CHƯƠNG II

ĐỒ THỊ DẠNG CÂY

(Chương này chỉ đề cập đến ***các đồ thị vô hướng***)

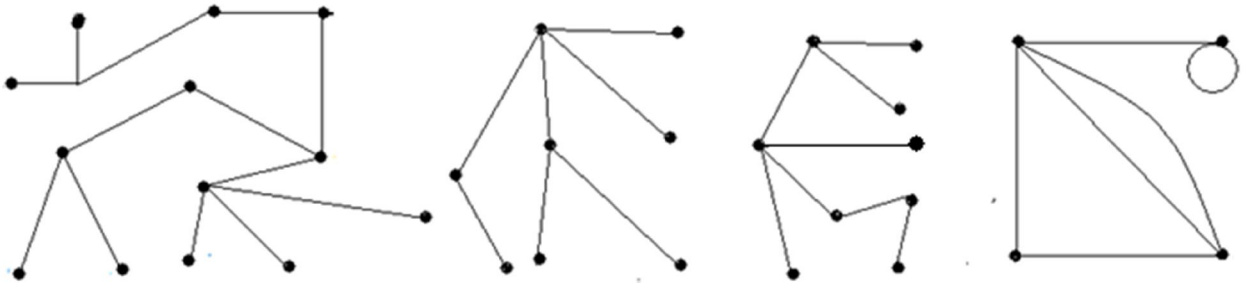
I. CÁC KHÁI NIỆM CƠ BẢN:

1.1/ ĐỊNH NGHĨA:

a) *Cây tự do* là một đồ thị *vô hướng, liên thông* và *không chứa chu trình*. Từ nay về sau, ta nói gọn cây tự do là *cây*. Như vậy mỗi cây là *một đơn đồ thị*.

b) Rừng là một đồ thị *vô hướng, rời rạc và không chứa chu trình*. Nói khác đi, rừng là phần hội của *nhiều cây rời nhau từng đôi một*. Mỗi cây là *một thành phần liên thông* (nghĩa là *một đồ thị con liên thông tối đại*) của rừng.

Ví dụ:



G **H** **K** **L**

G, H, K là các cây (tự do) và L không phải là cây (tự do) vì L chứa chu trình.

G, H và K hội lại thành một rừng gồm 3 thành phần liên thông rời nhau đôi một.

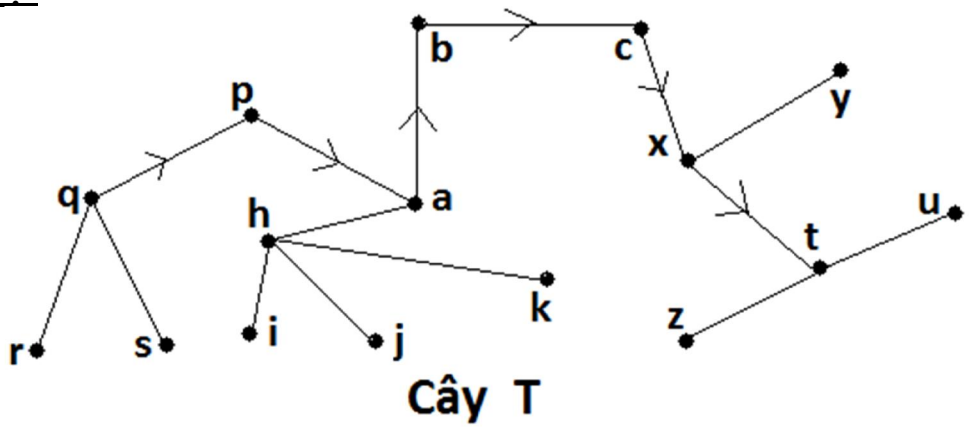
1.2/ MỆNH ĐỀ: Cho *cây* T = (V, E) cùng với hai đỉnh *khác nhau* a và b. Khi đó :

a) Tồn tại duy nhất một đường (P_{ab}) trong T nối a và b.

b) Đặt L(a, b) = *độ dài* của đường (P_{ab}) trong T.

Ta nói L(a, b) là *khoảng cách* giữa a và b ở trong T.

Ví dụ:



Cây T

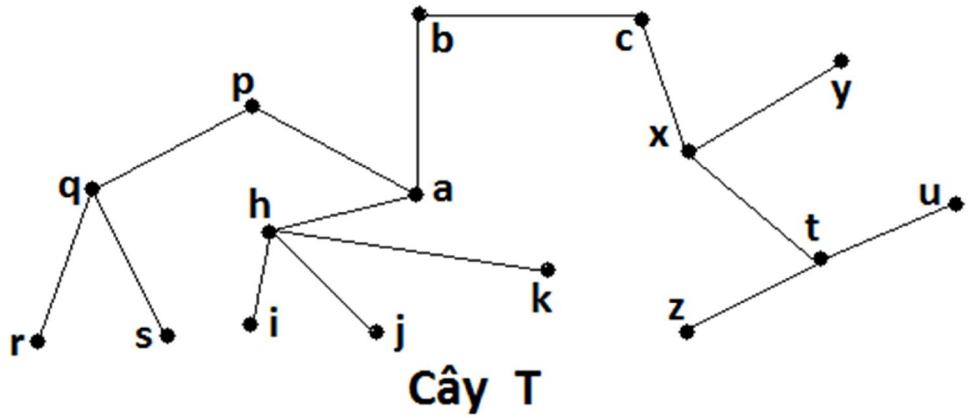
Xét 2 đỉnh khác nhau tùy ý trong T, chẳng hạn là 2 đỉnh q và t.

Đường duy nhất trong G nối 2 đỉnh q và t là (P) : $\overline{qpabcxt}$ có L(P) = 6 nên ta cũng nói khoảng cách giữa các đỉnh q và t là L(q, t) = 6.

1.3/ MỆNH ĐỀ: Cho cây T = (V, E) .

Nếu | V | = n thì | E | = n – 1.

Ví dụ:



Cây T

Cây T có 16 đỉnh nên T có (16 – 1) = 15 cạnh.

1.4/ MỆNH ĐỀ: Cho đồ thị T = (V, E) có | V | = n .

Các phát biểu sau đây là *tương đương với nhau*:

a) T là *một cây*.

b) T *liên thông* và | E | = n – 1.

c) T *liên thông* vả khi bỏ một cạnh tùy ý của T (không bỏ đỉnh) thì đồ thị mới T' *không còn liên thông* (T' *rời rạc*) .

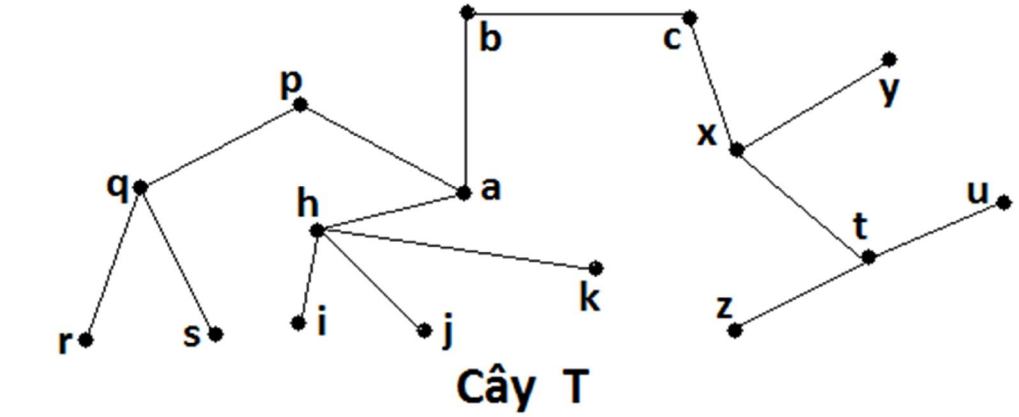
d) T *không có chu trình* và | E | = n – 1.

e) T *không chứa chu trình* và khi thêm một cạnh mới nối hai đỉnh bất kỳ của T, đồ thị mới T'' *chứa chu trình*.

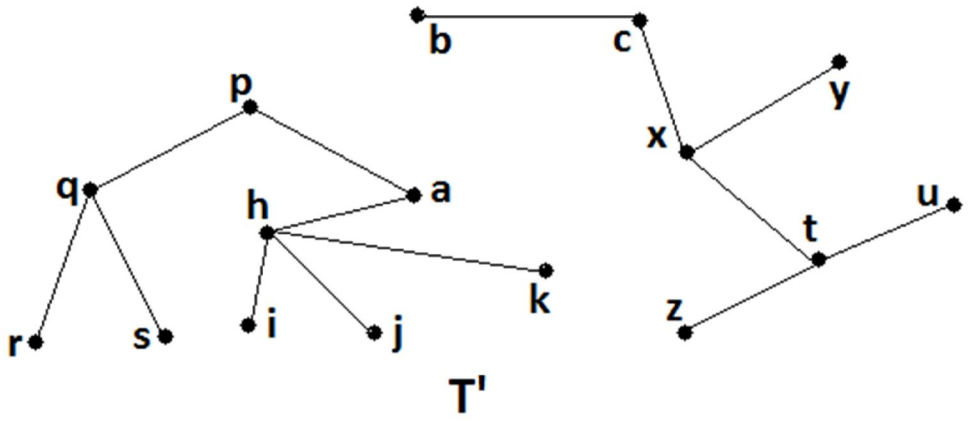
f) Trong T, giữa hai đỉnh khác nhau bất kỳ luôn luôn có *một đường nối duy nhất*.

