한글 유니코드 텍스트의 효율적인 탐색을 위한 컴팩트 바이너리 트라이의 재구성*

정 규 철** · 이 종 찬*** · 박 상 준**** · 김 병 기*****

Reconstitution of Compact Binary trie for the Efficient Retrieval of Hangul UniCODE Text

Jung, Kyu Cheol · Lee, Jong Chan · Park, Sang Joon · Kim, Byung Gi

〈Abstract〉

This paper proposes RCBT(Reduced Compact Binary trie) to correct faults of CBT (Compact Binary trie). First, in the case of CBT, a compact structure was tried for the first time, but as the amount of data was increasing, that of inputted data gained and much difficulty was experienced in insertion due to the dummy nodes used in balancing trees.

On the other hand, if the HCBT realized hierarchically, given certain depth to prevent the map from increasing onthe right, reached the depth, the method for making new trees and connecting to them was used. Eventually, fast progress could be made in the inputting and searching speed, but this had a disadvantage of the storage space becoming bigger because of the use of dummy nods like CBT and of many tree links. In the case of RCBT in this thesis, a capacity is increased by about 60% by completely cutting down dummy nods.

Key Words : Compact Binary Trie, Binary Trie, Treemap, Leafmap, Dummy Node

Ⅰ. 서론

모바일 데이터베이스는 기존의 데이터베이스를 기준으로 변형된 것이기 때문에 용량이 상대적으로 크고 속도 또한 느린 단점이 있다[1]. 이러한 데이터베이스는 모바일 웹을 지원하기 위해 설계되어 서버 클라이언트 환경 하에서 작동한다. 그러므로 파일관리를 위해서는 기존의 알고리즘을 이용하여 직접 레코드 주소와 키를 매핑해야 하는데 모바일환경에 맞도록 필요한 디스크 공간을 가능한 한 작게 사용하여야 한다[5]. 이러한 매핑을 수행하기 위한 기존의 방법으로 해싱(Hashing), 확장성

* 이 논문은 2005년 정부(교육인적자원부)의 재원으로 한국학술진흥재단의 지원을 받아 수행된 연구임 (KRF-2004-005-D00147)
** 숭실대학교 정보미디어연구소 연구교수
*** 군산대학교 컴퓨터정보공학과 부교수(교신저자)
**** 군산대학교 컴퓨터정보공학과 조교수
***** 숭실대학교 컴퓨터학부 교수

해싱(Extensible hashing), 동적해싱(Dynamic hashing) 등이 있다. 특히 사전탐색이 잦은 자연어 처리 분야에서는 입력문자에 의해 찾는 트라이 (trie; 이하 T)를 채택하고 있다[2-4]. 최근에는 T의 단점을 보완한 CBT (Compact Binary Trie)와 HCBT(Hierarchical Compact Binary Trie) 채택하기도 하였다. 여기서 HCBT는 map이 오른쪽으로 증가하는 것을 막기 위해 일정 깊이를 주어 깊이에 다다르면 새로운 트리를 만들어 연결시키는 방법을 이용하였다. 결과적으로 입력과 검색 속도를 상당히 빠르게 진전시킬 수 있었으나 CBT와 마찬가지로 더미노드를 사용하고 여러 트리의 링크를 사용하기 때문에 저장공간이 커지는 단점을 안고 있다[6, 7].

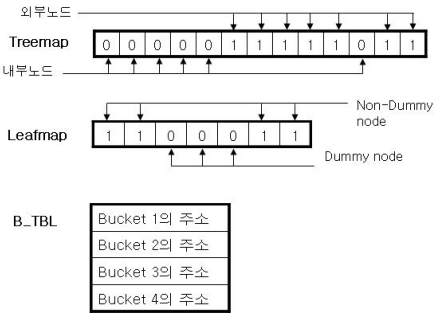
본 논문의 목적은 CBT를 이용하여 모바일환경에 적합한 탐색이 빠르고 색인 용량이 적은 알고리즘으로 재설계하는 것이다. 논문의 구성은 다음과 같다. 제 2장에서는 기존에 연구된 이진트라이와 CBT에 대해 살펴본다. 제 3장에서는 CBT의 더미노드를 제거하여 트리 길이를 크게 줄이기 위해 제안한 RCBT를 설명한다. 제 4장에서 CBT와 제안한 RCBT와 비교 실험, 마지막 제 5장에서 결론 및 향후 발전방향에 대해 기술한다.

