجول السابق للترتيب بأسلوب حفزي ولا نحيدة يكون بالخروج من يسار العقدة وأن لا يقطع الرابط مع الابن وفق NLR وتجمع من اليسار لليمين عند نهايات الاسهم.



الشكل 5-6- التجول السابق
للترتيب لشجرة مرتبة PLR

هناك تطبيق هام لخوارزمية التجول السابق للترتيب، وهو يولد تمثيلاً لشجرة كاملة على شكل سلسلة محرفية.

لنفترض مجدداً، من أجل كل عنصر e مخزن في شجرة T ، أن استدعاء $e.print()$ يعيد سلسلة محرفية مرتبطة بـ e . فإن التمثيل الأبوي على شكل سلسلة محرفية **parenthetic string representation** يمكن ان يعرف عودياً كمايلي:

- إذا كانت الشجرة T مؤلفة من عقدة واحدة v فإن $P(T) = v.element().print()$

- وإلا فإن $P(T) = v.element().print() + "(" + P(T_1) + "," + P(T_2) + "," + \dots + P(T_k) + ")"$

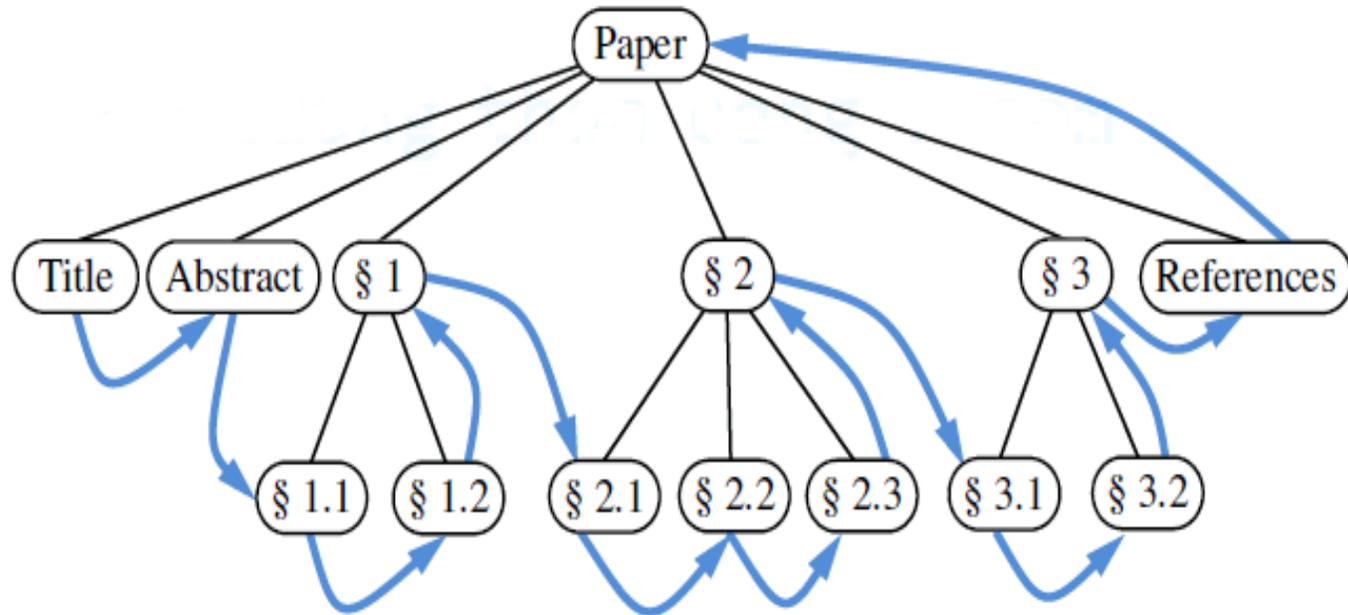
حيث إن v هي جذر T و T_1, T_2, \dots, T_k هي الأشجار الفرعية التي جذورها هي أبناء v .

بناء عليه فإن التمثيل الأبوي على شكل سلسلة محرفية للشجرة المبينة في [الشكل 2-5](#) هو على النحو التالي:

Electronics R'Us (R&D Sales (Domestic International (Canada S. America
 Overseas (Africa Europe Asia Australia))
 Purchasing Manufacturing (TV CD Tuner))

الشكل 5-7- التمثيل الأبوي على شكل سلسلة محرفية.

خوارزمية التجول اللاحق للترتيب postorder traversal. يمكن النظر إلى هذه الخوارزمية على أنها معاكسة لخوارزمية التجول السابق للترتيب، وذلك لأنها تتجول عودياً على الأشجار الفرعية التي جذورها أبناء لجذر الشجرة أولاً ومن ثم تزور الجذر. إن أصل التسمية "التجول اللاحق للترتيب postorder traversal" يأتي من حقيقة أن طريقة التجول ستقوم بزيارة v بعد أن تكون قد قامت بزيارة جميع العقد في الشجرة الفرعية التي جذورها v (عقد الابناء).



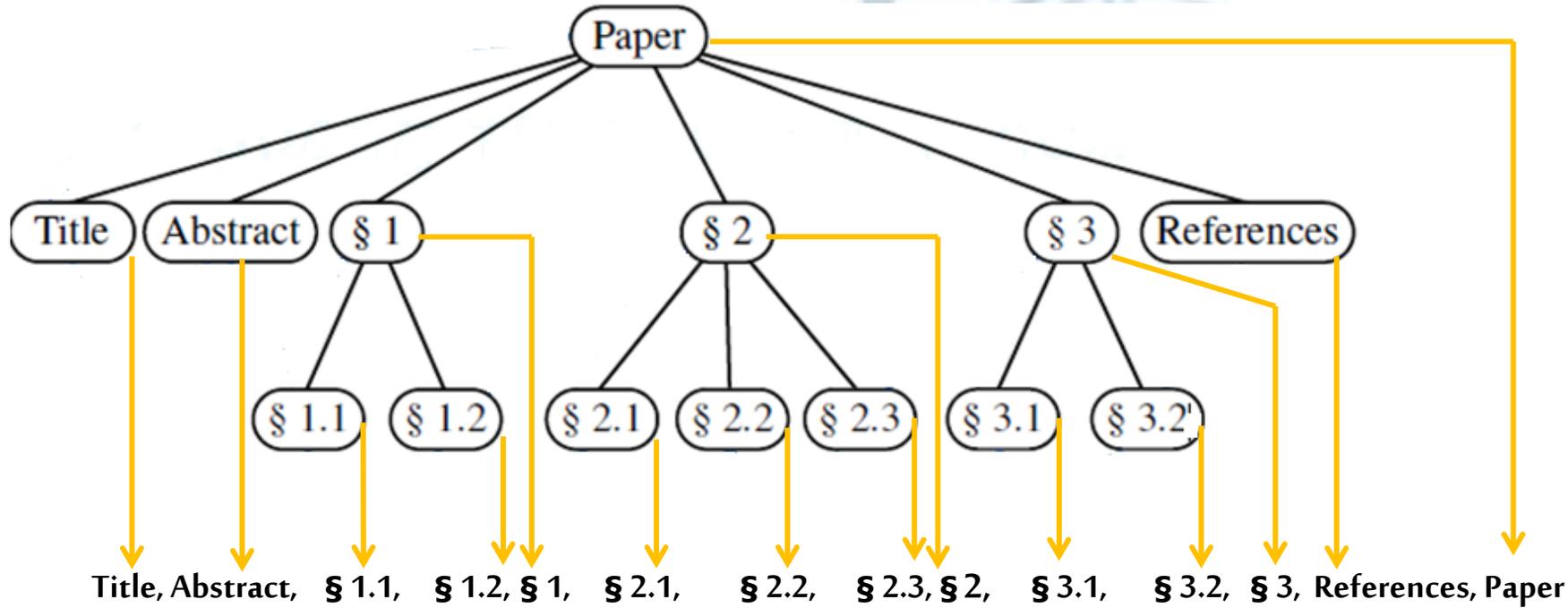
يبين الشكل 5-8 هذا التجول المماثل للشكل 5-6:

فإن التجول اللاحق للترتيب لشجرة T ذات n عقدة يستغرق زمناً $O(n)$ وذلك بفرض أن زيارة كل عقدة تتطلب زمناً $O(1)$. وبالتالي، فإن التجول اللاحق للترتيب تنفذ في زمن خطي.

الشكل 5-8- التجول اللاحق للترتيب عبر شجرة مرتبة. LRP

Title, Abstract, § 1.1, § 1.2, § 1, § 2.1, § 2.2, § 2.3, § 2, § 3.1, § 3.2, § 3, References, Paper

إن خوارزمية التجول اللاحق للترتيب بأسلوب حفزي ولا نخبذة يكون بالخروج من يمين العقدة وأن لايقطع الرابط مع الابن وفق LRN وتجمع من اليسار لليمين عند نهايات الاسهم.



