خوارزميات المخططات البيانية Graph Algorithms

تعريف 4-1: **المخطط البياني المُوَجّه** Directed Graph

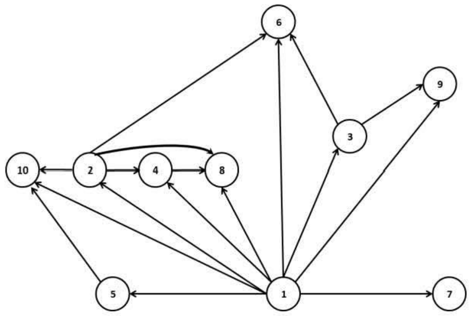
المخطط البياني (أو الرسم البياني) المُوَجَّه (directed graph, or **digraph)** هو زوج $G = (V, E)$ (a pair)، حيث **V** هى مجموعة عناصرها تدعى "رؤوساً" (vertices) و **E** هى مجموعة من **أزواج مرتبة (ordered pairs)** من عناصر **V**. والرؤوس كثيرا ما تدعى أيضاً "عُقَداً" (nodes). وعناصر المجموعة **E** يطلق عليها "أحرف" (edges) أو "أحرف مُوَجَّهَة" (directed edges) ، أو "أقواس" (arcs). وبالنسبة إلى الحرف المُوجّه (v, w) فى المجموعة **E** فإن **v** تُعَدُّ "ذيله" (its tail) و **w** تُعَدُّ "رأسه" (its head). والزوج (v, w) يُمثَّل فى الرسوم البيانية (diagrams) بالسهم $v \rightarrow w$ (the arrow)، ونحن هنا سنكتبه ببساطة **vw**.

مثال 4-1: نفرض أن **R** هى العلاقة الثنائية (binary relation) المعرفة على المجموعة $S = \{1, 2, \ldots, 10\}$ والمكونة من الأزواج المرتبة (x, y) التى تحقق الشرط أن **x** هى عامل حقيقى / فعلى من عوامل **y** (a proper factor of)، أى أن $(x \neq y)$، وباقى (y / x) (remainder of) يساوى صفرا 0. وعادةً نكتب $x R y$ كاصطلاح بديل للانتماء $(x, y) \in R$.

فى المخطط البياني المُوجّه (digraph) التالى (شكل 4—1): النقاط

هى عناصر المجموعة S، وهناك سهم (arrow) من x إلى y إذا وفقط إذا كانت x R y (if and only if). لاحظ أن العلاقة R متُعَدية (transitive)؛ أى أنه اذا كانت x R y & y R z متحققتين فإن x R z تكون أيضا متحققة. ونرمز لهذا المخطط البياني المُوجّه بالزوج $G = (V, E)$، حيث

$$V = \{1, 2, \dots, 10\},$$
$$E = \{(1,2), \dots, (1,10), (2,4), (2,6), (2,8), (2,10), (3,6), (3,9), (4,8), (5,10)\}$$



شكل 4–1

مخطط بياني موجه يمثل العلاقة الثنائية R: " x عامل فعلى من عوامل y"

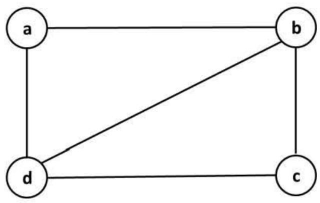
The relation R representing "x is a proper factor of y"

تعريف 4-2: **المخطط البياني غير المُوجّه** Undirected graph

المخطط البياني غير الموجه هو زوج $G = (V, E)$، حيث V هى مجموعة عناصرها تُدعى "رؤوسا"، و E هى مجموعة من **ازواج غير مرتبة (unordered pairs)** من عناصر متمايزة (distinct elements) من عناصر V. والرؤوس كثيرا ما تدعى أيضاً "عقدا" (nodes). وعناصر E يطلق عليها "أحرف" **أو أحرف غير مُوَجّهة (undirected edges)** للتأكيد. وأى حرف يمكن اعتباره

مجموعة جزئية من V (subset of) تحتوى على عنصرين، وبالتالى فإن $\{v, w\}$ يرمز إلى حرف غير موجه. وفى الرسوم البيانية فإن هذا الحرف سيكون هو الخط $v - w$ (line)، ونحن هنا سنكتبه ببساطة vw. وبالطبع فبالنسبة للمخططات البيانيه غير المُوجَّهة فإن $vw = wv$.

مثال 4-2: فى المخطط البياني غير الموجه فى الشكل التالى شكل (2–4):



شكل 4–2

مخطط بياني غير موجه

$V = \{a, b, c, d\}$

$E = \{(a, b), (a, d), (b, c), (b, d), (c, d)\}$

اصطلاحات Notations

- عادةً سنستخدم الاصطلاح (i, j) ليشير إلى الحرف الواصل بين الرأسين i, j بدلا من الاصطلاح $\{i, j\}$.
- سنرمز عادةً لرؤوس المخططات البيانية بالأعداد الطبيعية (natural 1, 2, 3, . . . numbers).
- سنستخدم الحرف n ليشير إلى عدد رؤوس المخطط البياني $n = |V|$، والحرف m ليشير إلى عدد أحرفه $m = |E|$. فمثلا فى المخطط البياني فى شكل 4-2:

عدد الرؤوس: $n = |V| = 4$

عدد الأحرف: $m = |E| = 5$

تعريف 4–3–أ) المخطط البياني الجزئي Subgraph

يُقال إن المخطط البياني $G' = (V', E')$ هو مخطط بياني جزئى (subgraph) من المخطط البياني $G = (V, E)$ إذا كان $V' \subseteq V \;\; \& \; E' \subseteq E$. ومن تعريف المخطط البياني فيجب أيضاً أن يكون $E' \subseteq V' \times V'$.

ب) المخطط البياني المُوجّه المتماثل Symmetric digraph

هو مخطط بياني موجه (directed graph) بحيث أنه لكل حرف **vw** يوُجدَ ايضاً الحرف المعاكس **wv** (the reverse edge). وأى مخطط بياني غير موجه (undirected graph) له مخطط بياني موجه متماثل مقابل (corresponding symmetric digraph) وذلك بتفسير (interpreting) أى حرف غير موجه (undirected edge) باعتباره زوجاً من حرفين موجهين (a pair of directed edges) فى إتجاهين متعاكسين (in opposite directions).

جـ) المخطط البياني الكامل Complete graph

هو مخطط بياني (عادةً غير موجه) فيه حرف (an edge) بين كل زوج من الرؤوس (between each pair of vertices).

د) يُقال إن الحرف VW يحدث عند / يقع على (incident upon) الرأسين **v,w**، وكذلك العكس صحيح، أي يقال إن الرأسين **v, w** يحدثان عند/ يقعان على الحرف **vw**.

تعريف 4-4 **علاقة التجاور** "A" Adjacency relation

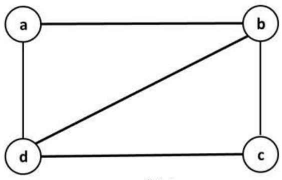
نفرض أن $G = (V, E)$ مخطط بياني (موجه أو غير موجه)، وأن **v, w** عنصران في V. يُقال إن **"w مجاورة لـ v"** (**"w is adjacent to v"**)، وتكتب **vAw**، إذا وفقط إذا كان $vw \in E$ (if and only if **vw** is in E). وبأسلوب آخر فإن **vAw** تعني أن **w** يمكن الوصول اليها من **v** بالتحرك (moving) عبر حرف واحد (along one edge) من حروف G. وإذا كان المخطط **G** غير موجه فإن العلاقة **A** تكون متماثلة (symmetric). أي أن **wAv** إذا وفقط إذا كان **vAw** (if and only if).

تعريف 4-5: **المسار في مخطط بياني** Path in a graph

المسار من **v** إلى **w** في مخطط بياني $G = (V, E)$ هو متتابعة من الأحرف $v_0v_1, v_1v_2, \dots, v_{k-1}v_k$ (a sequence of edges) بحيث أن $v = v_0$ & $v_k = w$.

