論文
3-2-13

수량적 접근 방법에 의한 이진 검색 트리 불균형도에 따른 검색 성능 비교 분석

(Quantitative approach to analyze searching efficiencies varying degrees of imbalance in a binary search tree)

김 숙 영*
(Sook-Young Kim)

요 약

트리 재균형 과정을 최소화 하기 위하여 이진 검색 트리의 불균형도에 따른 검색 성능에 관한 수량적 정보를 얻기 위한 실험이 행하여졌다.

트리를 구성하는 노드들의 좌, 우 서브트리 높이 차 인 균형 인수에 의하여 불균형도를 수량화 한 결과 불균형도가 심해질수록 검색성능이 저하됨이 실험 자료들에 의하여 확률적으로 증명 되었고 ($p<0.01$), 노드 개수와 평균 검색 횟수 관계를 설명하기 위한 모형으로는 로그 모형 보다 선형 모형이 적합한 경향을 보였다.

그러나 균형 인수 크기가 3 미만인 노드들 만으로 구성된 이진 검색 트리의 성능은 높이 균형 이진 트리에 비하여 저하되지 않는 것으로 평가된다.

본 연구 결과들은 이진 트리를 자료 구조를 사용하는 소프트웨어 관리에 적용될 수 있을 것이다.

ABSTRACT

To minimize restructuring cost of a tree, experiments were conducted to collect quantitative information of searching efficiencies varying degrees of imbalance in a binary search tree. Degrees of tree imbalance were measured by a balance factor, an absolute value of height difference of left subtree and right subtree in a binary search tree.

The average number of comparisons increased ($p<0.01$), and searching efficiency of O(n) was more appropriate rather than O(logn), as degrees of imbalance in a binary search tree deteriorated.

However, there were no significant differences of searching efficiencies in height balanced trees and trees with subtrees to have height 3 less than the other ($p>0.05$).

Therefore, the findings would be applicable to maintain searching efficiency of a software with a binary search tree.

* 정회원 : 안산공과대학 컴퓨터정보과 조교수

논문접수 : 2002. 2. 5.
심사완료 : 2002. 2. 23.

1. 서론

정보 관리에 필수적인 검색(search), 추출(retrieval),삽입(insert),수정(modify), 삭제(delete) 등의 기본연산들 중 연산에 필요한 레코드를 찾는 검색 연산이 가장 핵심적 이다. 따라서 검색 연산이 효율적으로 수행될 수 있는 자료 구조 표현에 대한 연구가 필요하다. 검색할 자료들의 집합이 선언되어 있고 새로운 레코드삽입이나 기존 레코드 삭제가 허용되지 않는 정적인 테이블을 이용한 검색에서는 이진 검색이나 피보나치 검색 알고리즘을 적용할 수 있다. 그러나 검색 자료의 삽입, 삭제연산이 임의로 발생하는 동적 테이블을 이용한 검색 에서는 검색의 키가 되는 자료와 트리의 노드상에 존재하는 자료들 간의 반복적인 비교 연산에 의한 이진 검색트리 구조가 많이 적용된다 [3].

이진 검색 트리는 트리를 구성하는 모든 노드들의 좌,우 서브트리 또한 모두 이진 검색 트리이며 좌 서브트리에 있는 모든 자료들은 루트 노드의 자료보다 수치적 또는 알파벳 순에서 적으며 우,서브트리에 있는 모든 자료들은 루트의 자료보다 커야 하는 속성이 항상 유지 되어야 한다. 따라서 이진 검색 트리에서 자료를 검색하기 위하여 좌,우 서브트리들의 높이 차이가 나지 않도록 최소의 높이를 유지하는 균형을 이룬 트리가 생성 되어야 한다. 이론적으로 트리를 구성하는 모든 노드들의 좌,우 서브트리 높이 차이가 최대 1인 트리를 높이 균형 트리로 정의하며 검색 성능이 O(logn) 으로 향상 된다는 주장이 존재한다 [4].

