Ngày 8/8/2017

Bài 1. Tree Data Structure – Cấu trúc dữ liệu cây

* Cách tốt nhất để hiểu cấu trúc dữ liệu cây là thông qua hình sau đây

Hình 1. Hình ảnh mô tả cấu trúc dữ liệu của cây

Hình trên hiển thị a cây với 5 levels (5 mức độ). Nút root (Nút gốc là level 0) và bạn di chuyển xuống chiều sâu của cây, mỗi mức tang là 1.

Cây có thể giúp bạn giải quyết nhiều vấn đề quan trọng, bao gồm

* Đại diện cho một mối quan hệ kế thừa giữa các đối tượng
* Thực hiện tìm kiếm nhanh và hiệu quả
* Cung cấp một danh sách sắp xếp của dữ liệu
* Bộ nguồn tiền tố trùng khớp trong trường văn bản

Thuật ngữ

Trước tiên hãy giới thiệu một số thuật ngữ quan trọng bạn nên hiểu về Tree

* Root

Root hay gọi là nút gốc của cây đề cập đến level thứ 0 của cây. Bạn cũng có thể nghĩ nó như là điểm vào tới cấu trúc dữ liệu cây

Hình 2. Mô tả node gốc của cây

* Node (Hay còn gọi là nút trên cây)

Node là một khối của dữ liệu trong cấu trúc cây. Dữ liệu mà một node chứa phụ thuộc vào kiểu dữ liệu của cây bạn đang làm. Root cũng là một node trong cây.

Hình ảnh các node

* Left (Node lá)

Đôi khi được giới thiệu như là một node cuối của cây, node là một node đơn giản không có node con

Hình ảnh node lá

Sự thực hiện cây trong Swift

Trong phần này, chúng ta sẽ thực hiện 1 cây mục đích chung. Điều này là cách ưa thích để nói một cây không có bất kì hạn chế nào

Hãy nhớ rằng, một cây được tạo ra bởi các node. Vì vậy để bắt đầu, hãy tạo ra một class Node cơ bản. Chúng ta hãy tạo nó trong playground của Xcode và thêm 1 class trống như sau

Code:

Class Node {

}

Tất nhiên, một node sẽ phải chứa các giá trị. Để đơn giản cây này của bạn sẽ quản lý dữ liệu là chuỗi. Cập nhật cây của bạn như sau

Class Node {

Var value: String

//Khởi tạo cây ban đầu với một giá trị

init(value: String) {

selft.value = value

}

}

giải thích : Ở trên bạn khai báo thuộc tính tên là value có kiểu là String. Bạn cũng khai báo 1 phương thức khởi tạo, nó được yêu cầu khởi tạo cho các thuộc tính non-optional trong class. Theo mình hiểu thì các thuộc tính đó không được phép là nil.

Children

Ngoài các giá trị ra, mỗi node cần có 1 danh sách các node con

Thực hiện cập nhật class của bạn như sau

Class Node {

var value: String

var children: [Node] = []

//Khởi tạo cây ban đầu với một giá trị

init(value: String) {

selft.value = value

}

}

Bạn khai báo những node con như là một mảng các node. Mỗi node con thể hiện một node có độ sâu hơn node hiện tại là 1.

Parent

Đôi khi nó tiện dụng cho mỗi node để có một liên kết tới node cha của nó. Children là những node ở dưới một Node cho trước, Node Parent là node ở trên. Một node chỉ có một node Parent nhưng có thể có nhiều Node Child.

Cập nhật lại Class Node của bạn như sau:

Class Node {

var value: String

var children: [Node] = []

weak var parent: Node?

//Khởi tạo cây ban đầu với một giá trị

init(value: String) {

selft.value = value

}

}

Chú ý rằng chúng ta vừa tạo một thuộc tính parent là một optional. Điều này là vì không phải tất cả các Node đều có các node Parent, chẳng hạn như node root. Cũng tương tự không phải tất cả các node đều có các node con, như node lá (leaf) chính vì điều đó mình đã khởi tạo 1 mảng mặc định là rỗng.