Ví dụ:

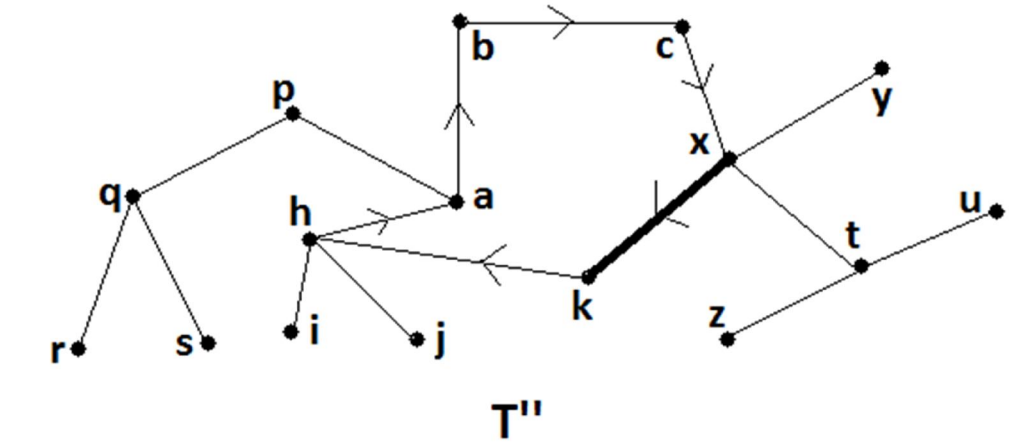
a)



Cây T



T'



T''

Giữa 2 đỉnh khác nhau bất kỳ của T luôn luôn có đường nối duy nhất trong T.

Xóa cạnh $\overline{ab}$ (không xóa 2 đỉnh a và b) của T thì đồ thị mới T' không còn liên thông (T' rời rạc).

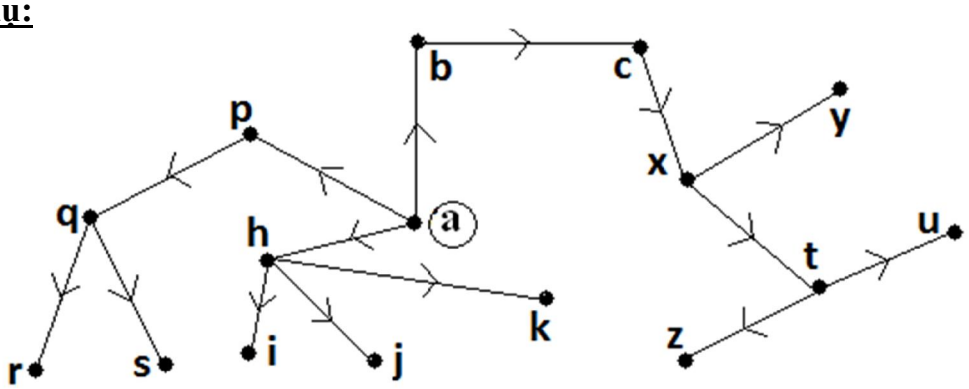
Thêm cạnh mới $\overline{kx}$ vào T thì đồ thị mới T'' chứa chu trình (C) : $\overline{abcxkha}$.

b) Một cây có ít nhất hai đỉnh treo. Thật vậy, nếu cây T = (V, E) với | V | = n và có không quá một đỉnh treo (bậc 1) thì 2 | E | = (Tổng bậc của n đỉnh) ≥ 2(n – 1) + 1, nghĩa là | E | > (n – 1) : mâu thuẫn với b) của **(1.4)**.

1.5/ <u>CÂY CÓ GỐC:</u> Cho cây (tự do) T = (V, E) . Chọn tùy ý đỉnh a của T.

Ta định hướng trên mỗi cạnh của T sao cho ta đều có các đường đi từ đỉnh a đến các đỉnh khác của T theo hướng đã định. Lúc này ta nói T là *một cây có gốc* là a và ta có thể xem T = (V, E) như *một đồ thị có hướng*.

<u>Ví dụ:</u>

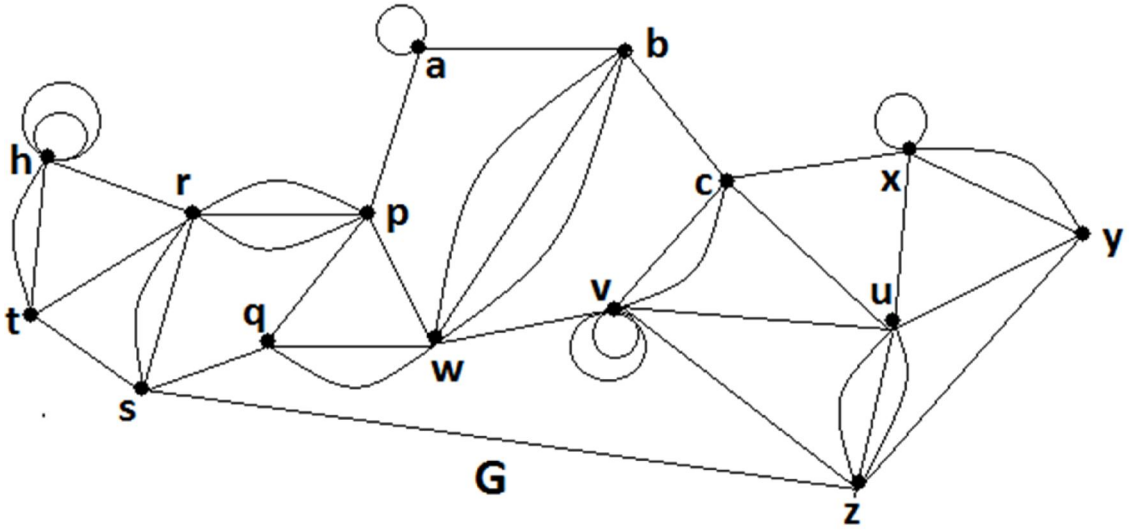


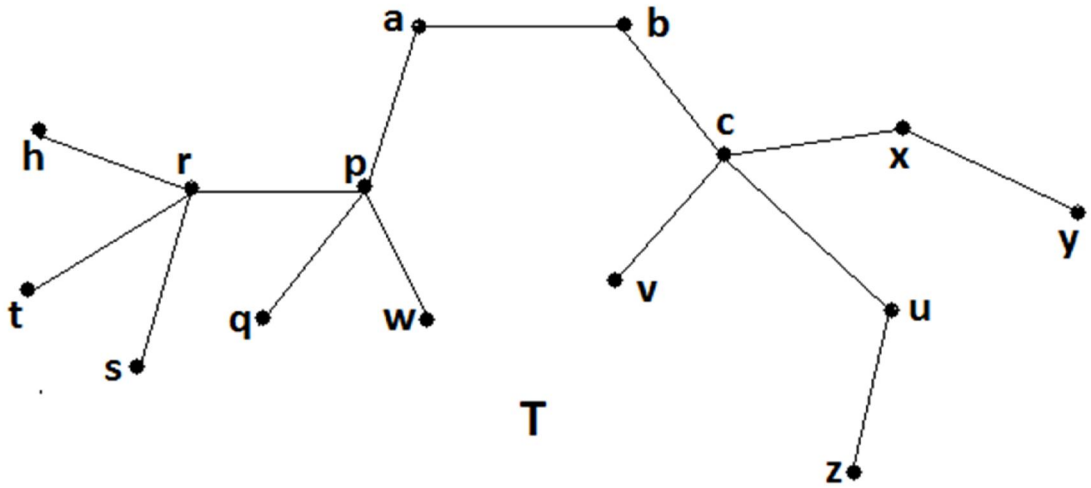
Cây T (gốc a)

II. <u>CÂY KHUNG TRONG ĐỒ THỊ LIÊN THÔNG:</u>

2.1/ <u>ĐỊNH NGHĨA:</u> Ta nói T là *một cây khung* (spanning tree = ST) của đồ thị G = (V, E) nếu T là một cây trong G và T chứa mọi đỉnh của G.

<u>Ví dụ:</u> Đồ thị G dưới đây có một cây khung T như sau:





2.2/ <u>MỆNH ĐỀ:</u> Cho G = (V, E). Khi đó

a) G *có ít nhất một cây khung* $\Leftrightarrow$ G *liên thông*.

b) Nếu G là *một cây* thì G là *cây khung duy nhất* của chính G.

c) Giả sử G *liên thông có chứa chu trình* và T = (V, E') là *một cây khung* trong G với $E' \subset E$ và $E' \neq E$.