وطول المسار (length of the path) هو عدد الأحرف في المسار أي هو k. وأي رأس **v** بمفرده (alone) تعتبر مساراً طوله صفر من **v** إلى نفسها. و"**المسار البسيط**" (**a simple path**) هو مسار بحيث أن الرؤوس $v_0, v_1, \dots, v_k$ جميعها متمايزة (are all distinct). **ويُقال إن رأسا w يمكن الوصول إليها من الرأس v (reachable from)** إذا كان هناك مسار من **v** إلى W.

مثال 4-3: شكل 4-3 يبيّن المسار (a, b), (b, d), (d, c). وعادةً نرمز لأى مسار بذكر متتابعة الرؤوس (sequence of vertices) التى يمر بها (through which it passes). ولكن تذكر أن طول أى مسار هو عدد الأحرف التى اجتازها (number of edges traversed). فالمسار فى شكل 4- 3 هو a, b, d, c وطوله ثلاثة.



شكل 4-3

مسار من a إلى c

تعريف 4-6: **المخطط البياني المُتّصل** Connected graph

أ) يُقال إن المخطط البياني غير المُوجَّه (undirected graph) "**مُتّصل**" **(connected)** إذا وفقط إذا (if and only if) وُجِدَ لكل زوج **v, w** من الرؤوس مسار (a path) من **v** إلى **w**.

ب) يُقال إن المخطط البياني المُوجّه (directed graph) **قَويُّ الاتصال / مُتّصل بقوة (strongly connected)** إذا وفقط إذا (if and only if) وُجِدَ لكل زوج **v, w** من الرؤوس مسار (a path) من **v** إلى **w**.

ملاحظة: السبب فى ذكر تعريفين منفصلين مع أن الشرط المذكور هو نفسه، هو أنه فى حالة المخطط البياني غير المُوجّه: إذا كان هناك مسار من **v** إلى **w**، فإنه تلقائياً (automatically) يوُجدَ مسار من **w** إلى **v** ، بينما فى حالة المخطط البياني المُوجّه قد لا يكون هذا صحيحا، ولذلك فقد استُخدم القيد "بقوة "strongly (qualifier) ليشير إلى أن الشرط (condition) أقوى (stronger). وإذا نظرنا إلى المخطط غير المُوجّه على أنه نظام من شوارع ثنائية الاتجاه (a system of two- way streets)، والمخطط المُوجّه على أنه نظام من شوارع أحادية الاتجاه (one-way streets)، فشرط "قوة الاتصال" / "الاتصال القَوِي" (strong connectivity) يعنى أنه يمكننا أن نصل من أى موضع إلى أى موضع (from anywhere to anywhere) بالسير فى الشوارع أحادية الاتجاه فى اتجاهاتها الصحيحة. ومن الواضح ان هذا شرط أكثر صرامة / تشددا / تضييقاً (more stringent condition) مما لو كانت الشوارع ثنائية الاتجاه.

تعريف 4-7: الدورة فى مخطط بياني Cycle in a graph

أ) فى المخطط البياني المُوجّه، **الدورة (cycle)** هى مجرد مسار غير خالى (just a nonempty path) بحيث أن الرأس الأولى والرأس الأخيرة (first and last vertices) متطابقتان (identical). **والدورة البسيطة (simple cycle)** هى دورة لا تتكرر فيها أى رأس (no vertex is repeated)، باستثناء أن الرأسين الأولى والأخيرة متطابقتان.

ب) فى المخطط البياني غير المُوجّه، تعريفا الدورة والدورة البسيطة كما هما فى المخطط المُوجّه ولكن مع إضافة الشرط: إذا ظهر أى حرف (edge) أكثر من مرة (appears more than once) فيجب أن يظهر دائما فى التوجيه نفسه (with the same orientation)، بمعنى أنه إذا كانت

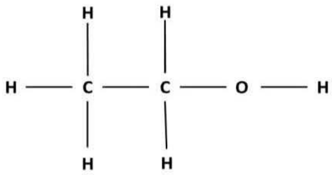
$$v_i = x \;\&\; v_{i+1} = y \qquad \text{for } 0 \le i < k$$

فلا يمكن أن توجد j بحيث أن $v_j = y \;\&\; v_{j+1} = x$.

جـ) يُقال إن المخطط البياني "**لادَوْرَوِيّ**" (**acyclic**) إذا لم يحتو على أى دورة.

د) المخطط البياني غير المُوجّه اللادوروي (undirected acyclic graph) يطلق عليه "**غابة غير موجهه**" (**undirected forest**). وإذا كان المخطط البياني مُتّصلاً (connected) أيضاً، فيطلق عليه **شجرة حرة** (a **free tree**) أو "**شجرة غير موجهة**" (undirected tree). [انظر مثلا شكل 4–4].

هـ) غالبًا ما يُختصر اسم المخطط البياني المُوجّه اللادوروي (directed acyclic graph) إلى "**DAG**" لولا يُفترض أن يحقق المخطط DAGأى شرط اتصالية (any connectivity condition)].



شكل 4-4

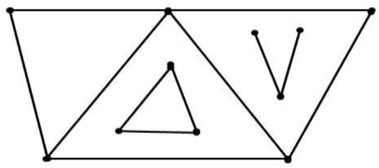
شجره حرة / شجرة غير موجهة (تمثل جزىء الكحول)

تعريف 4-8: **المُرَكِّبَة المتصلة** Connected component

المُرَكّبَة المتصلة فى مخطط بياني غير موجه G هى مخطط بياني متصل "أعظَمي" جزئي من G (a maximal connected subgraph of). ويقال لمخطط بياني إنه "أعظمى" ضمن مجموعة ما من مخططات بيانية إن لم يكن مخططا بيانيا جزئيا فعليا (a proper subgraph) من أى مخطط بياني فى هذه المجموعة.

وأى مخطط بياني غير موجه إن لم يكن متصلا، فإنه يمكن تجزئته

(partitioned) إلى مُرَكّبَات متصلة منفصلة، (separate connected components)، وهذه التجزئة وحيدة (unique). والشكل التالى (شكل 4-5) يبين مخططا بيانيا له ثلاث مركبات متصلة.



شكل 4-5

مخطط بياني له ثلاث مُرَكّبَات متصلة

تعريف 4-9: **المخطط البياني المُوَزَّن** Weighted graph

المخطط البياني المُوَزَّن هو ثلاثى (V, E, W) (a triple) حيث (V, E) مخطط بياني (a graph) (موجه أو غير موجه)، وW دالة من E إلى R حيث R هى مجموعة الأعداد الحقيقية (the reals). وبالنسبة لأى حرف e (edge) فإن $W(e)$ يطلق عليها "**وَزْن** e" (**weight** of e).

ملاحظة: عادةً يشير وَزْن أى حرف e إلى خاصية من خصائص (properties) الحرف تبعا للتطبيق (application) المستخدم فيه هذا المخطط البياني. فمثلا إذا كان الحرف e يمثل طريقاً أو شارعاً أو مسافة بين بلدين أو نقطتين، فوَزْنه $W(e)$ قد يشير إلى تكاليف اجتياز هذا الطريق أو سرعة اجتيازه أو طوله أو زمن قطع هذه المسافة. ولذلك قد نستخدم أياً من المصطلحات (terms) التالية: وَزْن (weight)، وتكاليف (cost)، وطول (length) وسعة (capacity) هذا الحرف.

طرق تمثيل المخططات البيانية Graph Representations

رأينا طريقتين لتمثيل أى مخطط بياني على الورق: الأولى برسم صورة (drawing a picture) تمثل فيها الرؤوس بنقاط (points) والأحرف بخطوط (lines) أو أسهم (arrows)، والثانية بذكر عناصر مجموعتى الرؤوس والأحرف (by listing the vertices and edges). وفيما يلى ندرس بإذن الله بنى المعطيات /هياكل البيانات (data structures) التى تفيدنا فى تمثيل المخططات البيانية فى برامج الحاسوب. نفرض أن $G = (V, E)$ مخطط بياني حيث

$n = |V|$ ، $m = |E|$ ، $V = \{v_1, v_2, \dots, v_n\}$.