II. 이진트라이와 CBT, HCBT

트라이는 탐색 트리이므로 여기서는 키를 버킷주소로 맵핑 하기위해 이진트라이를 사용한다. 이진코드의 순서열은 문자들의 변환 값으로 되어있고 키 값처럼 사용되므로 왼쪽 간선(edge)은 0으로 레이블(label) 되고 오른쪽 간선은 1로 레이블 된다. 이는 키 집합 K에 대한 이진트라이로서 외부노드의 수가 내부노드 수보다 하나 더 많다.

이진트라이가 구현될 때 트리의 노드보다 훨씬 더 많은 저장 공간을 요구한다. 따라서 Jonge은 집적한 비트스트림(bit stream)으로 이진트라이를 압축하는 방법인 CBT를 제안하였다[9, 10]. 이 T는 3개의 요소- treemap, leafmap과 B_TBL-로 구성된다. treemap은 트리의 상태를 표현하고 전위 순회(preorder traversing)로 산정되는데, 모든 내부노드 방문에 대해서는 0을 보내고 모든 외부노드방문에 대해 1을 내보낸다. leafmap은 각 외부노드의 상태 즉 더미인지 아닌지를 표현하고 전위 순회함으로 얻어진다.

만일 외부노드가 더미이면 일치하는 비트를 "0"으로 바꾸고 그렇지 않으면 "1"로 설정한다. B_TBL은 버켓주소를 저장한다. <그림 1>는 CBT를 보인다. 이진T에서 저장 공간은 버켓에 대한 포인터를 제외하고 24바이트 = 192비트가 요구한다. 왜냐하면 한 포인터 당 2바이트가 소요되는데 이 트리는 12포인터를 가지고 있기 때문이다. 그러나 treemap과 leafmap의 길이가 20비트이기 때문에 CBT를 사용할 경우 단지 20비트만 요구된다.

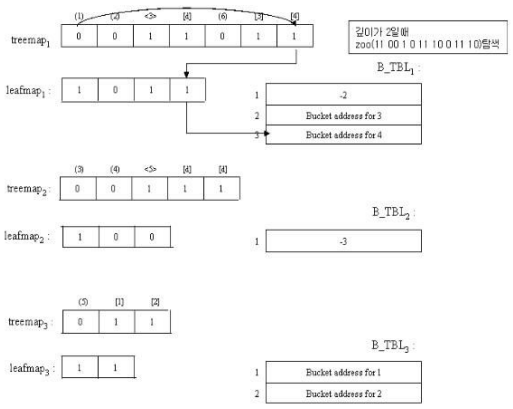


<그림 1> CBT 표현

그러나 CBT의 문제점은 삽입·삭제 시 비트 스트링의 잦은 좌우 이동에 있다. 이를 해결하기 위해 트리에 일정의 깊이(depth)를 두어 그 이상 늘어나게 되면 트리를 분리하는 기법을 적용하였다[11, 12]. 이 깊이를 분할깊이라고 부르며, 이 작은 트리를 분할된 트리라고 부른다. 이 같은 방법으로 분리된 트리는 계층적 이진 트라이며, 이 트라이를 CBT구조에 기반을 둔 표현한 기법을 HCBT(Hierarchical Compact Binary Trie)이다. 계층적 이진 트라이에서 i번째 분리된 트리에 일치하는 HCBT는 treemap-i, leafmap-i와 B_TBL-i로 구성하지만, 분리된 다

음 트리의 포인터가 되는 외부노드는 특별한 외부노드로 간주된다.

<그림 2>에서 처럼 하위 트리에 존재할 경우 복잡하게 보이지만 키 "zoo"를 탐색할 경우에는 오직 분리된 트리 1의 HCBT 사용에 의해 탐색될 수 있다. 따라서 탐색의 시간비용은 CBT보다 더 우수하다.



