Đại Học Quốc Gia TP.HCM
Trường Đại học Bách Khoa
Khoa KH&KT Máy Tính

Vietnam National University – HCMC
Hochiminh University of Technology
Faculty of Computer Science and Engineering

CẤU TRÚC DỮ LIỆU & GIẢI THUẬT
ASSIGNMENT 02 - TREE SET

1 Giới thiệu

Trong bài tập lớn này sinh viên sẽ hiện thực cấu trúc dữ liệu tập hợp TreeSet[1]. Cụ thể, cấu trúc dữ liệu tập hợp này sẽ được hiện thực dựa trên cây AVL đã được học trên lớp. Việc hiện thực này phải đảm bảo thời gian thực thi trong trường hợp xấu nhất (worst case) là log(n) cho các phép toán cơ bản như thêm phần từ (add), xóa phần tử (remove), và các phép toán tìm kiếm. Ở bài tập lớn này, dữ liệu kiểm tra sẽ có kích thước rất lớn, do đó, sinh viên cần lưu ý tối ưu hóa mã nguồn để đảm bảo thời gian thực thi.

2 Hiện thực

Phần hiện thực trong bài tập lớn này được tổ chức trong file **main.cpp** và 2 lớp: **AVLNode** and **TreeSet** và được chia thành bốn file riêng biệt: **AVLNode.h**, **TreeSet.h**, **TreeSet.cpp** và **main.cpp**. Cụ thể như sau:

2.1 AVLNode.h

File này chứa phần khai báo và định nghĩa cấu trúc dữ liệu **AVLNode** để hiện thực cấu trúc dữ liệu cây AVL.

```
class AVLNode {
public:
  int key;       // data
  AVLNode* left;     // left child
  AVLNode* right;    // right child
  int balance;     // balance factor

  AVLNode(int key) {
    this->key = key;
    left = right = NULL;
    balance = 0;
  }
  AVLNode(int key, int balance) {
    this->key = key;
    this->balance = balance;
    left = right = NULL;
  }
};
```

2.2 TreeSet.h

File này chứa phần khai báo các hàm (method) cho lớp **TreeSet**. Phần prototype của lớp này là dựa trên hiện thực Java của cấu trúc dữ liệu TreeSet.

```
class TreeSet
{
private:
  AVLNode * root;
  int count;

protected:
  void clearRec(AVLNode*root);

```

[1] https://docs.oracle.com/javase/7/docs/api/java/util/TreeSet.html

```
10
11 public:
12   TreeSet();
13   ~TreeSet();
14   void clear();
15   // print out the set in ascending order
16   friend ostream& operator<<(ostream& os, const TreeSet& t);
17
18   // YOUR TASKS START HERE
19   int add(int val);
20   bool contains(int val);
21   void copy(const TreeSet& set);
22   int first();
23   int higher(int val);
24   int last();
25   int lower(int val);
26   int remove(int val);
27   TreeSet* subSet(int fromVal, int toVal);
28   int size();
29   // END HERE
30 };
```

2.3 TreeSet.cpp

File này chứa phần các prototypes cho tất cả các hàm mà sinh viên cần hiện thực trong bài tập lớn này. Phần mô tả chi tiết cho những hàm này được cung cấp ở phần 3.

2.4 main.cpp

File này chứa hàm **main()** được dùng để kiểm tra hiện thực cấu trúc dữ liệu **TreeSet** của sinh viên. Hàm này sẽ đọc tuần từ các lệnh từ file **input.txt** để gọi các hàm trong lớp **TreeSet**. Chi tiết định dạng của file **input.txt** và nghĩa của các lệnh này được định nghĩa như sau:

No	Command	Parameters	Activate
1	a	val (int)	add(val)
2	c	val (int)	contains(val)
3	d		copy()
4	f		first()
5	h	val (int)	higher(val)
6	l		last()
7	o	val (int)	lower(val)
8	p		print()
9	r	val (int)	remove(val)
10	s	fromVal (int), toVal (int)	subSet(fromVal, toVal)
11	z		size()

Dưới đây là một ví dụ của file input:

```
a 4
a 5
a 7
a 10
a 20
c 9
f
l
p
```

2.5 Yêu cầu

Yêu cầu của bài tập lớn này là hiện thực cấu trúc dữ liệu tập hợp **TreeSet** được mô tả ở phần trước. Cấu trúc dữ liệu này sẽ chứa các phần tử là các số nguyên không âm. Cụ thể, sinh viên cần hoàn tất các hàm được cung cấp trong lớp **TreeSet** (**TreeSet.h** và **TreeSet.cpp**). Phần code cho sẵn sẽ giúp sinh viên hoàn tất bài tập lớn dễ dàng hơn.

