BÀI 5

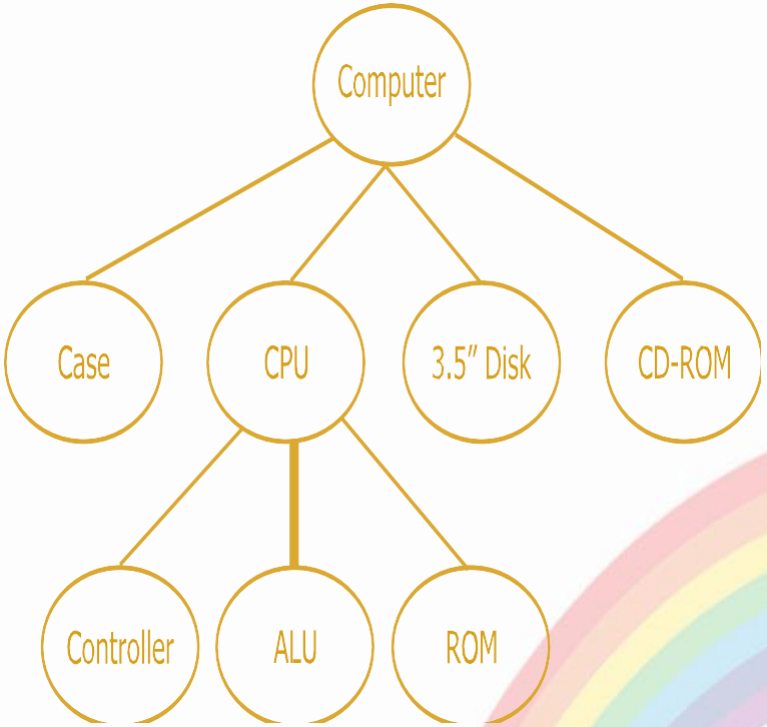
CẤU TRÚC CÂY

MỤC TIÊU

- Mô tả các khái niệm về cây, cây nhị phân, cây tổng quát một cách chính xác;
- Trình bày các cách cài đặt về cây tổng quát, cây nhị phân và thực hiện cài đặt các thao tác trên cây nhị phân một cách chính xác bằng một ngôn ngữ lập trình C;
- Xác định đúng một số ứng dụng của cây nhị phân và cây trò chơi;
- Sử dụng cấu trúc dữ liệu dạng cây cho phù hợp để giải quyết một số bài toán.

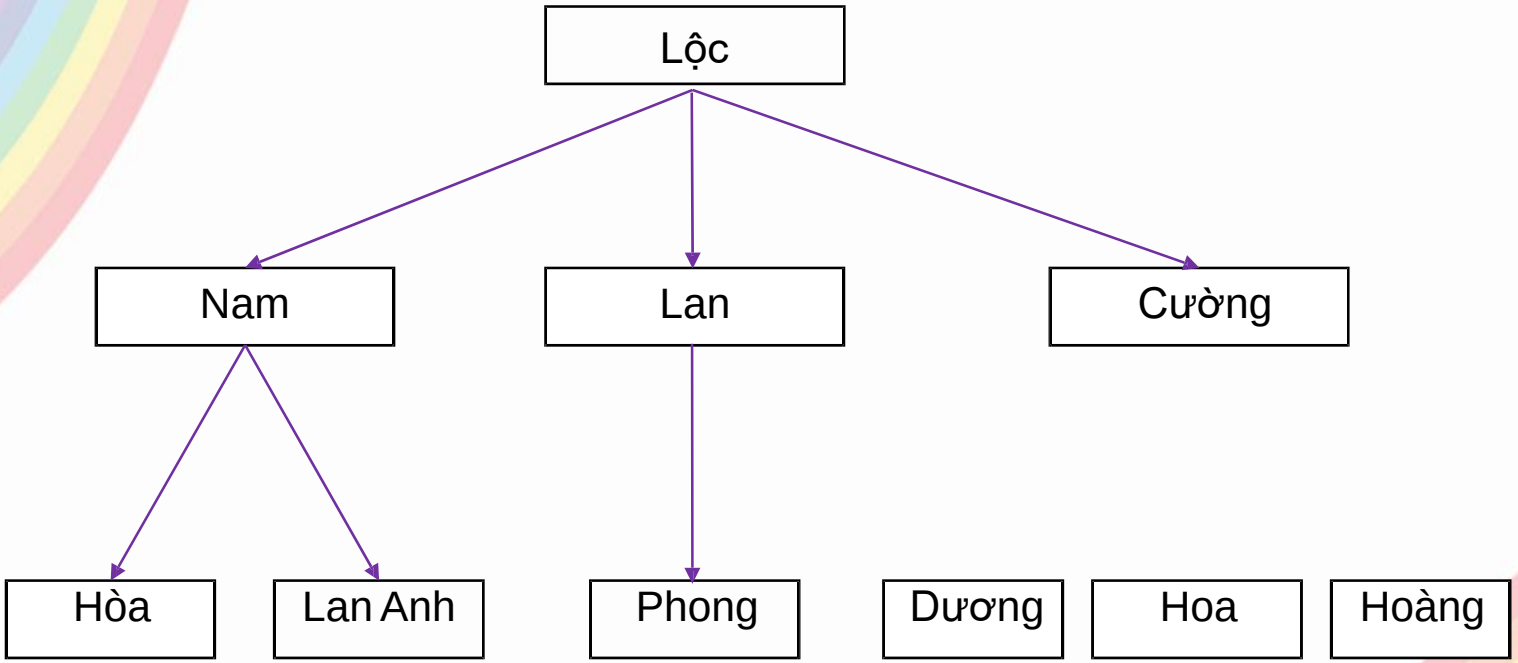
NỘI DUNG

- Các khái niệm cơ bản về cây;
- Cây nhị phân;
- Ứng dụng của cây nhị phân;
- Cây tổng quát;
- Cây biểu diễn trò chơi.



1. CÁC KHÁI NIỆM VỀ CÂY

1. CÁC KHÁI NIỆM VỀ CÂY



Cây là một cấu trúc dữ liệu trừu tượng gồm một tập hữu hạn các phần tử gọi là nút (hay đỉnh), giữa các nút có một quan hệ phân cấp gọi là quan hệ "cha – con" và một tập hợp hữu hạn những cạnh nối các cặp nút cha – con với nhau. Nếu cây không rỗng, có một nút gọi là nút gốc (root).

1. CÁC KHÁI NIỆM VỀ CÂY

Ví dụ 5.1. Xét một cây có dạng như sau:

Tập hợp T gồm 11 phần tử, T = {a, b, c, d, e, f, g, h, i, j, k}.

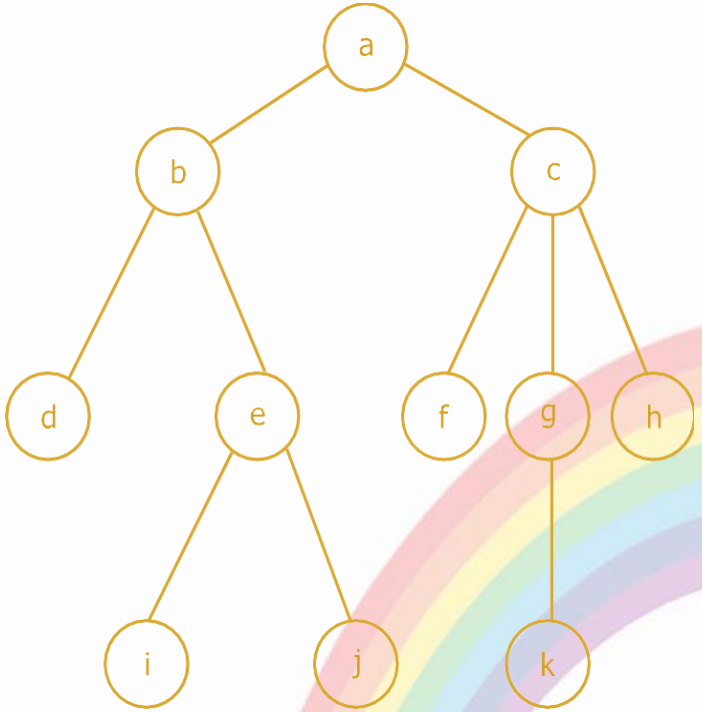
Các phần tử của T được gọi là các đỉnh của cây T;

Tập T có cấu trúc như sau: Các đỉnh của T được phân thành các lớp không cắt nhau:

Lớp thứ nhất gồm một đỉnh duy nhất a, đỉnh này gọi là **gốc** của cây; Lớp thứ hai gồm các đỉnh b, c;

Lớp thứ ba gồm các đỉnh d, e, f, g, h;

Lớp cuối cùng gồm các đỉnh i, j, k, mỗi đỉnh thuộc một lớp (trừ gốc), có một cung duy nhất nối với một đỉnh nào đó thuộc lớp kề trên.