(أ) التمثيل بمصفوفة التجاور

Adjacency (Incidence) Matrix Representation

يمكننا تمثيل المخطط G بمصفوفة n x n مربعة $A = (a_{ij})$ تدعى "مصفوفة التجاور" للمخطط G (Adjacency matrix for). وتعرف المصفوفة A هكذا:

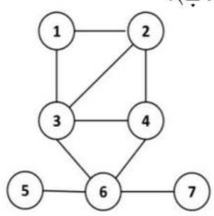
$$a_{ij} = \begin{cases} 1 & \text{if } v_i v_j \in E \quad \text{إذا كان} \\ 0 & \text{otherwise} \quad \text{ما عدا ذلك} \end{cases} \qquad \text{for } 1 \le i, j \le n$$

وبالنسبة للمخطط البياني غير المُوجّه فإن مصفوفة التجاور تكون متماثلة (symmetric) [ولذلك فإننا نحتاج لتخزين (storing) نصفها فقط]. وإذا كان $G = (V, E, W)$ مخططا مُوَزّنا فإنه يمكن تخزين الأوزان فى مصفوفه التجاور بتعديل تعريفها كما يلي:

$$a_{ij} = \begin{cases} W(v_i v_j) & \text{if } v_i v_j \in E \quad \text{اذا كان} \\ c & \text{otherwise} \quad \text{ما عدا ذلك} \end{cases} \qquad \text{for } 1 \le i, j \le n$$

حيث c ثابت تعتمد قيمته على تفسير الأوزان والمسألة المطلوب حلها. وإذا نظرنا إلى الأوزان على أنها تكاليف (costs)، فقد نختار (∞) (أو أى عدد كبير جداً) كقيمة الثابت c لأن تكلفة اجتياز حرف غير موجود

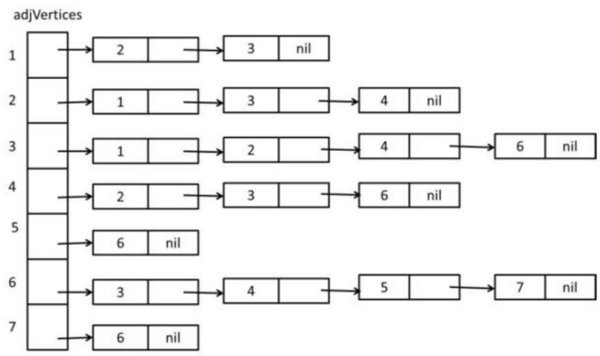
(cost of traversing a nonexistent edge) تكون عالية جدا. وإذا كانت الأوزان سعات (capacities)، فإن اختيار c = 0 يكون مناسباً لأنه لا يمكن لأى شىء أن يتحرك عبر حرف غير موجود. انظر مثلا شكلي 4–6 (أ، ب)، و 4–7(أ، ب).



(أ) مخطط بيانى غير موجه
An undirected graph

$$\begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

(ب) مصفوفة التجاور للمخطط
its Adjacency (incidence) matrix



its adjacency- list structure (جـ) بنية قائمة التجاور للمخطط

شكل 4–6

تمثيلان لمخطط بياني غير موجَّه بدون أوزان للحروف

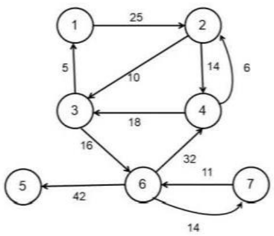
(مصفوفة التجاور، و منظومة قوائم التجاور)

ويمكن أيضًا أن يكون مخططًا بيانيًّا مُوجّها متماثلاً

$$\begin{pmatrix} 0 & 25.0 & \infty & \infty & \infty & \infty & \infty \\ \infty & 0 & 10.0 & 14.0 & \infty & \infty & \infty \\ 5.0 & \infty & 0 & \infty & \infty & 16.0 & \infty \\ \infty & 6.0 & 18.0 & 0 & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & 0 & \infty & \infty \\ \infty & \infty & \infty & 32.0 & 42.0 & 0 & 14.0 \\ \infty & \infty & \infty & \infty & \infty & 11.0 & 0 \end{pmatrix}$$

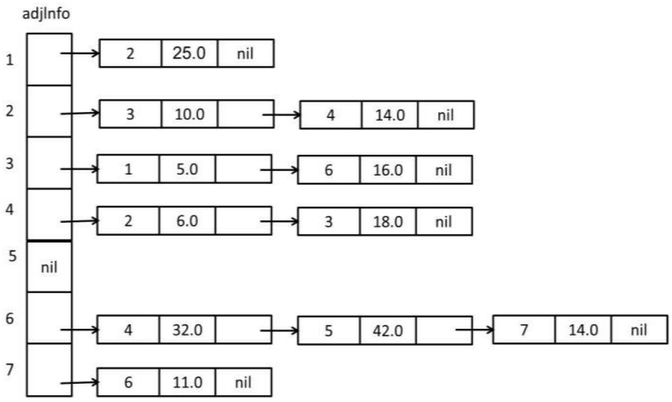
(ب) مصفوفة التجاور للمخطط

Its adjacency matrix



(أ) مخطط بياني مُوجَّه مُوَزَّن

A weighted digraph



(جـ) بنية قائمة التجاور للمخطط its adjacency- list structure

شكل 4–7

تمثيلان لمخطط بياني مُوجّه مُوَزَّن

وتتطلب خوارزميات حل بعض مسائل المخططات البيانية فحص (examining) وتشغيل (processing) كل حرف (edge) بطريقةٍ ما على الأقل مرة واحدة. وإذا استخدمنا تمثيل المخطط البياني بمصفوفة التجاور

فيمكننا أيضاً أن ننظر إلى المخطط على أنه يشتمل على أحرف بين جميع أزواج الرؤوس المتمايزة (edges between all pairs of distinct vertices)، لأن خوارزميات عديدة تقوم بفحص كل عنصر فى المصفوفة لتحديد أى الأحرف موجودة فعلاً (which edges really exist). ونظرا لأن عدد الأحرف المحتملة هو n^2 فى مخطط بياني موجه، أو $n(n-1)/2$ فى مخطط بياني غير مُوجه، فدرجة تعقيد (complexity) مثل هذه الخوارزميات ستكون فى $\Omega(n^2)$.

(ب) التمثيل بمنظومة قوائم التجاور

Array of Adjacency Lists Representation

يمكننا تمثيل المخطط البياني G بمنظومة (array) مؤشرة برقم الرأس (indexed by vertex number) وتحتوى على قوائم مترابطة (linked lists) تُدعى " قوائم التجاور " (adjacency lists). ولكل رأس v_i (vertex) يحتوى العنصر رقم i فى المنظومة the i-th array (entry) على قائمة (a list) بمعلومات (information) عن جميع أحرف المخطط G التى تغادر v_i (leave). وفى المخطط البياني الموجه (directed graph) هذا يعنى أن v_i هى ذيل (tail) الحرف، بينما فى المخطط غير المُوجّه هذا يعنى أن الحرف يقع على v_i (incident upon). وقائمة v_i (list for) تحتوي على عنصر واحد لكل حرف (one element per edge) . وإذا أطلقنا على هذه المنظومة الاسم adjInfo فيمكننا تعريفها كما يلي:

```
list [ ] adjInfo = new list[n+1];
```

وسنستخدم المؤشرات $1, 2, \ldots, n$ (indexes)، وبالتالى سنحجز (allocate) عدد $n + 1$ من المواضع (positions)، ولا نستخدم الموضع رقم

(the 0-th 0 position). وهكذا سيكون adjInfo[i] قائمة بمعلومات عن الأحرف التى تغادر الرأس v_i.