<그림 2> 깊이가 2인 HCBT에서 "zoo" 탐색

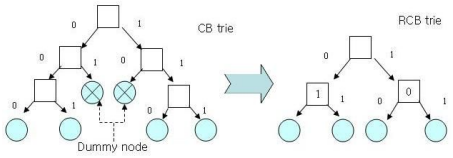
이와 같이, HCBT는 CBT의 삽입 삭제 시 비트열의 이동을 최소화하고 트리의 오른쪽에 위치한 키의 검색 시간을 단축시키기 위한 알고리즘이다. 이는 새로운 트리를 만들어 트리의 오른쪽을 탐색할 경우 분리한 트리는 탐색에서 제외시키는 방법을 취하였다. 또한 깊이가 정해진 크기를 넘지 않는 한 CBT와 같은 방법으로 삽입되기 때문에 CBT에서 변환도 가능하게 된다. 따라서 CBT에 비해 평균탐색 시간 또한 단축을 시킨다. 그러나 버킷테이블에 분리된 트리의 주소를 저장해야 하므로 주기억장치에 저장된다면 빠른 속도의 삽입·탐색을 할 수 있지만 보조기억장치에 버킷테이블을 저장한다면 분리된 트리를 참조할 때 마다 디스크를 접근해야한다. 버킷테이블을 주기억장치에 저장할 경우 실제 데이터의 주소이외에 링크 주소까지 모두 저장되므로 기억장치에 오버헤드가 발생한다. 따라서 대량의 사전인 경우 이 HCBT는 적합하지 않다.

Ⅲ. RCBT의 구성

3.1 RCBT의 개요

CBT는 더미노드를 효율적으로 이용한 집약적인 이진 탐색 트리의 표현법이다. 이 더미노드의 역할은 트리의 균형을 맞추고 트리 순회를 자연스럽게 하기 위해 제안된 방법이다. 삽입 시 더미노드에 도달할 경우 버킷이 비어있는 것으로 간주하여 트리의 구조에는 영향을 주기 않고 외부메모리에만 공간을 확보하여 쉬운 삽입을 가능케 한다[3].

그러나 대량의 데이터가 다루어야할 경우 균형을 위해 과도하게 생성된 더미노드로 인해 탐색 시 많은 더미노드를 순회하여야 하며, 삽입·삭제 시 더미노드로 인해 과다한 Shift 연산이 발생한다. 이러한 문제를해결하기 위해 제안된 HCBT는 트리를 분리하여 treemap과 leafmap이 길어지는 것을 방지하였다. 그러나 주기억장치에 로드 되는 데이터를 줄이기 위해 버킷테이블이 주기억장치가 아닌 보조기억장치에 위치할 경우 링크된 트리 주소를 탐색하기 위한 잦은 입출력으로 오히려 더 많은 시간을 요구하게 된다[2].



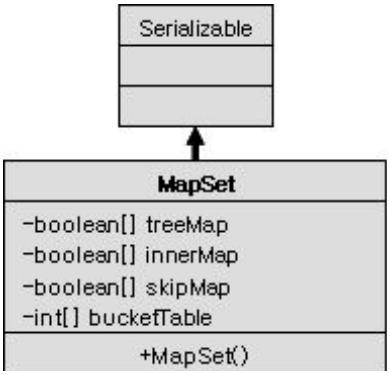
<그림 3> 트리 구조의 비교

또한 CBT와 HCBT 모두 한글과 같이 16비트의 이진 코드를 가진 유니코드들의 경우에 더미노드가 방대하게 증가한다.

본 장에서는 더미노드를 별도로 관리하기 위한 맵의 사용으로 treemap의 길이를 줄이고, 보조기억장치를 한 번의 접근으로 탐색하여 잦은 입출력으로 인한 시간 증가를 억제하기 위한 새로운 트리구조인RCBT(Reduced

CBT)를 제안한다.

RCBT에 대한 데이터 객체는 <그림 4>과 같이 정의한다.