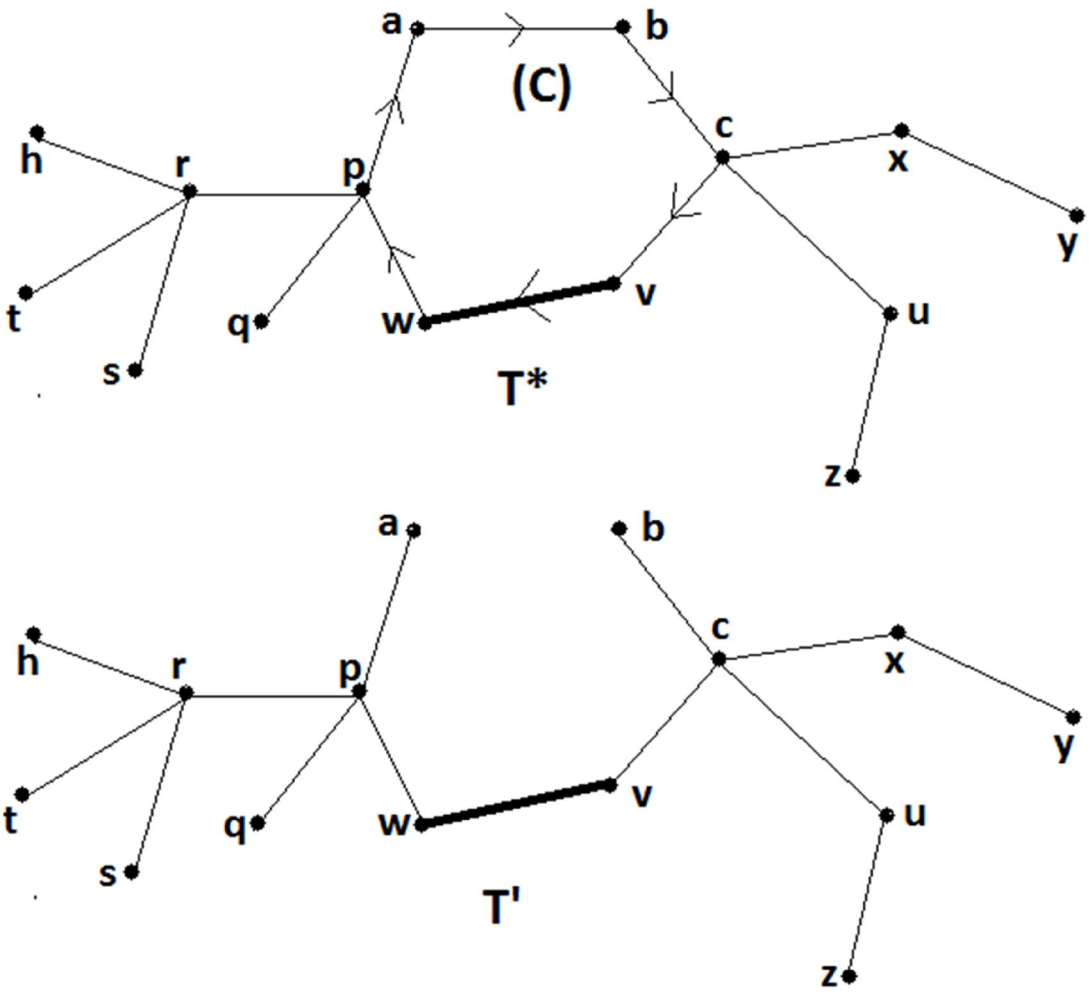
Chọn tùy ý $\alpha \in E \setminus E'$ thì $T^* = (V, E^* = E' \cup \{\alpha\})$ có chu trình (C) qua α.

Chọn tùy ý $\beta \in (C) \setminus \{\alpha\}$ thì $T' = [\, V, E'' = E^* \setminus \{\beta\} = (E' \cup \{\alpha\}) \setminus \{\beta\}]$ là một cây khung mới của G.

Như vậy cây khung của G *không nhất thiết duy nhất*.

Ví dụ: Xét lại đồ thị liên thông G = (V, E) cùng với cây khung T = (V, E') trong **Ví dụ (2.1)**. Chọn $\alpha = \overline{vw} \in E \setminus E'$ thì $T^* = (V, E^* = E' \cup \{\alpha\})$ có chu trình (C) : $\overline{abcvwpa}$ qua α.

Chọn $\beta = \overline{ab} \in (C) \setminus \{\alpha\}$ thì $T' = [V, E'' = E^* \setminus \{\beta\} = (E' \cup \{\alpha\}) \setminus \{\beta\}]$ là một cây khung mới của G.



2.3/ ĐỒ THỊ CÓ TRỌNG SỐ: Cho G = (V, E).

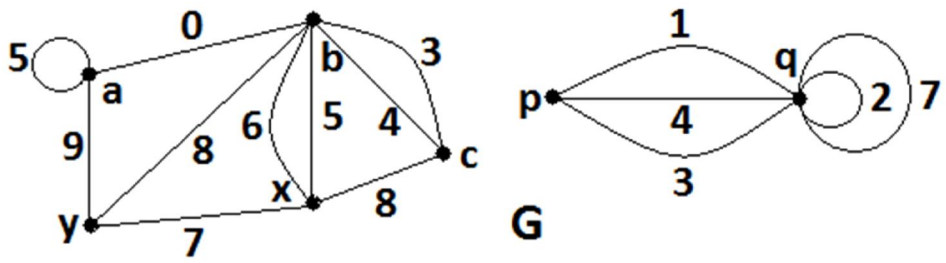
a) Giả sử mỗi cạnh của G được gán với một số thực và ta gọi số thực này là *trọng số* của cạnh tương ứng đã được gán.

Lúc này ta nói G là *một đồ thị có trọng số*.

b) Khi G là một đồ thị *có trọng số*, ta gọi tổng trọng số của tất cả các cạnh trong G là trọng số của G. Tương tự, trọng số của một đồ thị con (hay cây, đường,

chu trình) trong G là tổng trọng số của tất cả các cạnh mà đồ thị con (hay cây, đường, chu trình) đó đi qua. Ký hiệu w(G) là trọng số của đồ thị G.

Ví dụ: Xét đồ thị G có trọng số như sau:



Ta có w(G) = 5 + 9 + 0 + 8 + 6 + 5 + 7 + 3 + 4 + 8 + 1 + 4 + 3 + 2 + 7 = 72.

2.4/ ĐỊNH LÝ CAYLEY : $\forall n \geq 2$, đồ thị K_n có số cây khung là n^{n-2}.

Ví dụ: K_2 có $2^{2-2} = 1$ cây khung. K_3 có $3^{3-2} = 3$ cây khung.

K_4 có $4^{4-2} = 16$ cây khung. K_5 có $5^{5-2} = 125$ cây khung.

2.5/ XÂY DỰNG CÂY KHUNG ĐỒ THỊ LIÊN THÔNG (THUẬT TOÁN WFS):

Cho G = (V, E) liên thông. Thuật toán Width – First – Search (WFS) xây dựng một cây khung T của G bằng phép duyệt theo chiều rộng như sau :

- Chọn $a \in V$ và a là gốc của cây rỗng T_o.
- Thêm các đỉnh và các cạnh kề với a vào T_o sao cho đồ thị mới tạo ra không chứa chu trình. Đồ thị mới này được chọn là cây T_1.
- Thêm các đỉnh và các cạnh kề với các đỉnh của T_1 vào T_1 sao cho đồ thị mới tạo ra không chứa chu trình. Đồ thị mới này được chọn là cây T_2.
- Tiếp tục quá trình này cho đến khi tất cả các đỉnh của G được ghép vào cây T_k ở bước thứ k. Lúc đó T_k chính là một cây khung T có gốc a của G.

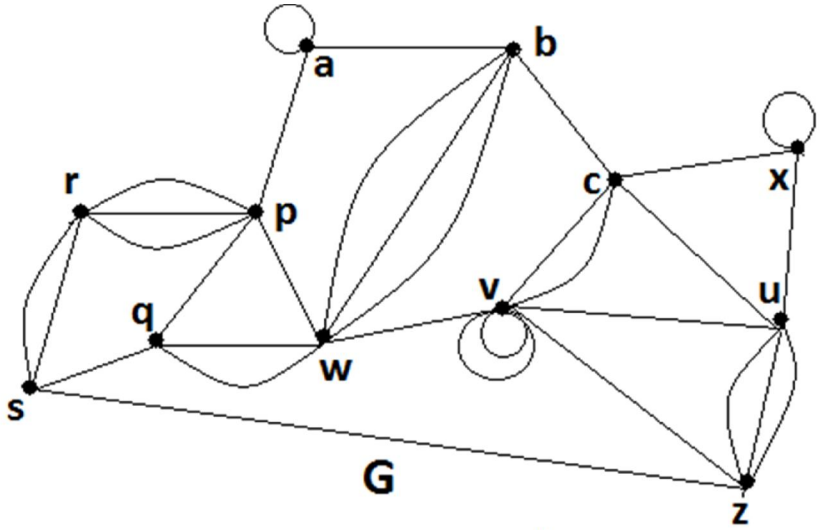
Ví dụ: Xây dựng một cây khung cho đồ thị G dưới đây bằng thuật toán BFS :

$T_o = (V_o = \{ a \}, E_o = \varnothing)$, $T_1 = (V_1 = \{ a, b, p \}, E_1 = \{ \overline{ab}, \overline{ap} \})$,

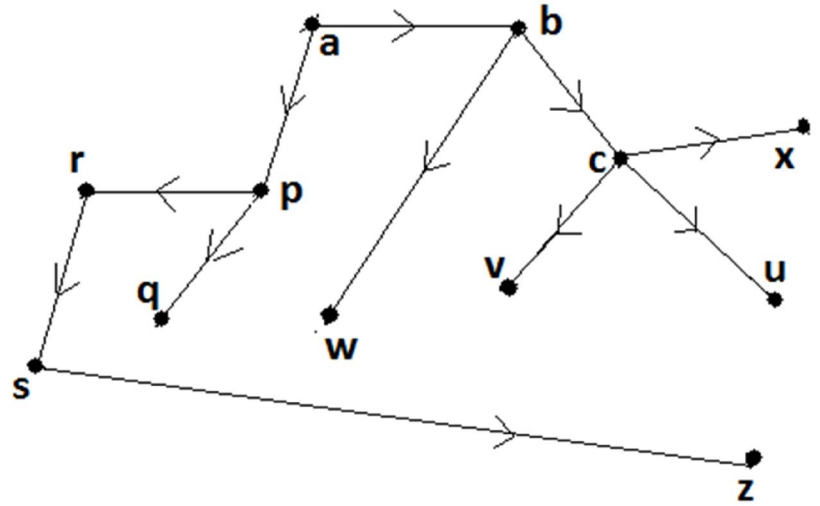
$T_2 = (V_2 = \{ a, b, p, c, w, q, r \}, E_2 = \{ \overline{ab}, \overline{ap}, \overline{bc}, \overline{bw}, \overline{pq}, \overline{pr} \})$,

$T_3 = (V_3 = \{a, b, p, c, w, q, r, s, u, v, x\}, E_3 = \{\overline{ab}, \overline{ap}, \overline{bc}, \overline{bw}, \overline{pq}, \overline{pr}, \overline{rs}, \overline{cx}, \overline{cu}, \overline{cv}\})$,

$T_4 = (V_4 = \{a, b, p, c, w, q, r, s, u, v, x, z\}, E_3 = \{\overline{ab}, \overline{ap}, \overline{bc}, \overline{bw}, \overline{pq}, \overline{pr}, \overline{rs}, \overline{cx}, \overline{cu}, \overline{cv}, \overline{sz}\})$.