وتتميز بنية قائمة التجاور (adjacency–list structure) بأن الأحرف التى لا توجد فى المخطط G لا توجد أيضاً فى التمثيل. وإذا كان G هشاً / متناثراً (sparse) [أى أن عدد أحرفه أقل بكثير من n^2]، فإنه يمكن معالجته (processed) بسرعة. ويلاحظ أنه إذا كانت العناصر فى أى قائمة تجاور تظهر بترتيب مختلف (appear in a different order)، فإن البنية (structure) ستظل تمثل المخطط نفسه، ولكن أي خوارزمية تستخدم القائمة ستواجه أو ستقابل (encounter) العناصر بترتيب مختلف وقد تتصرف بطريقةٍ ما مختلفة. فأي خوارزمية يجب ألا تفترض أي ترتيب معين [إلا – بالطبع – إذا كانت الخوارزمية نفسها تنشئ (constructs) القائمة بطريقة خاصة].

ويبين شكل 4–6 مثالاً لبنية المعطيات (data structure) هذه لمخطط غير موجه (وممكن أيضا لمخطط موجه متماثل). بينما يبيّن شكل 4–7 مثالاً لهذه البنية لمخطط موجه مُوَزّن.

ويلاحظ أنه فى أى مخطط غير موجه يُمثَّل كل حرف مرتين، فمثلا إذا كان vw حرفا فى المخطط، فسيوجد عنصر للرأس w على قائمة تجاورv، وعنصر للرأس v على قائمة تجاورw. وهكذا يوجد عدد 2m من عناصر قائمة التجاور (2m adjacency-list elements) وعدد n من قوائم التجاور (n adjacency lists). وبالنسبة لأى مخطط موجه فإن كل حرف – كونه موجها – سيُمثَّل مرة واحدة. ولاحظ أن تمثيلات قوائم التجاور للمخططات غير المُوجَّهه، والمخططات المُوجَّهه المتماثلة المقابلة (corresponding symmetric digraphs) متطابقة (identical).

اجتياز المخططات البيانية

Traversing Graphs

تقوم معظم الخوارزميات المستخدمة لحل مسائل المخططات البيانية بفحص أو تشغيل كل رأس (vertex) وكل حرف (edge). وهناك استراتيجيتان لاجتياز المخططات البيانية تقدم كل منهما طريقة ذات كفاءة عالية لزيارة (visiting) كل رأس وكل حرف مرة واحدة بالضبط (exactly once). وهاتان الاستراتيجيتان هما:

- **البحث بالعرض أولاً (breadth-first search)**، وتُدعَى أيضاً **الاجتياز بالعرض أولاً (breadth-first traversal)**.
- **البحث بالعمق أولاً (depth-frist search)**، وتُدعَى أيضاً **الاجتياز بالعمق أولاً (depth-frist traversal)**.

وبناءً عليه فإن خوارزميات عديده – مبنية على أسس هاتين الاستراتيجيتين – يتم تشغيلها في زمن (run in time) يتزايد خطيًّا (grows linearly) مع حجم مخطط الإدخال (input graph).

أولاً : البحث بالعمق أولاً Depth-First Search (DFS)

نفرض أن $G = (V, E)$ مخطط بياني غير موجَّه (an undirected graph)، وأننا سنقوم بزيارة رؤوسه بالطريقة التالية:

- نختار أولاً رأس بداية **v** (select a starting vertex) و"نزورها".
- ثم نختار أي حرف (v, w) (any edge) واقع على **v** (incident upon) ونزور w .

- وعمومًا إذا فرضنا أن **x** هى أحدث رأس تمت زيارتها (the most recently visited vertex)، فإن البحث (search) يستمر باختيار حرف ما **(x, y)** لم يتم استكشافه بعد (some unexplored edge) واقع على **x**:

(i) فإن كانت y قد تمت زيارتها سابقاً (has been previously visited)، فإننا نبحث عن حرف جديد آخر (another new edge) واقع على x.

(ii) وإن كانت **y** لم تتم زيارتها من قبل، فإننا نزور **y**، ونبدأ البحث من جديد مبتدئين عند الرأس **y**.

- بعد إكمال البحث عبر جميع المسارات (paths) التى تبدأ عند **y**، فإن البحث يعود إلى **x**، وهى الرأس التى وصلنا منها إلى **y** أول مرة.
- تستمر عملية اختيار الأحرف غير المستكشفة (unexplored edges) الواقعة على **x**، حتى يتم استنفاد (exhausting) قائمة (list) هذه الأحرف .

يطلق على هذه الطريقة لزيارة رؤوس مخطط بياني غير موجَّه "البحث بالعُمق أولاً" (DFS) نظراً لأننا نستمر فى البحث متجهين إلى الأمام (إلى العُمق) (in the forward / deeper direction) قدر الاستطاعة (as long as possible).

ويمكن تطبيق طريقة **DFS** على المخطط البياني الموجَّه أيضا كما يلى: عند الرأس **x** نختار فقط الأحرف الموجَّهه **(x, y)** الخارجة من **x** (edges directed out **of**). وبعد استنفاد جميع الأحرف الخارجة من **y**، فإننا نعود

إلى x حتى لو كانت هناك أحرف أخرى موجَّهة نحو y لم يتم بحثها بعد (have not yet been searched).

وإذا طبقت طريقة DFS على أى مخطط بياني غير موجَّه وكان متصلاً (connected)، فمن السهل إثبات أن كل رأس ستزار، وكل حرف سيفحص (examined). وإن كان المخطط غير متصل، فإنه سيتم البحث فى مُرَكِّبَة متصلة فى المخطط. وعند إكمال (completion) مُرَكِّبَة متصلة نختار رأساً لم تتم زيارتها بعد لتكون رأس البداية الجديدة (the new starting vertex)، ونبدأ بحثاً جديداً.

وعندما تطبق طريقة البحث بالعمق أولاً DFS على أى مخطط بياني غير موجَّه $G = (V, E)$ فإنها تقوم بتجزئة أحرف E الى مجموعتين T, B، حيث نضع أى حرف (v, w) فى المجموعة T إذا كان الرأس w لم تتم زيارته من قبل عندما نكون عند الرأس v ونأخذ فى الاعتبار الحرف (v, w)، بينما نضع الحرف (v, w) فى المجموعة B إذا زرنا w من قبل. وأحرف المجموعة T يطلق عليها "**أحرف شجرة**" (**Tree edges**)، بينما أحرف B يطلق عليها "**أحرف خلفية /عكسية**" (**Back edges**). والمخطط البياني الجزئى (V, T) هو غابة غير موجَّهة (undirected forest) يطلق عليها "**غابة مُولِّدة للمخطط G بالعمق أولاً**" (a **depth-first spanning forest** for G) .

وفى حالة ما إذا كانت الغابة تتكون من شجرة واحدة (a single tree)، فإن الغابة (V, T) يطلق عليها "**شجرة مُولِّدة بالعمق أولاً**" (a depth-first spanning tree). ويلاحظ أنه إذا كان المخطط G متصلاً (connected) فإن الغابة المُولِّدة بالعمق أولا ستكون شجرة. وفى الغابة

المُولِّدة بالعمق أولاً نعتبر أن جذر أى شجرة فى الغابة هو تلك الرأس التى بدأنا عندها البحث بالعمق أولا فى هذه الشجرة.