<그림 4> RCBT의 객체

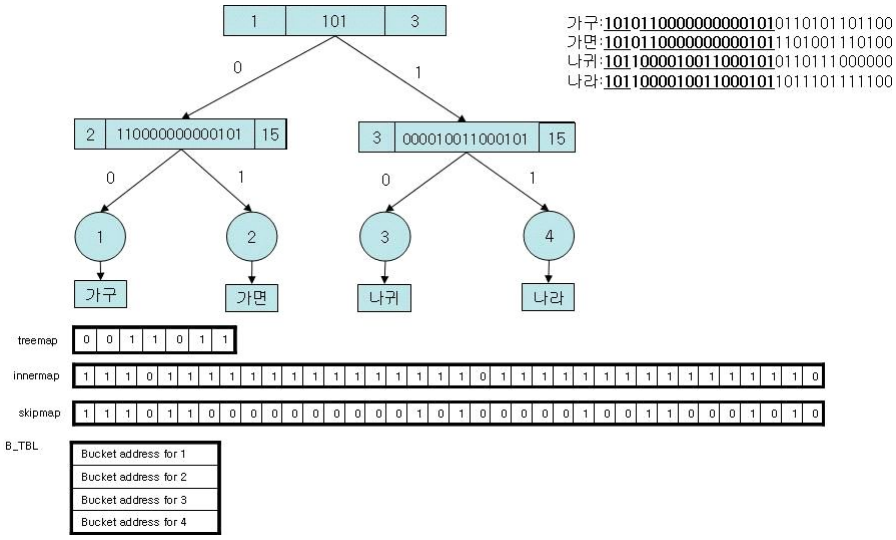
CBT에서는 초기상태로서 treemap이 '011', leafmap이 '00'이었으나 RCBT에서는 모두 '0'으로 초기화하였다. 왜냐하면 CBT는 더미노드를 허용하면서 이진트리의 구조를 유지해야 하기 때문인데, RCBT에서는 둘이상이 삽입이 되어야 이진트리를 구성할 수 있기 때문이다.

3.2 RCBT의 탐색

RCBT는 더미노드를 가지고 있지 않은 treemap을 표현하기 때문에 더미노드의 상태를 표현한 leafmap은 더 이상 필요하지 않다. 반면 내부노드에 생략된 비트를 표현하기 위한 innermap, 수집된 비트들의 정보를 가지고 있는 skipmap을 정의한다. 이 innermap은 수집된 양에 관한 내용을 가지며 skipmap은 수집된 비트들의 내용을 담고 있다. 그리고 삽입키의 물리적 위치를 가지고 있는 B_TBL을 포함한다.

<그림 5>과 <그림 6>은 각각 RCBT의 구조와 탐색 알고리즘을 보인다.

예를 들어 "나귀"를 탐색하는 절차를 보자. <그림 5>에서 먼저 treemap의 첫 비트가 '0'이고 innermap의 위치 포인터인 ip가 0인 비트가 '1'이면 첫 번째 내부노드에 하나 이상의 비트들이 존재한다는 의미이므로 innermap의 비트가 '0'인 3번 비트까지 ip를 하나씩 전진 시킨다. 그리고 kp도 skipmap의 위치포인터인 sp와 같기 때문에 함께 전진한 후 treemap의 위치 포인터인 tp를 전진한다. 즉, 3개의 맵을 동시에 전진 시킨다.



<그림 5> RCBT 구조

(begin)
Step-0 : kp: 입력키의 위치 포인터, tp:treemap의 위치 포인터, ip: innermap의 위치 포인터, sp:skipmap의 위치 포인터.
Step-1 : kp=0, tp=0, ip=0, sp=0.
Step-2 : **If** tp가 가리키는 treemap의 비트가 '0'이면
Goto Step-3,
Else Goto Step-5
Step-3 : **If** ip가 가리키는 innermap의 비트가 '1'이면 ip의 비트가 '0'일 때까지 ip의 비트와 sp의 비트를 하나씩 증가하면서 kp가 가리키는 삽입 키의 비트와 sp가 가리키는 skipmap의 비트를 비교.
If kp의 비트와 sp의 비트가 서로 다르면 False.
If ip가 가리키는 innermap의 비트가 '0'이면
Goto Step-4.
Step-4 : kp를 하나 증가.
If kp가 가리키는 비트가 '0'이면 Goto Step-2,
Else treemap에서 현재 tp의 위치부터 앞으로 전진하여 '0'과 '1'비트의 수가 같은 비트 다음으로 이동.
ip를 현재 tp까지 0의 수와 같은 곳으로 이동.
If ip가 가리키는 비트의 이전 비트가 1이면
Goto Step-3,
Else Goto Step-2.
Step-5 : **If** 입력키와 버킷안의 데이터가 같으면 Return True,
Else Return False.
(End)