Bảng 1: Mô tả các yêu cầu

No	Method	Description
1	int add(int val)	Thêm một phần tử mới vào trong tập hợp. Nếu phần tử đó đã tồn tại trong tập hợp, tập hợp không thay đổi và trả về false; ngược lại phần tử sẽ được thêm vào trong tập hợp và trả về true. Parameters: - val: phần tử cần được thêm vào trong tập hợp.
2	bool contains(int val)	Trả về true nếu tập hợp chứa phần tử val; ngược lại trả về false. Parameters: - val: phần tử cần kiểm tra xuất hiện trong tập hợp.
3	int copy(const TreeSet& set)	Sao chép dữ liệu từ tập hợp set (nguồn) sang tập hợp hiện tại (đích). Lưu ý: Hai tập hợp nguồn và đích sẽ là đối tượng khác nhau (không chia sẻ không gian lưu trữ) nhưng chứa dữ liệu giống nhau. Parameters: - set: tập hợp chứa dữ liệu nguồn cần sao chép
4	int first()	Trả về phần tử nhỏ nhất trong tập hợp Throws: - NoSuchElementException : nếu tập hợp rỗng
5	int higher(int val)	Trả về phần tử nhỏ nhất trong tập hợp lớn hơn phần tử val. Lưu ý: phần tử trả về không được phép bằng phần từ val. Trả về giá trị -1 nếu không tồn tại phần tử nào trong tập hợp thỏa mãn yêu cầu (giả sử rằng tập hợp chỉ chứa các giá trị nguyên không âm). Parameters: - val: phần tử dùng để kiểm tra.
6	int last()	Trả về phần tử lớn nhất trong tập hợp Throws: - NoSuchElementException : nếu tập hợp rỗng
7	int lower(int val)	Trả về phần tử lớn nhất trong tập hợp nhỏ hơn phần tử val. Lưu ý: phần tử trả về không được phép bằng phần từ val. Trả về giá trị -1 nếu không tồn tại phần tử nào trong tập hợp thỏa mãn yêu cầu (giả sử rằng tập hợp chỉ chứa các giá trị nguyên không âm). Parameters: - val: phần tử dùng để kiểm tra.
8	int remove(int val)	Xóa phần tử val trong tập hợp. Nếu phần tử val không tồn tại trong tập hợp thì tập hợp sẽ không thay đổi và trả về false; ngược lại phần tử val sẽ bị xóa khỏi tập hợp và trả về true. Parameters: - val: phần tử cần được xóa khỏi tập hợp.
9	TreeSet* subSet(int fromVal, int toVal);	Trả về con trỏ đến tập hợp con của tập hợp hiện tại. Tập hợp con này chứa các phần tử từ [fromVal, toVal) (tính phần tử fromVal nhưng không tính phần tử toVal). Lưu ý: tập hợp con là đối tượng tách biệt với tập hợp hiện tại (tập hợp con không chia sẻ không gian lưu trữ với tập hiện tại).
10	int size()	Trả về số lượng phần tử của tập hợp

Sinh viên không được sử dụng các thư viện nào khác ngoài các thư viện đã được dùng trong code mẫu.

3 Quy định

3.1 Đánh giá

Output từ chương trình sẽ được so trùng với output kì vọng. Một testcase là đạt nếu toàn bộ output trùng khớp với output kì vọng, và không bị vi phạm ràng buộc thời gian chạy. Từng testcase sẽ được chạy và số lượng testcase đạt sẽ tương ứng với số điểm của assignment.

3.2 Nộp bài

Sinh viên sẽ nộp bài qua hệ thống và sẽ được hướng dẫn cụ thể qua thông báo trên Sakai.

Lưu ý: sinh viên chỉ cần nộp 2 file **TreeSet.h** và **TreeSet.cpp** chứa mã nguồn (code) hiện thực của sinh viên. Vì vậy, sinh viên không được phép thay đổi mã nguồn trong những file còn lại bao gồm: **AVLNode.h**, và **main.cpp**.

Hệ thống sẽ build code trên Linux. Do đó sinh viên không sử dụng các hiện thực đặc biệt nào phụ thuộc hệ điều hành hay trình biên dịch. Sinh viên phải kiểm tra chương trình của mình trên Linux trước khi nộp để chắc chắn code chạy được.

Hạn chót của assignment 2 là **14-12-2018**. Hệ thống chấm sẽ được mở trước ngày 14-12-2018.

3.3 Xử lý gian lận

Bài tập lớn phải được sinh viên TỰ LÀM. Sinh viên sẽ bị coi là gian lận nếu:

- Có sự giống nhau bất thường giữa mã nguồn của các bài nộp. Trong trường hợp này, tất cả các bài nộp đều bị coi là gian lận. Do vậy sinh viên phải bảo vệ mã nguồn bài tập lớn của mình. Các bài làm của các sinh viên ở các học kỳ trước cũng sẽ được dùng để kiểm tra gian lận.
- Sinh viên không hiểu mã nguồn do chính mình viết, trừ những phần mã được cung cấp sẵn trong chương trình khởi tạo. Sinh viên có thể tham khảo từ bất kỳ nguồn tài liệu nào, tuy nhiên phải đảm bảo rằng mình hiểu rõ ý nghĩa của tất cả những dòng lệnh mà mình viết. Trong trường hợp không hiểu rõ mã nguồn của nơi mình tham khảo, sinh viên được đặc biệt cảnh báo là KHÔNG ĐƯỢC sử dụng mã nguồn này; thay vào đó nên sử dụng những gì đã được học để viết chương trình.

Trong trường hợp bị kết luận là gian lận, sinh viên sẽ bị **điểm 0 cho toàn bộ môn học** (không chỉ bài tập lớn). **Không có NGOẠI LỆ nào được chấp nhận.**

Sau mỗi bài tập lớn được nộp, sẽ có một số sinh viên được gọi phỏng vấn ngẫu nhiên để chứng minh rằng bài tập lớn vừa được nộp là do chính mình làm.