**Cây AVL**

Sinh viên tuân thủ theo quy định trong file “**QUY ĐỊNH ĐỐI VỚI MÔN HỌC LẬP TRÌNH C++**”

Do thời gian có hạn, sinh viên tự tìm hiểu về 2-3-4 tree, B tree,...

0. Khởi tạo

1. Tạo class Node dùng để chứa thông tin của node trong cây. Node này chỉ chứa thông tin duy nhất là: key, có kiểu dữ liệu integer (sinh viên có thể thêm biến chiều cao của node nếu cần thiết).
2. Ta sẽ quản lý cây AVL bằng node root.

Các key sẽ không trùng nhau (theo tài liệu của MIT http://web.mit.edu/jlai321/Public/old\_class\_files/1.00/LectureSlides/Lecture28.pdf)

1. Thêm node vào cây:
   1. Viết hàm thêm node vào cây AVL.
   2. Lần lượt thêm các node: 10, 100, 20, 30, 40, 50 vào cây AVL. Để đơn giản, sinh viên có thể cài đặt cứng mà không cần nhập std in.
   3. Viết hàm in ra giá trị của cây, theo thứ tự inorder (left - root - right)
   4. In ra giá trị toàn bộ cây AVL.

| **Std in** | **Std out** |
| --- | --- |
|  | 10 20 30 40 50 100 |

1. Xóa node khỏi cây:
   1. Viết hàm xóa node khỏi cây AVL.
   2. Lần lượt thêm các node: 10, 100, 20, 30, 40, 50 vào cây AVL. Để đơn giản, sinh viên có thể cài đặt cứng mà không cần nhập std in.
   3. Nhập vào key của giá trị cần xóa, với giá trị key = 10.
   4. Viết hàm in ra giá trị của cây, theo thứ tự inorder (left - root - right)
   5. In ra giá trị toàn bộ cây AVL.

| **Std in** | **Std out** |
| --- | --- |
| 10 | 20 30 40 50 100 |

1. Kiểm tra node trong cây:
   1. Viết hàm kiểm tra, cây AVL có tồn tại node có giá trị key = x hay không? Nếu có, in ra màn hình “exist”, ngược lại in “non-exist”
   2. Lần lượt thêm các node: 10, 100, 20, 30, 40, 50 vào cây AVL. Để đơn giản, sinh viên có thể cài đặt cứng mà không cần nhập std in.
   3. Tìm xem node = 50 có tồn tại trong cây AVL không?

| **Std in** | **Std out** |
| --- | --- |
| 50 | exist |

1. Tổng hai node trong cây:
   1. Viết hàm kiểm tra, cây AVL có tồn tại hai node, sao cho tổng của hai node đó có giá trị bằng sum cho trước không? Nếu có, in ra màn hình “exist”, ngược lại in “non-exist”
   2. Lần lượt thêm các node: 10, 100, 20, 30, 40, 50 vào cây AVL. Để đơn giản, sinh viên có thể cài đặt cứng mà không cần nhập std in.
   3. Kiểm tra cây AVL có hai node có tổng bằng 80 hay không?

* Lưu ý: sinh viên phải hoàn thành bài tập này với độ phức tạp về mặt thời gian là O(n)

| **Std in** | **Std out** |
| --- | --- |
| 80 | exist |

1. Giá trị lớn thứ k:
   1. Viết hàm tìm giá trị lớn thứ k trong cây AVL.

Ví dụ: [10, 20, 30, 40], thì giá trị lớn thứ 2 là 30.

* 1. Lần lượt thêm các node: 10, 100, 20, 30, 40, 50 vào cây AVL. Để đơn giản, sinh viên có thể cài đặt cứng mà không cần nhập std in.
  2. Tìm giá trị lớn thứ k = 2 trong cây AVL và in ra màn hình.
* Lưu ý: sinh viên phải hoàn thành bài tập này với độ phức tạp về mặt thời gian là O(n)

| **Std in** | **Std out** |
| --- | --- |
| 2 | 50 |

1. Sinh viên lập bảng báo cáo, dùng để tổng kết một số cấu trúc cây đã được học. Hoàn thành và lưu vào file ex\_7.pdf

* Binary search tree:
  + Ưu điểm
  + Nhược điểm
  + Đặc điểm (nếu có).
* Cây AVL tree
* Red black tree
* B tree