تعريف 4-10: **شجرة البحث بالعمق أولاً، غابة البحث بالعمق أولاً**

Depth-first search tree, depth-first search forest

الأحرف التى تؤدى إلى رؤوس "جديدة" لم تتم زيارتها أو اكتشافها من قبل (vertices undiscovered / unvisited / "new") أثناء البحث بالعمق أولا فى مخطط بياني موجَّه G (a digraph) تُكوِّن شجرة ذات جذر (a rooted tree) يطلق عليها "**شجرة بحث بالعمق أولاً**" **(a depth-first search tree)** أو "**شجرة مُولِّدة بالعمق أولاً**" **(a depth-first spanning tree)**، وتُختصر – فى أى من الحالتين – إلى "**شجرة DFS**" (**DFS tree**). وإذا لم نستطع الوصول (reaching) إلى جميع الرؤوس من رأس البداية (الجذر)، فإن اجتيازا كاملا للمخطط G سيُجزّئ (partitions) الرؤوس إلى عدة أشجار يُسمّى مجموعها الكلى (entire collection) "**غابة البحث بالعمق أولا**" **(depth-first search forest)** أو "**غابة مُولِّدة بالعمق أولاً**" **(depth-first spanning forest)**، وتُختصر – فى أى من الحالتين – إلى "**غابة DFS**" (**DFS forest**).

وفيما يلى خوارزمية للبحث بالعمق أولا فى مخطط بياني.

الخوارزمية 4-1: **خوارزمية البحث بالعمق أولا فى مخطط بياني غير موجّه.**

Depth-first search Algorithm of an undirected graph

المدخلات: مخطط بياني $G = (V, E)$ تمثله قوائم التجاور:

$L[v]$, for $v \in V$

المخرجات: تجزئة E الى مجموعتين:

T: مجموعة أحرف شجرة (a set of tree edges)،

B: مجموعة أحرف عكسية / خلفية (a set of back edges).

الإجـــراء:

يقوم الإجراء الارتدادي التالي DFS(v) بإضافة الحرف (v, w) إلى المجموعة T إذا كنا سنصل إلى الرأس w أول مرة أثناء البحث عن طريق حرف من الرأس v. ونفرض أن جميع رؤوس المخطط ستُعطَى في البداية العلامة "new" (initially marked).

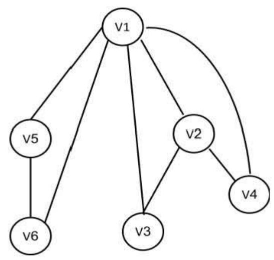
```
void DFS (v)
  {
     mark v "old";
     for ( each vertex w on L [v] )
       if ( w is marked "new" ) {
          add (v, w) to T
          DFS (w)
       }
  }
```

وفيما يلى الخوارزمية الكلية [البرنامج الرئيسى (main program)]:

```
   {
     T = Ø;
     for (all v in V ) mark v "new";
     while ( there exists a vertex v in V marked "new" )
          DFS (v)
   }
```

جميع أحرف E التى لم توضع فى T نعتبرها فى B. ويلاحظ أنه إذا كان الحرف (v,w) فى E، فإن w ستكون على L[v] و v ستكون على L[w]. وبناء عليه فإننا لا نستطيع أن نضع ببساطة الحرف (v, w) في B إذا كنا عند الرأس v وكان الرأس w مُعنوناً "old" نظراً لأن w قد يكون والد v (father of).

مثال 4-4: نفرض أن لدينا المخطط البياني G غير الموجَّه المبين فى شكل 4–8.



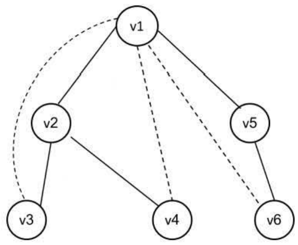
شكل 4–8

مخطط بياني G غير موجَّه

طبِّق خوارزمية البحث بالعمق أولا لاجتياز هذا المخطط وتجزئة مجموعة أحرفه E إلى مجموعتين: T مجموعة أحرف شجرة (Tree edges)، و B مجموعة أحرف خلفية (Back edges).

الحـــل: سنستخدم اصطلاح بيان أحرف الشجرة فى T بخطوط متصلة / مصمتة (solid)، والأحرف الخلفية فى B بخطوط متقطعة (dashed). وكذلك سنرسم الشجرة (أو الأشجار) بوضع الجذر

(root) (وهو رأس البداية (starting vertex) الذى تم اختياره فى السطر 8) فى أعلى (top) الشجرة، ورسم أبناء (sons) أى رأس من اليسار إلى اليمين بالترتيب الذى أضيفت به أحرفهم فى السطر 4 فى خوارزمية **DFS**. بناء على هذا فإن إحدى التجزئات الممكنة للمخطط المعطى **G** إلى شجرة وأحرف خلفية نتيجة تطبيق بحث بالعمق أولا تظهر فى شكل 4–9.



شكل 4–9

تجزئة المخطط G (فى شكل 4–8) إلى T و B بتطبيق DFS

فى البداية تكون جميع الرؤوس "new". نفرض أننا سنختار v_1 عند السطر 8. عندما ننفذ $DFS(v_1)$ فقد نختار $w = v_2$ فى السطر 2. ونظراً لأن v_2 معنونة "new" فإننا نضيف (v_1, v_2) إلى T ونستدعى $DFS(v_2)$. الإجراء $DFS(v_2)$ قد يختار v_1 من $L[v_2]$، ولكن v_1 قد تم عنونتها "old". فنفرض أننا نختار $w = v_3$. نظراً لأن v_3 معنونة "new"، فإننا نضيف (v_2, v_3) إلى T ونستدعى $DFS(v_3)$. والآن جميع الرؤوس المجاورة للرأس v_3 معنونة "old"، ولذلك فإننا نرجع إلى $DFS(v_2)$.

وبمتابعة $DFS(v_2)$ نجد الحرف (v_2, v_4) ونضيفه إلى **T**، ونستدعى $DFS(v_4)$. ولاحظ أننا رسمنا v_4 إلى يمين v_3 وهو الابن الذى وجدناه سابقاً

للرأس v_2. والآن لا توجد أى رؤوس "new" مجاورة للرأس v_4، وبالتالى نرجع إلى $DFS(v_2)$. وحيث أننا لا نجد الآن أى رؤوس "new" مجاورة للرأس v_2، فإننا نرجع إلى $DFS(v_1)$. وبمواصلة البحث فإن $DFS(v_1)$ يجد v_5، و $DFS(v_5)$ يجد v_6. وهكذا تصبح جميع الرؤوس على الشجرة ومعنونة "old"، وبالتالى تصل الخوارزمية إلى نهايتها. وإذا لم يكن المخطط البياني متصلا فإن عروة السطرين 9 , 8 ستتكرر مرة واحدة لكل مُرَكّبَة.

تحليل خوارزمية البحث بالعمق أولا Analysis of DFS

نود أن نحصل على درجة تعقيد الخوارزمية فى أسوأ حالة (worst case complexity) كدالة فى"n" عدد رؤوس المخطط البياني و "m" عدد أحرفه. نلاحظ أن **العمليتين الأساسيتين basic operations** فى الخوارزمية هما: اجتياز أحرف المخطط (traversing graph edges) وأوامرالإسناد لمتجه العلامات/العناوين (assignments to the mark vector). وبالنسبة **لعملية إسناد العناوين (mark assignments)** نلاحظ أنه يتم إجراء **n** عملية (n operations) فى السطر 7 فى البرنامج الرئيسى، وكذلك **n** عملية فى السطر 1 فى الإجراء **DFS**. وأما بالنسبة لعملية **اجتياز الأحرف (edge traversals)** نجد أولاً أن الاجراء DFS يُستدعى مرة واحدة (called once) لكل رأس إما من الخوارزمية الرئيسية (main algorithm) أو ارتداديا (recursively) من الإجراء DFS نفسه. ونجد ثانيا أنه داخل DFS (within) يتم اجتياز قائمة التجاور لأى رأس (adjacency list of a vertex) مرة واحدة بالضبط (exactly once). ومن أولاً وثانيا نجد أن لدينا O(m) عمليات اجتياز كلية [a total of O(m) traversals].

ومما سبق نستنتج أن درجة التعقيد (complexity) هى $O(m + n)$،

والتى تكون عادة $O(m)$ نظراً لأن n تكون عادة $O(m)$. وأما بالنسبة لباقي الخطوات فى الخوارزمية فإنها تتطلب عددا مساوياً أو أقل من العمليات (operations).