<그림 6> 탐색 알고리즘

Keyword의 위치포인터인 kp가 4인 곳의 비트가 1이므로 tp를 오른쪽으로 이동시키기 위해 tp를 3으로 이동시킨다. 이때 현재 위치에서 0의 개수가 3개이므로 ip를 '0'이 3개 있는 곳으로 이동시킨다. 이는 treemap의 0의 개수는 내부노드의 수와 동일하므로 innermap 에서도 동일한 위치로 이동시켜준다. 이때 이동한 위치의 바로 앞 비트가 1이므로 현재 내부노드에서는 수집된 비트가 있기 때문에 위 방법과 동일하게 반복하여 innermap 의 비트가 '0'인 20번 비트까지 ip를 이동시킨다. 그리고 kp도 skipmap의 sp와 같기 때문에 이동 시킨후 tp를 전진한다. kp가 20일 때 비트가 0이므로 treemap을 하나 전진시킨 비트가 1이다. 현재 1의 개수가 2이므로 3번째 버킷에서 입력 값과 같은 것을 발견하여 탐색을 종료한다.

3.3 RCBT의 삽입

키의 삽입은 CBT와 유사한 방법으로 삽입된다. 단, 중복되는 키 값이 있을 경우 CBT는 서로 다른 비트가 보일 때 까지 더미 노드를 계속해서 생성하지만 RCBT는 비트가 동일할 경우 내부노드에 수집한다는 것이다. 그런데 CBT는 무조건 삽입하고자하는 노드가 더미노드이면 더미노드에 해당하는 버킷에 키를 삽입하고 버킷이 차있을 경우에만 새로운 기본 트리를 삽입하므로 삽입 과정은 매우 간단하다. 하지만 RCBT에서는 다음과 같은 사항을 고려하여야 한다.

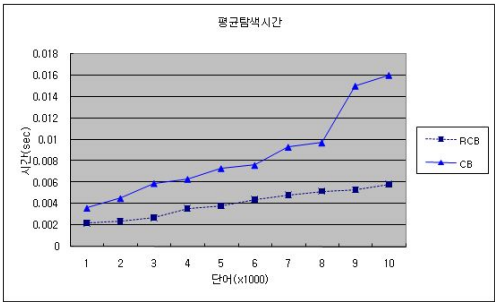
① 모아진 내부노드 안에서 분기가 일어날 경우
② 기존 버킷에 다른 키 값이 존재하여 서브트리를 생성해야할 경우

먼저 내부노드에 모아진 비트에서 분기가 일어나는 경우, 삽입되는 키의 분기 위치 비트가 '0', 또는 '1'인지에 따라 다르게 설정하여야 하고 다음 버킷에 다른 키 값이 존재하는 경우 서브트리를 생성한 후 기존 키의 현재 위치 비트와 삽입하고자 하는 키 값의 현재 위치비트를 비교하여 생성하는 서브트리의 내부 노드중복되는 비트를 모아두고 다른 비트들의 크기를 비교하여 '0'이면 왼쪽, '1'이면 오른쪽으로 이동시킨다.

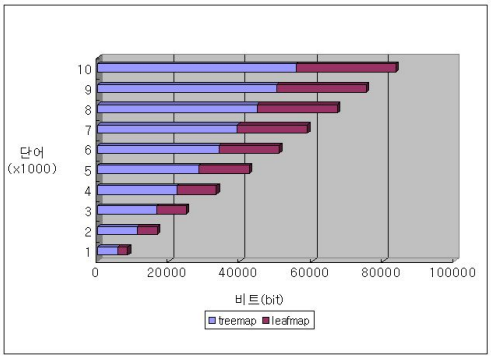
IV. 실험 및 평가

탐색 실험을 위해 형태소 분석기용 명사사전[13]을 한글 유니코드로 변환 하여 1만 단어를 추출하고 기본데이터를 만들어 CBT와 RCBT를 탐색 시간과 생성되는 맵의 크기를 실험 하였다. 단, 본 논문의 취지인 주기억장치에서 소용량의 색인 공간이기 때문에 보조기억장치를 사용하는 HCBT는 실험에서 제외하였다. 개발 언어는 범용성을 고려하여 Java를 이용하였다[8].

우선 실험에 앞서 전제가 되어야 하는 것은 CBT는 참고 자료를 이용하여 직접 프로그램을 작성하였고 실험



<그림 7> 평균탐색 시간 비교



<그림 8> CBT에서 맵크기 변화

데이터는 참고 저자들이 제시한 데이터가 아닌 RCBT를 구성하면서 만든 데이터를 사용했기 때문에 실제 참고 저자들의 실험과결과가 다를 수 있다는 것을 전제로 하고자 한다. 또한 실험은 빠른 결과를 위해 모바일용으로 직접 구현하지 않고 데스크 탑에서 속도와 테이블의 크기를 계산하였다.