그러나 이진 검색 트리는 동적인 환경하에서 검색이 수행 되므로 노드의 삽입, 삭제 등으로 트리의 균형을 잃게 된다. 실제 6-7개의 노드를 가진 트리 생성 과정에서 높이 균형 검색 트리가 생성될 확률은 10% 정도에 불과하다 [2]. 트리 생성 과정 중 균형을 잃은 트리에 대하여는 불균형 방향으로 LL 회전, LR 회전, RL 회전, RR 회전 등 재균형 알고리즘을 적용하여 높이 균형 트리로 변환할 수 있으나 최악의 경우에는 트리 전체를 재구성해야 하므로 많은 노력을 필요로 한다 [5]. 즉 재균형 과정에서 많은 비교 연산들을 필요로 하는 경우에는 이진 검색 트리의 검색 성능이 향상된다 평가하기 어려울 것이다.

따라서 본 연구 에서는 트리 불균형 정도에 따른 검색 성능 저하를 수량적으로 분석 하여 재균형 과정이 검색 성능 향상에 도움이 되지 않는 불균형도 크기를 찾기 위한 목표로 다음 가설들을 설정 하였다.

첫째, 트리의 불균형도 크기 및 불균형 방향에 따른 평균 검색 횟수의 차이가 존재하는가?

둘째, 트리 검색 성능이 로그형에서 선형으로 변하는 트리의 불균형도 크기를 추정한다.

본 연구 결과들은 현재 까지 문헌에서 발견되기 어려운 높이 균형 이진 검색 트리 이론에 관한 실험적 근거를 제공하며, 트리 불균형도 크기별로 검색 성능을 평가한 결과들은 트리 검색 성능 연구에 유익한 정보들을 제공할 수 있을 것이다.

2. 방법

본 연구 실험들은 3 개 이상 7개 이하의 노드들을 가진 이진 검색 트리를 대상으로 행하여 졌다.

2.1 이진 검색 트리 생성

검색 자료의 입력 순서에 따라 여러 형태의 이진 검색 트리들이 생성될 수 있으므로 본 실험 에서는 n 개 노드들을 가진 트리의 자료값을 {1,2,,,,n} ($3 \leq n \leq 7$)로 n ! 개의 모든 가능한 이진 검색 트리들을 생성하였다.

2.1.1 트리의 자료 구조

트리 에서의 노드 위치는 이차원 배열(node[i][j])에 의하여 선언 되었고 트리의 레벨을 배열의 행으로, 레벨 내에서 최 좌측 노드로 부터의 순서를 배열의 열로 표현하였다. node[i][j] 는 완전 이진 트리 (i=0,1, ...,n, j:0,1,...2^n-1)의 크기를 확보하며 삽입된 노드 위치에 대응되는 배열의 위치에는 자료값을, 노드가 삽입되지 않은 위치에 대응되는 배열 원소들은 0의 값을 가진다.

트리에 삽입된 노드들의 자료값, 좌 서브트리 높이 및 우 서브트리 높이 등의 정보는 크기가 n 인 일차원 배열들로 표현 되었다.