G



T = T_4 (một cây khung của G vì V_4 = V)

2.6/ XÂY DỰNG CÂY KHUNG ĐỒ THỊ LIÊN THÔNG (THUẬT TOÁN DFS) :

Cho G = (V, E) liên thông. Thuật toán Depth – First – Search (DFS) xây dựng một cây khung T của G bằng phép duyệt theo chiều sâu như sau :

- Chọn $a \in V$ và a là gốc của cây rỗng T_o.
- Tạo một đường sơ cấp (càng dài càng tốt) từ gốc a sao cho nó không chứa chu trình. Đồ thị mới này được chọn là cây T_1.
- Tạo một đường sơ cấp (càng dài càng tốt) từ gốc a sao cho nó không phải là đồ thị con của T_1 và khi thêm nó vào T_1 thì đồ thị mới tạo ra không chứa chu

trình. Đồ thị mới này được chọn là cây T_2.

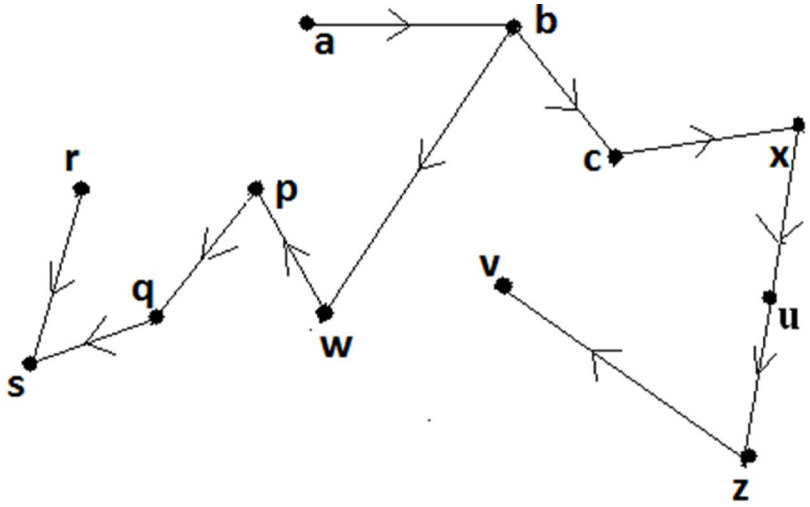
- Tiếp tục quá trình này cho đến khi tất cả các đỉnh của G được ghép vào cây T_k ở bước thứ k. Lúc đó T_k chính là một cây khung T có gốc a của G.

Ví dụ: Xây dựng một cây khung cho đồ thị G của **(2.6)** bằng thuật toán DFS.

(P_1) : $\overline{abwpqsr}$, $T_1 = (V_1 = \{a, b, w, p, q, s, r\}, E_2 = \{\overline{ab}, \overline{bw}, \overline{wp}, \overline{pq}, \overline{qs}, \overline{sr}\})$,

(P_2) : $\overline{abcxuzv}$. $(P_1) \cup (P_2)$ không chứa chu trình.

$T_2 = (V_2 = \{a, b, w, p, q, s, r, c, x, u, z, v\}, E_2 = \{\overline{ab}, \overline{bw}, \overline{wp}, \overline{pq}, \overline{qs}, \overline{sr}, \overline{bc}, \overline{cx}, \overline{xu}, \overline{uz}, \overline{zv}\})$

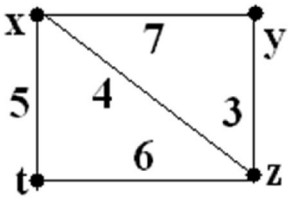


$T = T_2$ (một cây khung của G vì $V_2 = V$)

2.7/ MA TRẬN KHOẢNG CÁCH (MA TRẬN TRỌNG SỐ) :

Cho G = (V, E) là đơn đồ thị có trọng số với $V = \{v_1, v_2, \ldots, v_n\}$. Với $1 \le i, j \le n$, ký hiệu $w(v_i, v_j)$ là trọng số của cạnh $\overline{v_i v_j}$ và đặt $d_{ij} = 0$ (nếu i = j), $d_{ij} = w(v_i, v_j)$ (nếu có $\overline{v_i v_j}$) và $d_{ij} = \infty$ (nếu không có $\overline{v_i v_j}$). Khi đó $D = (d_{ij})_{1 \le i, j \le n}$ gọi là *ma trận khoảng cách* (hay *ma trận trọng số*) của G. Ma trận D là ma trận đối xứng.

Ví dụ: Cho đơn đồ thị có trọng số



G = (V, E)

G có ma trận khoảng cách D_G là

D_G	x	y	z	t
x	0*	7	4	5
y	7	0*	3	∞
z	4	3	0*	6
t	5	∞	6	0*

2.8/ CÂY KHUNG NHỎ NHẤT VÀ CÂY KHUNG LỚN NHẤT:

Cho G = (V, E) *liên thông và có trọng số*.

G có thể có nhiều cây khung khác nhau với các trọng số khác nhau.

a) Nếu T là một cây khung *có trọng số nhỏ nhất* thì ta nói T là *một cây khung nhỏ nhất* (minimal spanning tree = MST) của G.

b) Nếu T là một cây khung *có trọng số lớn nhất* thì ta nói T là *một cây khung lớn nhất* (maximal spanning tree = M'ST) của G.

2.9/ THUẬT TOÁN PRIM TÌM CÂY KHUNG NHỎ NHẤT VÀ LỚN NHẤT:

Cho G = (V, E) *liên thông có trọng số* và $|V| = n \geq 1$.

Ta tìm một MST cho G. Việc tìm M'ST cho G hoàn toàn tương tự bằng cách thay các từ '' *nhỏ nhất* '' bằng các từ '' *lớn nhất* ''.

* Chọn tùy ý $a \in V$. Đặt $T_1 = (V_1 = \{a\}, E_1 = \varnothing)$.

* Nếu $|V_1| = 1 = n$ thì T_1 là một MST của G.

* Nếu $|V_1| = 1 < n$, chọn $a \in V_1$, $b \in V \setminus V_1$ sao cho $\overline{ab} \in E$ và $\overline{ab}$ có trọng số nhỏ nhất có thể được. Đặt $T_2 = (V_2 = V_1 \cup \{b\}, E_2 = E_1 \cup \{\overline{ab}\})$.

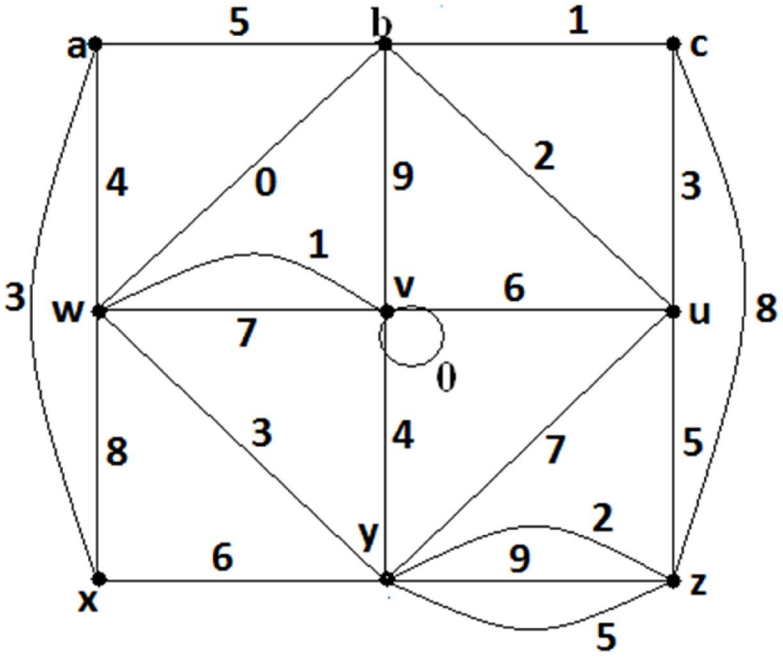
* Nếu $|V_2| = 2 = n$ thì T_2 là một MST của G.

* Nếu $|V_2| = 2 < n$, chọn $x \in V_2$, $y \in V \setminus V_2$ sao cho $\overline{xy} \in E$ và $\overline{xy}$ có trọng số nhỏ nhất có thể được. Đặt $T_3 = (V_3 = V_2 \cup \{y\}, E_3 = E_2 \cup \{\overline{xy}\})$.

Tiếp tục như trên cho đến khi có $T_n = (V_n, E_n)$ với $|V_n| = n$ và $V_n = V$.

Ta có T_n là một MST của G.

Ví dụ: Cho đồ thị G = (V, E) liên thông có trọng số như sau :



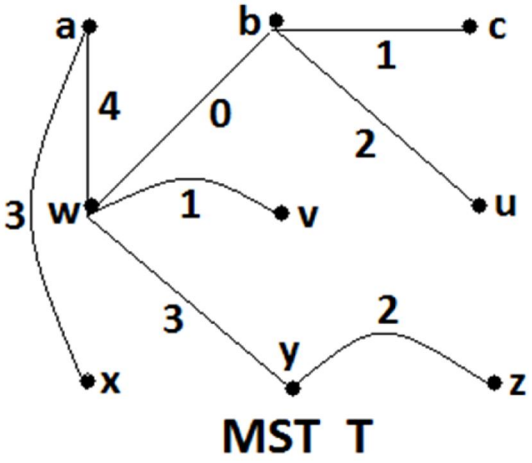
Hãy xác định cây khung nhỏ nhất T và cây khung lớn nhất Z của G sao cho Z đi qua $\overline{bw}$ và không đi qua $\overline{bv}$. Tính w(T) và w(Z).