ويمكننا أن نلخص ما سبق من تحليل خوارزمية DFS فى النظرية التالية.

نظرية 4-1: الخوارزمية 4-1 (خوارزمية البحث بالعمق أولاً) لاجتياز مخطط بياني غير موجَّه عدد رؤوسه n وعدد أحرفه m تتطلب خطوات (steps) عددها $O(\max\ (n, m))$.

البرهان: إذا عملنا قائمة (list) من الرؤوس ومسحناها (scanned) مرة واحدة، فإن السطر 7 فى الخوارزمية والبحث عن رؤوس "new" يتطلبان $O(n)$ من الخطوات. والزمن الذى نستغرقه فى الإجراء $DFS(v)$ - باستثناء استدعاءاته الارتدادية لنفسه (exclusive of recursive calls to itself) - يتناسب طردياً مع (proportional to) عدد الرؤوس المجاورة للرأس v. والإجراء $DFS(v)$ يتم استدعاؤه مرة واحدة فقط لرأس معطى v، وذلك لأن الرأس v يُعنون "old" حينما نستدعى $DFS(v)$ أول مرة. وهكذا فإن الزمن الكلى الذى نستغرقه فى DFS يساوى $O(\max\ (n, m))$.

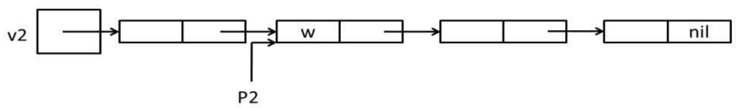
من الواضح أن خوارزمية DFS خوارزمية ذات كفاءة عالية لاستكشاف (exploring) مخطط بياني، حيث أنها تحتاج لرؤية مدخلاته من الرؤوس والأحرف (its input vertices and edges) مرة واحدة فقط.

ملاحظة:

كى يكون تحليلنا السابق صحيحاً (valid)، فإن خوارزميتنا DFS يجب أن تتذكر (remember) من أين تستمر/ تستأنف (where to continue / resume) اجتياز الأحرف فى السطر 2 فى الإجراء:

for (each vertex **w** on L [v])
    if (**w** is marked "new") {
        add (v, w) to T
        DFS (w)
    }

فمثلا اذا كان لدينا التمثيل البياني التالى (منظومة قوائم التجاور):



المؤشر الحالى
current pointer

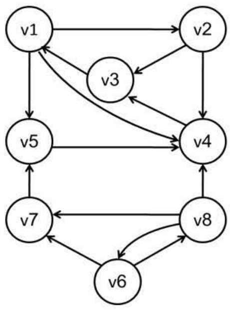
فهذا يعنى اننا نحتاج إلى منظومة مؤشرات (array of pointers): مؤشرا لكل رأس (vertex). غير أن هذا يتحقق لنا تلقائيا (automatically done for us) فى أى نظام (system) يسمح بالارتداد (recursion).

البحث بالعمق أولا فى مخطط بياني موجه

Depth-First Search of a directed graph

يمكننا تطبيق الخوارزمية 4- 1 أيضاً لإيجاد غابة مُولِّدة موجهة (a directed spanning forest) لمخطط بياني موجه $G = (V, E)$ اذا عَرَّفنا القائمة $L[v]$ وهى قائمة الرؤوس "المجاورة" "adjacent" للرأس **v** على أنها $\{w \mid (v, w) \text{ is an edge}\}$، أى أن $L(v)$ هى قائمة الرؤوس (vertices) التى أىٌّ منها هو رأس حرف (head of an edge) ذيله **v** **(tail)** .

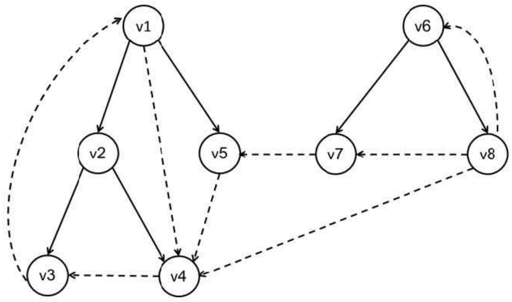
مثال 4-5: نفرض أن لدينا المخطط البياني الموجه **G** المبين في شكل 4-10.
طبِّق خوارزمية البحث بالعمق أولاً لاجتياز المخطط **G**، وإيجاد غابة مُولِّدة له.



شكل 4-10

مخطط بياني موجّه G

الحــل: باستخدام اصطلاح بيان أحرف الشجرة (tree edges) بخطوط مصمتة وغيرها بخطوط متقطعة - كما فعلنا سابقاً - نحصل على غابة مولّدة للمخطط **G** مبينة في شكل 4-11.



شكل 4-11

غابة مولّدة للمخطط G (شكل 4-10)

لإنشاء غابة مولّدة سنبدأ عند الرأس v_1. الإجراء $DFS(v_1)$ يستدعى $DFS(v_2)$ الذى يستدعى $DFS(v_3)$. وهذا الأخير يتوقف دون أي إضافات للشجرة لأن الحرف الوحيد الذى ذيله v_3 يتجه إلى v_1 وهذا قد سبقت عنونته "old". وهكذا نعود إلى $DFS(v_2)$ الذى يضيف عندئذ v_4 كإبنٍ ثانٍ للرأس v_2. الآن $DFS(v_4)$ يتوقف ولا يقوم بأى إضافات إلى الغابة نظراً لأن v_3 قد سبق عنونتها "old". ثم يتوقف $DFS(v_2)$ لأن جميع الأحرف الخارجة من v_2 قد تم أخذها فى الاعتبار. وهكذا نعود أدراجنا إلى v_1 الذى يستدعى $DFS(v_5)$، وهذا يتوقف دون أى إضافات إلى الشجرة، وكذلك لا يستطيع $DFS(v_1)$ إضافة أى جديد.

والآن نختار v_6 كجذر لشجرة مولدة بالعمق أولا جديدة. وإنشاء هذه الشجرة مماثل لما سبق ونترك للقارئ الكريم متابعته. ولاحظ أن الترتيب الذى اخترناه لزيارة الرؤوس هو $v_1, v_2, \ldots, v_8$.

تعريف 4-11: يقال إن رأساً v هو **سَلَف** / **جَدّ** (**ancestor**) لرأس w فى شجرة ما إذا وقع v على المسار (path) من الجذر (root) إلى w. ويقال إن v **سلف فعلى** (a **proper ancestor**) للرأس w إذا كان v سلفاً للرأس w و$v \neq w$. وأقرب سلف فعلى للرأس v هو **والد** v (**parent of**)، وإذا كان v سلفا فعليا للرأس w ، فإن w **سليل / حفيد فعلى من ذرية** v (a **proper descendant** of).

تعريف 4-12: أحـــرف أى مخطـــــط بيانـــي موجـــــه G تُصنـــف / تجزأ

(classified / partitioned) إلى أربع فئات (categories) بالبحث فى G بالعمق أولاً بناءً على كيفية استكشافها (how they are explored) أى اجتيازها فى اتجاهها للأمام (traversed [in their forward direction)؛

(1) **أحرف شجرة (Tree edges)**: وهى الأحرف التى تؤدى إلى رؤوس جديدة أثناء عملية البحث. أى أنه إذا كان **w** رأساً جديداً عند استكشاف الحرف **vw**، فإن **vw** يطلق عليه حرف شجرة، ويصبح **v** والد **w** (parent of) .

(2) **أحرف خلفية / عكسية (Back edges)**: وهى الأحرف التى تتجه من السلالة / الذرية من الأبناء والأحفاد (descendants) إلى السلف من الآباء والأجداد (ancestors) [مع احتمال اتجاه الحرف من رأس إلى نفسه]. أى أنه إذا كان **w** سلفا للرأس **v**، فإن **v w** يسمى حرفاً خلفياً (وذلك يشمل أيضا **v v**).