الشكل 5-8- التجول اللاحق
للترتيب عبر شجرة مرتبة LRP.

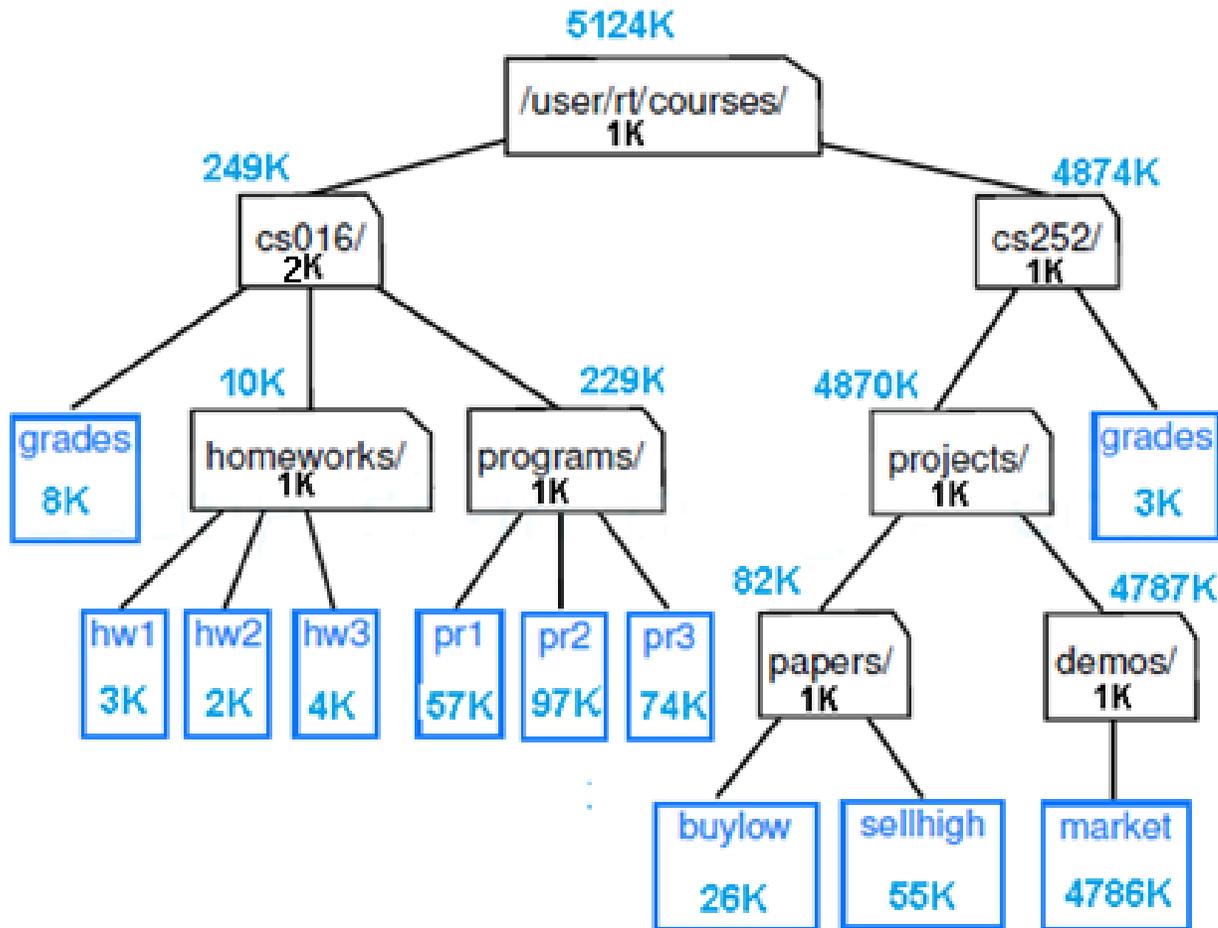
إن طريقة التجول اللاحق للترتيب مفيدة في حل المسائل التي نرغب فيها بحساب صفة ما لكل عقدة في شجرة، إلا أن حساب تلك الصفة ل v يتطلب أن نكون قد حسبنا نفس الصفة لأبناء v .

كمثال على هذه الحالة، ليكن لدينا شجرة نظام الملفات T ، حيث تمثل العقد الخارجية الملفات والعقد الداخلية المجلدات. وبفرض أننا نرغب بحساب مساحة القرص التي يشغلها مجلد ما، فإن هذا الأمر يحسب عودياً على أنه مجموع:

- حجم المجلد بحد ذاته.
 - حجم الملفات في المجلد.
 - الحجم المشغول بالمجلدات الأبناء.
- بالنظر إلى الشكل 5-9، إن هذا الحساب يمكن أن يتم تنفيذه بالتجول اللاحق للترتيب عبر الشجرة T ، فبعد أن يتم التجول عبر الأشجار الفرعية لعقدة داخلية، نقوم بحساب الحجم المستخدم من قبل v من خلال جمع حجوم المجلد v بحد ذاته والملفات المحتواة في v إلى الحجم المشغول من قبل كل عقدة ابن داخلية ل v ، والتي حسبت من خلال التجولات العودية اللاحقة للترتيب لأبناء v .

DFT Postorder Traversal LRP

التجول اللاحق للترتيب وفق 5 DFT



الشكل 5-9- شجرة الملفات مرفقة بحجوم الملفات.

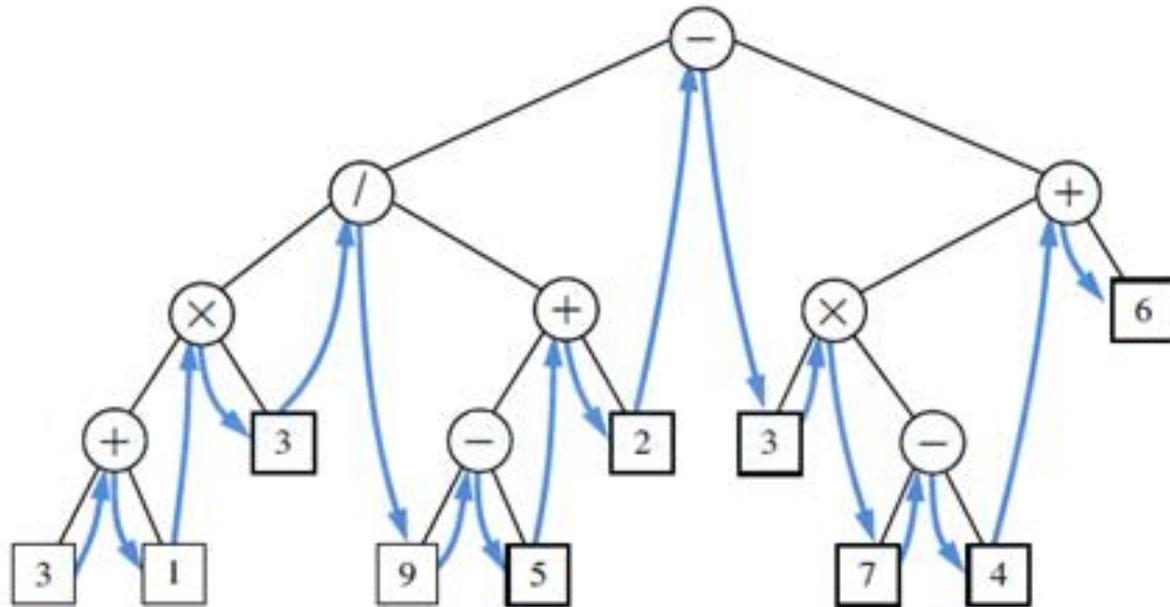
ويمكن كتابة الطريقة diskSpace لتستخدم التجول اللاحق للترتيب لشجرة نظام الملفات T، لطباعة أسماء الملفات ومساحة القرص المستخدم من قبل المجلد المرفق بكل عقدة داخلية من T. عندما يتم استدعاؤها على جذر الشجرة T فإنها تنفذ في زمن $O(n)$ حيث n هو عدد العقد في T.

النوع الثالث للتجوال InOrder LNR سيتم دراسته لاحقاً

DFT Inorder Traversal of a Binary Tree

التجول المرتب عبر شجرة ثنائية وفق DFT 1

طريقة إضافية للتجول عبر شجرة ثنائية، هي التجول المرتب inorder traversal. في هذا التجول، نقوم بزيارة عقدة بين التجولات العودية من أشجارها اليسارية واليمنية. يبين المقطع التالي خوارزمية هذا النوع من التجول:



الشكل 5-10- التجول المرتب عبر شجرة ثنائية LPR.

