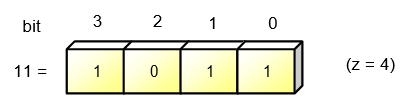
**9.7. CÂY TÌM KIẾM SỐ HỌC (DIGITAL SEARCH TREE - DST)**

Xét dãy khoá k1, k2, …, kn là các số tự nhiên, mỗi giá trị khoá khi đổi ra hệ nhị phân có z chữ số nhị phân (bit), các bit này được đánh số từ 0 (là hàng đơn vị) tới z - 1 từ phải sang trái.

Ví dụ:

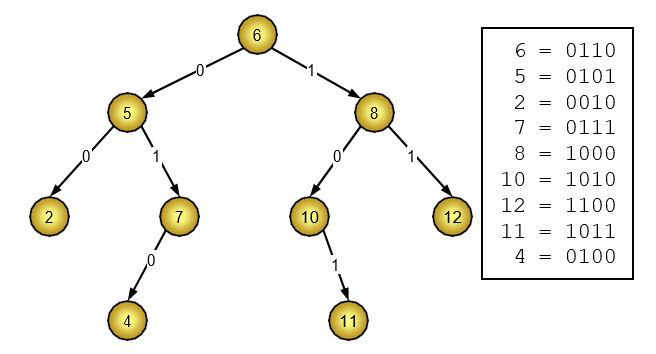


**Đánh số các bit**

Cây tìm kiếm số học chứa các giá trị khoá này có thể mô tả như sau: Trước hết, nó là một cây nhị phân mà mỗi nút chứa một giá trị khoá. Nút gốc có tối đa hai cây con, ngoài giá trị khoá chứa ở nút gốc, tất cả những giá trị khoá có bit cao nhất là 0 nằm trong cây con trái, còn tất cả những giá trị khoá có bit cao nhất là 1 nằm ở cây con phải. Đối với hai nút con của nút gốc, vấn đề tương tự đối với bit z - 2 (bit đứng thứ nhì từ trái sang).

So sánh cây tìm kiếm số học với cây nhị phân tìm kiếm, chúng chỉ khác nhau về cách chia hai cây con trái/phải. Đối với cây nhị phân tìm kiếm, việc chia này được thực hiện bằng cách so sánh với khoá nằm ở nút gốc, còn đối với cây tìm kiếm số học, nếu nút gốc có mức là d thì việc chia cây con được thực hiện theo bit thứ d tính từ trái sang (bit z - d) của mỗi khoá.

Ta nhận thấy rằng những khoá bắt đầu bằng bit 0 chắc chắn nhỏ hơn những khoá bắt đầu bằng bit 1, đó là điểm tương đồng giữa cây nhị phân tìm kiếm và cây tìm kiếm số học: Với mỗi nút nhánh: Mọi giá trị chứa trong cây con trái đều nhỏ hơn giá trị chứa trong cây con phải.



**Cây tìm kiếm số học**

Giả sử cấu trúc một nút của cây được mô tả như sau:

type

PNode = ^TNode; {Con trỏ chứa liên kết tới một nút}

TNode = record {Cấu trúc nút}

Info: TKey; {Trường chứa khoá}

Left, Right: PNode; {con trỏ tới nút con trái và phải, trỏ tới nil nếu không có nút con trái (phải)}

end;

Gốc của cây được lưu trong con trỏ Root. Ban đầu nút Root = nil (cây rỗng)

Với khoá tìm kiếm X, việc tìm kiếm trên cây tìm kiếm số học có thể mô tả như sau: Ban đầu đứng ở nút gốc, xét lần lượt các bit của X từ trái sang phải (từ bit z - 1 tới bit 0), hễ gặp bit bằng 0 thì rẽ sang nút con trái, nếu gặp bit bằng 1 thì rẽ sang nút con phải. Quá trình cứ tiếp tục như vậy cho tới khi gặp một trong hai tình huống sau:

* Đi tới một nút rỗng (do rẽ theo một liên kết nil), quá trình tìm kiếm thất bại do khoá X không có trong cây.
* Đi tới một nút mang giá trị đúng bằng X, quá trình tìm kiếm thành công