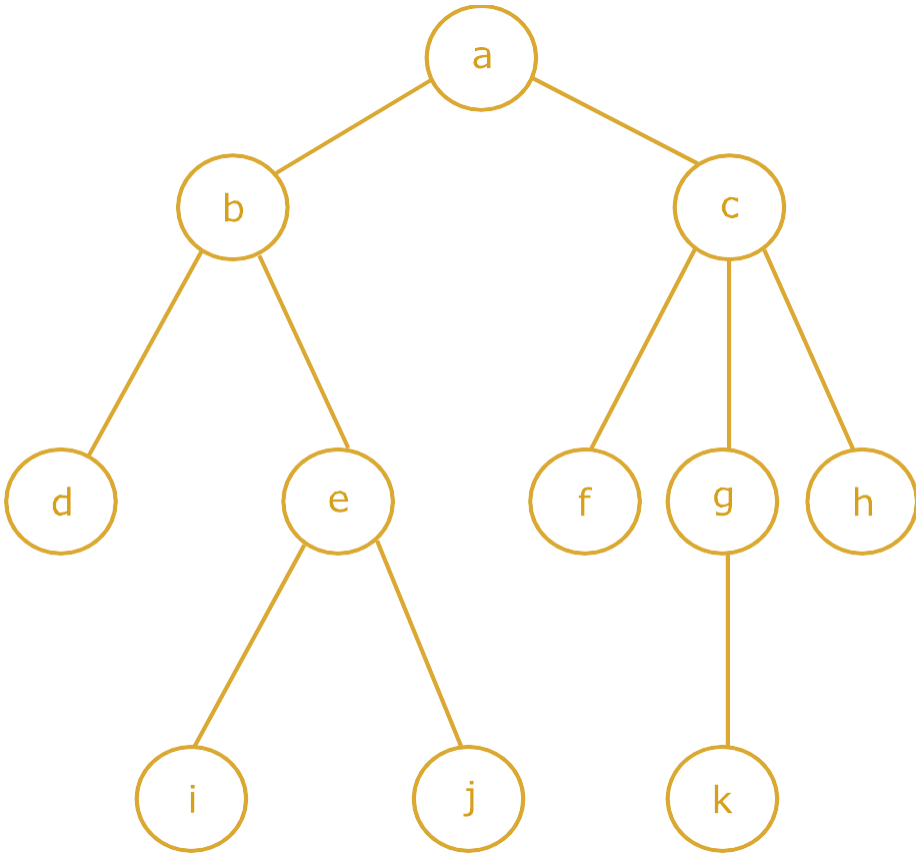


1. CÁC KHÁI NIỆM VỀ CÂY

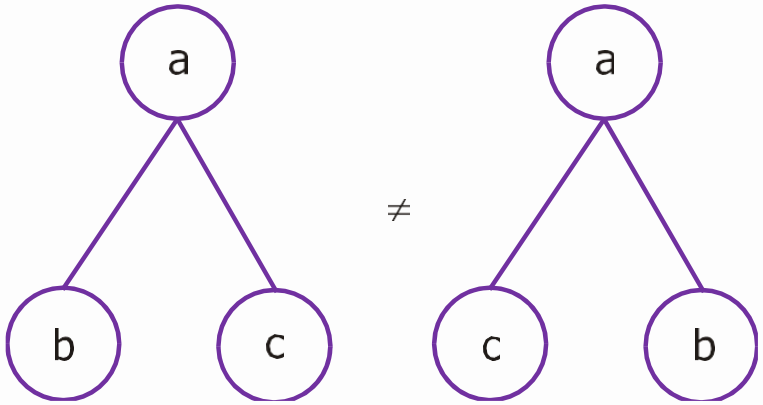
Một số khái niệm cơ bản về cây:

- Mỗi đỉnh của cây là gốc của các cây con của nó;
- Số các cây con của một đỉnh gọi là bậc của đỉnh đó;
- Các đỉnh có bậc bằng không được gọi là lá của cây;
- Các đỉnh có ít nhất một con được gọi là đỉnh trong;
- Các đỉnh có cùng một cha được gọi là anh em;
- Một dãy các đỉnh a_1, a_2, ... a_n ($n \geq 1$), sao cho a_i ($i = 1, 2, ..., n-1$) là cha của a_{i+1} được gọi là đường đi từ a_1 đến a_n;
- Chiều dài đường đi này là số cành trên nó đó là $n - 1$. Ta có nhận xét rằng, luôn luôn tồn tại một đường đi duy nhất từ gốc tới một đỉnh bất kỳ trong cây. Mỗi nút có đường đi chiều dài bằng 0 đến chính nó.

1. CÁC KHÁI NIỆM VỀ CÂY



1. CÁC KHÁI NIỆM VỀ CÂY



Hình 5.2: Hai cây được sắp khác nhau

- Trong một cây, độ cao của một đỉnh a là độ dài của đường đi dài nhất từ a đến các lá của nó. Độ cao của gốc được gọi là độ cao của cây;
- Mức (độ sâu) của đỉnh a là độ dài của đường đi từ gốc đến a. Như vậy gốc có mức 0;
- Cây được sắp: Trong một cây, nếu các cây con của mỗi đỉnh được sắp theo một thứ tự nhất định, thì cây được gọi là cây được sắp.

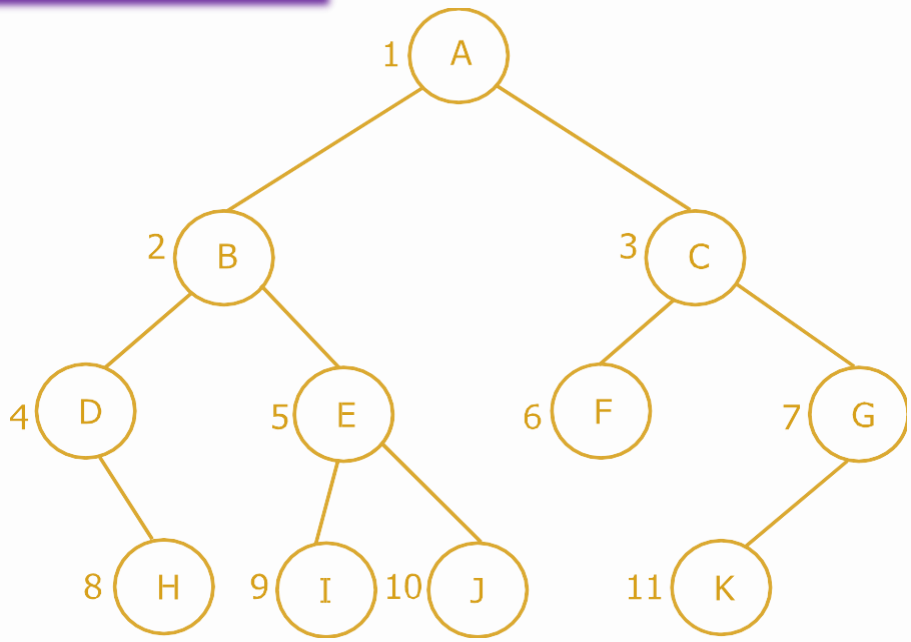
1. CÁC KHÁI NIỆM VỀ CÂY

- Cây gắn nhãn: Là cây mà mỗi đỉnh của nó được gắn với một giá trị (nhãn) nào đó;
- Rừng: Một rừng F là một danh sách các cây:

$F = (T_1, T_2, ..., T_n)$ trong đó $T_i (i = 1, ..., n)$ là cây (cây được sắp).

2. CÂY NHỊ PHÂN

2. CÂY NHỊ PHÂN



Hình 5.3. Một cây nhị phân

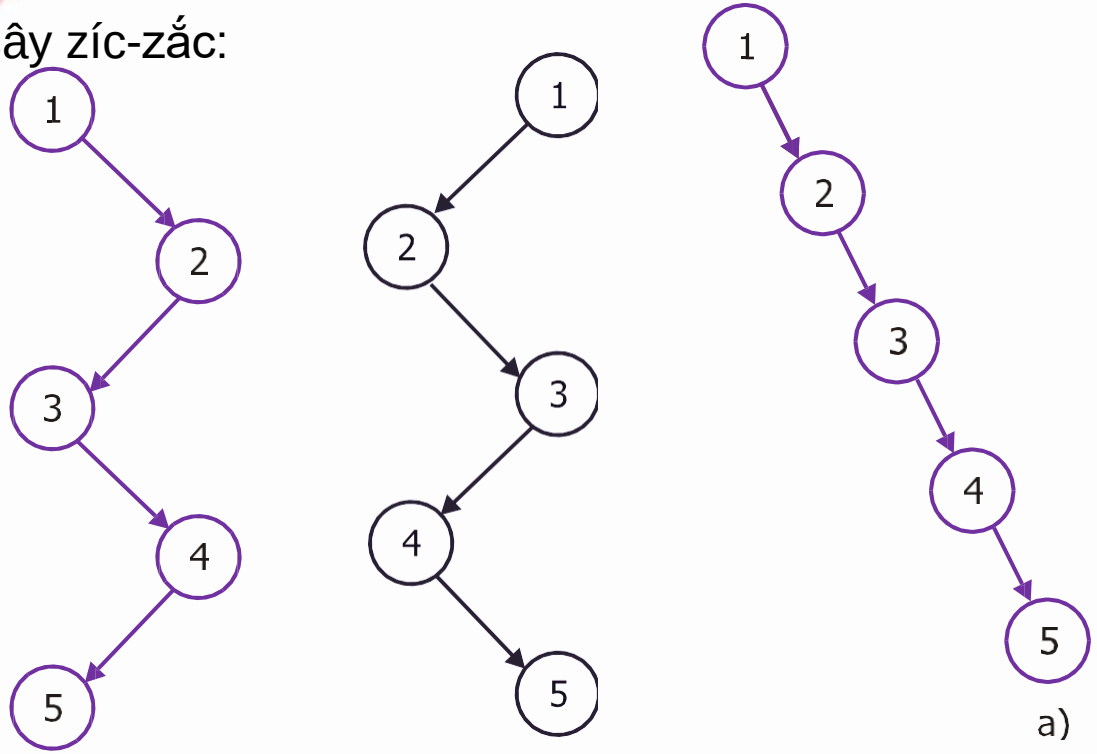
Khái niệm: Cây nhị phân là một cây mà trong đó mỗi nút có nhiều nhất hai con.

- Mỗi đỉnh của cây nhị phân chỉ có nhiều nhất là hai đỉnh con.
- Một đỉnh con bên trái (đó là gốc của cây con trái) và một đỉnh con bên phải (đó là gốc của cây con phải).