Algorithm inorder(T,v):

if v has a left child u in T **then**

 inorder(T,u) {recursively traverse left subtree}

 perform the "visit" action for node v

if v has a right child w in T **then**

 inorder(T,w) {recursively traverse right subtree}

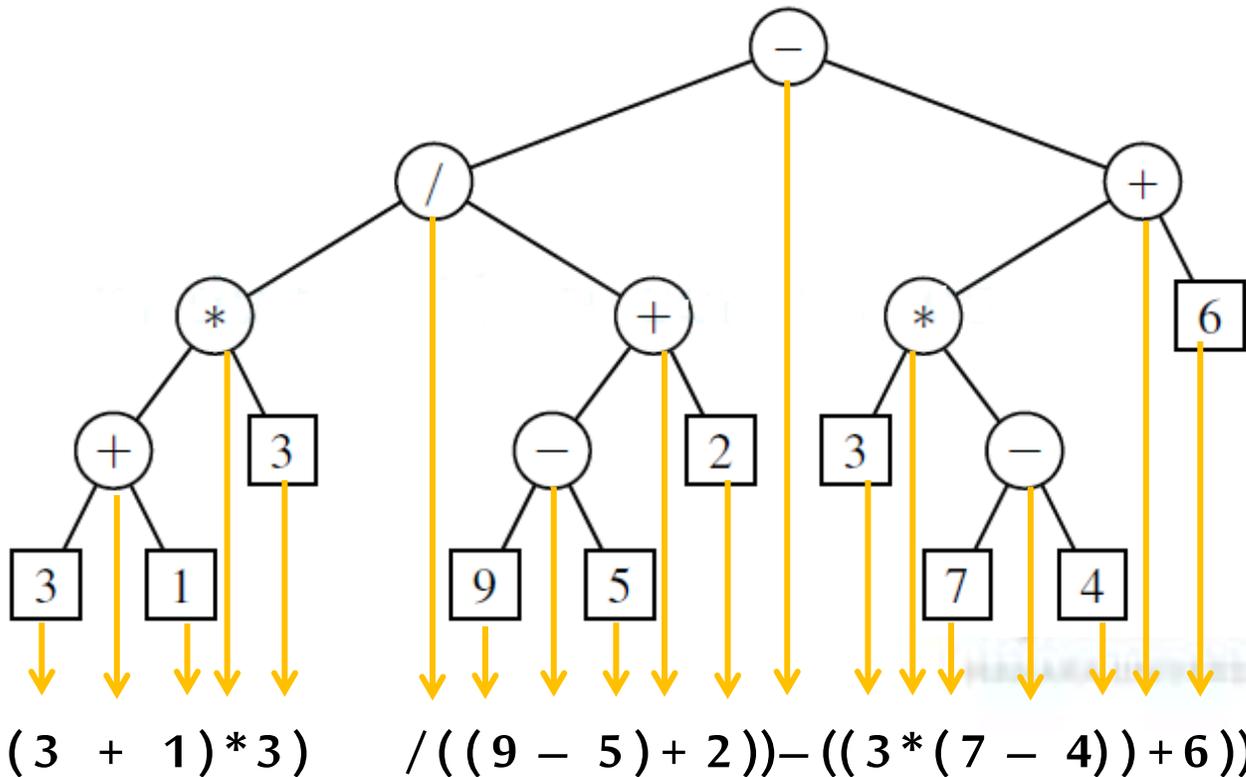
$$(((3 + 1) * 3) / ((9 - 5) + 2)) - ((3 * (7 - 4)) + 6)$$

خوارزمية inorder

DFT Inorder Traversal of a Binary Tree

التجول المرتب عبر شجرة ثنائية وفق DFT 2

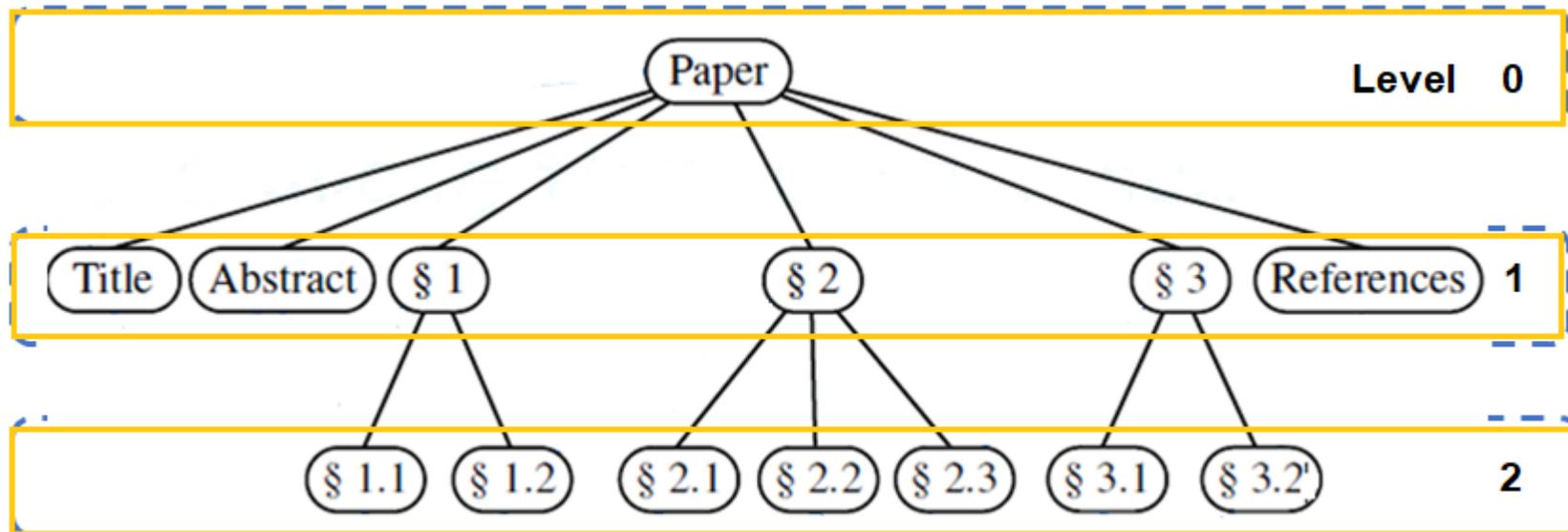
إن خوارزمية التجول الترتيب بأسلوب حفظي ولا
نحبذة يكون بالخروج من يسار العقدة وأن لايقطع
الرابط مع الابن وفق LNR وتجمع من اليسار لليمين
عند نهايات الاسهم.



الشكل 5-10- التجول المرتب عبر شجرة ثنائية LPR.

Breadth-first traversal

التجوال بالعرض



Paper, Title, Abstract, § 1, § 2,, § 3, References § 1.1, § 1.2, § 2.1, § 2.2, § 2.3, § 3.1, § 3.2,

BFT ()

if (!IsEmpty()) Q= new queue , Q.enqueue(root)

While (!Q.IsEmpty()) {Node n=Q.dequeue() , print n,

if(n.left !=null) Q.enqueue (n.left) ,

if(n.right !=null) Q.enqueue(n.right))

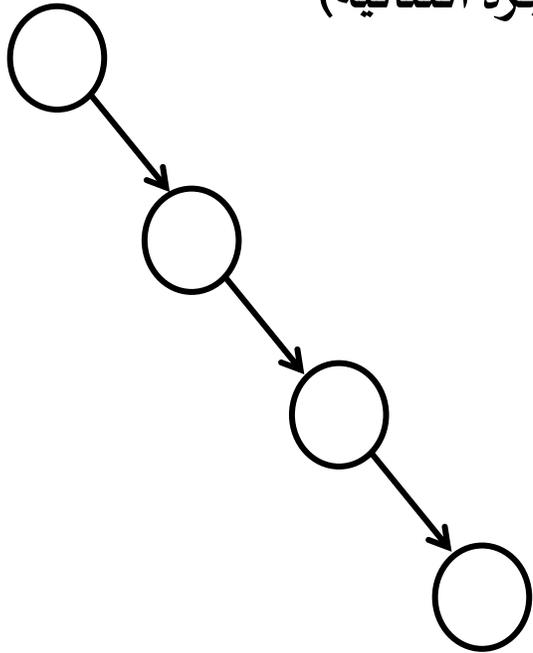
- التجوال بالعرض: Breadth-first traversal
- يتم طباعة المستوى 0 ثم المستوى 1 من اليسار لليمين ثم المستوى 3 بنفس الأسلوب حتى النهاية.