Tìm T : a, $\overline{ax}$, w, $\overline{aw}$, b, $\overline{bw}$, c, $\overline{bc}$, v, $\overline{vw}^{1}$, u, $\overline{bu}$, y, $\overline{wy}$ và z, $\overline{yz}^{2}$.

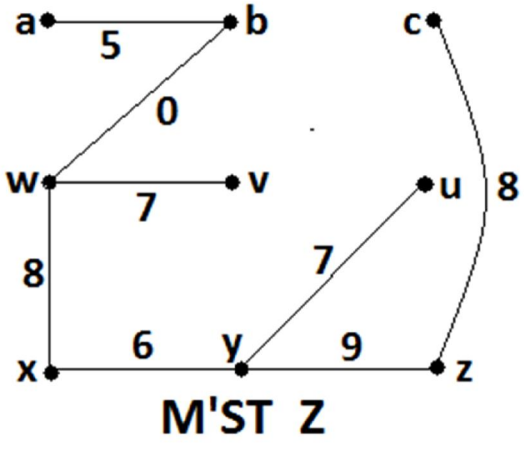
w(T) = 3 + 4 + 0 + 1 + 1 + 2 + 3 + 2 = 16.

Tìm Z : a, $\overline{ab}$, w, $\overline{bw}$, x, $\overline{wx}$, v, $\overline{vw}^{7}$, y, $\overline{xy}$, z, $\overline{yz}^{9}$, c, $\overline{cz}$ và u, $\overline{yu}$.

w(Z) = 5 + 0 + 8 + 7 + 6 + 9 + 8 + 7 = 50.



MST T



M'ST Z

2.10/ THUẬT TOÁN KRUSKAL TÌM CÂY KHUNG NHỎ NHẤT VÀ CÂY KHUNG LỚN NHẤT:

Cho $G = (V, E)$ *liên thông có trọng số* và $|V| = n \geq 1$.

Ta tìm một MST cho G. Việc tìm M'ST cho G hoàn toàn tương tự bằng cách thay các từ '' *nhỏ nhất* '' bằng các từ '' *lớn nhất* ''.

* Đặt $T_1 = (V, E_1 = \varnothing)$.

* Nếu $|E_1| = 0 = n - 1$ thì T_1 là một MST của G.

* Nếu $|E_1| = 0 < (n - 1)$, chọn $\alpha \in E \setminus E_1$ sao cho α có trọng số nhỏ nhất có thể được và $E_2 = E_1 \cup \{\alpha\} = \{\alpha\}$ không chứa chu trình. Đặt $T_2 = (V, E_2)$.

* Nếu $|E_2| = 1 = n - 1$ thì T_2 là một MST của G.

* Nếu $|E_2| = 1 < n - 1$, chọn $\beta \in E \setminus E_2$ sao cho β có trọng số nhỏ nhất có thể được và $E_3 = E_2 \cup \{\beta\}$ không chứa chu trình. Đặt $T_3 = (V, E_3)$.

Tiếp tục như trên cho đến khi có $T_n = (V, E_n)$ với $|E_n| = n - 1$.

Ta có T_n là một MST của G.

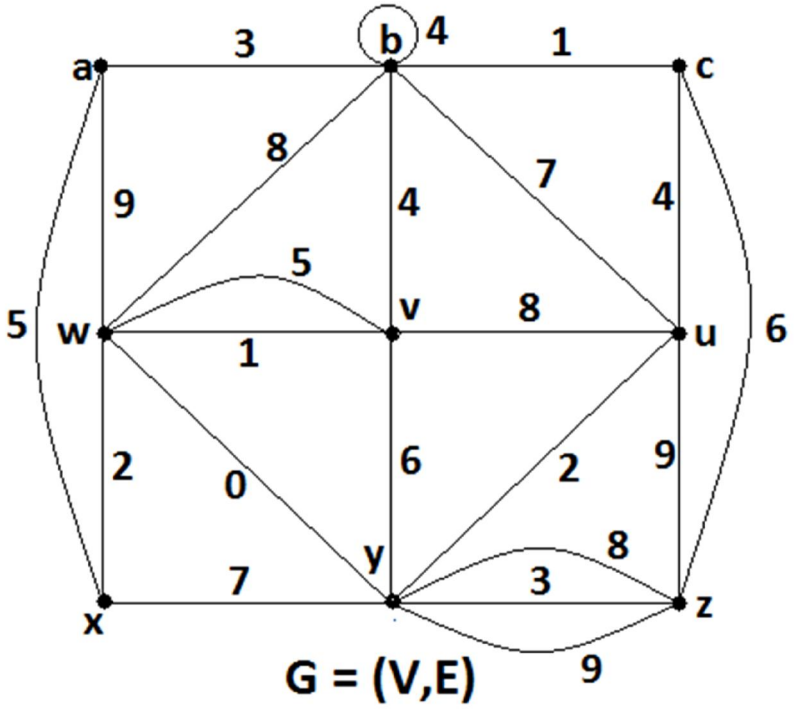
2.11/ NHẬN XÉT VỀ THUẬT TOÁN PRIM VÀ KRUSKAL:

a) Trong thuật toán PRIM, ở mỗi bước ta chọn *một cạnh mới* có trọng số *nhỏ nhất* (nếu tìm MST) hoặc *lớn nhất* (nếu tìm M'ST) *nối từ một đỉnh cũ nào đó với một đỉnh mới*. Cách làm này luôn luôn bảo đảm được *sự liên thông* và *không có chu trình* của cây trong quá trình xây dựng từ khi bắt đầu đến khi kết thúc. Khi đồ thị đang xây dựng có đủ số đỉnh thì thuật toán hoàn thành.

b) Trong thuật toán KRUSKAL, từ *các đỉnh hoàn toàn rời rạc* đã chọn lúc ban đầu, ta thêm lần lượt các cạnh có trọng số *tăng dần* (nếu tìm MST) hoặc có trọng số *giảm dần* (nếu tìm M'ST) để được các đồ thị dần dần " *bớt rời rạc* "

nhưng luôn luôn phải bảo đảm tính chất " *không chứa chu trình* ". Đến khi đồ thị đang xây dựng " *vừa hết rời rạc* " (nghĩa là " *vừa mới liên thông* ") thì thuật toán hoàn thành.

<u>Ví dụ</u> (của **<u>2.10</u>):** Cho đồ thị G liên thông có trọng số như sau:



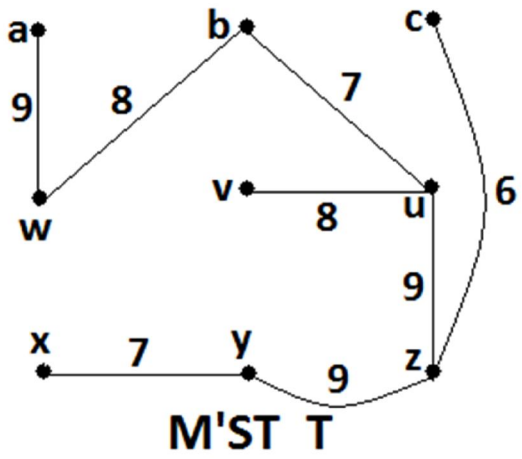
Hãy xác định cây khung lớn nhất T và cây khung nhỏ nhất Z của G sao cho Z đi qua $\overline{bw}$ và không đi qua $\overline{uy}$. Tính w(T) và w(Z).

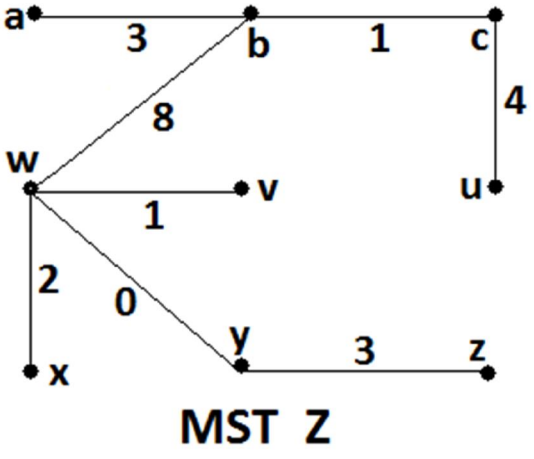
Tìm T : a, b, c, u, v, w, x, y z, $\overline{aw}$, $\overline{uz}$, $\overline{\overset{9}{yz}}$, $\overline{bw}$, $\overline{uv}$, $\overline{bu}$, $\overline{xy}$ và $\overline{cz}$.

w(T) = 9 + 9 + 9 + 8 + 8 + 7 + 7 + 6 = 63.

Tìm Z : a, b, c, u, v, w, x, y z, $\overline{bw}$, $\overline{wy}$, $\overline{bc}$, $\overline{\overset{1}{vw}}$, $\overline{wx}$, $\overline{ab}$, $\overline{yz}$ và $\overline{cu}$.

w(Z) = 8 + 0 + 1 + 1 + 2 + 3 + 3 + 4 = 22.