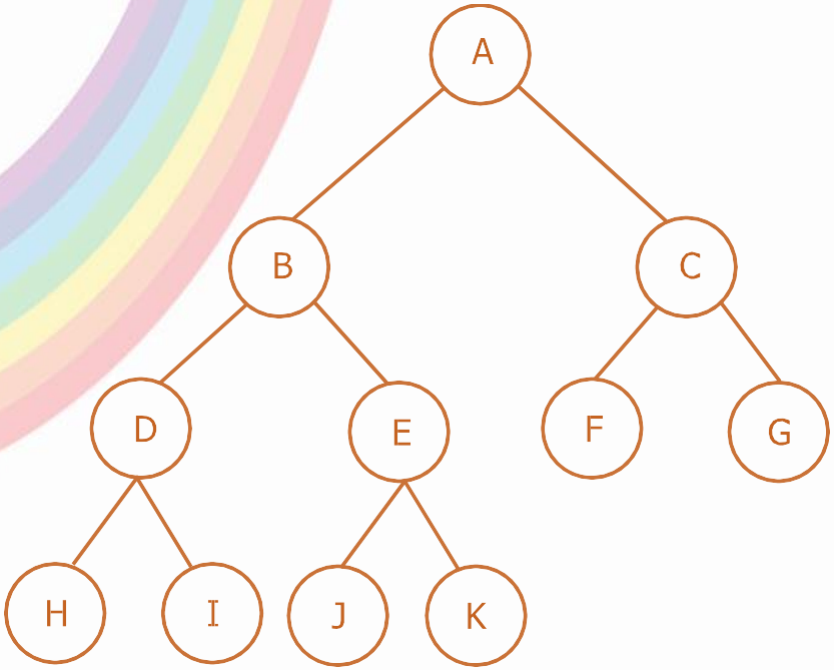
2. CÂY NHỊ PHÂN

Một số dạng đặc biệt của cây nhị phân:

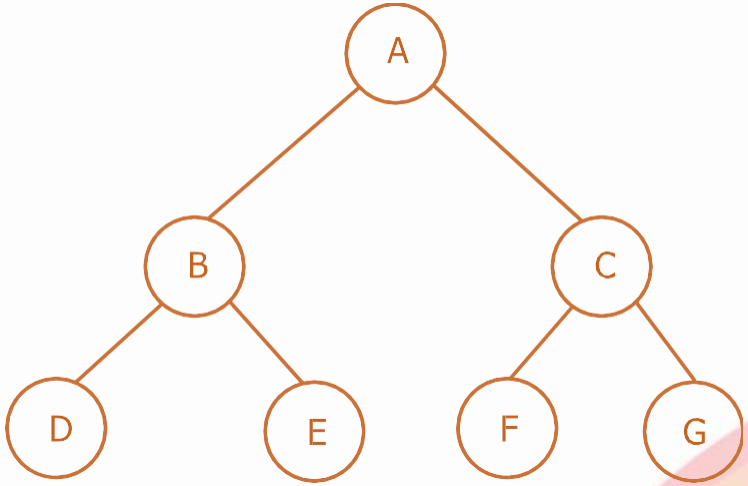
- Cây nhị phân hoàn chỉnh (Complete binary tree) là cây nhị phân mà mọi nút có mức < h – 1 đều có đúng 2 nút con với h là chiều cao của cây (Hay nói cách khác các nút trong đều có 2 con).
- Cây nhị phân lệch phải:
- Cây nhị phân lệch trái:
- Cây zíc-zắc:



2. CÂY NHỊ PHÂN



a)



b)

Cây nhị phân đầy đủ (Full binary tree): Cây nhị phân mà mọi nút có mức <= h – 1 đều có đúng 2 nút con với h là chiều cao của cây (Hay nói cách khác cây nhị phân đầy đủ là cây nhị phân hoàn chỉnh mà tất cả các cây con của nút gốc đều có độ cao như nhau).

2.1. CÀI ĐẶT CÂY NHỊ PHÂN BẰNG MẢNG

- Sử dụng một mảng để lưu giữ các đỉnh của cây nhị phân.
- Cấu trúc dữ liệu biểu diễn cây nhị phân được khai báo như sau:

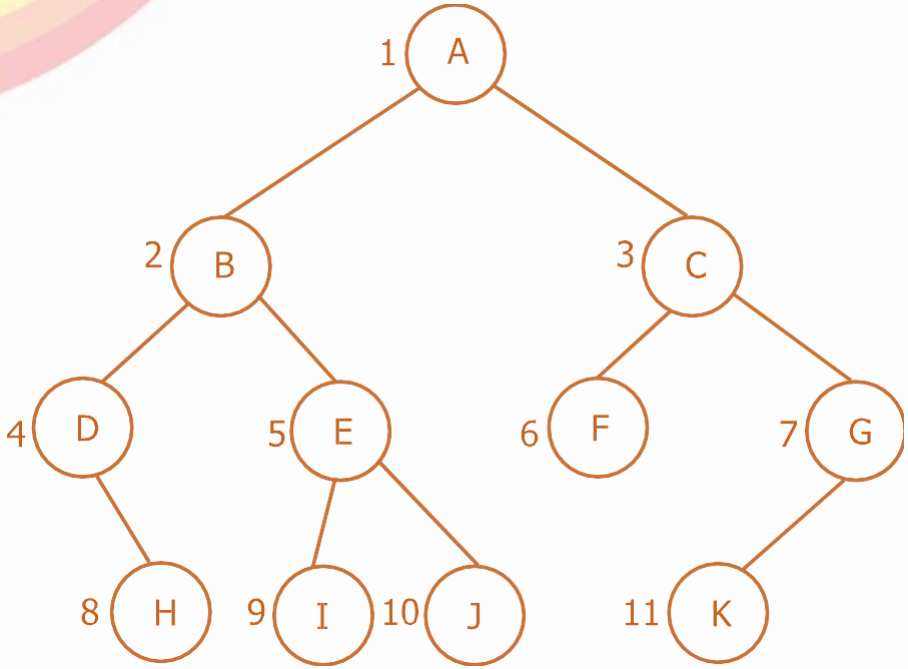
```
#define max N;
  typedef <ten_kieu_du_lieu> Item;
  typedef struct Node
  {
        Item infor;
        int left;
        int right;
  };
Node Tree[N];
```

2.1. CÀI ĐẶT CÂY NHỊ PHÂN BẰNG MẢNG

Được minh hoạ cấu trúc dữ liệu biểu diễn cây nhị phân bằng mảng:

Ví dụ:

Với cây nhị phân dưới đây:



	infor	left	right
1	A	2	3
2	B	4	5
3	C	6	7
4	D	0	8
5	E	9	10
6	F	0	0
7	G	11	0
8	H	0	0
9	I	0	0
10	J	0	0
11	K	0	0

2.2. CÀI ĐẶT CÂY NHỊ PHÂN CON TRỎ

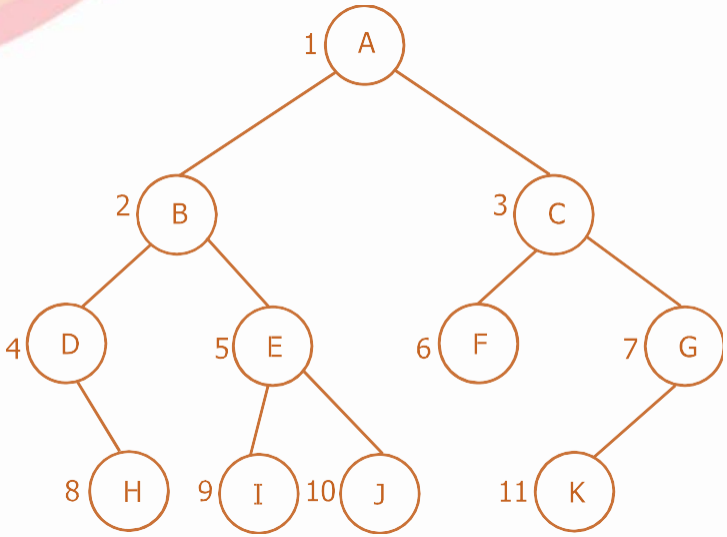
- Mỗi bản ghi biểu diễn một đỉnh của cây chứa hai con trỏ: con trỏ left trỏ tới đỉnh con trái, con trỏ right trỏ tới đỉnh con phải;
- Ta có khai báo sau đây cho cây nhị phân:

```
typedef <kieu_du_lieu> ElementType;
//cấu trúc một nút
struct Node
    {
        ElementType infor;
        struct TreeNode *left;
        struct TreeNode *right;
};
typedef struct Node *Tree;//định nghĩa một cây
Tree Root;//con trỏ trỏ tới nút gốc của cây
```

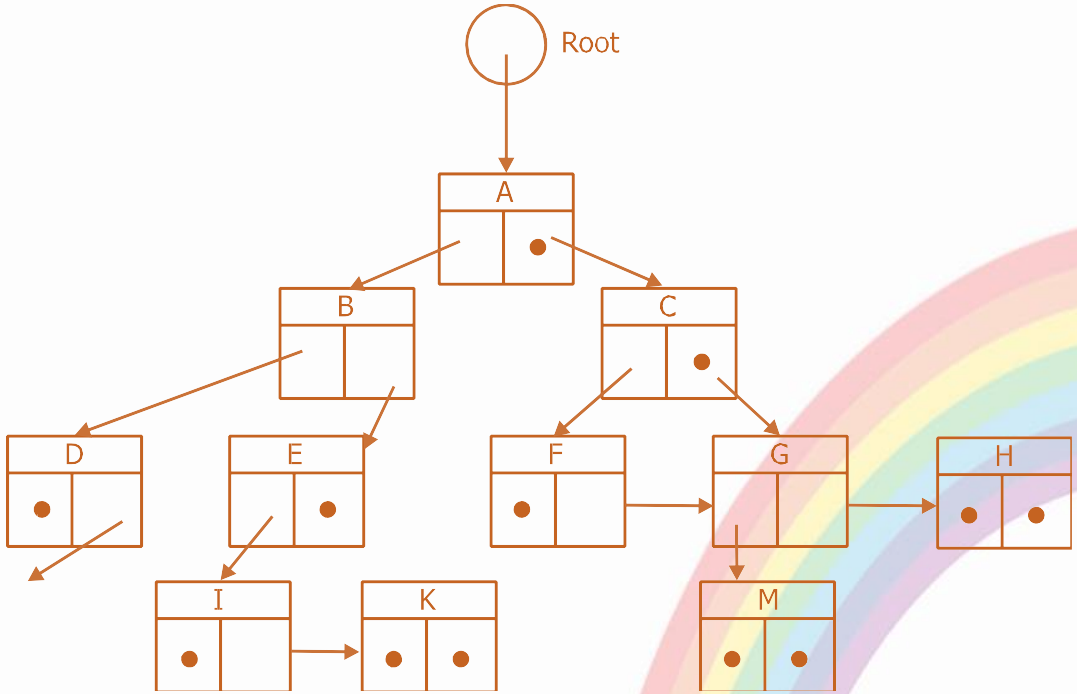
2.2. CÀI ĐẶT CÂY NHỊ PHÂN CON TRỎ

Ví dụ:

Với cây nhị phân dưới đây:



Được minh hoạ cấu trúc dữ liệu biểu diễn cây nhị phân con trỏ:



2.3. CÁC THAO TÁC TRÊN CÂY NHỊ PHÂN

- Khởi tạo cây nhị phân:

```
void BTree_Init(Tree *Root)
{
    (*Root)=NULL;
}
```

- Tạo một nút mới:

```
Node *CreateNode(ElementType NewData )
{
    Node *p;  p=(Node*)malloc(sizeof(struct Node));
    if(p!=NULL)
    {       p->left=NULL;
            p->right=NULL;
            p->Data=NewData;          }
return p;
}
```

2.3. CÁC THAO TÁC TRÊN CÂY NHỊ PHÂN

Duyệt các nút trên cây nhị phân:

- Phép duyệt cây là phép duyệt các nút trên cây một cách hệ thống sao cho mỗi nút chỉ được thăm một lần;
- Duyệt theo thứ tự trước: nút gốc sẽ được duyệt trước sau mới đến duyệt 2 nút gốc cây con:

```
void BTreeTravelling(N)
//duyệt nhánh nhận N làm nút gốc
{
    if (N!=NULL)
    {
        Thủ tục xử lý thông tin của nút N;
        BTreeTravelling(Nút bên trái của N);
        BTreeTravelling(Nút bên phải của N);
    }
}
```

2.3. CÁC THAO TÁC TRÊN CÂY NHỊ PHÂN

- Duyệt theo thứ tự nút gốc giữa: duyệt một trong hai cây con trước rồi duyệt nút gốc và sau đó duyệt cây con còn lại. Có hai cách duyệt theo thứ tự nút gốc giữa:
  - Duyệt cây con trái, duyệt nút gốc, duyệt cây con phải;
  - Duyệt cây con phải, duyệt nút gốc, duyệt cây con trái.
- Duyệt thứ tự nút gốc sau: theo cách duyệt này thì nút gốc sẽ được duyệt sau cùng so với duyệt hai nút cây con. Có hai cách duyệt thứ tự nút gốc sau:
  - Duyệt cây con trái, duyệt cây con phải, duyệt nút gốc;
  - Duyệt cây con phải, duyệt cây con trái, duyệt nút gốc.

2.3. CÁC THAO TÁC TRÊN CÂY NHỊ PHÂN

Hủy một nút trên cây nhị phân:

Việc hủy một nút trong cây có thể làm cho cây thành rừng. Vì vậy khi tiến hành hủy một nút nếu nút đó là lá thì không có điều gì xẩy ra song nếu hủy một nút không phải là lá thì chúng ta phải tìm cách chuyển các nút gốc cây con của nút cần hủy thành các nút gốc cây con của các nút khác rồi mới tiến hành hủy nút này.

3. ỨNG DỤNG CỦA CÂY NHỊ PHÂN

3. ỨNG DỤNG CỦA CÂY NHỊ PHÂN

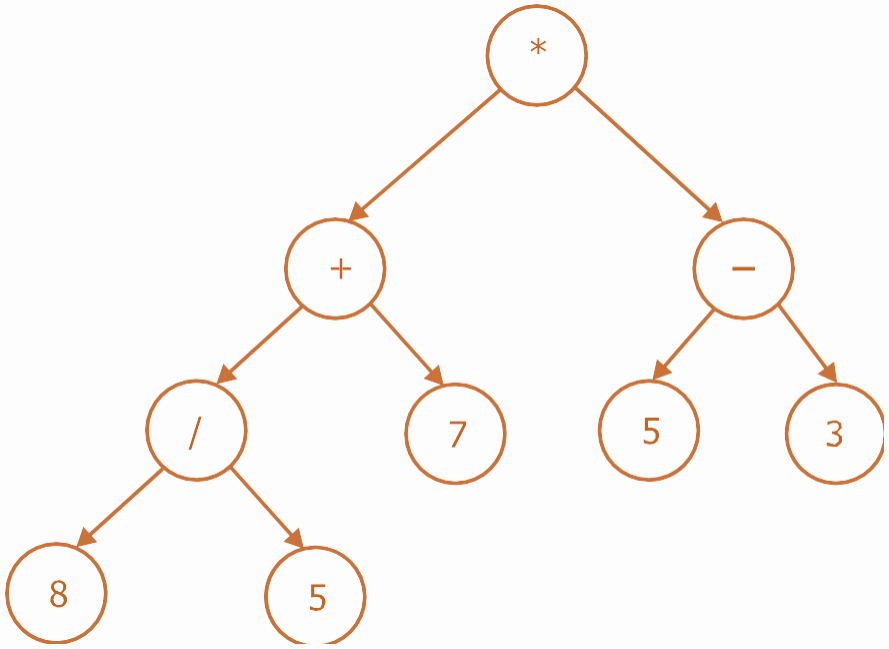
Cây biểu thức

Cây biểu thức là cây nhị phân gắn nhãn, biểu diễn cấu trúc của một biểu thức (số học hoặc logic). Trong đó:

- Các nút lá biểu thị các toán hạng (hằng hoặc các biến);
- Các nút không phải là lá biểu thị các toán tử (các phép toán số học);
- Mỗi phép toán trong một nút sẽ tác động lên hai biểu thức con nằm ở cây con bên trái và cây con bên phải của nút đó.

3. ỨNG DỤNG CỦA CÂY NHỊ PHÂN

Ví dụ 5.4. Xét cây nhị phân biểu diễn biểu thức (8/5 + 7)*(5 – 3)



3. ỨNG DỤNG CỦA CÂY NHỊ PHÂN

Để tính toán biểu thức, dùng một ngăn xếp Stack để lưu các kết quả trung gian, ta có bảng sau: (Duyệt theo thứ tự sau) 8 5 / 7 + 5 3 - *

Đọc	Xử lý	Stack
8	Đẩy vào Stack	8
5	Đẩy vào Stack	8,5
/	Lấy8 và 5 ra khỏi Stack tính được 8/5 = 1.6, đẩy 1.6 vào Stack	1.6
7	Đẩy vào Stack	1.6, 7
+	Lấy 1.6 và 7 ra khỏi Stack tính được 1.6 + 7 = 8.6, đẩy 8.6 vào Stack	8.6
5	Đẩy vào Stack	8.6, 5
3	Đẩy vào Stack	8.6, 5, 3
–	Lấy3 và 5 ra khỏi Stack tính được 5 – 3 = 2, đẩy 2 vào Stack	8.6, 2
*	Lấy2 và 8.6 ra khỏi Stack tính được 8.6 * 2 = 17.2, đẩy 17.2 vào Stack	17.2

3. ỨNG DỤNG CỦA CÂY NHỊ PHÂN

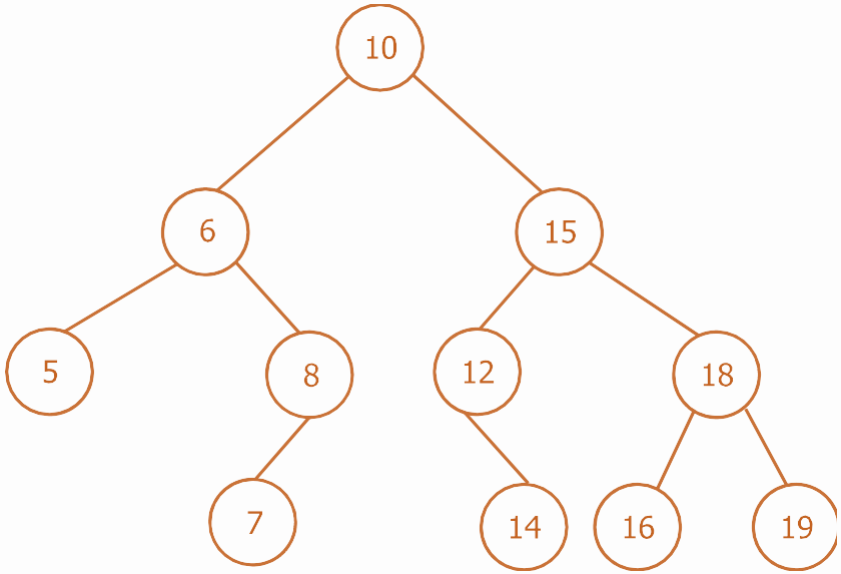
Cây nhị phân tìm kiếm:

Cây tìm kiếm nhị phân là cây nhị phân hoặc trống, hoặc thoả mãn các điều kiện sau:

- Khoá của các đỉnh thuộc cây con trái nhỏ hơn khoá của gốc;
- Khoá của gốc nhỏ hơn khoá của các đỉnh thuộc cây con phải của gốc;
- Cây con trái và cây con phải của gốc cũng là cây tìm kiếm nhị phân.

3. ỨNG DỤNG CỦA CÂY NHỊ PHÂN

Hình 5.11. Biểu diễn một cây tìm kiếm nhị phân, trong đó khoá của các đỉnh là các số nguyên.