خوارزمية التجوال بالعرض:
Breadth-first traversal

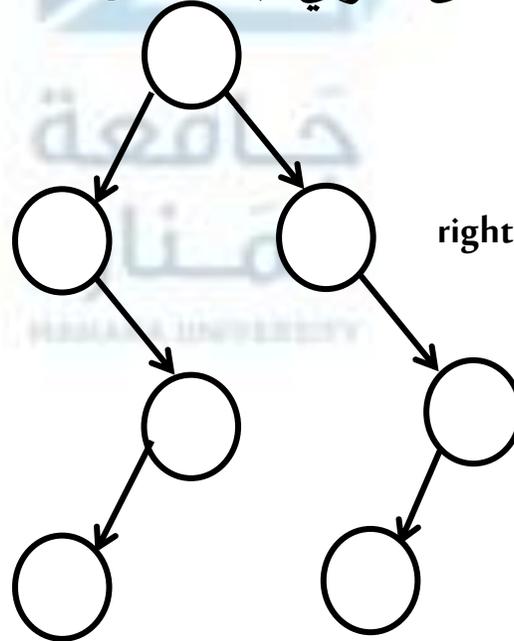
الشجرة الثنائية binary tree هي شجرة مرتبة تملك الخصائص التالية:

- كل عقدة لها على الأكثر إثنين. - كل عقدة ابن يشار إليها على أنها إما ابن يسار left child أو ابن يمين right child.
- الابن اليسار يسبق الابن اليمين في ترتيب أبناء العقدة.

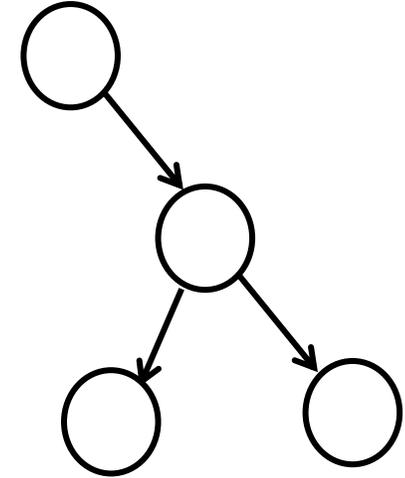
تدعى الشجرة الفرعية التي جذرها عند الابن اليسار أو اليمين لعقدة داخلية v بالاسم شجرة فرعية يسارية left subtree أو شجرة فرعية يمينية right subtree على التوالي. (الاشكال الثلاث تحقق شرط الشجرة الثنائية)



left subtree



right subtree

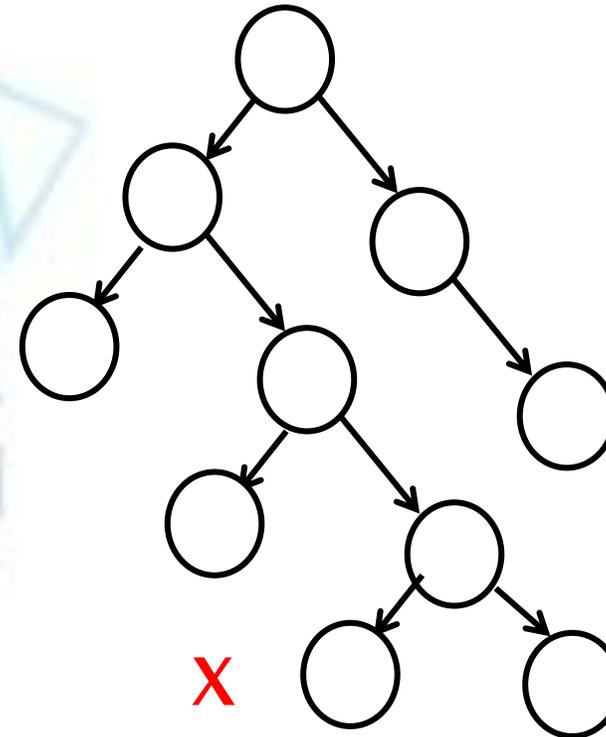
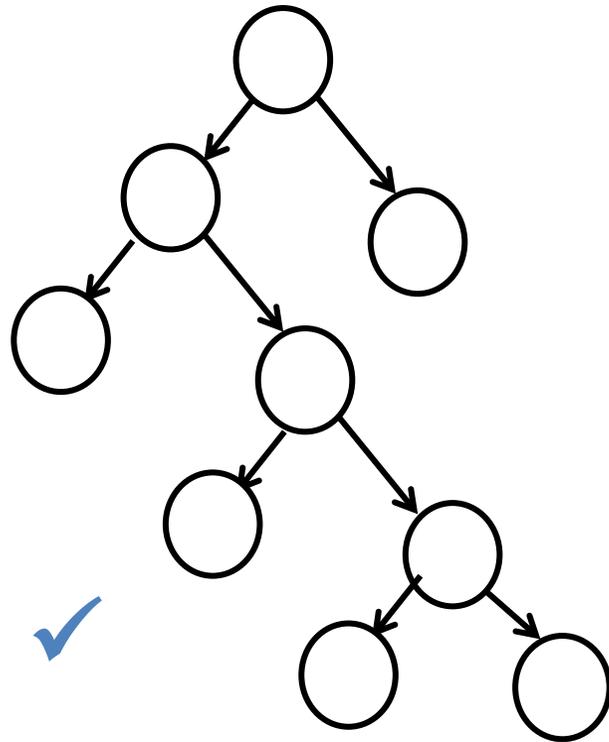


Binary Trees

الأشجار الثنائية 2

يقال عن شجرة ثنائية أنها ممتلئة (Strictly) Binary Tree full إذا حققت الشرط أن يكون لكل عقده عقدين أو صفر عقدة.

- if every node has zero or two children.

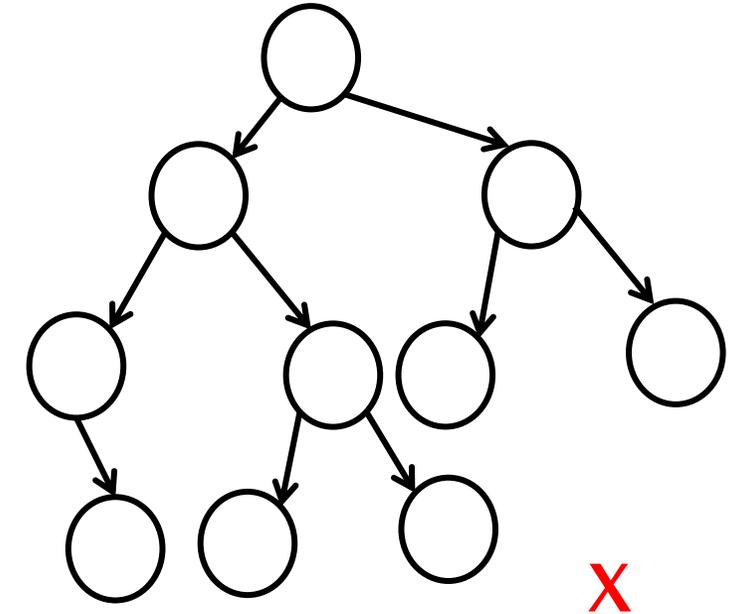
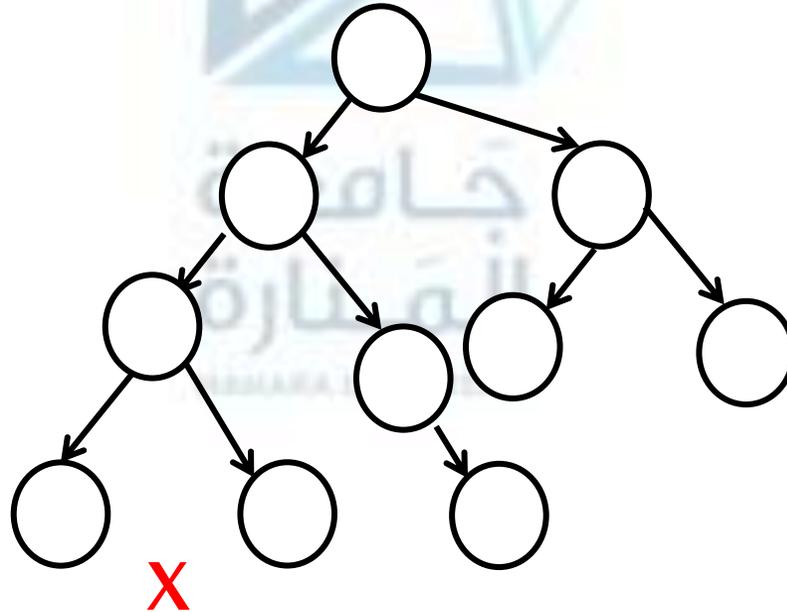
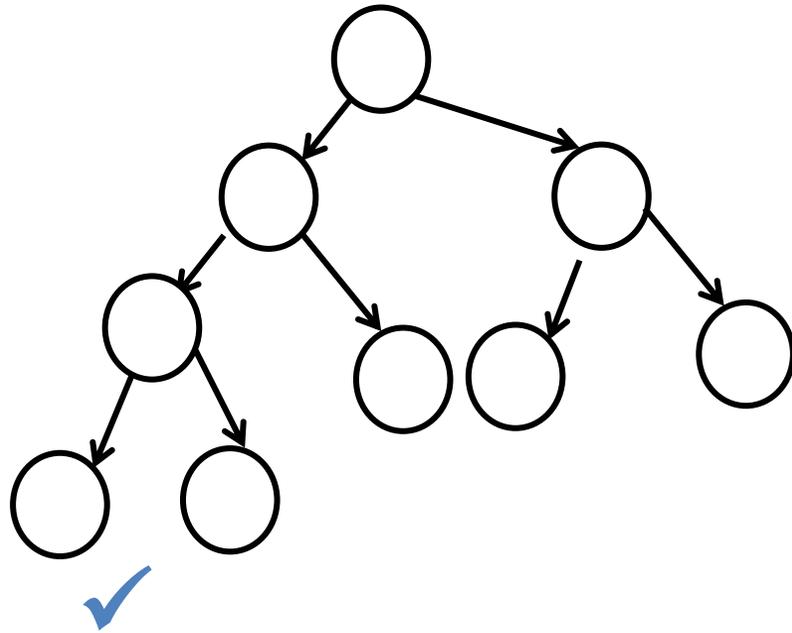


Binary Trees

الأشجار الثنائية 3

الشجرة الثنائية المكتملة Complete Binary Tree تكون كل عقدة غير ورقة تملك إثنين، جميع العقد في المستوى الأخير معتلة من اليسار، وتسمى عادة بالشجرة proper

1. In a complete tree, every non-leaf node must have exactly two children.
2. All Nodes as left as possible in last level.