2.1.2 노드의 삽입

트리 생성 시 n 개 노드 자료들의 집합에서 첫 번 째 원소는 루트 노드 (node[0][0])가 되고 새로운 원소가 삽입 될 때 마다 루트 노드로부터 시작하여 자료값을 비교하여 삽입될 노드의 자료값의 크기가 적으면 다음 레벨의 좌측으로, 자료값 크기가 크면 우측으로 이동하는 과정을 반복하여 부모가 될 노드 (node[i][j]) 위치를 찾은 후 삽입될 자료값이 크면 node[i][j*2+1] 위치에, 작으면 node[i+1][j*2] 위치에 원소를 삽입하여 트리를 생성하는 다음 알고리즘을 적용하였다.

```
node[0][0]=key_in[0];        /* 루트 노드 생성
  /* 루트 노드를 제외한 나머지 6개 노드들을 삽입
  for (k=1; k<7; k++)   {
    i=0;    j=0;
    insert_key = key_in[k];
    found='f';
    c_elmt=node[i][j];
   while ((node[i][j] != 0) && (found == 'f'))
    {
     iter[k]++;            /* 검색 횟수 증가
     /* 삽입될 노드 값이 크면 우측으로 이동
      if ((found == 'f') &&
     (insert_key > c_elmt) && (c_elmt !=0))   {
          /* 우측 서브트리로 생성
          if (node[i+1][j*2+1] == 0)    {
              node[i+1][j*2+1]=insert_key;
             found = 't';
              pos1[k]=i+1;  /* 배열의 행 위치
             pos2[k]=j*2+1; /* 배열의 열 위치
               }
               i++;
               j=j*2+1;
               c_elmt = node[i][j];
                     }
     /* 삽입될 노드 값이 작으면 좌측 으로 이동
      if ((found == 'f') &&
     (insert_key < c_elmt) && (c_elmt !=0))   {
                  /* 좌측 서브트리의 노드로 삽입됨
                   if (node[i+1][j*2] == 0)   {
                   node[i+1][j*2]=insert_key;
                   found = 't';
                   pos1[k]=i+1;
                   pos2[k]=j*2;
                            }
                   i++;
                   j=j*2;
                   c_elmt = node[i][j];   }
                   }                 }
```

2.1.3 트리 불균형도 추정

n 개 노드를 가진 트리가 생성된 후 트리의 가장 하위 레벨 노드 (node[i][j])부터 시작하여 node[i-1][j/2] 에 노드 자료 값이 존재하면 노드의 트리 높이를 증가 시켰다.

각 노드들은 좌,우 서브트리를 가지고 있으므로 j mod 2=0 이면 좌측 서브트리 높이를 1 증가 시켰고 j mod 2=1 이면 우측 서브 트리 높이를 1 증가 시키는 다음 알고리즘을 적용 하였다.

```
if ((spos1[n] == a-1) && (spos2[n] == b/2))  {
     /* 부모 노드를 찾았으면
    /* 좌, 우 서브트리 중 높은 높이 값을 찾음
       if (h1[m] >= h2[m])      c=h1[m];
       if (h1[m] < h2[m])       c=h2[m];
       if ((b % 2) == 0)

         /* 우측 서브트리 노드 인 경우
           h1[n]=1+c;
       if ((b % 2) == 1)
        /* 좌측 서브트리 노드인 경우
         h2[n]=1+c;
         }
```

트리 불균형도는 각 노드에서 [좌 서브트리 높이 - 우 서브트리 높이] 인 균형 인수로 수량화 되었다. 불균형 방향은 균형 인수의 부호로 + 인경우는 좌측으로 트리 구조가 치우치고, - 인 경우는 우측으로 트리 구조가 치우치고, 0 인 경우는 좌, 우측 균형되게 서브 트리가 생성됨을 표현한다 [10].

2.1.4 통계 분석

생성된 트리들을 노드 삽입 위치별로 그룹화 하여 노드 개수별로 상이한 트리 개수를 얻었다. 불균

형 크기 별로 검색 횟수들을 추정하였고 차이의 존재 유무를 일원 분산 분석법 및 Tukey 의 다중 비교에 의하여 확률적으로 증명하였다.

이진 검색 트리의 성능이 로그형에서 선형으로 저하되는 불균형도 크기를 찾기 위하여 불균형도 크기별로 트리 높이(X) 와 검색 횟수(Y) 관계를 설명하는 회귀 모형을 추정하였다 [1].

3. 결과

3.1 생성된 이진 트리 기술

생성된 이진 검색 트리의 균형도를 기술한 결과는 표 1 에 있다. 삽입된 노드 위치 별로 그룹화 하여 얻은 3개 이상 7개 이하 노드 개수에 대한 상이한 이진 트리의 개수는 5, 14, 42, 132, 429 개로 수학적으로 증명 가능한 이론상의 개수인

$\binom{2n}{n}\frac{1}{n+1}(n=3,4,5,6,7)$식에 의하여 얻은 결과와 일치하였다 [2,6].