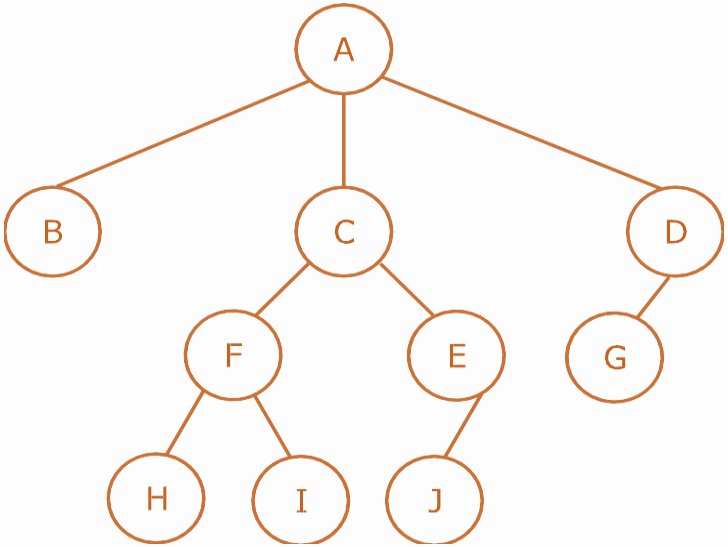
Hình 5.11. Một cây tìm kiếm nhị phân

Duyệt cây tìm kiếm nhị phân ở hình 5.11 theo thứ tự nút gốc giữa ta được dãy số sắp xếp theo thứ tự tăng dần: 5, 6, 7, 8, 10, 12, 14, 15, 16, 18, 19.

4. CÂY TỔNG QUÁT

4. CÂY TỔNG QUÁT

Cấu trúc cây mà trong đó mỗi nút có nhiều hơn hai cây con gọi là cây tổng quát.



Các phép toán trên cây tổng quát:

Ngoài các phép toán (thao tác) như cây nhị phân, cây tổng quát còn có một số phép toán cơ bản sau:

- Tìm nút cha của nút bất kỳ trên cây.;
- Tìm nút con trái nhất của một nút;
- Tìm các nút là anh em ruột của nút n của cây.

4.1. BIỂU DIỄN CÂY BẰNG DANH SÁCH CÁC CON CỦA MỖI ĐỈNH

- Với mỗi đỉnh của cây ta thành lập một danh sách các đỉnh con của nó theo thứ tự từ trái sang phải.
- Cài đặt bởi mảng khai báo cấu trúc dữ liệu biểu diễn cây như sau:

```
#define N…//N là số lớn nhất các đỉnh mà cây có thể có
{typedef <kiểu_dữ_liệu của đỉnh> Item;
 typedef struct Member
        int id;//id chỉ danh của đỉnh
};      Member *Next;//Next:trỏ tới em liền kề
typedef struct Node
{
        Item infor;
        Member *child;//con trỏ trỏ tới con đầu tiên trong danh sách các
    con của nó.
  };
typedef Node Tree[N];// cây là một mảng
Tree T;
```

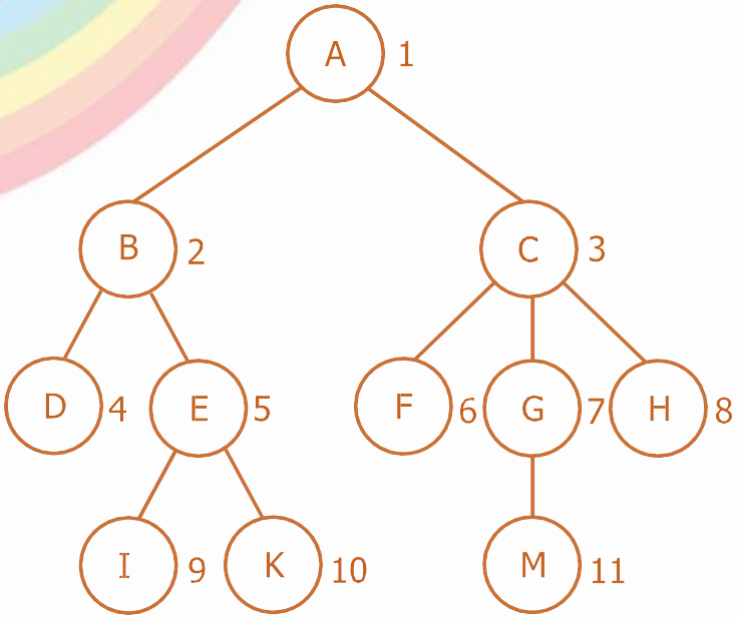
4.1. BIỂU DIỄN CÂY BẰNG DANH SÁCH CÁC CON CỦA MỖI ĐỈNH

Cài đặt bởi con trỏ:

```
#define K…//K là số tối đa các con của mỗi đỉnh
typedef <kiểu_dữ_liệu_của_đỉnh> Item;
typedef struct Node             //khai báo một đỉnh
{       Item info;//lưu thông tin của đỉnh
        Node *childs[K];//chứa các con của đỉnh
};
typedef struct Node *TreeNode;//khai báo một cây
TreeNode Root; //Root là con trỏ, trỏ tới gốc cây.
```

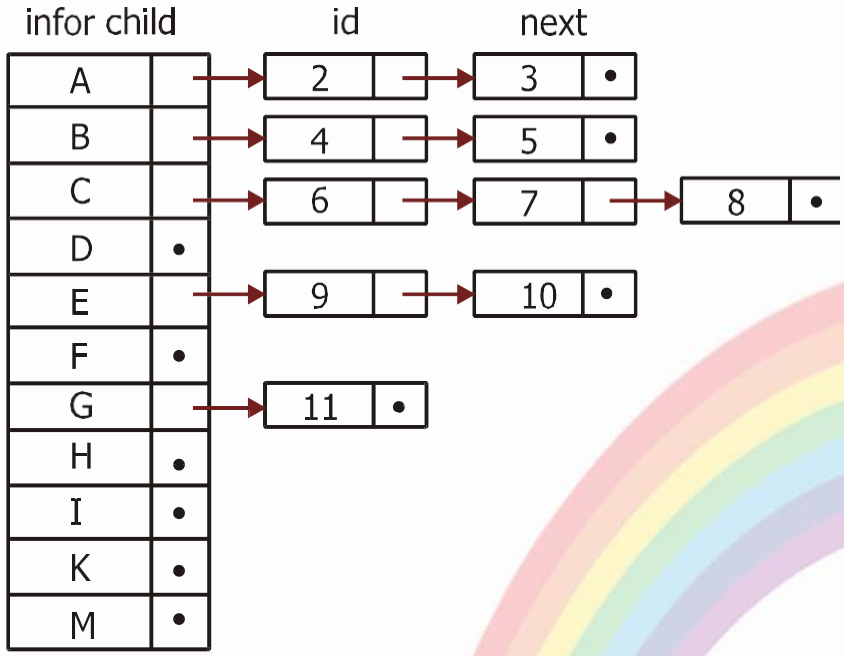
4.1. BIỂU DIỄN CÂY BẰNG DANH SÁCH CÁC CON CỦA MỖI ĐỈNH

Với cách cài đặt này, cấu trúc dữ liệu biểu diễn cây trong hình (a):



a)

Được minh hoạ trong hình (b).



b)

4.2. BIỂU DIỄN CÂY BẰNG CON TRƯỞNG VÀ EM LIỀN KỀ CỦA MỖI ĐỈNH

Cài đặt bởi con trỏ.

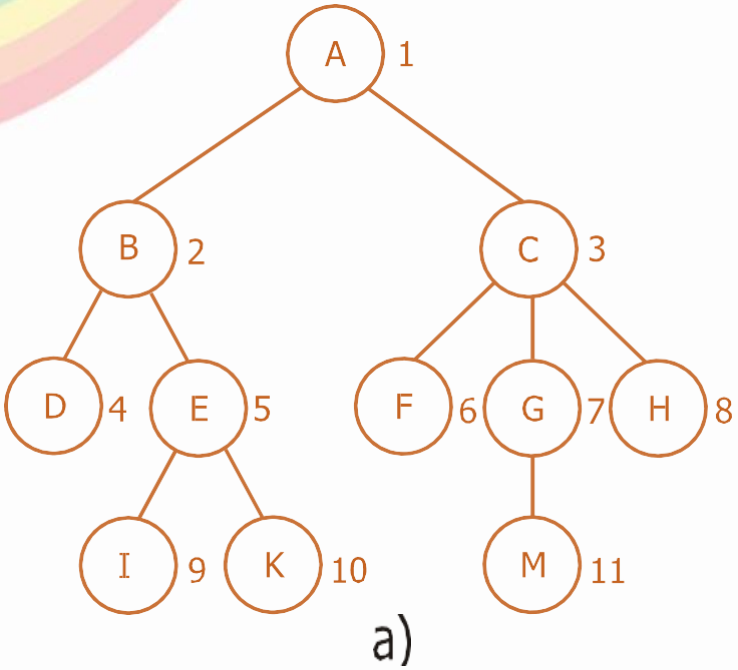
Cây sẽ được biểu diễn bởi cấu trúc sau:

```
typedef <kiểu_dữ_liệu> Item;  typedef
struct Node
    {Item infor;
     Node *EldestChild;//trỏ tới gốc cây con cả của đỉnh
     Node *NextSibling;//trỏ tới gốc em liền kề của đỉnh
    };
typedef struct Node *Tree;  Tree
Root;
```

4.2. BIỂU DIỄN CÂY BẰNG CON TRƯỞNG VÀ EM LIỀN KỀ CỦA MỖI ĐỈNH

Cài đặt bởi mảng.