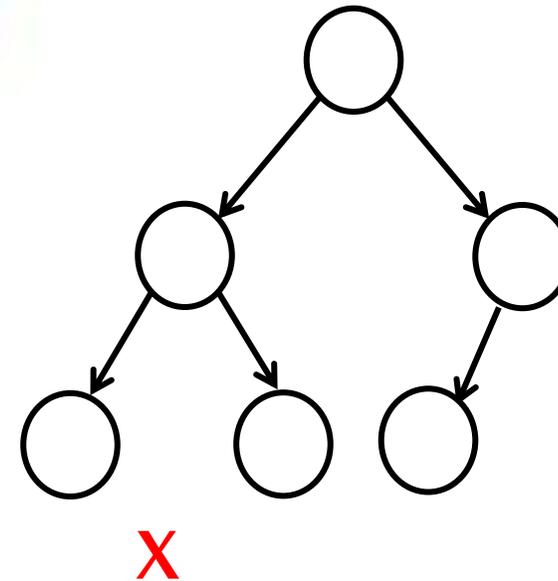
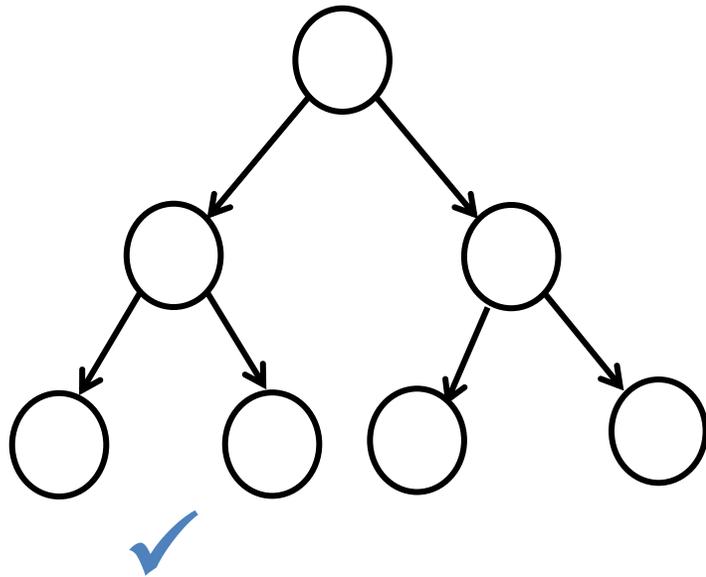


Binary Trees

الأشجار الثنائية 4

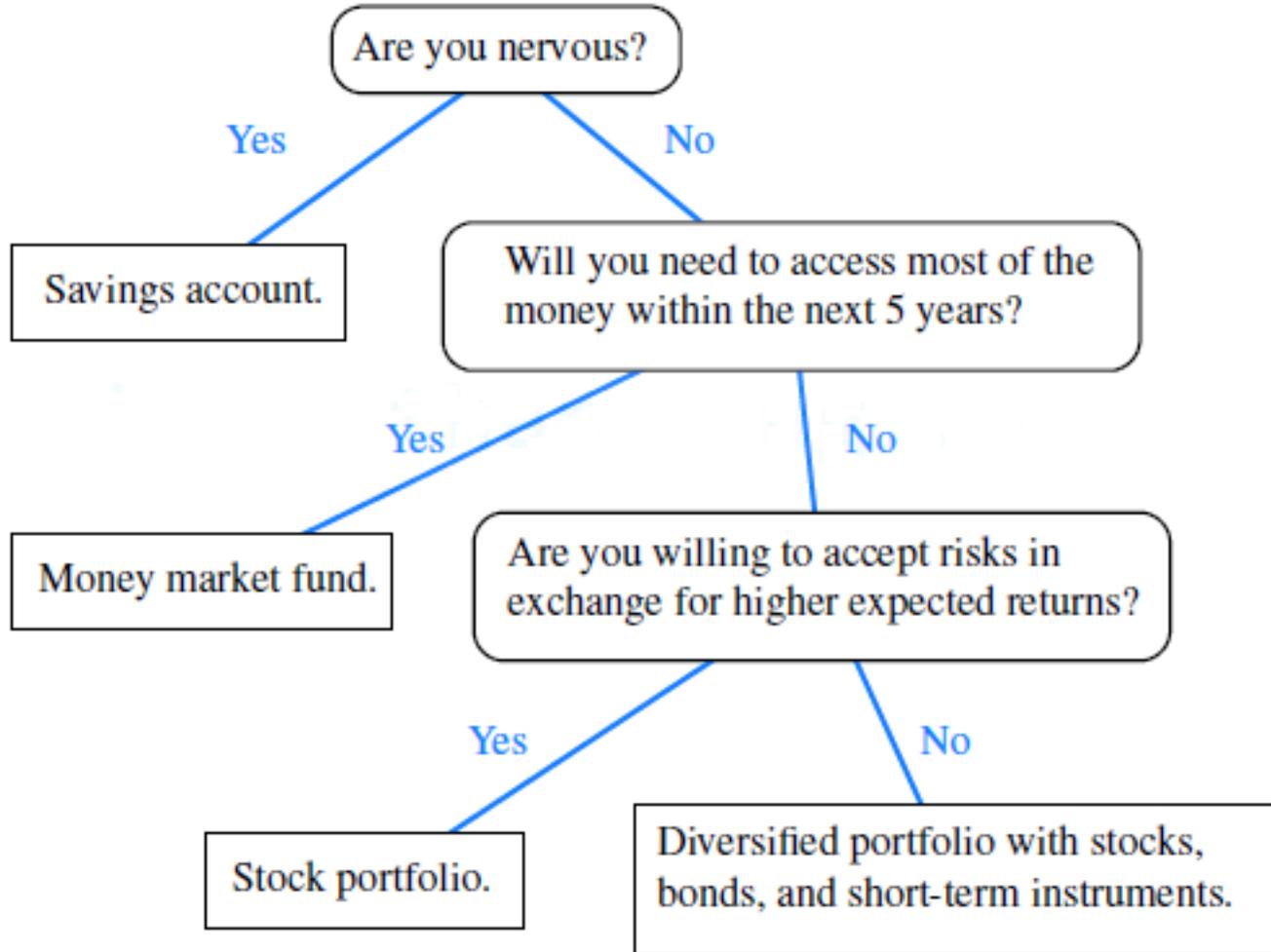
يقال عن شجرة ثنائية أنها مثالية (Perfect Binary Tree) (All levels in completely filled) إذا كانت كل عقدة داخلية تملك إبنين وجميع العقد الورقية في نفس المستوى.