수식적으로 높이 h 인 트리의 최대 노드 개수는 2^{h+1} -1이므로 n 개 노드의 트리에서는

2^{h+1} -1 ≥ n이며 h에 관하여 수식을 정리하면

$h \geq \lceil \log_2(n+1) \rceil - 1$ 이다 [8].

따라서 노드가 세 개인 트리의 최소 높이는 1, 네 개 이상 노드의 트리의 최소 높이는 2 인 <표 1> 결과들은 유도된 수식을 실험적으로 증명한다.

불균형도 크기를 평가하는 균형 인수 결과는 노드 개수가 3 및 7 일 때 완전 이진 트리가 생성되어 0 인 균형 인수가 존재 할 수 있으며 완전 사향 트리에서 균형 인수 크기가 최악의 트리 높이인[노드 개수 - 1] 로 된다는 이론을 증명하였다.

노드 개수를 독립 변수(X) 로, 검색 횟수를 종속 변수(Y) 로 회귀 모형을 추정한 결과최소의 검색 횟수는 Y=8.3 · log(X)-7.8, 최대 검색 횟수는 Y=-11.5X+4.5, 평균 검색 횟수는 Y=10.6 · log(X)-9.7 모형이 설명력이 높은 모형으로 추정되었다.

그러므로 이진 트리 검색의 성능은 평균 보다 우수한 경우는 O(logn), 최악의 경우는 O(n) 이론에 관한 근거를 제시하였다 [7].

3.2 트리 불균형도에 따른 이진 검색 트리 성능 비교

<표 2> 트리 불균형도에 따른 평균 검색 횟수

<Table 2> Average number of comparisons varying degrees of tree imbalance

노드수	불균형도	음수	양수	음, 양수	합
3	0				(n=2) 2.00±0.00
	1				
	2	(n=1) 3.00	(n=1) 3.00	(n=2) 2.50±0.50	(n=4) 2.75±0.25
4	0				
	1	(n=4) 4.00±0.00	(n=2) 4.00±0.00	(n=6) 3.67±0.21	(n=12) 3.83±0.11
	2	(n=2) 4.00±0.00	(n=2) 5.00±0.00		(n=4) 4.50±0.29
	3	(n=2) 6.00±0.00		(n=6) 5.33±0.42	(n=8) 5.50±0.33

<표 1> 생성된 이진 검색 트리의 균형도 기술

<Table 1> Description of balance for generated binary trees

노드 갯수	생성된 총 트리수	생성된 상이한 트리수	트리 높이			균형 인수 크기			검색 횟수		
			최소	최대	평균 ±표준오차	최소	최대	평균 ±표준오차	최소	최대	평균 ±표준오차
3	6	5	1	2	1.67±0.21	0	2	1.33±0.42	2	3	2.50±0.22
4	24	10	2	3	2.33±0.10	1	3	1.75±0.17	3	6	4.50±0.20
5	120	41	2	4	2.80±0.06	1	4	2.20±0.10	5	10	6.55±0.12
6	720	123	2	5	3.27±0.07	1	5	2.46±0.09	7	15	8.98±0.07
7	5040	332	2	6	3.64±0.01	0	6	2.55±0.02	9	21	11.61±0.03