(3) **أحرف أمامية/متجهة إلى الأمام (Forward edges)**: وهى الأحرف التى تتجه من السلف من الآباء والأجداد (ancestors) إلى السلالة الفعلية / الذرية الفعلية (proper descendants) ولكنها ليست أحرف شجرة. أى أنه إذا كان **w** من سلالة / ذرية **v**، ولكن **w** تم اكتشافه قبل (discovered earlier) وقت استكشاف **vw** (exploring) فإن **vw** يسمى حرفا أمامياً. والحرف الأمامى يسمى أيضا "**حرف نازل / هابط / منحدر من سلف** (a **descendant edge**)".

(4) **أحرف مستعرضة / متعارضة / متقاطعة (Cross**

edges) : وهى الأحرف التى تصل بين رؤوس ليس بينها علاقة آباء/ أبناء (no ancaster / descendant relationship).

أى أنه إذا لم يكن W سلفاً ولا سليلا للرأس V، فإن VW يسمى حرفاً مستعرضا.

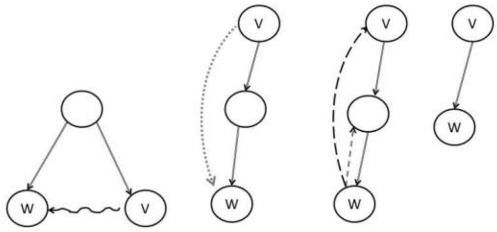
الشكل التالى (شكل 4-12) يوضح الاصطلاحات التى نستخدمها لهذه الأنواع الأربعة من الأحرف.

حرف شجرة: خط متصل

حرف خلفى: خط متقطع

حرف أمامي: خط منقط

حرف مستعرض: خط متعرج

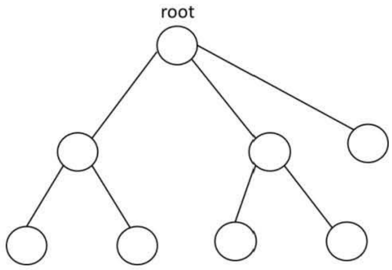


شكل 4-12

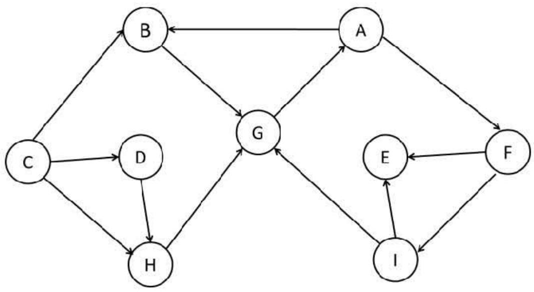
اصطلاحات الأحرف

والمثالان التاليان يوضحان هذه الأنواع من الأحرف [أحرف شجرة (tree edges)، وأحرف خلفية (back edges)، وأحرف أمامية (forward edges) وهى المتجهة من رأس إلى واحد من ذريتها ليس ابناً (not a child)، وأحرف مستعرضة (cross edges) وهى الأحرف بين رأسين ليس أى منهما من ذرية الآخر]. ولاحظ أن رأس وذيل (head and tail) الحرف المستعرض قد يكونان فى شجرتين مختلفتين.

ولتبسيط تصنيف الأحرف (edge classification) سنتبع اصطلاح رسم الأشجار كما هو مبين بالشكل التالى حيث الجذر (root) هو أعلى رأس وأبناء (children) أى رأس يظهرون فى المستوى الأدنى الذى يلى مستواه بترتيب زيارتهم (in order of visiting them) من اليسارإلى اليمين.



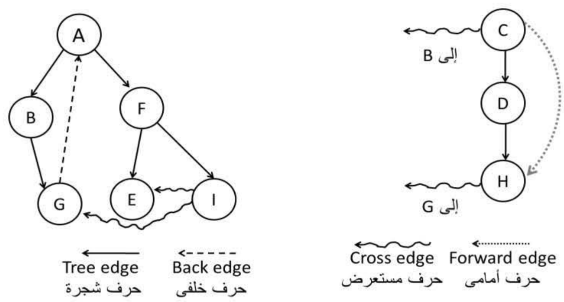
مثال 4-6: طبِّق خوارزمية البحث بالعمق أولاً لاجتياز المخطط البياني الموجه G المبين فى شكل 4-13 وبيان أشجار البحث بالعمق أولا مع تصنيف أحرف المخطط.



شكل 4-13

مخطط بياني موجه G

الحـــل:

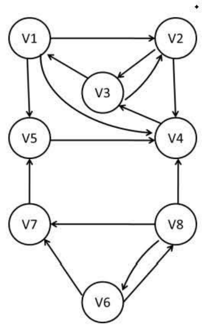


شكل 4-14

أشجار البحث بالعمق أولا للمخطط البياني الموجَّه G

Depth-first search trees for the digraph G

مثال 4-7: طبّق خوارزمية البحث بالعمق أولاً لاجتياز المخطط البياني الموجَّه G المبين في شكل 4-15 وبيان الغابة المولدة له مع تصنيف أحرف المخطط. وضح أيضاً أرقام / ترتيب زيارة رؤوس المخطط أول مرة بهذه الخوارزمية.



شكل 4-15

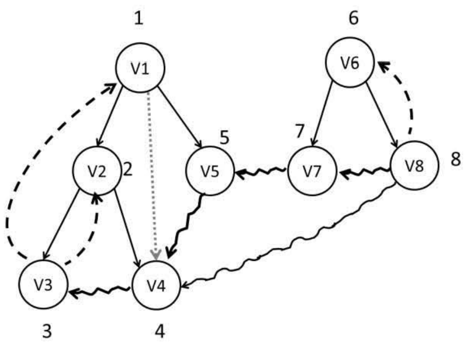
مخطط بياني موجَّه G

A digraph G

الحـــل: فيما يلى قائمة بأسماء رؤوس المخطط، وأمام كل رأس نذكر أسماء الرؤوس التى تتجه إليها أحرف من هذا الرأس.

$v_1 : v_2, v_4, v_5$	$v_5 : v_4$
$v_2 : v_3, v_4$	$v_6 : v_7, v_8$
$v_3 : v_1, v_2$	$v_7 : v_5$
$v_4 : v_3$	$v_8 : v_4, v_6, v_7$

ويوضح الشكل التالى (شكل 4-16) الغابة المولّدة لهذا المخطط مع بيان تصنيف أحرفه، وأرقام ترتيب زيارة رؤوسه. وقد استخدمنا الاصطلاحات نفسها التى ذكرناها سابقا لأنواع الأحرف الأربعة.



شكل 4-16

غابة مولدة للمخطط البياني الموجَّه G (شكل 4–15)

Spanning forest for the digraph G

ملاحظتان حول الأشجار Remarks regarding trees

ملاحظة 1: أى مخطط بياني غير موجه $G=(V,E)$(an undirected

يُعد graph) شجرة إذا كان متصلا (connected) ولا يحتوى على أى دورة (cycle).

ملاحظة 2: العبارات التالية جميعها متكافئة (equivalent) بفرض أن G مخطط بياني عدد أحرفه m ، وعدد رؤوسه n:

(1) المخطط G شجرة.

(2) المخطط G غير موجه، ومتصل، ولا يحتوى على أى دورة.

(3) المخطط G لا يحتوى على أى دورة، وعدد أحرفه $n-1$.

(4) المخطط G متصل، وعدد أحرفه n-1.

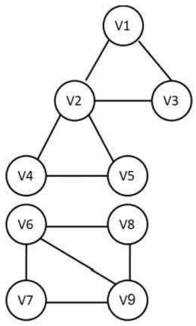
(5) المخطط G متصل، ولكن إذا حذف منه حرف (an edge is deleted) يصبح غير متصل (disconnected).

(6) المخطط G لا يحتوى على أى دورة، ولكن إذا أضيف إليه حرف (an edge is added) فإن دورة وحيدة تتكون (a unique cycle is formed).