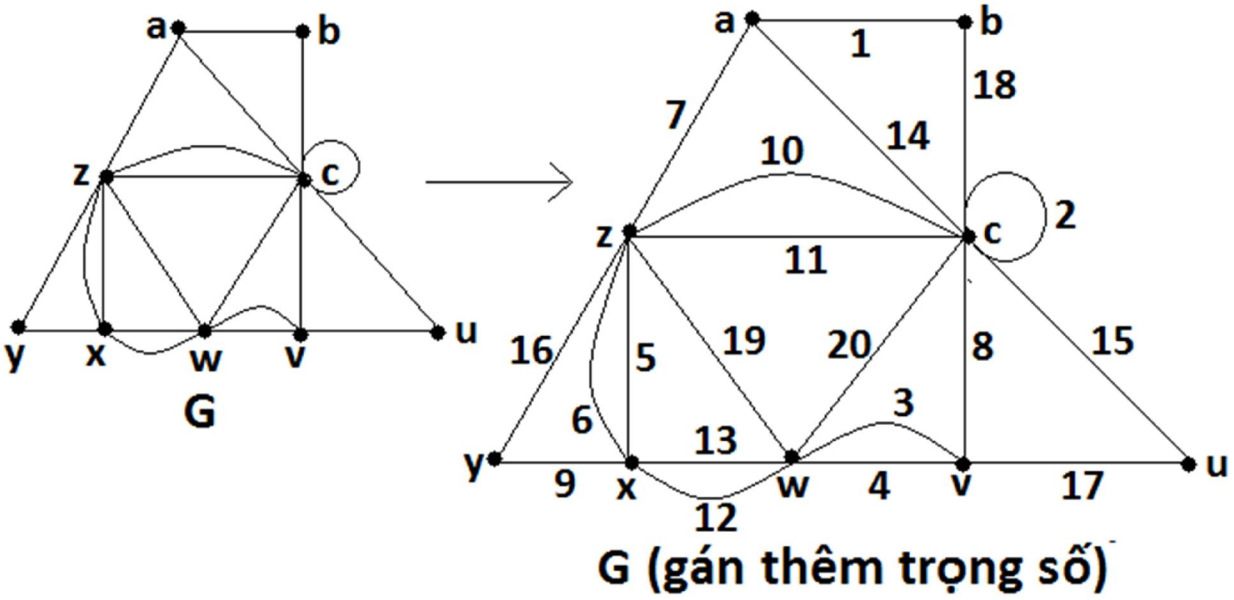
2.12/ THUẬT TOÁN VẼ CÂY KHUNG CHO ĐỒ THỊ LIÊN THÔNG:

Cho G = (V, E) liên thông và | E | = n.

Nếu G *không có chu trình* thì G chính là cây khung duy nhất của G.

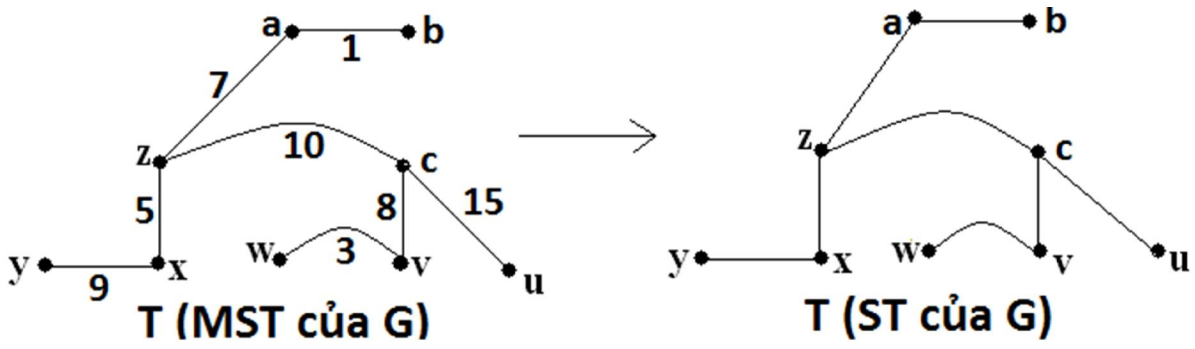
Nếu G *có chu trình*, dùng các số nguyên 1, 2, …, n để gán các trọng số khác nhau cho các cạnh của G. Dùng thuật toán PRIM (hay KRUSKAL) để tìm một MST là T cho G. Xóa các trọng số của G, ta được một cây khung T của G.

Ví dụ: Cho G = (V, E) liên thông và | E | = 20 như dưới đây. Sau đó ta gán trọng số khác nhau (từ 1 đến 20) cho 20 cạnh của G :



Dùng thuật toán KRUSKAL, ta được T là một MST của G như dưới đây.

Sau đó ta xóa bỏ trọng số của các cạnh của T để có một ST của G:



III. PHÉP DUYỆT CÂY:

3.1/ ĐỊNH NGHĨA: Cho cây T có gốc a (do đó T xem như là đồ thị có hướng).

a) Xét cạnh $\overline{uv}$ của T (u là đỉnh trước và v là đỉnh sau).

Ta nói u là *cha* của v hay ngược lại, v là *con* của u.

b) Các đỉnh có cùng cha được gọi là *các đỉnh anh em*.

c) Xét đường (P) : $\overline{v_1 v_2 \ldots v_{k-1} v_k}$ trong T. Ta nói $v_1, v_2, \ldots, v_{k-1}$ là *các tổ tiên* của v_k hay ngược lại, v_k là *hậu duệ* của $v_1, v_2, \ldots$ và v_{k-1}.

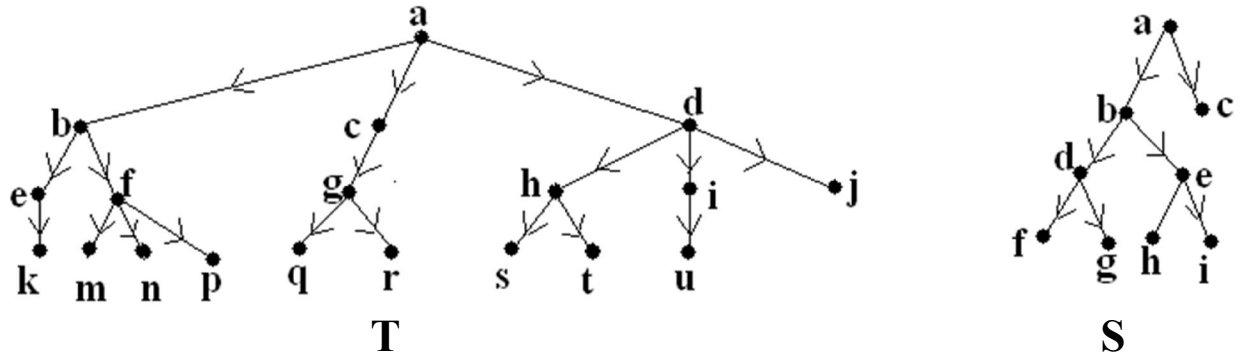
d) Đỉnh không có con (đỉnh bậc 1) của T gọi là *lá* (hay *đỉnh ngoài*).

Đỉnh có ít nhất một con của T gọi là *đỉnh trong*.

e) *Một cây con tại đỉnh* v của T là một cây có gốc là v và các đỉnh khác của nó đều là các hậu duệ của v trong T.

f) Xóa bỏ một đỉnh v của T (đương nhiên xóa luôn các cạnh đi qua gốc đó) thì phần đồ thị còn lại vẫn là *một cây* hoặc là *một rừng* (gồm nhiều cây rời nhau).

Ví dụ: Cho các cây T và S như sau:



Trong T, ta có b, c và d là các đỉnh anh em vì có đỉnh cha chung là a.

Trong S, xét đường (P): $\overline{abei}$. Ta thấy a, b, e là các tổ tiên của i và i là hậu duệ của a, b, e trong S.

T có các lá (đỉnh ngoài) là k, m, n, p, q, r, s, t, u và j.

S có các đỉnh trong là a, b, c, d và e.

Tại d, T có một cây con T' với đỉnh d và các đỉnh khác là h, i, j, s, t và u (chúng đều là các hậu duệ của d trong T).

Trong S, khi xóa đỉnh e (và xóa các cạnh $\overline{eb}$, $\overline{eh}$, $\overline{ei}$ đi qua e) thì phần đồ thị còn lại là một cây mới.

Trong T, khi xóa đỉnh a (và xóa các cạnh $\overline{ab}$, $\overline{ac}$, $\overline{ad}$ đi qua a) thì phần đồ thị còn lại là một rừng gồm 3 cây con rời nhau từng đôi một (đây là 3 thành phần liên thông của rừng).

3.2/ <u>ĐỊNH NGHĨA:</u> Cho cây T có gốc a (T xem như là một đồ thị có hướng).

a) Gốc a gọi là *đỉnh mức* 0 của T. Các đỉnh con của a gọi là *các đỉnh mức* 1 của T. Các đỉnh con của các đỉnh mức 1 gọi là *các đỉnh mức* 2 của T.

Các đỉnh con của các đỉnh mức 2 gọi là *các đỉnh mức* 3 của T.

Tương tự như vậy, ta có được *các đỉnh mức* k ($k \geq 4$) trong T.

b) Đỉnh v có mức k ($k \geq 1$) trong T $\Leftrightarrow$ Có một đường sơ cấp độ dài k đi từ gốc a đến v.

c) *Độ cao* của cây T chính là mức cao nhất của các đỉnh trong T.

<u>Ví dụ:</u> Cho các cây T và S như trong **Ví dụ (3.1)**.

Trong T: đỉnh mức 0 là a. Các đỉnh mức 1 là b, c và d. Các đỉnh mức 2 là e, f, g, h, i và j. Các đỉnh mức 3 là k, m, n, p, q, r, s, t và u.

Độ cao của T là 3. Trong S, đỉnh g có mức 3 vì có đường sơ cấp (Q): $\overline{abdg}$ có độ dài 3 đi từ a đến g.