5	0				
	1	(n=14) 5.43±0.14	(n=14) 5.57±0.14	(n=12) 5.50±0.15	(n=40) 5.50±0.08
	2	(n=4) 7.00±0.00	(n=4) 7.00±0.00	(n=24) 6.00±0.17	(n=32) 6.25±0.15
	3	(n=5) 7.40±0.25	(n=5) 8.40±0.25	(n=22) 6.86±0.20	(n=32) 7.19±0.18
	4	(n=1) 10.00	(n=1) 10.00	(n=14) 8.29±0.34	(n=16) 8.50±0.33
6	0				
	1	(n=40) 7.50±0.08	(n=40) 7.00±0.00		(n=80) 7.25±0.05
	2	(n=35) 8.57±0.09	(n=35) 9.00±0.13	(n=250) 7.90±0.05	(n=320) 8.09±0.05
	3	(n=19) 9.53±0.21	(n=19) 10.68±0.11	(n=122) 9.31±0.11	(n=160) 9.50±0.10
	4	(n=9) 11.78±0.28	(n=9) 12.78±0.28	(n=110) 10.46±0.14	(n=128) 10.72±0.13
	5	(n=1) 15.00	(n=1) 15.00	(n=30) 12.83±0.32	(n=32) 12.50±0.32
7	0				(n=80) 9,00±0.00
	1	(n=45) 10.00±0.00	(n=45) 10.00±0.00	(n=630) 9.71±0.03	(n=720) 9.75±0.03
	2	(n=188) 10.59±0.05	(n=174) 10.93±0.06	(n=1038) 10.14±0.03	(n=1400) 10.30±0.03
	3	(n=143) 12.41±0.09	(n=109) 13.25±0.08	(n=1300) 11.37±0.04	(n=1552) 11.60±0.04
	4	(n=61) 14.21±0.15	(n=55) 15.51±0.13	(n=812) 13.28±0.06	(n=928) 13.48±0.05
	5	(n=14) 17.14±0.29	(n=14) 18.14±0.29	(n=284) 15.64±0.10	(n=312) 15.82±0.09
	6	(n=1) 21	(n=1) 21	(n=46) 18.04±0.26	(n=48) 18.17±0.26

트리의 불균형도 크기와 방향에 따른 검색 횟수를 측정한 결과는 <표 2>에 있다.

노드 개수가 6 과 7 에서는 11% 및 16% 트리가 모든 노드의 균형 인수가 1 이하인 높이 균형 이진 트리가 생성 되었고 생성된 트리의 17% 및 16%가 루트 노드를 중심으로 좌 또는 우 의 한쪽 방향 으로만 노드들이 삽입되는 사향 트리가 생성 되었다.

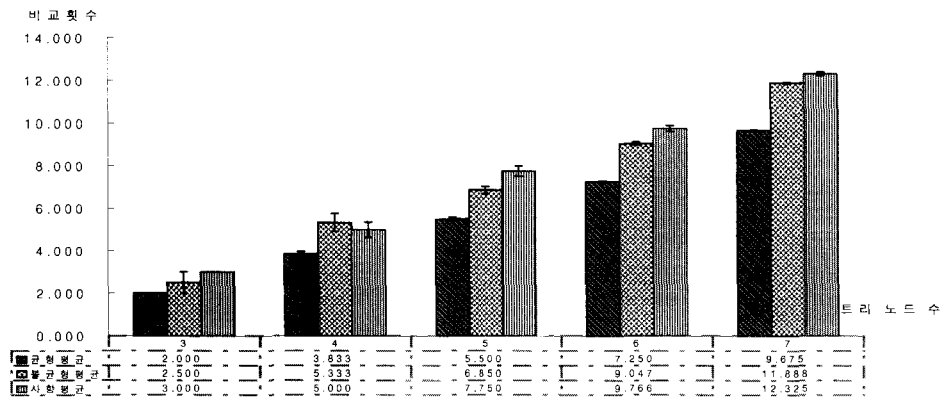
노드 개수가 6개 이상인 트리에서는 불균형도 크기가 증가할수록 검색 횟수가 증가하였다($p<0.01$).

불균형도 크기가 1에서 2로 증가할 때는 평균 검색 횟수가 0.75 회 정도 증가하였으나 2 이상의 불균형도 크기에서는 평균 검색 횟수의 증가율이 1회를 초과 하였다.

[그림 1] 의 불균형 방향에 따른 평균 검색 횟수의 비교 결과 5개 이상의 노드들을 가진 트리에서는 루트 노드를 중심으로 사향 트리인 경우 좌, 우로 불균형된 트리보다 검색 성능이 저하되는 결과가 얻어졌다($p<0.01$).

3.3 검색 트리 성능에 영향을 미치는 불균형 변수

<표 3>에는 검색 횟수에 영향을 미치는 요인들을 찾기 위하여 생성된 트리에서 좌, 우 서브트리 높이가 같은 노드 개수, 모든 노드들의 불균형도 합, 트리 높이의 양적 변수들과 검색 횟수의 상관 관계들을 산출하였다.