{Hàm tìm kiếm trên cây tìm kiếm số học, nó trả về nút chứa khoá tìm kiếm X nếu tìm thấy, trả về nil nếu không tìm thấy. z là độ dài dãy bit biểu diễn một khoá}

function DSTSearch(X: TKey): PNode;

var

b: Integer;

p: PNode;

begin

b := z; p := Root; {Bắt đầu với nút gốc}

while (p  nil) and (p^.Info  X) do {Chưa gặp phải một trong 2 tình huống trên}

begin

b := b - 1; {Xét bit b của X}

if <Bit b của X là 0> then p := p^.Left {Gặp 0 rẽ trái}

else p := p^.Right; {Gặp 1 rẽ phải}

end;

DSTSearch := p;

end;

Thuật toán dựng cây tìm kiếm số học từ dãy khoá k1, k2, …, kn cũng được làm gần giống quá trình tìm kiếm. Ta chèn lần lượt các khoá vào cây, trước khi chèn, ta tìm xem khoá đó đã có trong cây hay chưa, nếu đã có rồi thì bỏ qua, nếu nó chưa có thì ta thêm nút mới chứa khoá cần chèn và nối nút đó vào cây tìm kiếm số học tại mối nối rỗng vừa rẽ sang khiến quá trình tìm kiếm thất bại

{Thủ tục chèn khoá X vào cây tìm kiếm số học}

procedure DSTInsert(X: TKey);

var

b: Integer;

p, q: PNode;

begin

b := z;

p := Root;

while (p  nil) and (p^.Info  X) do

begin

b := b - 1; {Xét bit b của X}

q := p; {Khi p chạy xuống nút con thì q^ luôn giữ vai trò là nút cha của p^}

if <Bit b của X là 0> then p := p^.Left {Gặp 0 rẽ trái}

else p := p^.Right; {Gặp 1 rẽ phải}

end;

if p = nil then {Giá trị X chưa có trong cây}

begin

New(p); {Tạo ra một nút mới p^}

p^.Info := X; {Nút mới tạo ra sẽ chứa khoá X}

p^.Left := nil;

p^.Right := nil; {Nút mới đó sẽ trở thành một lá của cây}

if Root = nil then Root := p {Cây đang là rỗng thì nút mới thêm trở thành gốc}

else {Không thì móc p^ vào mối nối vừa rẽ sang từ q^}

if <Bit b của X là 0> then q^.Left := p

else q^.Right := p;

end;

end;

Muốn xoá bỏ một giá trị khỏi cây tìm kiếm số học, trước hết ta xác định nút chứa giá trị cần xoá là nút D nào, sau đó tìm trong nhánh cây gốc D ra một nút lá bất kỳ, chuyển giá trị chứa trong nút lá đó sang nút D rồi xoá nút lá.

{Thủ tục xoá khoá X khỏi cây tìm kiếm số học}

procedure DSTDelete(X: TKey);

var

b: Integer;

p, q, Node: PNode;

begin

{Trước hết, tìm kiếm giá trị X xem nó nằm ở nút nào}

b := z;

p := Root;

while (p  nil) and (p^.Info  X) do

begin

b := b - 1;

q := p; {Mỗi lần p chuyển sang nút con, ta luôn đảm bảo cho q^ là nút cha của p^}

if <Bit b của X là 0> then p := p^.Left

else p := p^.Right;

end;

if p = nil then Exit; {X không tồn tại trong cây thì không xoá được}

Node := p; {Giữ lại nút chứa khoá cần xoá}

while (p^.Left  nil) or (p^.Right  nil) do {chừng nào p^ chưa phải là lá}

begin

q := p; {q chạy đuổi theo p, còn p chuyển xuống một trong 2 nhánh con}

if p^.Left  nil then p := p^.Left

else p := p^.Right;

end;

Node^.Info := p^.Info; {Chuyển giá trị từ nút lá p^ sang nút Node^}

if Root = p then Root := nil {Cây chỉ gồm một nút gốc và bây giờ xoá cả gốc}

else {Cắt mối nối từ q^ tới p^}

if q^.Left = p then q^.Left := nil

else q^.Right := nil;

Dispose(p);

end;

Về mặt trung bình, các thao tác tìm kiếm, chèn, xoá trên cây tìm kiếm số học đều có độ phức tạp là O(log2n) còn trong trường hợp xấu nhất, độ phức tạp của các thao tác đó là O(z), bởi cây tìm kiếm số học có chiều cao không quá z + 1.