3.3/ <u>CÂY m - PHÂN:</u> Cho cây T có gốc a (T xem như là một đồ thị có hướng).

a) Ta nói T là *một cây m_ phân* ($m \geq 2$) nếu mỗi đỉnh của T có nhiều nhất m con và T phải có ít nhất một đỉnh có đúng m con. Cây 2_ phân còn được gọi là *cây nhị phân*. Cây 3_ phân còn được gọi là *cây tam phân*.

b) Ta nói T là *một cây m_ phân đủ* ($m \geq 2$) nếu mỗi đỉnh trong của T có đúng m con.

c) Một cây m_phân ($m \geq 2$) với độ cao h ($h \geq 1$) được gọi là *cân bằng* nếu mọi lá của nó đều ở mức h hoặc (h – 1).

<u>Ví dụ:</u> Cho các cây T và S như trong **Ví dụ (3.1)**.

T là một cây tam phân không đủ nhưng cân bằng (các lá đều có mức 2 hoặc mức 3).

S là một cây nhị phân đủ nhưng không cân bằng (các lá có mức 1 hoặc mức 3).

3.4/ <u>CÁC CÂY CON CỦA CÂY NHỊ PHÂN:</u>

Cho cây nhị phân T có gốc là a.

a) Nếu d(a) = 2 thì T có hai cây con ở bên trái và bên phải của đỉnh a ký hiệu lần lượt là T_L và T_R. Các đỉnh của T_L và T_R đều là các hậu duệ của đỉnh a.

b) Tương tự, nếu đỉnh v của T có bậc 2 thì đỉnh đó cũng có *cây con trái* và *cây con phải*. Nếu đỉnh v có bậc 1 thì nó chỉ có một cây con và ta có thể mặc định trước đó là *cây con trái* hoặc *cây con phải*. Các đỉnh của các cây con của v đều là các hậu duệ của đỉnh v.

3.5/ <u>PHÉP DUYỆT CÂY :</u> Một phép duyệt cây là một cách *nêu ra một danh sách liệt kê đầy đủ các đỉnh của cây theo một thứ tự nào đó* sao cho mỗi đỉnh xuất hiện đúng một lần.

a) Đối với cây nhị phân : Có 3 thuật toán thông thường dùng để duyệt cây nhị phân theo tinh thần đệ qui như sau:

– Thuật toán PREORDER SEARCH (tiền thứ tự):

* Gốc.

* Cây con bên trái (rồi lại dùng tiếp Preorder search).

* Cây con bên phải (rồi lại dùng tiếp Preorder search).

– Thuật toán INORDER SEARCH (trung thứ tự):

* Cây con bên trái (rồi lại dùng tiếp Inorder search).

* Gốc.

* Cây con bên phải (rồi lại dùng tiếp Inorder search).

– Thuật toán POSTORDER SEARCH (hậu thứ tự) :

* Cây con bên trái (rồi lại dùng tiếp Postorder search).

* Cây con bên phải (dùng lại dùng tiếp Postorder search).

* Gốc.

b) Đối với cây m_ phân ($m \geq 3$) : Ta chỉ dùng các thuật toán Preorder search và Postorder search tương tự như trên để duyệt cây.

– Thuật toán PREORDER SEARCH (tiền thứ tự):

* Gốc.

* Các cây con từ bên trái qua bên phải (trong mỗi cây con lại dùng tiếp Preorder search).

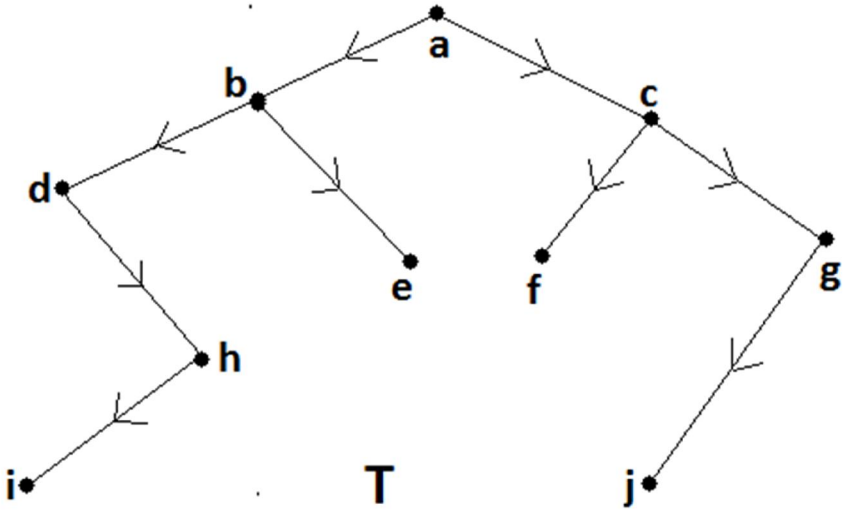
– Thuật toán POSTORDER SEARCH (hậu thứ tự) :

* Các cây con từ bên trái qua bên phải (trong mỗi cây con lại dùng tiếp Postorder search).

* Gốc.

Ví dụ:

a) Cho cây nhị phân T có gốc **a** như sau :



Mỗi đỉnh a, b, c có 2 cây con trái và phải. Mỗi đỉnh g, h chỉ có cây con trái.

Đỉnh d chỉ có cây con phải.

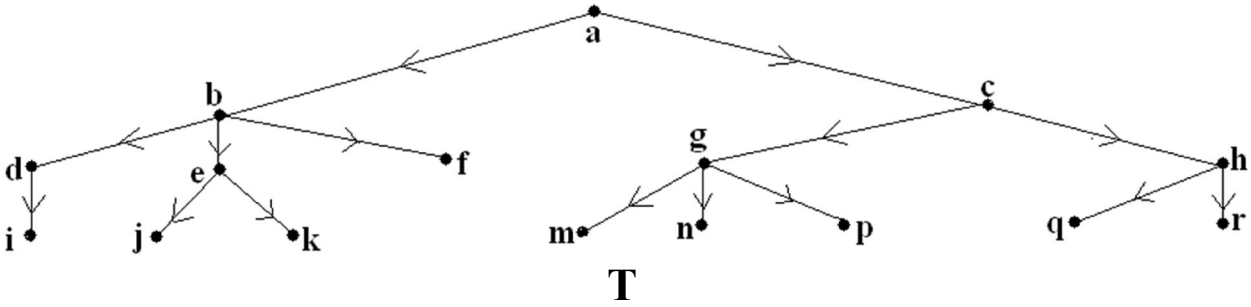
Các đỉnh e, f, i, j là các lá (vì chúng không có cây con).

– Duyệt cây theo thuật toán Preorder search : a, b, d, h, i, e, c, f, g, j.

– Duyệt cây theo thuật toán Inorder search : i, h, d, b, e, a, f, c, j, g.

– Duyệt cây theo thuật toán Postorder search : i, h, d, e, b, f, j, g, c, a.

b) Cho cây tam phân T có gốc **a** như sau :

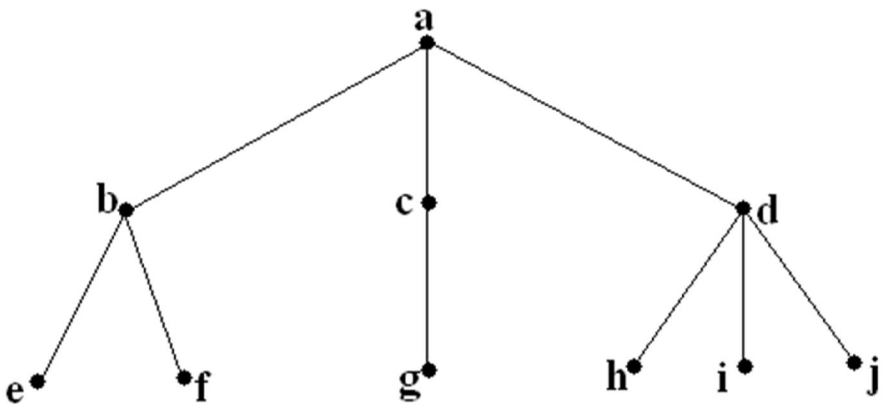


– Duyệt cây theo thuật toán Preorder search : a, b, d, i, e, j, k, f, c, g, m, n, p, h, q, r.

– Duyệt cây theo thuật toán Postorder search : i, d, j, k, e, f, b, m, n, p, g, q, r, h, c, a.

3.6/ BIỂU DIỄN CÂY BẰNG DANH SÁCH CÁC ĐỈNH CON:

Cho cây T như sau:



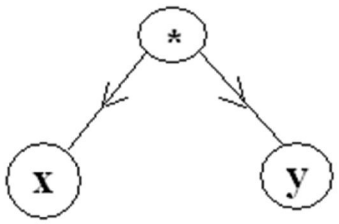
Chọn đỉnh a làm gốc của T để T là một cây có gốc.

Cây T được biểu diễn bằng danh sách các đỉnh con dưới đây:

Đỉnh	**a**	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**i**	**j**
Đỉnh con	b	e	g	h	∅	∅	∅	∅	∅	∅
Đỉnh con	c	f		i						
Đỉnh con	d			j						

3.7/ CÂY NHỊ PHÂN CỦA BIỂU THỨC:

a) Cho các biến số thực x và y cùng với phép toán đại số * (* có thể là một trong các phép toán cộng, trừ, nhân, chia, lũy thừa, …). Ta dùng một cây nhị phân đầy đủ để biểu diễn phép tính x*y như sau:



x và y gọi là *hai thành tố* của phép toán *. Chúng cũng ở vị trí hai lá của cây.

Ta viết * x y (theo phép duyệt Preorder search), x * y (theo phép duyệt Inorder search) hoặc x y * (theo phép duyệt Postorder search).

b) Cây nhị phân của một biểu thức đại số là một cây nhị phân đầy đủ trong đó

– Mỗi biến số được biểu diễn bằng *một lá*.