[그림 1] 트리의 불균형 방향에 따른 평균 검색 수

[Fig. 1] Average number of comparisons by signs of balance factors in a binary search tree

모든 노드 크기에서 균형 노드 개수가 증가할수록 검색 횟수는 감소하였으며 균형 인수 크기 합 및 높이가 증가할수록 검색 횟수도 증가하였다.

<표 3> 트리의 검색 횟수와 불균형도 변수들의 상관 계수

<Table 3> Correlation coefficients of number of comparisons and variables measuring imbalance in binary search trees

불균형도 변수	노드 갯수			
균형노드 갯수	n=4	n=5	n=6	n=7
균형인수합	-0.5067	-0.5764	-0.4134	-0.3882
트리 높이	0.7386	0.7895	0.7027	0.7213

3.4 트리 불균형도 크기에 따른 검색 성능 함수 추정

[그림 2]에는 모든 노드들의 균형 인수 크기가 1 이하인 높이 균형 이진 검색 트리에서 트리 높이에 따른 검색 횟수의 회귀 모형을 추정 하였다.

로그 모형 (R^2 =0.3498) 이 선형 모형 (R^2 = 0.3496) 보다 검색 성능을 설명하기에 적합한 모형으로 평가 되었다.

반면에 불균형도 크기가 3 이상인 트리들 에서는 검색 성능을 설명하기 위한 선형 모형의 적합성이 로그 모형보다 높은 경향을 보였다.

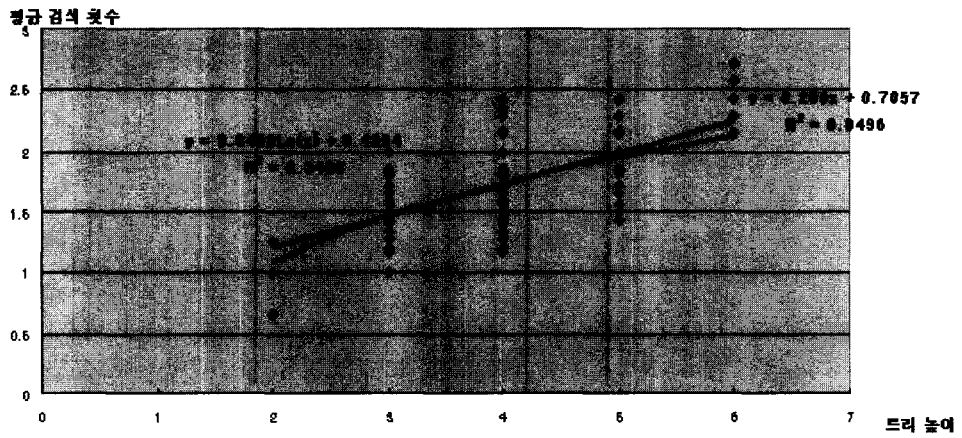
4. 결론

검색 트리 성능을 향상 시키기 위한 트리 재균형 과정은 때로는 트리 전체를 재구성하여야 하는 문제로 검색 성능을 오히려 저하 시키는 문제가 야기 될 수 있으므로 본 연구에서는 재균형 과정을 거치지 않고 검색 트리 성능이 저하되지 않는 트리 불균형도 크기 범위를 확률적으로 찾으려는 시도를 하였다.

노드 개수가 6개 이상일 때 좌, 우 서브 트리의 높이 차이가 1 이하인 노드들만으로 구성된 높이 균형 이진 검색 트리가 생성될 확률은 약 10% 정도로 측정 되었다.

Karlton 등의 논문 에서는 이진 검색 트리 생성 과정에서 임의의 노드 삽입 시 재균형이 필요 없을 확률을 0.5349 라는 결과를 발표 하였다 [9]. 본 실험에서 수집된 정보로부터는 트리 생성에 삽입된 총 노드들 중 균형 인수 크기가 1 이하인 확률은 0.5423으로 약간 높게 측정 되었다.