1. Every nodes two children.
2. All leaves are at the same level.



Binary Trees

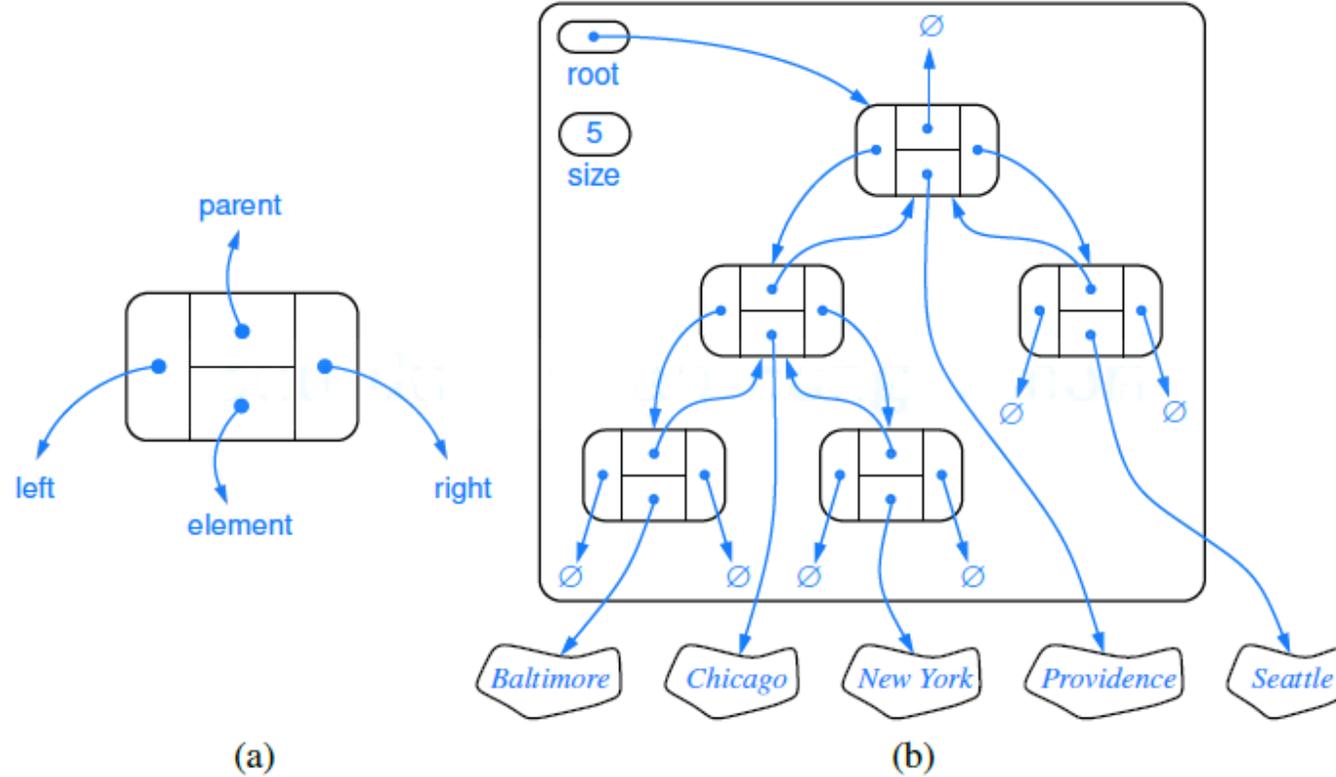
الأشجار الثنائية 5



مثال 1- أحد الأصناف الهامة للأشجار الثنائية أشجار القرار decision trees يمكن أن تنتج من الإجابة عن سلسلة من الأسئلة ذات الإجابات (نعم-لا). تكون كل عقدة داخلية مرتبطة بسؤال. بدءاً من الجذر، نحن نذهب إلى اليسار أو اليمين للعقدة الحالية، بحسب الجواب نعم أم لا. مثل هذه الأشجار الثنائية، حيث أن كل عقدة خارجية v في مثل هذه الشجرة تمثل قراراً بما يجب فعله. يبين الشكل 5-10 شجرة قرار تعطي توصيات للمستثمرين المحتملين:

الشكل 5-15- شجرة القرار.

A Linked Structure for Binary Trees



الشكل 5-17- البنية المترابطة لشجرة ثنائية،

البنية المترابطة للأشجار الثنائية 1

فإن الطريقة الطبيعية لتحقيق شجرة ثنائية T هي أن نستخدم بنية مترابطة linked structure، حيث نمثل كل عقدة v من T من خلال غرض (الشكل 5-17-a) يحوي حقولاً تتضمن ثلاث مراجع إلى الأب والابن اليسار والابن اليمين والقيمة المخزن في v .

إذا كانت v هي جذر T ، فإن حقل أب v هي المؤشر الصفري null. وإذا كانت v لا تملك أي ابن يسار Left child، فإن الحقل اليسار لـ v هو null، وبالمثل بالنسبة لابن اليمين. كما أننا قد نخزن أيضاً عدد العقد في T في متحول يدعى Size.

سندرس طرائق التحديث التالية:

- `addRoot(e)`: تنشيء وتعيد عقدة جديدة r تخزن العنصر e وتجعل r هي جذر الشجرة. يحدث خطأ إذا كانت الشجرة غير فارغة.
 - `insertLeft(v,e)`: تنشيء وتعيد عقدة جديدة w تخزن العنصر e ، وتضيف w كابن يسار لـ v وتعيد w . يحدث خطأ إذا كانت العقدة v تملك إبناً يساراً.
 - `insertRight(v,e)`: مثل سابقتهما، ولكن مع الابن اليمين.
 - `remove(v)`: تقوم بحذف العقدة v ، واستبدالها بابنها (إن وجد)، وتعيد العنصر المخزن في v . يحدث خطأ إذا كانت v تملك إبنيين.
 - `attach(v,T1,T2)`: تصل الشجرتين T_1 و T_2 على التوالي، كشجرتين فرعيتين يسارية ويمينية للعقدة الخارجية v . يحدث خطأ إذا كانت v ليست عقدة خارجية.
- يحتوي الصنف `LinkedBinaryTree` تابعاً بانياً بدون وسطاء يعيد شجرة ثنائية فارغة. وانطلاقاً من هذه الشجرة الثنائية الفارغة، يمكن أن نبني أي شجرة ثنائية من خلال إنشاء العقدة الأولى باستخدام الطريقة `addRoot` وتكرار استدعاء الطريقتين `insertLeft` و `insertRight` و/أو الطريقة `attach`.
- عندما يتم تمرير غرض v من النوع `position` كوسيط لواحدة من طرائق هذا الصنف، يتم التحقق من صلاحيته من خلال استدعاء الطريقة الإضافية `checkPosition(v)`.
- تتم زيارة قائمة العقد بالتجوال السابق للترتيب من خلال الطريقة العودية `preorderPositions`،

سندرس طرائق التحديث التالية:

- `addRoot(e)`: تنشيء وتعيد عقدة جديدة r تخزن العنصر e وتجعل r هي جذر الشجرة. يحدث خطأ إذا كانت الشجرة غير فارغة.
 - `insertLeft(v,e)`: تنشيء وتعيد عقدة جديدة w تخزن العنصر e ، وتضيف w كابن يسار لـ v وتعيد w . يحدث خطأ إذا كانت العقدة v تملك إبناً يساراً.
 - `insertRight(v,e)`: مثل سابقتها، ولكن مع الابن اليمين.
 - `remove(v)`: تقوم بحذف العقدة v ، واستبدالها بابنها (إن وجد)، وتعيد العنصر المخزن في v . يحدث خطأ إذا كانت v تملك إبنيين.
 - `attach(v,T1,T2)`: تصل الشجرتين T_1 و T_2 على التوالي، كشجرتين فرعيتين يسارية ويمينية للعقدة الخارجية v . يحدث خطأ إذا كانت v ليست عقدة خارجية.
- يحتوي الصنف `LinkedBinaryTree` تابعاً بانياً بدون وسطاء يعيد شجرة ثنائية فارغة. وانطلاقاً من هذه الشجرة الثنائية الفارغة، يمكن أن نبني أي شجرة ثنائية من خلال إنشاء العقدة الأولى باستخدام الطريقة `addRoot` وتكرار استدعاء الطريقتين `insertLeft` و `insertRight` و/أو الطريقة `attach`.
- عندما يتم تمرير غرض v من النوع `position` كوسيط لواحدة من طرائق هذا الصنف، يتم التحقق من صلاحيته من خلال استدعاء الطريقة الإضافية `checkPosition(v)`.
- تتم زيارة قائمة العقد بالتجوال السابق للترتيب من خلال الطريقة العودية `preorderPositions`،

Performance of the Linked Binary Tree Implementation



أداء الشجرة الثنائية المترابطة

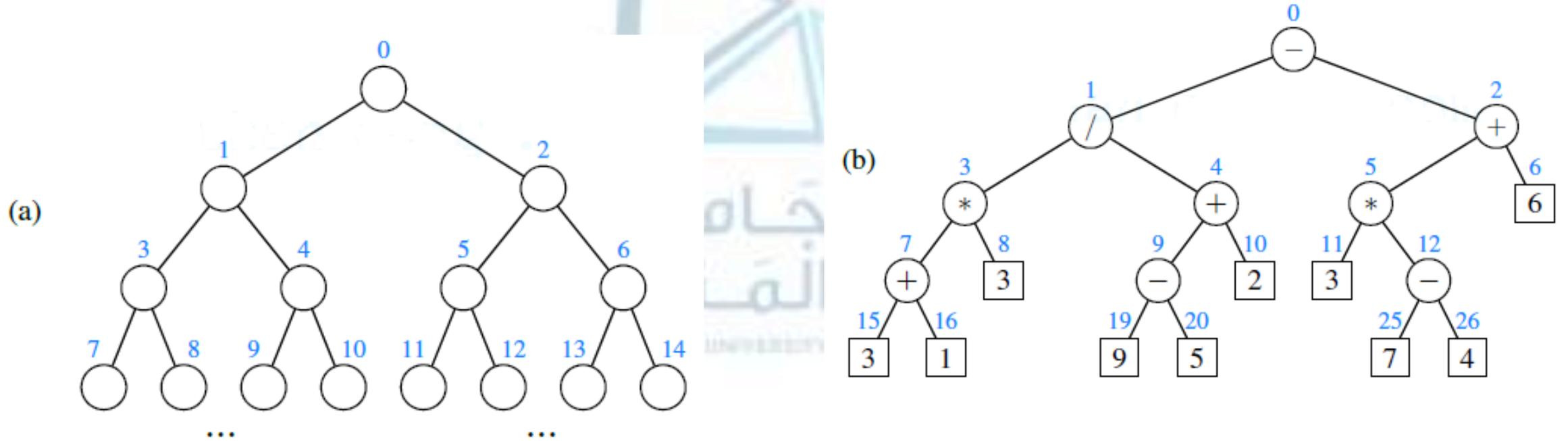
يبين الجدول التالي تلخيصاً لأزمنة التنفيذ لطرائق الصنف `LinkedBinaryTree`:

الطريقة	زمن التنفيذ
size ، isEmpty	$O(1)$
iterator ، position	$O(n)$
replace.	$O(1)$
root ، parent ، children ، left ، right ، sibling	$O(1)$
hasLeft ، hasRight ، isInternal ، isExternal ، isRoot	$O(1)$
insertLeft ، insertRight ، attach ، remove	$O(1)$

Performance of the Linked Binary Tree Implementation

أداء الشجرة الثنائية المترابطة

إن تابع التقييم p يعرف باسم ترقيم المستويات $level\ numbering$ للعقد في شجرة ثنائية T ، فهو يرقم العقد في كل مستوى بترتيب تصاعدي من اليسار إلى اليمين، على الرغم من أنه يتجاوز بعض الأرقام. يبين الشكل 5-14 ترقيم المستويات:



الشكل 5-18- ترقيم المستويات في شجرة ثنائية (a : الأسلوب العام، b : مثال).

كما في حال الأشجار العامة، غالباً ما تتطلب أي عمليات معالجة لعناصر الأشجار الثنائية تجولاً عبر هذه الأشجار. التجول السابق للترتيب لشجرة ثنائية: بإمكاننا تبسيط الخوارزمية في حال التجول على شجرة ثنائية لتصبح كما يلي:

Algorithm binaryPreorder(T,v):

perform the “visit” action for node v

if v has a left child u in T **then** binaryPreorder(T,u) {recursively traverse left subtree}

if v has a right child w in T **then** binaryPreorder(T,w) {recursively traverse right subtree}

خوارزمية binaryPreorder

Algorithm Preorder (Node* r) // **root -> left->right** : ويمكن كتابته Preorder بطريقه أقرب للكود من خلال :

```
{ if (r == NULL) return;
```

```
print r->data ;
```

```
Preorder (r->left);
```

```
Preorder (r->right);
```

```
}
```

Traversals of Binary Trees



التجول عبر الأشجار الثنائية 1

التجول اللاحق للترتيب لشجرة ثنائية:

Algorithm binaryPostorder(T,v):

```
if v has a left child u in T then    binaryPostorder(T,u) {recursively traverse left subtree}
if v has a right child w in T then   binaryPostorder(T,w) {recursively traverse right subtree}
perform the "visit" action for node v
```

خوارزمية Postorder

ويمكن كتابتها بطريقه أقرب للكود من خلال :

Algorithm Postorder(Node* r) // left--> right->root

```
{ if (r == NULL) return;
```

```
    Postorder(r->left);
```

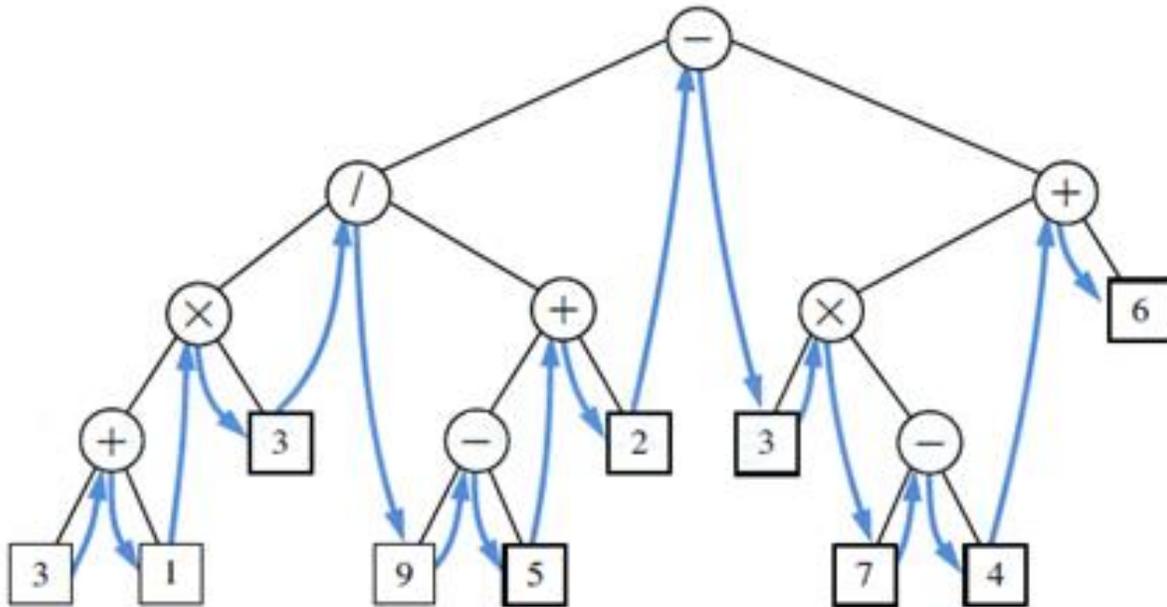
```
    Postorder(r->right);
```

```
    print r->data ; }
```

Inorder Traversal of a Binary Tree

التجول المرتب عبر شجرة ثنائية

طريقة إضافية للتجول عبر شجرة ثنائية، هي التجول المرتب inorder traversal. في هذا التجول، نقوم بزيارة عقدة بين التجولات العودية من أشجارها اليسارية واليمنية. يبين المقطع التالي خوارزمية هذا النوع من التجول:



الشكل 5-19- التجول المرتب عبر شجرة ثنائية LPR.

```
Algorithm Inorder(Node* r) // left->root -> right
```

```
{ if (r == NULL) return;
```

```
  Inorder(r->left);
```

```
  cout << r->data << "\t";
```

```
  Inorder(r->right); }
```

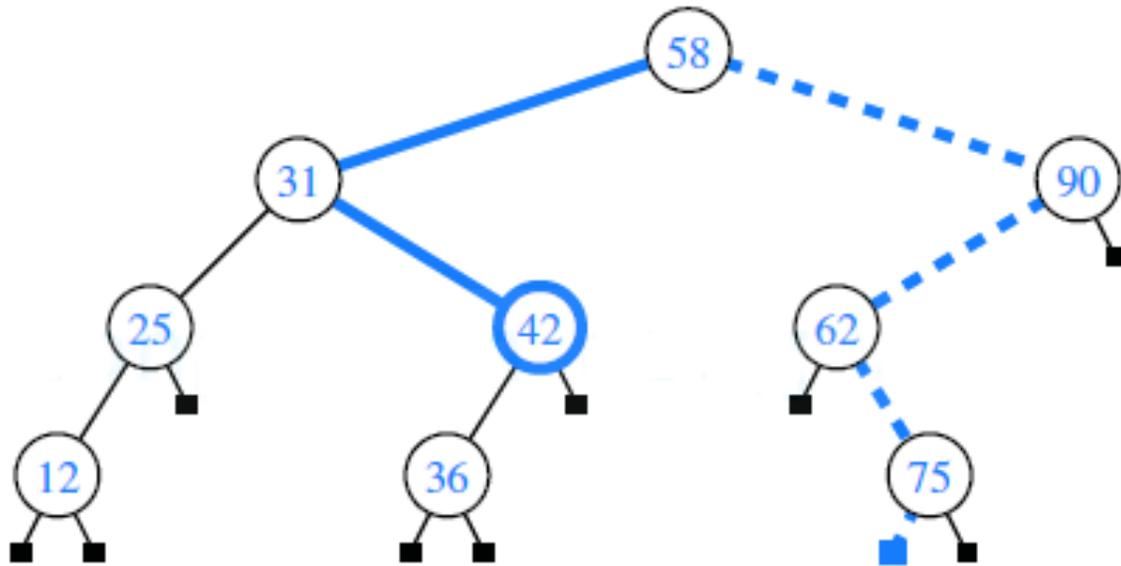
```
}
```

خوارزمية inorder

Binary Search Trees

أشجار البحث الثنائية 1

القيمة في كل عقدة أكبر من القيم الموجودة في شجرتها الفرعية اليسرى (إن وجدت) وأقل من القيم الموجودة في شجرتها الفرعية اليمنى (إن وجدت). الشجرة الثنائية التي تتمتع بهذه الخاصية تدعى شجرة البحث الثنائية (BST) binary search tree



الشكل 5-20- شجرة بحث ثنائية.

