Final Project Report

Bees - C++ Inplementation of the Java STL

上海交通大学致远学院 11 级 ACM 班刘爽

April 30, 2012

Abstract

利用 C++ 模板功能实现部分的 JAVA 的 STL 的功能。在给定函数体框架的情况下补全每个函数体内容，最终需要实现 ArrayList,LinkedList, HashSet,HashMap,TreeSet,TreeMap 六个文件。

Contents

Chapter 1

Project Summary

1.1 Summary

利用 C++ 的 template 功能实现部分 Java 的 STL 功能。

在给定一个 Utility.h 头文件以及其余六个头文件的函数定义框架的情况下补全每个函数体内部的内容。

最终需要实现线性查找表 ArrayList 与 LinkedList，哈希查找表 HashSet 与 HashMap, 树形查找表 TreeSet 与 TreeMap 六个文件。

1.2 Motivation

首先，进行这次大作业是为了完成本学期数据结构课的学习任务。

其次，作为对深入了解 Java 与 C++ 的一条途径，我把这次作业看成是对 Java 与 C++ 的语法与内部架构的复习。

再次，这次作业也是为了给以后写大型工程打好基础。

Chapter 2

Work Description

2.1 Preparing

- 阅读任务文档，熟悉任务要求
- 阅读 JAVA 文档及源码，选择了 GNU CLASSPATH
- 了解 JAVA 实现方法，并研究如果运用在 C++ 中会遇到什么困难

2.2 Implementation

2.2.1 ArrayList

如下的三段程序分别是 ArrayList 的成员变量，ArrayList::Iterator 的成员变量，以及在空间不够时用来 double space 的 ensureCapacity 函数。

```
class ArrayList {
    private:
    static const int DEFAULT_CAPACITY = 10;
    int sz, cap;
    E* data;

class Iterator {
    private:
    int pos, size, last;
    ArrayList* arr;

void ensureCapacity(int minCapacity) {
    if (minCapacity > cap) {
        E* newData = new E[cap = getMax(cap * 2, minCapacity)];
        memmove(newData, data, sz * sizeof(E));
        delete [] data;
        data = newData;
    }
}
```

2.2.2 LinkedList

如下程序是 linkedList 的成员函数。

```
class LinkedList {
  private:
  class Entry {
    public:
    T data;
    Entry *next, *previous;
    Entry() {}
    Entry(T _data) {
      data = _data;
      next = previous = NULL;
    }
  };
  Entry *first, *last;
  int sz;
```

2.2.3 HashMap

HashMap 采用拉链的 hash 表实现，有一个叫做 loadFactor 的参数，以及叫做 threshold 的值，threshold 表示当 size 达到 threshold 的时候就需要 double space 扩大 hash 表的 capcity，而 threshold 等于 capcity 乘以 loadFactor, 在本程序中，loadFactor 设为 0.75，这也是 Java 在某个版本后一直沿用的 Default LoadFactor。

```
class HashMap {
  public:
  static const int DEFAULT_CAPCITY = 11;
  static const double DEFAULT_LOAD_FACTOR = 0.75;
  private:
  template <class K2, class V2>
  class HashEntry: public Entry<K2, V2> {
    public:
    HashEntry<K2, V2>* next;
    HashEntry() {}
    HashEntry(K2 _key, V2 _value): Entry<K2, V2>(_key, _value) {}
  };
  int threshold, cap;
  double loadFactor;
  HashEntry<K, V>** buckets;
  int sz;

V put(const K& key, const V& value) {
  if (++sz > threshold) {
    rehash();
    idx = hash(key);
  }
```

```
void rehash() {
    cap = cap * 2 + 1;
    threshold = (int)(cap * loadFactor);
```

2.2.4 HashSet

HashSet 内部直接调用 HashMap，即 HashSet 可以看作是 value 为空的 HashMap。

```
    class HashSet {
        private:
        HashMap<T, bool, H>* map;

    class Iterator {
        public:
        typename HashMap<T, bool, H>::Iterator mItr;
```