Ta có thể khai báo như sau:



a)

Được minh hoạ trong hình sau:

	Infor	EldestChild	NextSibling
1	A	2	0
2	B	4	3
3	C	6	0
4	D	0	5
5	E	9	0
6	F	0	7
7	G	11	8
8	H	0	0
9	I	0	10
10	K	0	0
11	M	0	0

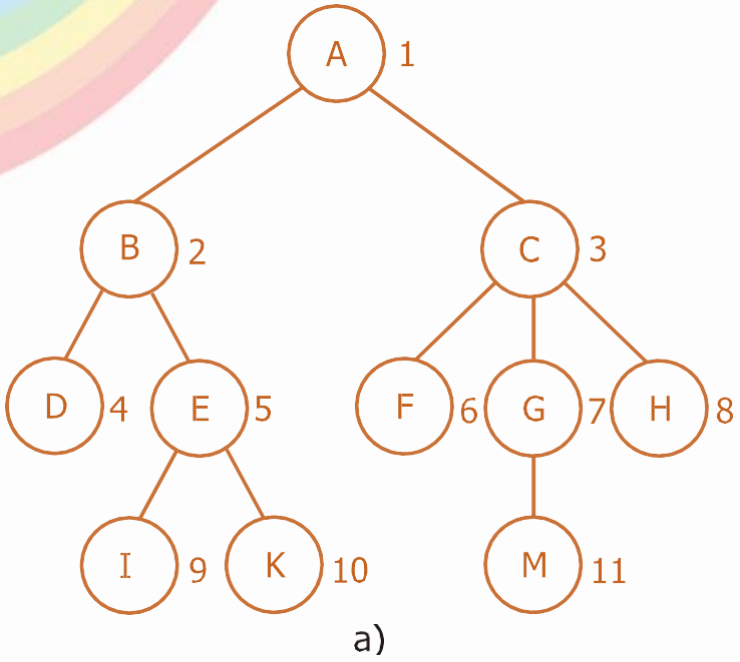
4.3. BIỂU DIỄN CÂY BỞI CHA CỦA MỖI ĐỈNH

Cây được biểu diễn bởi cấu trúc sau:

```
#define N…
typedef <kiểu_dữ_liệu_của_đỉnh> Item;  typedef
struct Node
        {           Item infor;  int
                    parent;
        } ;
```

4.3. BIỂU DIỄN CÂY BỞI CHA CỦA MỖI ĐỈNH

Với cấu trúc dữ liệu biểu diễn cây trong hình (a):



a)

Được minh hoạ trong hình sau:

	infor	Parent
1	A	0
2	B	1
3	C	1
4	D	2
5	E	2
6	F	3
7	G	3
8	H	3
9	I	5
10	K	5
11	M	7

5. CÂY BIỂU DIỄN TRÒ CHƠI

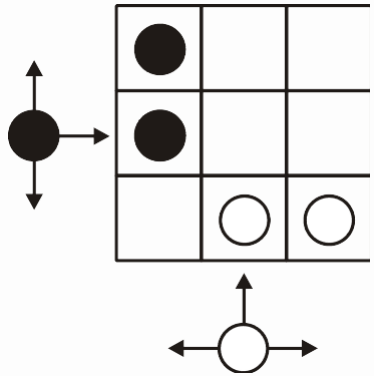
5. CÂY BIỂU DIỄN TRÒ CHƠI

Cây trò chơi:

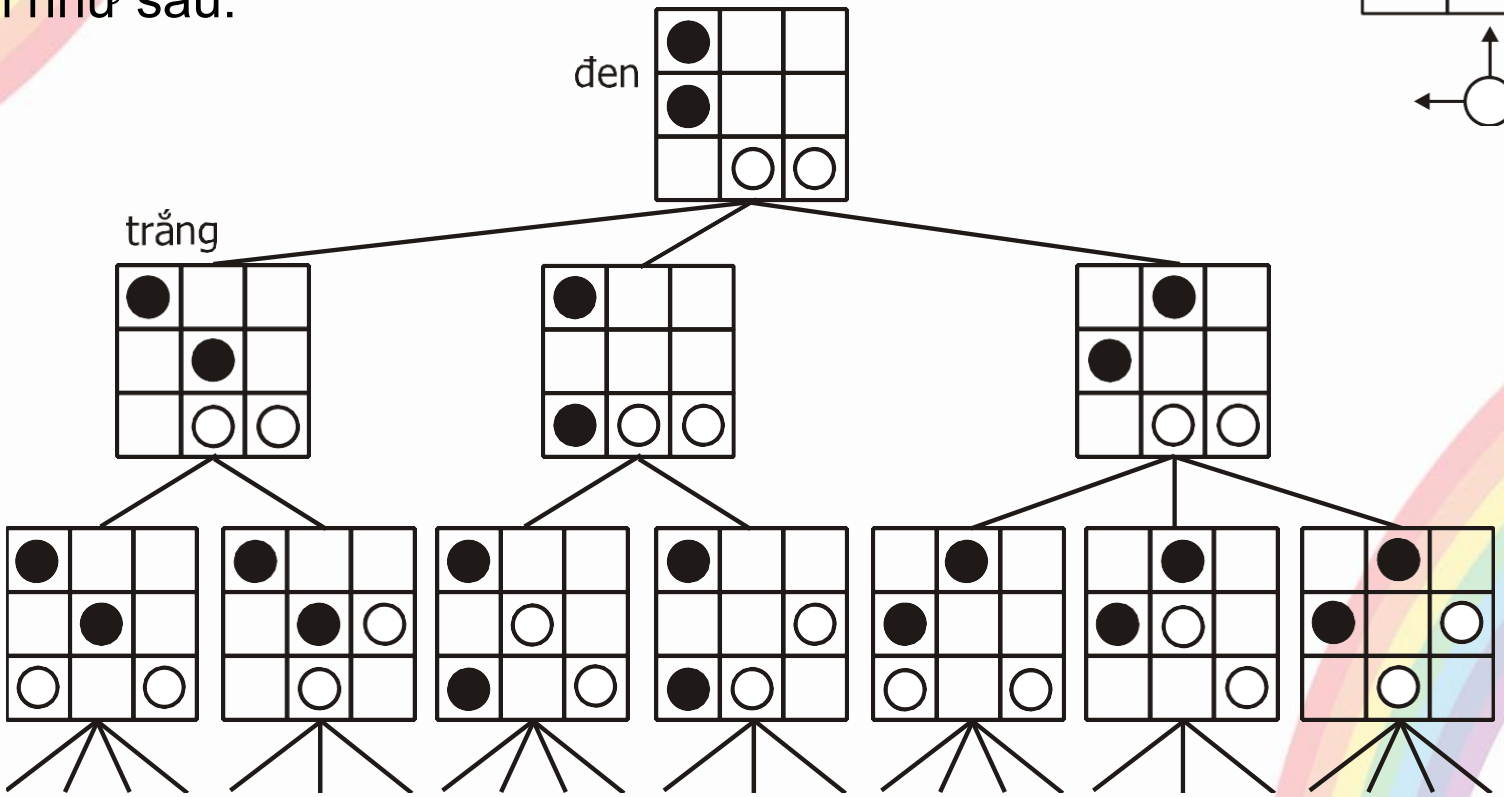
- Trong nghiên cứu các trò chơi có hai người tham gia (gọi là **Trắng – Đen)**;
- Để thuận lợi cho việc nghiên cứu các chiến lược chọn nước đi, chúng ta biểu diễn không gian trạng thái trên dưới dạng cây gọi là **cây trò chơi**;
- **Cây trò chơi** được xây dựng như sau:
  - Gốc của cây ứng với trạng thái ban đầu;
  - Nếu một đỉnh là Trắng (Đen) ứng với trạng thái u, thì các đỉnh con của nó là tất cả các đỉnh biểu diễn trạng thái v, v nhận được từ u do Trắng (Đen) thực hiện nước đi hợp lệ nào đó;
  - Do đó, trên cùng một mức của cây các đỉnh đều là Trắng hoặc đều là Đen, các lá của cây ứng với các trạng thái kết thúc trò chơi.

5. CÂY BIỂU DIỄN TRÒ CHƠI

Ví dụ 5.7. Xét trò chơi Dodgen (được tạo ra bởi Colin Vout).



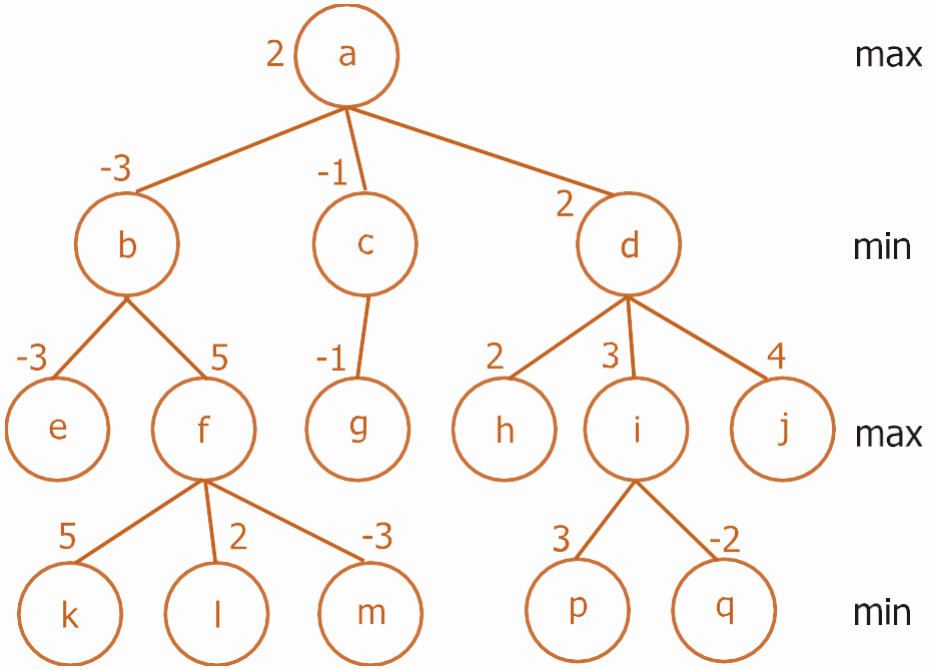
Giả sử Đen đi trước, ta có cây trò chơi được biểu diễn như sau:



5.1. PHƯƠNG PHÁP MINIMAX

- Giả sử Trắng cần tìm nước đi tại đỉnh u. Nước đi tối ưu cho Trắng là nước đi dẫn tới đỉnh con của v là đỉnh tốt nhất (cho Trắng) trong số các đỉnh con của u;
- Đối thủ chọn nước đi từ v, Đen cũng sẽ chọn nước đi tốt nhất cho anh ta;
- Đỉnh có giá trị càng lớn càng tốt cho Trắng;
- Đỉnh có giá trị càng nhỏ càng tốt cho Đen;
- Để xác định giá trị các đỉnh của cây trò chơi gốc u, ta đi từ mức thấp nhất lên gốc u.

5.1. PHƯƠNG PHÁP MINIMAX



Hình 5.20. Gán giá trị cho các đỉnh của cây trò chơi

Ví dụ 5.8. Xét cây trò chơi trong **hình 5.20**, gốc a là đỉnh Trắng. Giá trị của các đỉnh là số ghi cạnh mỗi đỉnh. Đỉnh i là Trắng, nên giá trị của nó là max(3, –2) = 3, đỉnh d là đỉnh Đen, nên giá trị của nó là min(2, 3, 4) = 2.