لتكن S هي عبارة عن مجموعة عناصرها تملك علاقة ترتيب. مثلاً يمكن لـ S أن تكون مجموعة من الأعداد الصحيحة. عندها فإن شجرة البحث الثنائية binary search tree هي عبارة عن شجرة ثنائية تحوي:

- كل عقدة داخلية v من T تخزن عنصراً من عناصر S ويرمز له $x(v)$.
 - من أجل كل عقدة داخلية v من T ، فإن العناصر المخزنة في الشجرة اليسارية لـ v هي أقل من أو تساوي $x(v)$ والعناصر المخزنة في الشجرة اليمينية لـ v هي أكبر $x(v)$.
 - لا تخزن العقد الخارجية لـ T أي عنصر.
- يقوم التجول المرتب على العقد الداخلية في شجرة البحث الثنائية T بزيارة العناصر بترتيب غير تنازلي (كما يبين الشكل المرفق):

يمكن استخدام شجرة بحث ثنائي T من أجل مجموعة S من أجل إيجاد فيما إذا كانت قيمة بحث ما y موجودة ضمن S أم لا، وذلك من خلال التجول عبر مسار هابط من الشجرة بدءاً من الجذر (الشكل 5-17 الذي يظهر مسار البحث عن القيمة 36 بخط غامق (عملية ناجحة) ومسار البحث عن لقيمة 70 بخط منقط (نتيجة سلبية)).

تم عملية البحث كمايلي: عند كل عقدة داخلية تتم مصادفتها، نقارن قيمة المبحوث عنه y مع العنصر $x(v)$ المخزن في v ، فإن كانت القيمتان متساويتان ينتهي البحث، أما إن كانت قيمة البحث أصغر من $x(v)$ عندها يستمر البحث في الشجرة الفرعية اليسارية وإلا يستمر في الشجرة الفرعية اليمينية. فإذا وصلنا إلى عقدة خارجية بدون أن نجد القيمة التي نبحث عنها، تكون القيمة ليست ضمن الشجرة.

هناك العديد من أشكال التجول الأخرى عبر الأشجار الثنائية إلا أننا نكتفي بالمقدار الذي تم عرضه في هذا البحث، ونترك للمهتمين التعرف على الأشكال الأخرى.

انتهت محاضرة الأسبوع 8

توجد طريقتان للتجول:

- التجول بالعمق (Depth-first traversal (DFT)

- التجول بالعرض: (Breadth-first traversal (BFT)

خوارزميات إنجاز عمليات التجول بالعمق على شجرة من خلال الوصول إليها من خلال طرائق الشجرة كنمط بيانات مجرد.

نعرف العمق والارتفاع Depth and Height: لتكن v عقدة في الشجرة T ، عمق $depth$ العقدة v هو عدد الآباء لـ v ، على سبيل المثال، في الشجرة المبينة في الشكل 2-5، عمق العقدة التي تخزن International هو 2، لاحظ أن هذا التعريف يعني أن عمق

جذر الشجرة T هو 0 وعمق الشجرة الفارغة هو 1.

يمكن تعريف عمق عقدة v عودياً كما يلي:

- إذا كانت v هي الجذر، عندئذ فإن العمق هو 0.

- وإلا، عمق v هو عمق العقدة الأب لـ v مضافاً إليه واحداً.

بناء على هذا التعريف، تمثل الخوارزمية العودية التالية حساب عمق عقدة v في شجرة T .

Algorithm depth(T,v)

if v is the root of T then return 0 else return $1+depth(T,w)$, where w is the parent of v in T

المقطع 2-5- خوارزمية حساب العمق.

إن زمن التنفيذ للخوارزمية $depth(T,v)$ هو $O(d_v)$ حيث تمثل d_v عمق العقدة v في الشجرة T . وذلك لأن الخوارزمية تنفذ خطوة عودية ذات زمن ثابت من أجل كل أب l لـ v . وبالتالي، فإن الخوارزمية $depth(T,v)$ تنفذ في زمن $O(n)$ في الحالة الأسوأ، حيث n هو العدد الكلي للعقد في T ، وذلك بما أن عقدة ما في الشجرة T يمكن أن تكون ذات عمق $n-1$ في الحالة الأسوأ.

عرف ارتفاع $height$ عقدة v في شجرة T هو عدد الحدود الأعظمي للوصول إلى العقدة الخارجية، وبشكل عودي على النحو:

- إذا كانت v عقدة خارجية، عندئذ فإن ارتفاعها هو 0.
 - وإلا، إرتفاع v هو الارتفاع الأعظمي لابن العقدة v مضافاً إليه واحداً.
- إن ارتفاع شجرة غير فارغة T هو ارتفاع جذر الشجرة، على سبيل المثال، الشجرة في الشكل 2-5 ارتفاعها هو 4. يمكن أيضاً النظر إلى الارتفاع $height$ لشجرة غير فارغة T على أنه العمق $depth$ الأعظمي لعقدة خارجية من T . نبين فيما يلي خوارزمية حساب ارتفاع شجرة.

Algorithm height1(T)

$h \leftarrow 0$

for each head v in T do if v is an external node in T then $h \leftarrow \max(h, depth(T,v))$

return h

المقطع 4-5- خوارزمية حساب الارتفاع.

توجد لدينا ثلاث طرق قياسية:

- التجول السابق للترتيب (Node, Left, Right NLR) Pre-order Traversal .
- التجول اللاحق للترتيب (Left, Right, Node LRN) Post-order Traversal .
- التجول بالترتيب (Left, Node, Right LNR) In-order Traversal .

نبدأ بالتجول السابق للترتيب preorder traversal لشجرة T ، يتم زيارة جذر الشجرة T أولاً ومن ثم يتم التجول عودياً عبر الأشجار الفرعية والتي جذورها هي أبناء جزر الشجرة T . إذا كانت الشجرة مرتبة ordered عندئذ فإن الأشجار الفرعية يتم التجول عليها وفقاً لترتيب الأبناء. إن الفعل المحدد المرافق لعملية زيارة عقدة ما v يتوقف على التطبيق الخاص بالتجول، وقد يتضمن أي شيء بدءاً من مزايدة عداد وانتهاء بالقيام ببعض العمليات الحسابية المعقدة على v . خوارزمية التجول السابق للترتيب .

Algorithm preorder(T,v):

perform the “visit” action for node v

for each child w of v in T **do**

preorder(T,w) {recursively traverse the subtree rooted at w }

المقطع 5-8- خوارزمية التجول preorder.

خوارزمية هامة أخرى للتجول عبر الشجرة هي خوارزمية التجول اللاحق للترتيب postorder traversal. يمكن النظر إلى هذه الخوارزمية على أنها معاكسة لخوارزمية التجول السابق للترتيب، وذلك لأنها تتجول عودياً على الأشجار الفرعية التي جذورها أبناء لجذر الشجرة أولاً ومن ثم تزور الجذر.

إنها شبيهة بخوارزمية التجول السابق للترتيب في أننا نستخدمها لحل مسائل خاصة من خلال تخصيص الفعل المرافق لعملية الزيارة للعقدة v .

وأيضاً، وكما في حالة التجول السابق للترتيب، إذا كانت الشجرة مرتبة، فإننا نجري استدعاءات عودية لأبناء العقدة v بحسب ترتيبها المحدد.

المقطع التالي يوضح شبه الشيفرة لخوارزمية التجول اللاحق للترتيب:

Algorithm postorder(T,v):

for each child w of v in T **do**

 postorder(T,w) {recursively traverse the subtree rooted at w }

perform the “visit” action for node v

المقطع 5-11- خوارزمية التجول postorder.

Properties of Binary Trees

- $h+1 \leq n \leq 2^{h+1} - 1$
- $1 \leq n_E \leq 2^h$
- $h \leq n_I \leq 2^h - 1$
- $\log(n+1) - 1 \leq h \leq n-1$

وإذا كانت الشجرة T تامة proper فإننا نضيف الخصائص التالية:

- $2h+1 \leq n \leq 2^{h+1} - 1$
- $h+1 \leq n_E \leq 2^h$
- $h \leq n_I \leq 2^h - 1$
- $\log(n+1) - 1 \leq h \leq (n-1)/2$

$$n_E = n_I + 1$$

وأخيراً، إذا كانت الشجرة الثنائية تامة وغير فارغة فإن:

حيث إن:

- T شجرة ثنائية غير فارغة.
- n عدد العقد.
- n_E عدد العقد الخارجية.
- n_I عدد العقد الداخلية.
- h ارتفاع الشجرة.