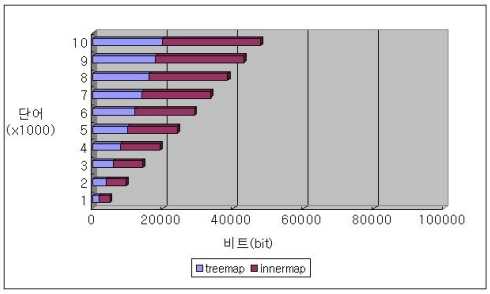
4.1 탐색 시간 비교 실험

비교실험은 색인파일의 크기를 1000개씩 늘리면서 그 중 무작위로 100개의 단어를 선정한 후 탐색 시간을 측정하였다. 그리고 측정단위가 미세한 밀리 초가 되어 시스템의 상태에 따라 다른 결과가 나올 수 있으므로 정확한 측정을 위해 같은 조건에서 3번의 탐색을 더하여 평균탐색 시간을 계산하였다.

<그림 7>에서 보듯이 CBT에 비해서는 RCBT 속도가 평균 60%정도 향상되었다. 이 결과는 더미노드가 없어지면서 트리의 크기가 전체적으로 줄어든 것에 따른 효과라고 볼 수 있다.

4.2 색인 용량의 비교 실험

<그림 8>은 CBT에서 실험한 데이터를 가지고 맵의 크기를 살펴본 결과이다. treemap이 leafmap의 2배 정도 많은 공간을 차지하는 것을 볼 수 있다. 더미노드의 비율은 leafmap의 64%에 해당할 정도로 많은 부분을 차지하고 있다.

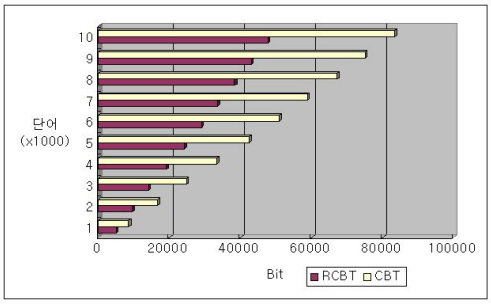


<그림 9> RCBT의 treemap과 innermap 크기 변화

이 때문에 탐색과 삽입 삭제 시 많은 시스템의 부담을 초래하고 있다. RCBT의 경우 탐색을 위해서는 treemap과 innermap만 필요하기 때문에 주기억장치에 로드되는 두 트리의 합만 계산하였다. 만약 여기에 skipmap의 크기를 포함하더라도 innermap만큼의 용량이 더 필요하므로 실험에서는 제외하였다. 위의 CBT와 RCBT의 생성 크기를 비교해보면 평균 57%정도 적게 생성된다. 또한 다음 트리를 탐색해나가기 위해 버킷테이블을 찾아야 하므로 기억장치에는 treemap과 leafmap이외에 버킷테이블이 동시에 적재된다. 이 때문에 treemap과 leafmap의 크기만으로는 의미가 없다.

그리고 CBT는 <그림 10>에서 처럼 83, 000비트 즉, 10Kbyte면 트리를 다 표현할 수 있고, RCBT는 47, 000비

트 즉, 5Kbyte면 트리를 전부 표현이 가능하다. 따라서 RCBT가 월등히 적은 공간을 필요로 하는 것을 알 수 있으며, 약 50%정도의 공간을 절약하였다.



<그림 10> CBT와 RCBT의 map 크기

V. 결론 및 향후 과제

본 논문에서는 이진트라이를 축약된 배열 구조로 표현한 CBT의 단점을 보완하여 한 번의 보조기억장치의 접근으로 빠르게 탐색할 수 있고 트리 길이에 부담을 주는 더미노드를 제거하기 위하여 RCBT를 제안하였다. 이 RCBT는 더미노드를 직접 외부노드에 표현하지 않고 내부노드에 표현하는 방식을 채택하였다. 기존의 트리와는 달리 leafmap이 아닌 innermap을 사용하여 내부노드에 더미노드에 해당하는 중복되는 비트를 보관할 경우 '1'로 설정하고 내부노드는 '0'으로 설정한다. 이렇게 하면 더미노드는 만들어지지 않고 트리의 깊이도 상당히 줄어듦을 알 수 있다. 또한 탐색 시간도 CBT에 비해 향상되어 소규모의 모바일 환경이나 임베디드 환경에 적용이 가능하다.