5.1. PHƯƠNG PHÁP MINIMAX

Việc gán giá trị cho các đỉnh được thực hiện bởi các hàm đệ quy MaxVal và MinVal.

- Hàm MaxVal xác định giá trị cho các đỉnh Trắng:

```
int MaxVal(u)
//xác định giá trị cho các đỉnh trắng
{
if(u là đỉnh kết thúc)MaxVal(u)=f(u);
else    MaxVal(u)=Max{MinVal(v)|v là đỉnh con
        của u}
};
```

- Hàm MinVal xác định giá trị cho các đỉnh Đen:

```
int MinVal(u)
//xác định giá trị cho các đỉnh đen
{
if(u là đỉnh kết thúc)MinVal(u)=f(u);
else    MinVal(u)=Min{MaxVal(v)|v là đỉnh
con của u}
};
```

5.1. PHƯƠNG PHÁP MINIMAX

- Trong các hàm đệ quy trên, f(u) là giá trị của hàm kết cuộc tại đỉnh kết thúc u;
- Trong thủ tục Minimax(u,v), v là biến lưu lại trạng thái mà Trắng đã chọn đi tới từ u;
- Thủ tục chọn nước đi như trên gọi là **phương pháp Minimax;**
- Thuật toán Minimax là thuật toán tìm kiếm theo độ sâu, ở đây ta đã cài đặt thuật toán Minimax bởi các hàm đệ quy.

```
void Minimax(u,v)
    {  val ←        -∞;
       for mỗi w là đỉnh con của u
       {            if (val <= MinVal(w))
                    {val ←     MinVal(w); v ←        w}
       }
    }
```

5.1. PHƯƠNG PHÁP MINIMAX

- Nếu cây có độ cao m, và tại mỗi đỉnh có b nước đi thì độ phức tạp về thời gian của thuật toán Minimax là O(bm);
- Để có thể tìm ra nhanh nước đi tốt (không phải là tối ưu) thay cho việc sử dụng hàm kết cuộc và xem xét tất cả các khả năng dẫn tới các trạng thái kết thúc, chúng ta sẽ sử dụng hàm đánh giá và chỉ xem xét một bộ phận của cây trò chơi.

5.1. PHƯƠNG PHÁP MINIMAX

Hàm đánh giá:

- Hàm đánh giá eval ứng với mỗi trạng thái u của trò chơi với một giá trị số eval(u):
  - Trạng thái u càng thuận lợi cho Trắng thì eval(u) là số dương càng lớn;
  - u càng thuận lợi cho Đen thì eval(u) là số âm càng nhỏ;
  - eval(u) ≈ 0 đối với trạng thái không lợi thế cho ai cả.
- Chất lượng của chương trình chơi cờ phụ thuộc rất nhiều vào hàm đánh giá;
- Hàm đánh giá chính xác sẽ đòi hỏi rất nhiều thời gian tính toán, mà người chơi lại bị giới hạn bởi thời gian phải đưa ra nước đi.

5.1. PHƯƠNG PHÁP MINIMAX

Phương pháp minimax với cơ chế cắt tỉa alpha-beta:

- Phương pháp cắt cụt alpha-beta cho phép ta cắt bỏ các nhánh không cần thiết cho sự đánh giá đỉnh u;
- Tư tưởng của kỹ thuật cắt cụt alpha-beta như sau:
  - Giả sử trong quá trình tìm kiếm ta đi xuống đỉnh a là đỉnh Trắng, đỉnh a có người anh em v đã được đánh giá;
  - Giả sử cha của đỉnh a là b và b có người anh em u đã được đánh giá, và giả sử cha của b là c;
  - Khi đó ta có giá trị đỉnh c (đỉnh Trắng) ít nhất là giá trị của u, giá trị của đỉnh b (đỉnh Đen) nhiều nhất là giá trị v;
  - Do đó, nếu eval(u) > eval(v), ta không cần đi xuống để đánh giá đỉnh a nữa mà vẫn không ảnh hưởng gì đến đánh giá đỉnh c. Hay nói cách khác ta có thể cắt bỏ cây con gốc a.

5.1. PHƯƠNG PHÁP MINIMAX

Để cài đặt kỹ thuật cắt cụt alpha-beta:

- Đối với các đỉnh nằm trên đường đi từ gốc tới đỉnh hiện thời, ta sử dụng tham số α để ghi lại giá trị lớn nhất trong các giá trị của các đỉnh con đã đánh giá của một đỉnh Trắng;
- Tham số β ghi lại giá trị nhỏ nhất trong các đỉnh con đã đánh giá của một đỉnh Đen;
- Giá trị của α và β sẽ được cập nhật trong quá trình tìm kiếm;

Υ α và β được sử dụng như các biến địa phương trong các hàm MaxVal(u, α, β) (hàm xác định giá trị của đỉnh Trắng u) và Minval(u, α, β) (hàm xác định giá trị của đỉnh Đen u).

5.1. PHƯƠNG PHÁP MINIMAX

Hàm MaxVal(u, α, β):

```
int MaxVal(u, α, β);
{   if (u là lá của cây hạn chế hoặc u là đỉnh kết thúc)
MaxVal ← eval(u)
    else
    {       for (mỗi đỉnh v là con của u)
                    { α   ←  max[α, MinVal(v, α, β)];
                     // Cắt bỏ các cây con từ các đỉnh v còn lại
                     if (α ≥ β) exit;
                    }
    }
    MaxVal ←    α;
}
```

5.1. PHƯƠNG PHÁP MINIMAX

Hàm Minval(u, α, β)

```
int MinVal(u, α, β);
{
    if (u là lá của cây hạn chế hoặc u là đỉnh kết thúc)
    MinVal ← eval(u);
    else for mỗi đỉnh v là con của u do
{
      β ← min[β, MaxVal(v, α, β)];
    // Cắt bỏ các cây con từ các đỉnh v còn lại
    if (α ≥ β) exit;
};
    MinVal ←     β;
}
```

5.1. PHƯƠNG PHÁP MINIMAX

Thuật toán tìm nước đi cho Trắng sử dụng kỹ thuật cắt cụt alpha-beta, được cài đặt bởi thủ tục Alpha_beta(u,v), trong đó v là tham biến ghi lại đỉnh mà Trắng cần đi tới từ u.

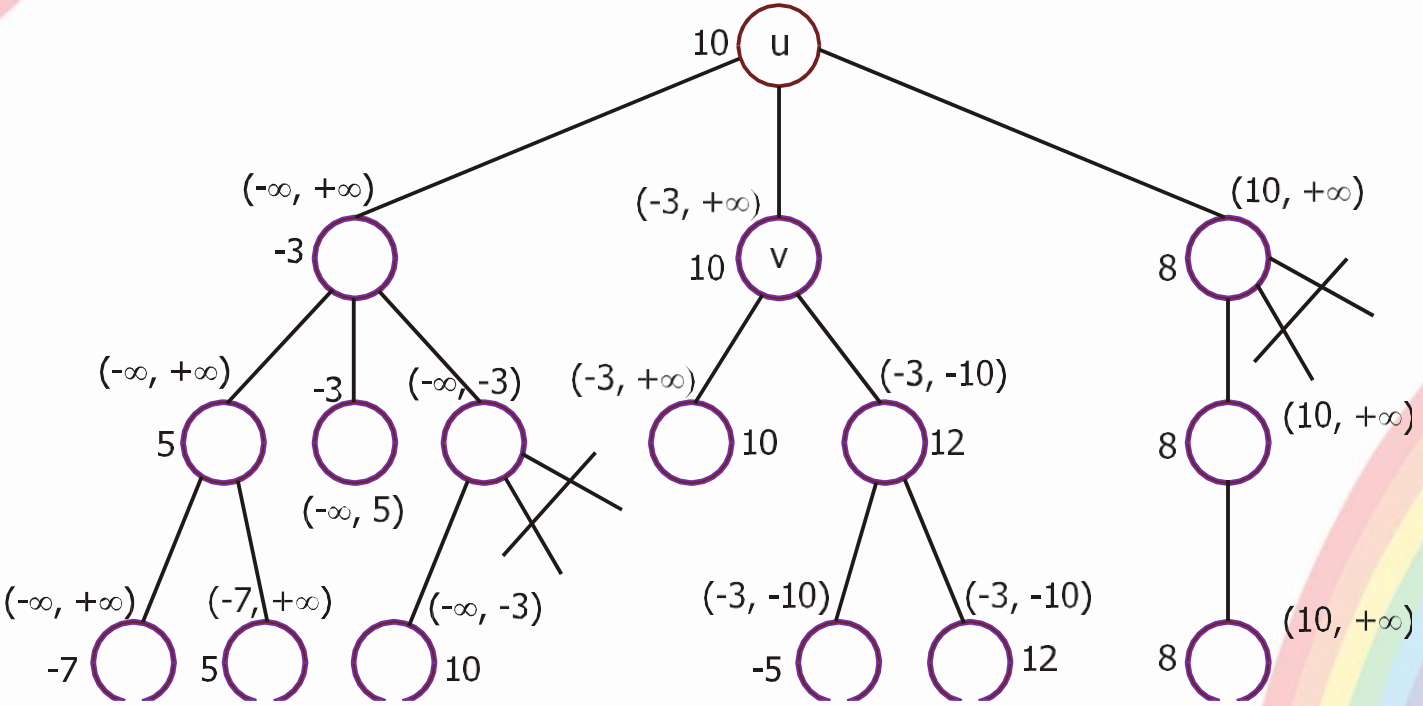
```
void Alpha_beta(u,v)
{   α ← -∞;
    β ← ∞;
    for (mỗi đỉnh w là con của u)
        if (α ≤ MinVal(w, α, β))
        {α ← MinVal(w, α, β);
         v ← w;
        }
}
```