Nếu các bạn có thắc mắc tại sao lại phải khai báo weak trước thuộc tính parent mình sẽ giải thích kĩ ở 1 bài khác vì vấn đề này liên quan đến retain cycle. Hiện tại thì hãy tập trung xây dựng liên qua dến Tree.

Insertion

Để xử lý sự thêm vào cây, bạn cần khai báo 1 phương thức add (child: Node) trong class của bạn

Hãy cập nhật cây như sau

Class Node {

Var value: String

Var children:[Node] = []

Var paren: Node?

Init(value: String) {

Self.value = value

}

//Method add

function add(child: Node) {

selft.children.append(child)

child.parent = selft

}

}

Giải thích phương thức add: Bạn chỉ cần hiểu đơn giản khi bạn thêm một Node child tới node hiện tại thì đồng nghĩa với việc bạn thực hiện 2 việc sau đây:

Thêm node child đang thêm vào mảng children của Node hiện tại// selft.children.append(child)

Và node cha của Node đang thêm(child) chính là node hiện tại// selft.children.append(child)

Như vậy chúng ta đã xây dựng xong class Node. Bây giờ hãy thựa hiện như sau để các bạn có thể hiểu rõ nhưng gì chúng ta vừa làm

. Cho hình ảnh sau. Các bạn hãy xây dựng cây đó

Trước khi đi xây dựng chúng ta hãy cùng nhau phân tích về cây này một chút. Cây này có Node root có giá trị là “beverages”, node root có 2 node child là hot và cold. Đồng thời 2 node con đó không có node con nào nên nó là Node leaf luôn

Thực hiện

1. Khởi tạo 1 Node root

let beverages = Node(value: “beverages”)

1. Khởi tạo 2 node leaf hay còn gọi là 2 node child của node root

let hotBeverages = Node(value: “hot”)

let coldBeverages = Node(value: “cold”)

1. ở tren chung ta mới chỉ khai báo các node là rời rạc chưa thể hiện mối quan hệ giữa chúng. Muốn 2 node leaf là node con của node root chúng ta hãy sử dụng phương thức add

beverages.add(hotBeverages)

beverages.add(coldBeverages)

Đến đây chúng ta đã hoàn thành được cây ở trên

Sau đây là thử thách cho các bạn xây dựng một cây khó hơn

Hình ảnh thử thách xây dựng cây

Printing Trees

Chúng ta hãy đặt ra một câu hỏi làm thế nào để có thể in được cả một cấu trúc cây ra console???

Các bạn hãy thử dung câu lệnh:

Print(beverages)// liệu có được không??? :D

Kết quả nhận được là Node. Để giải quyết vấn đề này bạn hãy thực hiện như sau

*// 1*

extension Node: CustomStringConvertible {

*// 2*

var description: String {

*// 3*

var text = "\(value)"

*// 4*

if !children.isEmpty {

text += " {" + children.map { $0.description }.joined(separator: ", ") + "} "

}

return text

}

}

Giải thích các câu lệnh trên.

1. Bạn khai báo 1 extension tới class Node của bạn và kế thừa lại protocol CustomStringConvertible. Protocol này mong rằng bạn implement một thuộc tính với tên là description với kiểu là String.
2. Bạn khai báo 1 thuộc tính description. Nó là một thuộc tính chỉ đọc và trả về một String
3. Khai báo 1 biến text. Nó sẽ giữ một giá trị kiểu string, bây giờ bạn sẽ nhân được giá trị của node hiện tại
4. Ngoài việc in giá trị của node hiện tại, bạn cũng sẽ cần để in các node con của chúng, và node con của node con. Để thực hiện bạn thực hiện đệ quy.

Bây giờ bạn hãy thực hiện print ra node gốc. Kết quả nhận được là

**"beverages {hot {tea {black, green, chai} , coffee, cocoa} , cold {soda {ginger ale, bitter lemon} , milk} } \n"**