– Mỗi phép toán được thể hiện tại *một đỉnh trong* của cây nhị phân với *hai thành tố* là hai giá trị của hai cây con bên trái và bên phải của đỉnh ấy.

c) Khi duyệt cây nhị phân của một biểu thức đại số, kết quả thu được sẽ gọi là

– Trung tố (nếu dùng phép duyệt Inorder search).

– Tiền tố (nếu dùng phép duyệt Preorder search).

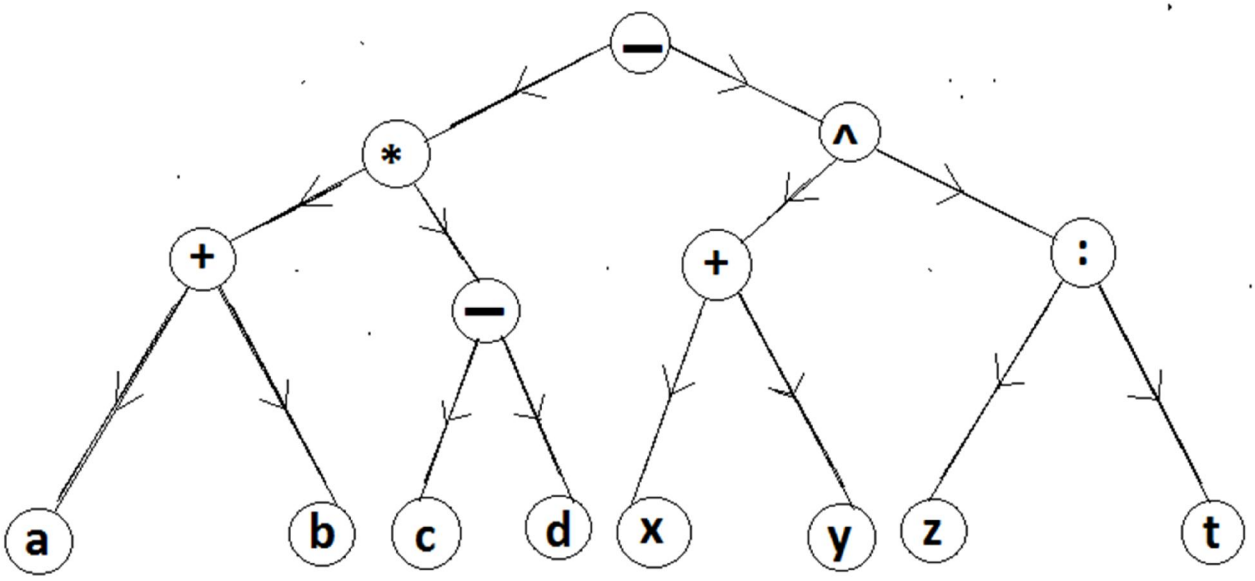
– Hậu tố (nếu dùng phép duyệt Postorder search).

d) Tiền tố còn được gọi là *ký pháp Ba Lan* (Polish Notation) của biểu thức đại số và viết tắt là PN.

Hậu tố còn được gọi là *ký pháp Ba Lan nghịch đảo* (Reverse Polish Notation) của biểu thức đại số và viết tắt là RPN.

Ví dụ:

a) Cho biểu thức đại số A = (a + b)(c – d) – $(x+y)^{\frac{z}{t}}$ trong đó a, b, c, d, x, y, z và t là các số nguyên dương. Cây nhị phân sau đây thể hiện cách tính giá trị của biểu thức A.



A = a + b * c – d – x + y ∧ z : t (trung tố).

A = – * + a b – c d ^ + x y ÷ z t (tiền tố = dạng thức ký pháp Ba Lan).

A = a b + c d – * x y + z t ÷ ^ – (hậu tố = dạng thức ký pháp Ba Lan nghịch đảo).

b) Viết các dạng thức trung tố, tiền tố và hậu tố và vẽ cây nhị phân biểu diễn cho các biểu thức đại số $B = (x + 2)^3(y – \frac{x}{4}) – 5$ và $C = (4 + 1)^{2 \times 2} – (9 – 3)\frac{48}{6}$.

Ta có B = x + 2 ∧ 3 * y – x : 4 – 5 (trung tố).

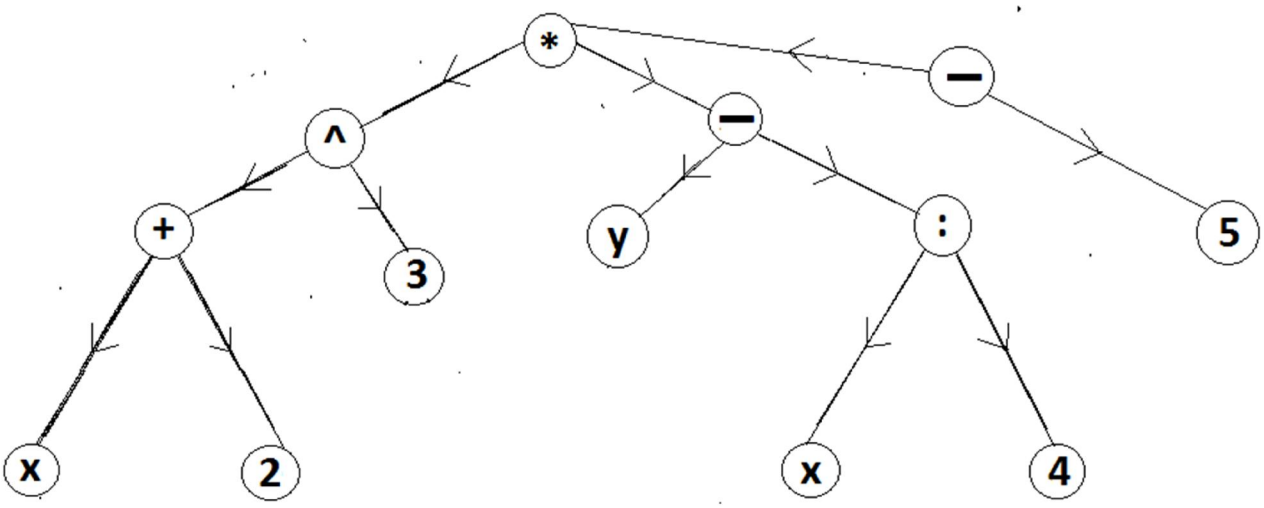
B = – * ^ + x 2 3 – y ÷ x 4 5 (tiền tố).

B = x 2 + 3 ^ y x 4 ÷ – * 5 – (hậu tố).

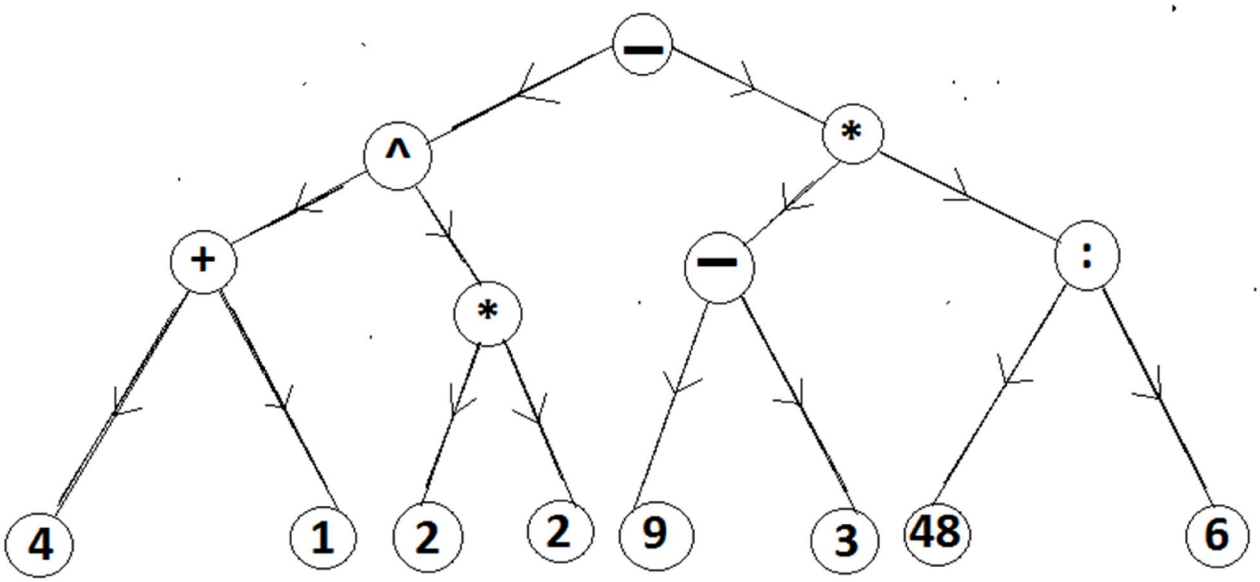
C = 4 + 1 ∧ 2* 2 – 9 – 3 * 48 : 6 (trung tố).

C = – ∧ + 4 1 * 2 2 * – 9 3 ÷ 48 6 (tiền tố).

C = 4 1 + 2 2 * ∧ 9 3 – 48 6 : * – (hậu tố).



Cây nhị phân biểu diễn B.



Cây nhị phân biểu diễn C.

Ta có $C = 5^4 - (6 \times 8) = 625 - 48 = 577$.

3.8/ <u>GHI CHÚ:</u> Các dạng thức trung tố, ký pháp Ba Lan và ký pháp Ba Lan nghịch đảo của một biểu thức giúp ta biểu diễn và tính toán được biểu thức một cách chính xác mà không cần phải dùng thêm các dấu ngoặc (đơn, vuông, móc).