참고문헌

[1] 강성윤외 2, "자바 모바일 프로그래밍", 대림, 2002.

[2] 김철수, "절단검색을 지원하는 전자사전 구조", 한국정보과학회, 제9-C권, 제1호, 2003.

[3] Aho A. V., Hopcroft J. E., & Ullman J. A. (1983). "DataStructuresandAlgorithms." Reading, MA: Addison-Wesley.

[4] Aoe J., Morimoto K., Shishibori M., & Park K. (1996). "A Trie Compaction Algorithm for a Large Set of Keys." IEEE Transactions on Knowledge and DataEngineering, Vol. 8, No. 3, pp. 476-491.

[5] 리처드 헌터, "유비쿼터스 공유와 감시의 두얼굴", 21세기 북스, 2003.

[6] 문봉재, "모바일 디바이스에서 닷넷 애플리케이션 구축하기", Microsoft Press, 2003.

[7] 정규철, 이진관, 장혜숙, 박기홍, "CBDS 트리를 이용한 모바일 기기용 저용량 사전 구축에 관한 연구", 한국컴퓨터정보학회, 제10권 제5호, 2005.

[8] 김윤명, "뇌를 자극하는 Java 프로그래밍", 한빛미디어, 2006.

[9] Masami, S. Aoe, J. "An Order Searching Algorithm of Extensible Hashing" Inter. j. Computer Math., Vol. 63, 1997, pp. 179-201.

[10] Aoe, J, Park K(1996), A trie compaction algorithm for a large set of keys, IEEE Trans. on Knowledge and Data Engineering, 8(3).

[11] Jonge, W. D(1987), Two access methods using compact binary tree, IEEE Trans. on Software Engineering, 13(7), 7999-809.

[12] Jung. M, Shishibori. M, Tanaka. A, J. Aoe: "ADynamicConstructionAlgorithmfortheCompactPatriciaTrieUsingtheHierarchicalStructure", Information Processing & Management, Vol. 38, No. 2, pp. 221-236.

[13] 고려대학교 자연어처리 연구실: nlp.korea.ac.kr, 2002.

논문접수일 : 2009년 5월 1일
수 정 일 : 2009년 5월 15일
게재확정일 : 2009년 5월 20일

■ 저자소개 ■

정 규 철
Jung, Kyu Cheol

2008년 4월~현재 숭실대학교 정보미디어기술연구소 연구교수
2006년 2월 군산대학교컴퓨터과학과 (이학박사)
1999년 2월 군산대학교컴퓨터과학과 (이학석사)
1995년 2월 군산대학교 컴퓨터과학과 (이학사)

관심분야 : 정보검색, 센서네트워크
E-mail : kcjungs@lycos.co.kr

이 종 찬
Lee, Jong Chan

2005년 3월~현재 군산대학교 부교수
2000년 10월 한국전자통신연구원 선임연구원
2000년 8월 숭실대학교컴퓨터학과 (공학박사)
1996년 8월 숭실대학교컴퓨터학과(공학석사)
1994년 2월 군산대학교 컴퓨터과학과 (이학사)

관심분야 : 이동통신, 원격제어시스템
E-mail : chan2000@kunsan.ac.kr

박 상 준
Park, Sang Joon

2007년 3월~현재 군산대학교 조교수
2005년~2007년 숭실대학교 정보미디어기술연구소 연구교수
2002년~2003년 런던대 ISG박사후 과정
2002년 숭실대학교 컴퓨터학과 (공학박사)
1998년 숭실대학교 컴퓨터학과 (공학석사)
1996년 동국대학교 전산학과 (공학사)

관심분야 : B3G 이동통신, 센서네트워크
E-mail : lubimia@kunsan.ac.kr

김 병 기
Kim, Byung Gi

1982년 3월~현재 숭실대학교 컴퓨터학부 교수
1979년 3월 경북대학교 전자공학과 전임강사
1997년 한국과학기술원 전산학과 (이학박사)
1979년 한국과학기술원 전산학과 (이학석사)
1977년 서울대학교 전자공학과 (공학사)

관심분야 : 유비쿼터스 디지털방송, HMIPv6
E-mail : bgkim@ssu.ac.kr