(7) أى رأسين v , w فى المخطط G متصلان بمسار وحيد (a unique path).

المثال التالي **يوضح تطبيق خوارزمية البحث بالعمق أولا على مخطط بياني غير موجه وغير متصل.**

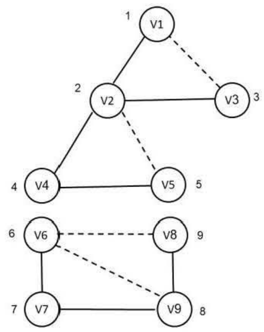
مثال 8-4: طبّق خوارزمية البحث بالعمق أولا لاجتياز المخطط البياني G غير الموجه وغير المتصل المبين فى شكل 4-17، والحصول على غابة مولّدة للمخطط، مع بيان أرقام ترتيب زيارة رؤوسه.



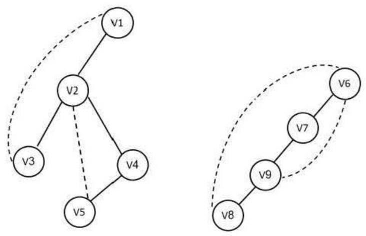
شكل 4-17
مخطط بياني G غير موجَّه وغير متصل
A disconnected undirected graph G

الحـــل: سنبدأ بإذن الله بإعطاء جميع رؤوس المخطط العلامة (mark) "new". وسنختار v_1 كرأس بداية (a starting vertex). ثم نتابع تطبيق خطوات الخوارزمية 4-1، والشكل التالى (شكل 4-18) يوضح نتيجة تطبيق هذه الخطوات، مع ملاحظة أن كلا من الشكلين (4-18- أ) و(4-18- ب) يعطى الغابة المولدة التى حصلنا عليها، ولكن الشكل (4-18- ب) يتبع اصطلاح رسم الأشجار الذى ذكرناه سابقاً حيث الجذر هو أعلى رأس، وأبناء أى رأس يظهرون فى المستوى الأدنى الذى يلى مستواه بترتيب زيارتهم من اليسار إلى اليمين. وتظهر فى الشكل (4-18- أ) أرقام ترتيب زيارة رؤوس المخطط G. وفى الغابة المولّدة التى حصلنا عليها نجد أن

$$T = \{ (v_1,v_2), (v_2,v_3), (v_2,v_4), (v_4,v_5), (v_6,v_7), (v_7,v_9), (v_9,v_8) \}$$



(أ)



(ب)

شكل 4-18

غابة مولّدة للمخطط البياني G غير الموجّه وغير المتصل (شكل 4-17)

spanning forest for the disconnected undirected graph G

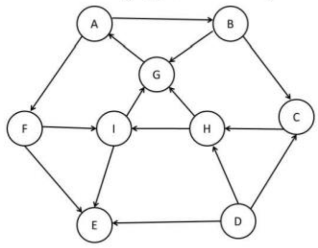
ثانياً: البحث بالعرض أولاً Breadth-First Search (BFS)

رأينا سابقا فى طريقة البحث بالعمق أولا لاجتياز مخطط بياني أنه عند زيارة رأس جديدة "new" ما **v** ، فإننا نستمر فى البحث متجهين إلى العمق (إلى الأمام) قدر الاستطاعة، ثم نرجع (back up) عبر آخر حرف تم اجتيازه

لتتفرع (branch out) فى اتجاه آخر. وهذا يؤدى إلى زيارة جميع الرؤوس الموجودة فى مخطط بياني جزئي مجاور للرأس **v** (subgraph adjacent to) قبل الذهاب إلى مخطط بياني جزئي جديد (new subgraph) مجاور للرأس **v**، وهذا يماثل الاجتياز سابق الترتيب للأشجار (preorder traversal of trees) الذى يقوم بزيارة جميع الرؤوس فى شجرة فرعية ما (one subtree) قبل الذهاب إلى الشجرة الفرعية التالية. ولذلك فإن البحث بالعمق أولا يُعدَ تعميما (generalization) للاجتياز سابق الترتيب للأشجار.

أما فى طريقة البحث بالعرض أولا فإننا نزور الرؤوس بترتيب المسافات المتزايدة (in order of increasing distance) من نقطة البداية **v** (starting point) مثلاً، حيث المسافة ببساطة هى عدد الأحرف فى أقصر مسار (number of edges in a shortest path). ونقصد بأقصر مسار من رأس **v** إلى رأس **w** يمكن الوصول إليه من v (reachable from v) فى مخطط G، مساراً (a path) يحتوى على أقل عدد من الأحرف (smallest number of edges). أى أن الخوارزمية تكتشف (discovers) / تزور (visits) جميع الرؤوس التى تبعد مسافة تساوى d من الرأس **v** قبل اكتشاف / زيارة أى رؤوس تبعد مسافة تساوى d+1 من **v**. فالخطوة الأساسية/ المركزيه (central step) فى طريقة البحث بالعرض أولاً - ابتداءً بمسافة $\mathbf{d = 0}$ وتكراريا (repeated) حتى لا نجد أى رؤوس جديدة - هى أن نأخذ فى الاعتبار بالترتيب (consider in turn) كل رأس "x" واقعة على بُعد d من رأس البداية **v**، وبفحص (examining) جميع الأحرف الواقعة على x (all edges incident with) ثم نجد ونزور / نشغّل (find and process) جميع الرؤوس الواقعة على بُعد $d + 1$ من **v**.

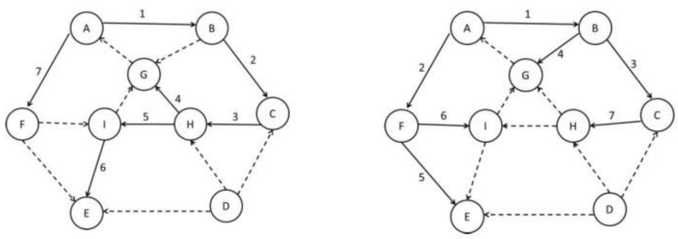
مثال 4-9: طبّق كلاً من خوارزمية البحث بالعرض أولا و خوارزمية البحث بالعمق أولا لاجتياز المخطط البياني الموجه G المعطى فى شكل 4-19، مبتدئاً عند الرأس A.



شكل 4-19

مخطط بياني موَجَّه A digraph G

الحـــل: الشكل التالى (شكل 4-20) يعطى نتيجة اجتياز المخطط G بتطبيق كل من خوارزميتي البحث بالعرض أولا والبحث بالعمق أولاً، وقد رُقّمت الأحرف فى الشكل بترتيب اجتيازها (edges are numbered in the order traversed).



(أ) البحث بالعرض أولاً BFS

Breadth-First search

(ب) البحث بالعمق أولاً DFS

Depth-First search

شكل 4-20

ونلاحظ من الشكل أن ترتيب زيارة الرؤوس (order in which vertices are visited) عند البحث بالعرض أولا (شكل 4-20- أ) هو: ABFCGEIH، بينما ترتيب زيارة الرؤوس عند البحث بالعمق أولاً (شكل 4-20- ب) هو: ABCHGIEF .

ملاحظة: رأينا سابقاً في (تعريف 4- 12) أن أحرف أي مخطط بياني موجَّه G (a digraph) تصنف إلى أربع فئات بالبحث في **G** بالعمق أولا. أما إذا كان **G** مخططاً بيانيا غير موجه (an undirected graph) فإن:

i) خوارزمية البحث بالعرض أولا **BFS** تُنتج (produces):

أحرف شجرة (tree edges)، وأحرفاً مستعرضة (cross edges) ولا تُنتج أي أحرف خلفية (back edges) أو أحرف أمامية (forward edges).

ii) خوارزمية البحث بالعمق أولا **DFS** تُنتج:

أحرف شجرة، وأحرفاً خلفية، ولا تُنتج أي أحرف مستعرضة أو أحرف أمامية.

❖ ❖ ❖