자료구조 기말시험 (Final Term Exam for Data Structure) – Phase 2

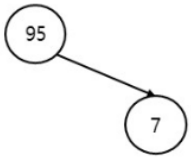
2021년 6월 17일

[Q] Radix search를 구현해 보자. Radix search는 radix sort처럼 radix로 검색을 하는 것이다. 즉, radix를 이용하여 데이터 요소를 binary tree로 표현한 후 검색을 한다. 이 문제에서는 정수를 bit-string으로 보고 각 자리수의 bit를 노드로 삼아 binary tree로 만든다. 예를 들어, 다음과 같은 정수 데이터가 순차적으로 주어진다고 가정하자.

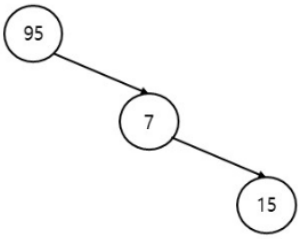
bit	MSB LSB
95	0000 0000 0101 1111
7	0000 0000 0000 0111
15	0000 0000 0000 1111
65	0000 0000 0100 0001
984	0000 0011 1101 1000
8	0000 0000 0000 1000
4	0000 0000 0000 0100
111	0000 0000 0110 1111
2	0000 0000 0000 0010
88	0000 0000 0101 1000
985	0000 0011 1101 1001
13	0000 0000 0000 1101

가장 먼저 95는 첫번째 데이터 요소이므로 root로 삼는다.

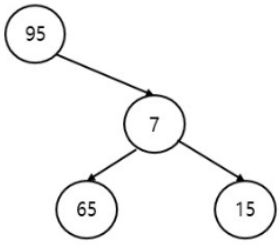
가장 오른쪽 비트(LSB)부터 왼쪽으로 검색 트리를 만들기 시작한다. 7은 첫번째 비트가 1이므로 95의 right child가 된다.



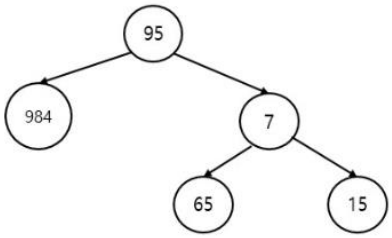
15는 첫번째 비트가 1이어서 95의 right child, 두번째 비트가 1이어서 7의 right child가 된다.



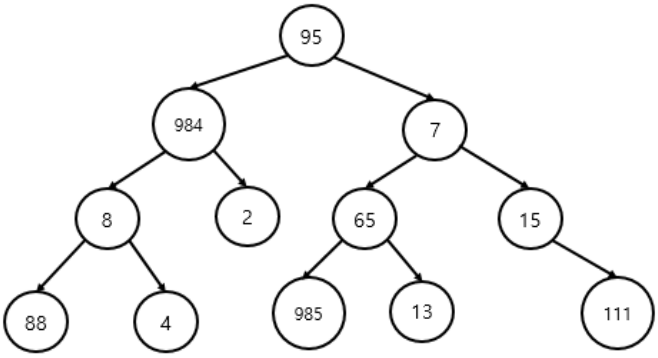
65는 첫번째 비트가 1이고 두번째 비트가 0이므로 7의 left child가 된다.



984는 첫번째 비트가 0이므로 95의 left child가 된다.



이와 같은 식으로 전체 tree를 다 그리면, 다음과 같다.



이 radix search tree를 위한 객체는 다음과 같다.

```
struct   Node {
           int        info;
           Node       *left;
           Node       *right;
};

class    RadixTreeType {
           Node       *root;
public:
           RadixTreeType();
           ~RadixTreeType();
           bool       RetrieveItem(int);
           void       InsertItem(int);
           void       DeleteItem(int);
           void       print();
};
```

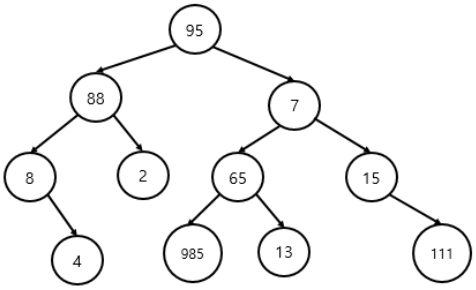
RetrieveItem(int v)는 v가 search tree에 있는지 확인하는 method이고,

InsertItem(int v)는 v를 search tree에 추가하는 method이고,

DeleteItem(int v)는 v를 이 search tree에서 제거하는 method이다.

DeleteItem(int v)는 지우고자 하는 노드에서 값을 지우고, 아무(left or right) subtree의 자식 노드로 치환하면 된다.

즉, 위의 트리에서 984를 지우면, 가장 왼쪽에 있는 leaf node인 88로 대치하면 된다.



print( )는 tree를 출력하는 method로 코드가 주어진다.

main 함수가 다음과 같이 주어지면,

```
int	main()
{
	int	data[] = {95, 7, 15, 65, 984, 8, 4, 111, 2, 88, 985, 13};
	RadixTreeType	tree;
	for(int i = 0;i < sizeof(data)/sizeof(int);i++)
		tree.InsertItem(data[i]);
	tree.print();
	cout << '₩n';
	cout << "Retrieval Test:₩n";
	cout << "₩t " << 983 << ": " << tree.RetrieveItem(983) << endl;
	cout << "₩t " << 777 << ": " << tree.RetrieveItem(777) << endl;
	cout << "₩t " << 985 << ": " << tree.RetrieveItem(985) << endl;
	cout << "₩t " << 12 << ": " << tree.RetrieveItem(12) << endl;
	cout << "₩t " << 15 << ": " << tree.RetrieveItem(15) << endl;
	cout << "₩t " << 13 << ": " << tree.RetrieveItem(13) << endl;
	cout << "₩t " << 6 << ": " << tree.RetrieveItem(6) << endl;
	cout << "₩t " << 111 << ": " << tree.RetrieveItem(111) << endl;
	tree.DeleteItem(4);
	tree.DeleteItem(7);
	cout << "After deleting 4 and 7:₩n";
	tree.print();
	cout << "Retrieval Test Again:₩n";
	cout << "₩t " << 983 << ": " << tree.RetrieveItem(983) << endl;
	cout << "₩t " << 777 << ": " << tree.RetrieveItem(777) << endl;
	cout << "₩t " << 985 << ": " << tree.RetrieveItem(985) << endl;
	cout << "₩t " << 12 << ": " << tree.RetrieveItem(12) << endl;
	cout << "₩t " << 15 << ": " << tree.RetrieveItem(15) << endl;
	cout << "₩t " << 13 << ": " << tree.RetrieveItem(13) << endl;
	cout << "₩t " << 6 << ": " << tree.RetrieveItem(6) << endl;
	cout << "₩t " << 111 << ": " << tree.RetrieveItem(111) << endl;
	cout << "Delete 95₩n";
	tree.DeleteItem(95);
	tree.print();
	return 0;
}
```

실행 결과는 다음과 같다.

```
95
984 7
8 2 65 15
88 4 985 13 111

Retrieval Test:
		983: 0
		777: 0
		985: 1
		12: 0
		15: 1
		13: 1
		6: 0
		111: 1
After deleting 4 and 7:
95
984 985
8 2 65 15
88 13 111

Retrieval Test Again:
		983: 0
		777: 0
		985: 1
		12: 0
		15: 1
		13: 1
		6: 0
		111: 1
Delete 95
88
984 985
8 2 65 15
13 111
```

RetrieveItem(int v), InsertItem(int v), DeleteItem(int v)을 구현하시오. 정수는 64비트라고 가정하시오.