2.2.5 TreeMap

TreeMap 我借鉴了 C++ 和 Java 的 STL，采用红黑树实现，但这也是这次作业碰到的比较大的困难之一。

```
class TreeMap {
    private:
    static const int RED = -1, BLACK = 1;
    template <class K2, class V2>
    class Node: public Entry<K2, V2> {
        public:
        int color;
        Node<K2, V2> *left, *right, *parent;
        Node(): Entry<K2, V2>(K(), V()) {
            color = BLACK;
            left = right = parent = this;
        }
        Node(K2 _key, V2 _value, int _color, Node<K2, V2>* _left,
                Node<K2, V2>* _right, Node<K2, V2>* _parent):
                        Entry<K2, V2>(_key, _value)
        {
            color = _color;
            left = _left; right = _right; parent = _parent;
        }
    };
    Node<K, V> *nil, *root; int sz;
```

```
class TreeMap {
  private:
    void rotateLeft(Node<K, V>* node);
    void rotateRight(Node<K, V>* node);
    void insertFixup(Node<K, V>* node);
    void deleteFixup(Node<K, V>* node, Node<K, V> *parent);
    void removeNode(Node<K, V>* node);
    Node<K, V>* getNode(K key);
    Node<K, V>* successor(Node<K, V>* node);
```

2.2.6 TreeSet

正如 Hashset 与 HashMap 的关系，TreeSet 也内部直接调用了 TreeMap 來实现。

```
class TreeSet {
  private:
    TreeMap<E, bool>* map;

class Iterator {
  public:
  typename TreeMap<E, bool>::Iterator mItr;
```

2.3 Testing Results

2.3.1 Script

作业写完了，如何检验质量呢。除了正确性，效率也是一个很关键的方面。既然是实现 Java 的 STL，不妨就和 Java 的 STL 进行对比。由于 Set 和 Map 的相似性，只测试了 ArrayList, LinkedList, HashSet, TreeSet 四个程序。为了避免 Java 读入过慢导致影响测试，采用 main 读入规模的方法來进行测试。四组程序分别测试数据范围由小到大的五个点。测试用的 bash 脚本及 4 组主程序见附录。

2.3.2 Tables

ArrayList

观察下表会发现，Cpp 随着规模增长，用时也稳步增长，但 Java 虽然固有速度很慢，但在数据规模达到 10^9 时，却是“惊人的”快了。

Test Case	Data Size	My Cpp STL	Official Java STL
1	10^5	0.06	0.86
2	10^6	0.03	0.10
3	10^7	0.30	1.12
4	10^8	3.21	23.80
5	10^9	29.07	39.04

LinkedList

从下表可以看出，Cpp 的用时增长和 ArrayList 一样十分稳定，但这回 Java 相反，当如据规模到达 10^8 时确实惊人的慢了。

Test Case	Data Size	My Cpp STL	Official Java STL
1	10^4	0.02	0.11
2	10^5	0.02	0.07
3	10^6	0.07	0.13
4	10^7	0.65	0.97
5	10^8	6.28	24.72

HashSet

这次有个明显的特点，当数据范围还不是很大时，Java 和 Cpp 的速度是相差无几的。

Test Case	Data Size	My Cpp STL	Official Java STL
1	10^4	0.01	0.09
2	10^5	0.02	0.11
3	10^6	0.16	0.18
4	10^7	1.46	1.10
5	10^8	14.77	40.58

TreeSet

也许是因为我第一次写红黑树，所以效率不高吧，快要跑不过 Java 了。

Test Case	Data Size	My Cpp STL	Official Java STL
1	10^3	0.00	0.09
2	10^4	0.01	0.09
3	10^5	0.09	0.12
4	10^6	1.25	0.92
5	10^7	18.92	18.99

Chapter 3

Final Conclusion

3.1 Gains

- 提高了 C++ 的编译能力，对 C++ 与 Java 的区别有了更感性的认识
- 加强了写工程的能力

3.2 Advices

- 希望大作业可以提供更高的的自由度
- 希望大作也可以有更加详尽的说明文档

Appendix A

Testing Scripts

该脚本是用 bash 编写的，每次读取当前要测试的主程序名（如 MainArrayList），初始数据规模（如 10000），最大数据规模 (如 100000000)，脚本对于每个数据规模分别运行 Cpp 与 Java 的主程序，输出运行时间。

```
#!/bin/bash

echo -n "Your program name: "
read program_name
echo -n "Begin Value: "
read beginValue
echo -n "End Value: "
read endValue

echo ""
echo "################################"
echo "          Testing starts...       "
echo ""

num=0;
for ((N=$beginValue; N<=$endValue; N=N*10))
do
        ((num++))
        echo "TEST CASE $num"
        echo "size: "$N
        echo -n "My Cpp STL: "
        (time -p ./$program_name $N) 2>&1 | grep real | sed 's/real/TIME/'
        echo -n "Offical Java STL: "
       (time -p java $program_name $N) 2>&1 | grep real | sed 's/real/TIME/'
        echo ""
done

echo "          All tests done.       "
echo "################################"
echo ""
```