5.1. PHƯƠNG PHÁP MINIMAX

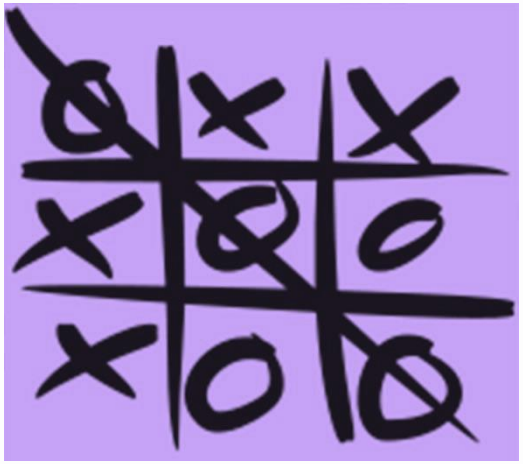
Ví dụ 5.11. Xét cây trò chơi gốc u (đỉnh Trắng) giới hạn bởi độ cao h = 3. Số ghi cạnh các lá là giá trị của hàm đánh giá;

Cạnh mỗi đỉnh ta cũng cho giá trị của cặp tham số (a, b);

Khi gọi các hàm MaxVal và MinVal để xác định giá trị của đỉnh đó. Các nhánh bị cắt bỏ được chỉ ra trong hình:



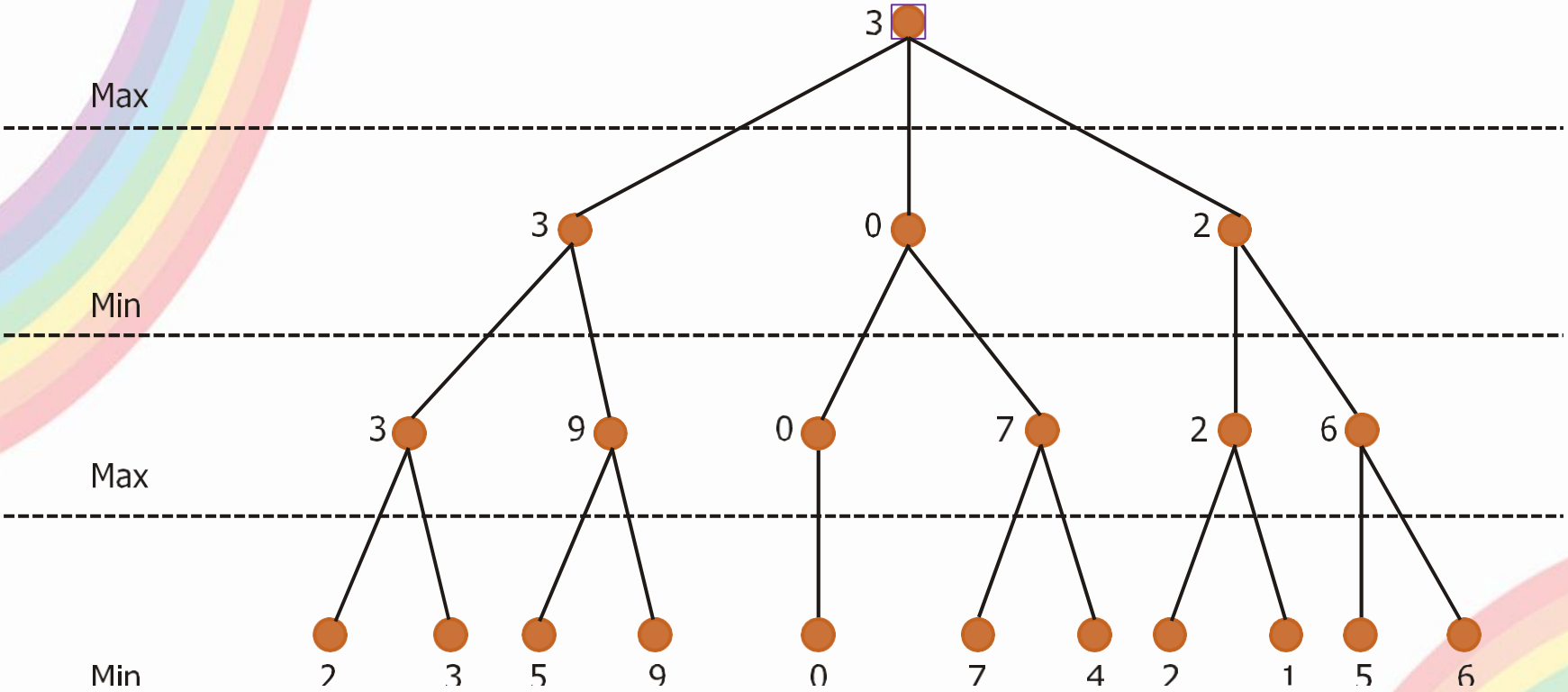
5.2. TRÒ CHƠI TIC-TAC-TOE



Hình 5.26: Trò chơi Tic-tac-toe với phần thắng thuộc về người O

Tic-tac-toe dùng viết trên bàn cờ giấy có 9 ô, 3x3 cho hai người chơi. Một trong hai đối thủ sẽ đi trước, đánh X (hoặc O) vào một ô trống bất kỳ trên bàn cờ. Hai người thay phiên nhau đánh vào các ô trống cho đến khi một người tạo được một dãy ba ký hiệu của mình, ngang dọc hay chéo đều được thì thắng. Nếu hết 9 ô cờ mà không người chơi nào có được 3 quân cờ của mình nằm trên một đường thẳng thì ván đó hòa.

5.2. TRÒ CHƠI TIC-TAC-TOE

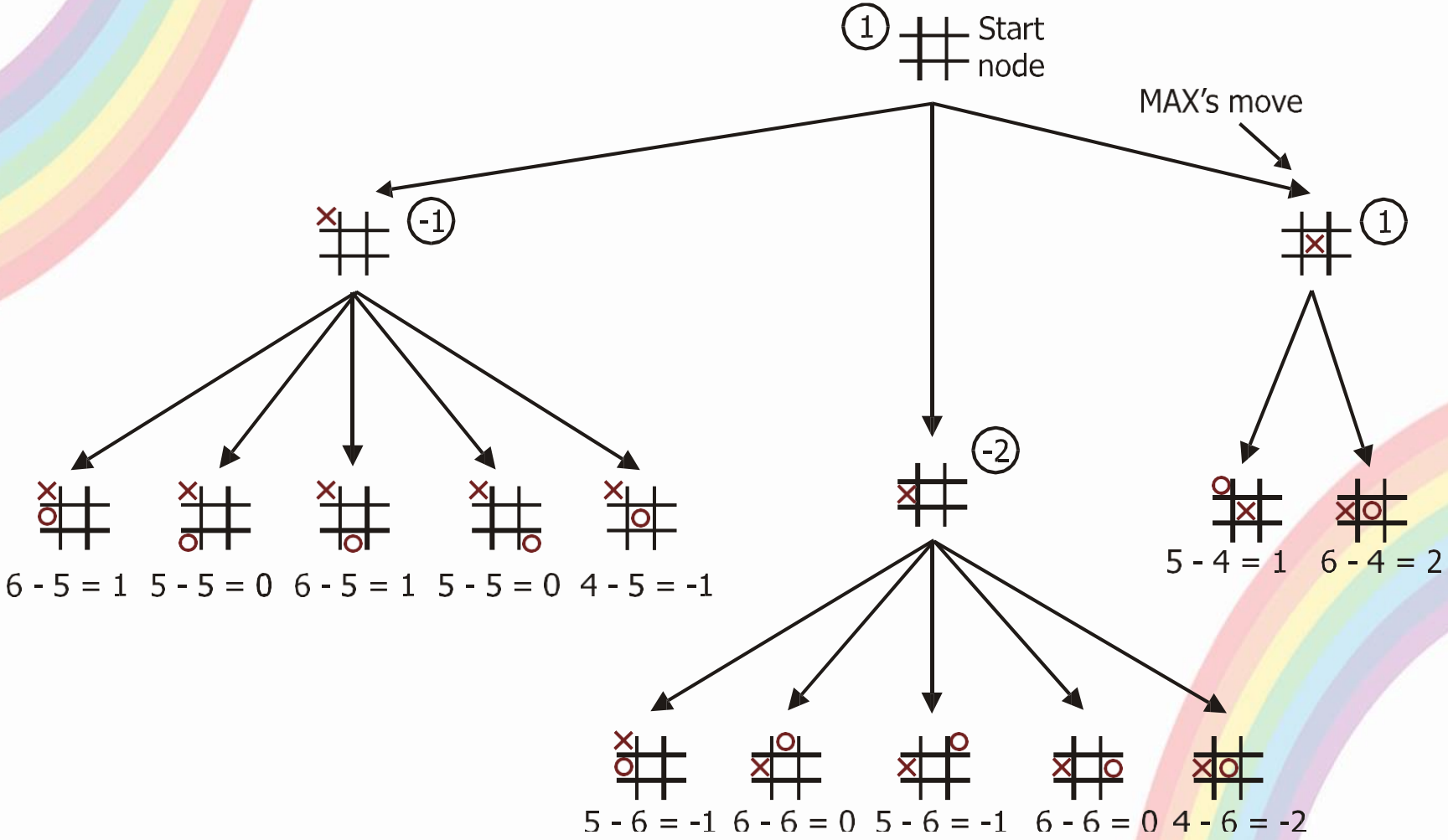


Hình 5.27. Cây thể hiện áp dụng giải thuật minimax cho trò chơi tic tắc toe

Để giải quyết bài toán này chúng ta sử dụng giải thuật minimax – cơ chế cắt tỉa alpha beta để tìm nước đi tốt nhất cho từng người chơi. Kết quả của việc áp dụng Minimax vào cho chơi Tic tac toe được thể hiện như hình 5.27.

5.2. TRÒ CHƠI TIC-TAC-TOE

Áp dụng vào trò chơi



TÓM LƯỢC CUỐI BÀI

- Trình bày đúng khái niệm về cây, cây con, độ cao mức của 1 đỉnh…;
- Trình bày đúng khái niệm cây nhị phân, cách cài đặt cây nhị phân, các thao tác trên cây nhị phân;
- Trình bày được các ứng dụng cây nhị phân trong giải quyết các bài toán.