본 실험 에서는 노드 개수를 7 이하로 제한하였으므로 검색 성능이 현저히 저하되는 불균형도 크기



[그림 2] 트리 높이와 평균 검색 횟수의 함수 관계 (높이 균형 검색 트리)

[Fig. 2] Functional relation to tree heights and average number of comparisons (height balanced tree)

를 확률적으로 증명하지 못하였다. 그러나 불 균형도 크기가 1에서 2로 증가될 때는 평균 검색 횟수의 증가가 1회 미만인 반면 균형인수 크기가 증가하여 불균형도가 심해질수록 검색 횟수의 증가율이 증대되는 결과로부터 좌,우 서브트리 높이 차이가 3 보다 적을때는 트리 재균형 과정을 거치지 않고도 검색 성능 저하 문제가 발생하지 않음을 의미한다.

<표 3> 의 검색 횟수와 불균형에 관한 변수들의 상관 관계 검정 결과들로부터 검색 성능 저하를 최소화 하기 위하여는 트리를 구성하는 전체 노드들의 불균형도 크기 합을 최소로 하고 트리의 높이를 최소로 함을 권장할 수 있다.

로그형의 검색 성능이 선형으로 변하는 불균형도 크기 변환점을 찾기 위한 본 연구 회귀 모형 추정에서는 자료가 불규칙(random) 하게 분포되어야 하는 통계 가정이 본 실험 자료에서는 만족하기 어려우므로 산출된 회귀 모형들의 설명력들이 낮게 계산되었고, 적합도 검정을 적용시킬 수 없었다.

그러나 불균형도 크기가 3 미만 에서는 로그 모형의 설명력이 높고, 불균형도 크기가 3 이상인 트리들 에서는 선형 모형의 설명력이 높게 계산된 본 연구 결과는 균형 인수 크기가 3 미만인 트리에서도 검색 성능은 로그형에서 선형으로 저하되지 않음을 의미한다.

본 연구 실험 결과들은 이진 트리를 자료 구조로 사용하는 소프트웨어 관리시 빈번한 자료의 수정 이나 삽입 과정에 의하여도 자료구조를 재균형 시키는 노력 없이 검색 성능을 유지 시킬 수 있는 정보를 제공할 수 있을 것이다.

보다 타당한 결론를 위하여는 회귀 모형의 통계적 가정을 위반하지 않도록 노드 개수 범위를 확장한 미래의 연구가 필요하다.

※ 참고문헌

[1] 김숙영. SAS 와 Minitab을 이용한 전산 통계학, 신하출판사. p.131-150, 1996.

[2] 김숙영. 확률적 이진 검색 트리 성능 추정, 한국 컴퓨터 산업 교육학회지, 2(2), pp.203-210, 2001.

[3] 박영근. 자료 구조 기초와 활용 21세기사, p.255-280, 1998.

[4] 이석호 역. C로 쓴 자료 구조론. 회중당. p.496-509, 1993.

[5] 임형근. C 언어로 구현한 자료 구조론, 기술연구사 p.326-352, 1996.

[6] C Baumgarten, "Probalistic Information Retrieval in a Distributed Heterogenous Environment", PHD Thesis, Dresden Univ, of Tech n., Accepted, 1999.

[7] H Chang and S Lyengra. Efficient algorithms to globally balanced binary search tree. Communications of the ACM, Vol 27, pp.695-702, 1984.

[8] T.Cormen, C.Leiserson and R.Rivest. "Introduction to Algorithms", McGraw-Hill, pp.254-262, 1990.

[9] PL Karlton, SH Fuller, RE Scroggs, and E B Koehler. Performance of the Height-Balanced Tree, CACM 19, 1976.

[10] R. Kruse, B Leung and C Tondo, "Data structures and program design in C", Prentice-Hall Inc, pp. 327-330, 1991.

김 숙 영

미국 오하이오 주립 대학교
컴퓨터 과학과 졸업
미국 오하이오 주립 대학교
대학원 통계학과 졸업
(응용 통계학 석사)
1995 - 현재 안산 공과 대학
컴퓨터 정보과 조교수
관심 분야 : 전산 통계
(데이터 분석).
자료 구조, 수치해석,