# Introduction

1. Data structure

* Cấu trúc dữ liệu là 1 cách cụ thể để lưu trữ và tổ chức dữ liệu trong máy tính để nó có thể sử dụng một cách hiệu quả
* Dựa vào cách tổ chức các phần tử, cấu trúc dữ liệu được chia làm 2 loại

|  |  |  |
| --- | --- | --- |
|  | Linear | Non-linear |
| Định nghĩa | Các phần tử được truy cập theo thứ tự tuần tự nhưng nó không bắt buộc phải lưu trữ tất cả các phần tử một cách tuần tự | Các phần tử của cấu trúc dữ liệu này được lưu trữ / truy cập theo thứ tự phi tuyến tính |
| Ví dụ | Linked list, stacks, queues | Tree. graph |

1. Data type

|  |  |  |
| --- | --- | --- |
|  | System-defined (Primitive) | User-define |
| Định nghĩa | Là kiểu dữ liệu do hệ thống định nghĩa, thường có trong các ngôn ngữ lập trình | Là kiểu dữ liệu cho programmer tạo ra. |
| Ví dụ | int, float, char, double, bool | Class trong Java, Struct trong C/C++ |

1. Abstract data type (ADTs)

* Với kiểu dữ liệu primitive (int, long, float,..), luôn có các operation như là +, -, \*, /.
* Vói user-defined, chúng ta cũng cần định nghĩa các operation của nó.
* ADT = data structure + operations.
* Ví dụ; cấu trúc dữ liệu Stack sử dụng operation là LIFO.

# Some data structure

|  |  |  |
| --- | --- | --- |
|  | Structure | Operation |
| Linked List | Data: dữ liệu của node đó  Next: trỏ tới node tiếp theo | Insert (head, tail, middle)  Delete(head,tail, middle) |
| Stack(LIFO) | Array: mảng chứa data  Top: vị trí của phần tử mới nhất | Push  Pop |
| Queue (FIFO) | Front: Vị trí của phần tử chờ Dequeue  Rear: vị trí của phần tử chờ Enqueue  Array: mảng chứa data | Enqueue  Dequeue |

# Tree

Là một dạng linked list đặc biệt. Mỗi node của tree có thể pointer đến nhiều node khác nhau.

1. Glossary

|  |  |  |  |
| --- | --- | --- | --- |
| Thuật ngữ | Định nghĩa | Ví dụ | Hình ảnh |
| Root of a tree | Là node có k có parent | Node A |  |
| Edge | Là có đường nối giữa các node parent, children. | Các link bên hình |
| Leaf node | Là node không có child | Node E, J, K, H and I |
| Siblings | Là các node có cùng parent | B, C, D <- A  E, F <- B |
| Ancestors | Nếu node p là ancestor của q khi có 1 path từ root tới q, và p nằm trên path đó. | A, C, G là ancestor of K  K is called as descendant. |
| Level | Các node có cùng depth | B,C,D là level 1  Root A có level 0 |
| Depth of a node | Số lượng edge từ root tới node đó | G có depth = 2.  (A-C-G) |
| Height of a node | Số lượng edge từ node đó tới leaf node (deepest node) | Height of B = 2 (B-F-J) |
| Height of tree | Max(Height of all node) |  |
| Size of a node | Total of descendant of that node including itself | Size© = 3 |

1. Binary Tree

Là cây cấu trúc mà mỗi node hoặc có 0 child, hoặc 1 child, hoặc 2 chidren. Empty tree also called binary tree.

1. Type

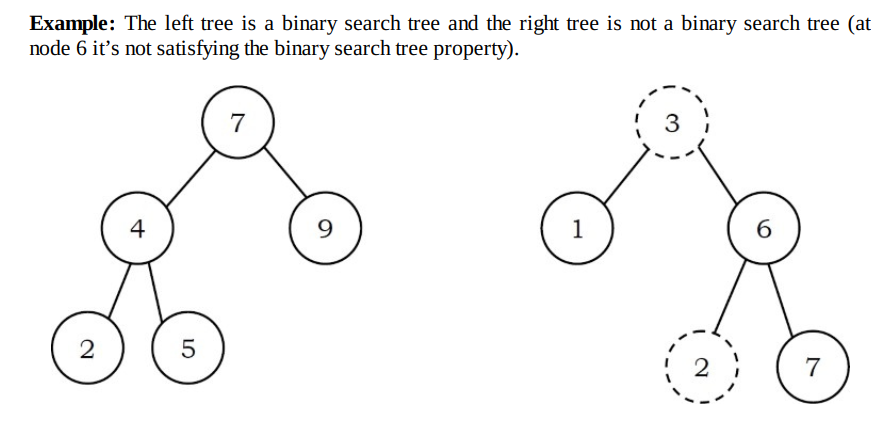
|  |  |  |
| --- | --- | --- |
| Type | Định nghĩa | Ví dụ |
| Strict binary tree | Mỗi node hoặc 0 child hoặc đúng 2 children |  |
| Full binary tree | Mỗi node có đúng 2 children và các leaf node có cùng level |  |
| Complete binary tree |  |  |

1. ADTs

* Xem code

1. Binary search tree (BST)

* Tương tự binary tree, nhưng có thêm những đặc điểm sau:
  + Các **left** subtree node data luôn **nhỏ hơn** root data.
  + Các **right** subtree node data luôn **lớn hơn** root data.
  + Mỗi left-right subtree cũng là 1 BST.



* ADTS

|  |  |
| --- | --- |
| Declare | Tương tự như Binary tree thông thường |
| Main Operation | 1. Tìm min/max element 2. Chèn 1 element 3. Delete |
| Note | 1. Dùng Inorder(LDR) để in ra sorted list. 2. Khi searching, nếu element lớn hơn root, skip the left. |

Hashing