Appendix B

Testing Main Programs

每个 Java 与 Cpp 的主程序都从 main 参数读入数据规模 N，然后运行。不执行 IO。

B.1 MainArrayList

B.1.1 MainArrayList.cpp

```cpp
int main(int argc, char **argv) {

    int N = atoi(argv[1]);
    int tot = N / 10;

    ArrayList<int> arr;
    for (int te = 0; te < 10; ++te) {
        for (int i = 1; i <= tot; ++i) arr.add(i);
        for (int i = 0; i < arr.size(); ++i) arr.get(i);
        while (!arr.isEmpty()) arr.removeIndex(arr.size() - 1);
    }

    return 0;

}
```

B.1.2 MainArrayList.java

```java
public class MainArrayList {
    public static void main(String args[]) {

        int N = Integer.parseInt(args[0]);
        int tot = N / 10;

        ArrayList<Integer> arr = new ArrayList<Integer>();
```

```
        for (int te = 0; te < 10; ++te) {
            for (int i = 1; i <= tot; ++i) arr.add(i);
            for (int i = 0; i < arr.size(); ++i) arr.get(i);
            while (!arr.isEmpty()) arr.remove(arr.size() - 1);
        }

    }
}
```

B.2 LinkedList

B.2.1 LinkedList.cpp

```
int main(int argc, char **argv) {

    int N = atoi(argv[1]);
    int tot = N / 10;

    LinkedList<int> lnk;
    for (int te = 0; te < 10; ++te) {
        for (int i = 1; i <= tot; ++i) lnk.add(i);
        while (!lnk.isEmpty()) lnk.removeLast();
    }

    return 0;

}
```

B.2.2 LinkedList.java

```
public class MainLinkedList {
    public static void main(String args[]) {

        int N = Integer.parseInt(args[0]);
        int tot = N / 10;

        LinkedList<Integer> lnk = new LinkedList<Integer>();
        for (int te = 0; te < 10; ++te) {
            for (int i = 1; i <= tot; ++i) lnk.add(i);
            while (!lnk.isEmpty()) lnk.removeLast();
        }
    }
}
```

B.3 HashSet

B.3.1 HashSet.cpp

```
int main(int argc, char **argv) {

    int N = atoi(argv[1]);
    int tot = N / 10;

    HashSet<int, Hashint> set;
    for (int te = 0; te < 10; ++te) {
        for (int i = 1; i <= tot; ++i) set.add(i);
        for (int i = 1; i <= tot; ++i) set.remove(i);
    }

    return 0;

}
```

B.3.2 HashSet.java

```
public class MainHashSet {
    public static void main(String args[]) {

        int N = Integer.parseInt(args[0]);
        int tot = N / 10;

        HashSet<Integer> set = new HashSet<Integer>();
        for (int te = 0; te < 10; ++te) {
            for (int i = 1; i <= tot; ++i) set.add(i);
            for (int i = 1; i <= tot; ++i) set.remove(i);
        }

    }
}
```

B.4 TreeSet

B.4.1 TreeSet.cpp

```
int main(int argc, char **argv) {

    int N = atoi(argv[1]);
    int tot = N / 10;

    srand(time(NULL));
    TreeSet<int> tree;
    for (int te = 0; te < 10; ++te) {
```

```
        for (int i = 1; i <= tot; ++i) tree.add(rand());
    }

    return 0;

}
```

B.4.2 TreeSet.java

```
public class MainTreeSet {
    public static void main(String args[]) {

        int N = Integer.parseInt(args[0]);
        int tot = N / 10;

        Random random = new Random();
        TreeSet<Integer> tree = new TreeSet<Integer>();
        for (int te = 0; te < 10; ++te) {
            for (int i = 1; i <= tot; ++i) tree.add(random.nextInt());
        }

    }
}
```