The Journal of The Institute of Internet, Broadcasting and Communication (IIBC)
Vol. 17, No. 5, pp.157-163, Oct. 31, 2017. pISSN 2289-0238, eISSN 2289-0246

https://doi.org/10.7236/JIIBC.2017.17.5.157
JIIBC 2017-5-22

블록 보간 탐색법

Block Interpolation Search

이상운*

Sang-Un Lee*

요 약 데이터 탐색법 중 가장 널리 알려진 이진법은 평균과 최악의 경우 $O(\log_2 n)$, 보간법은 평균 $O(\log_2 \log_2 n)$, 최악의 경우 $O(n)$의 수행 복잡도를 갖고 있다. 또한 기존의 보간탐색법은 사전정보없이 킷값이 확률적으로 위치한 정보에 근거하여 탐색을 한다. 본 논문에서는 데이터의 MSB 인덱스를 블록으로 하는 블록탐색법으로 해당 블록범위를 결정하고, 블록 내에서는 보간법을 적용하여 탐색하는 하이브리드 블록과 보간탐색 알고리즘을 제안한다. 제안된 알고리즘은 블록탐색법의 사전 정보를 활용하여 탐색범위를 축소시키고 축소된 탐색범위내에서 무정보 방법으로 탐색하는 방법으로 평균과 최악의 경우 모두 수행복잡도는 $O(\log_2 \log_2 n_i), n_i \simeq 0.1n$으로 보간탐색법의 평균 수행복잡도에 비해 10배 정도 시간을 단축시킬 수 있다.

Abstract The binary and interpolation search algorithms are the most famous among search area algorithms, the former running in $O(\log_2 n)$on average, and the latter in $O(\log_2 \log_2 n)$on average and $O(n)$ at worst. Also, the interpolation search use only the probability of key value location without priori information. This paper proposes another search algorithm, which I term a 'hybrid block and interpolation search'. This algorithm employs the block search, a method by which MSB index of a data is determined as a block, and the interpolation search to find the exact location of the key. The proposed algorithm reduces the search range with priori information and search the reduced range with uninformed situation. Experimental results show that the algorithm has a time complexity of $O(\log_2 \log_2 n_i), n_i \simeq 0.1n$ both on average and at worst through utilization of previously acquired information on the block search. The proposed algorithm has proved to be approximately 10 times faster than the interpolation search on average.

Key Words : Sequential search, Binary search, Fibonacci search, Interpolation search, Block search

Ⅰ. 서 론

최근 들어 IT분야의 화두는 단연 빅데이터(Big data)이다. 여기서 요점은 기하급수적으로 증가하고 있는 데이터 중에서 가치 있는 데이터는 소수에 불과하다는 점이다. 따라서 대용량 데이터를 처리하고, 의미 있는 데이터를 발굴하기 위해서는 빠른 정렬(sort)과 더불어 키 데이터(키워드) K에 의한 빠른 탐색(search) 기술이 요구된다.

n개의 레코드가 $x_1 < x_2 < \cdots < x_n$으로 정렬된 파일에

*정회원, 강릉원주대학교 과학기술대학 멀티미디어공학과
접수일자 : 2017년 5월 28일, 수정완료 : 2017년 10월 9일
게재확정일자 : 2019년 10월 13일

Received: 28 May 2017 / Revised: 9 October, 2017 /
Accepted: 13 October, 2017
*Corresponding Author: sulee@gwnu.ac.kr
Dept. of Multimedia Eng., Gangneung-Wonju National University, Korea

서 키 레코드 K를 검색하고자 한다면, 우리는 인덱스 i인 $x_i = K$를 가능한 빠르게 찾고자 하는 것이다. 이 경우 절단 인덱스(cut index, c) $1 \le c \le n$를 선택하여 K를 절단 값 x_c와 비교한다. 만약, $K = x_c$이면 성공적으로 찾아 알고리즘이 종료된다. 그러나 $K < x_c$이면 찾고자 하는 키 값 K는 $x_1, x_2, \cdots, x_{c-1}$에 존재하며, $K > x_c$이면 키 값 K는 $x_{c+1}, x_{c+2}, \cdots, x_n$에 존재하므로 이 부분파일 범위에서 다시 찾으면 된다.[1]

탐색법은 크게 무정보 탐색법(uninformed search)과 통계적 탐색법(statistical search)으로 분류될 수 있다. 무정보 탐색법에는 레코드들 n개 중에서 키 값 K의 위치를 사전에 알지 못한다고 가정하고 탐색하는 맹인 탐색방법(blind search)으로 순차적 탐색(sequential search), 이진법(binary search), 피보나치 탐색법(Fibonacci search)과 블록 탐색법(block search)이 있다.[2] 통계적 탐색법은 키 값 K가 전체 레코드 값 범위 $Hi - Lo (Hi = A[n], Lo = A[1])$ 중에서 $K - Lo$의 거리에 존재한다는 확률에 기반한 값 $c_k = \frac{K - Lo}{Hi - Lo} \times n$에 대해 K와 $x_c = A[Lo + \lfloor c_k \rfloor - 1]$를 비교하는 보간 탐색법(interpolation search)이 있다.[3,4]

이진 탐색법은 평균과 최악의 경우 $O(\log_2 n)$의 수행복잡도를 갖고 있으며, 보간탐색법은 균일분포(uniform distribution)인 경우 평균적으로 $O(\log_2 \log_2 n)$, 불균일분포인 최악의 경우 $O(n)$이 소요된다.[5-7]

본 논문에서는 키 값 K가 속한 범위에 대한 사전정보를 갖고 있는 상태(informed)에서 해당 범위 내에서 보간탐색법을 적용한 방법을 제안한다. 키 값 K가 속한 범위에 대한 사전정보는 데이터 정렬시 키 레코드의 MSB (most significant bit, most left bit) $i = 0, 1, \cdots, 9$에 대해 시작 인덱스를 가진 $S[i], i = 0, 1, \cdots, 9$를 만드는 방법을 활용한다. 따라서 키 값 K는 $K_{MSB} = i$인 $A[S[i], \cdots, S[i+1] - 1]$에 속함을 알 수 있다.

빅 데이터를 처리하기 위해서는 빠른 정렬뿐 아니라 빠른 탐색 또한 중요하다. 이는 실과 바늘의 관계와 같이 떨어질 수 없는 관계이다. 빠른 정렬에 대해서는 Hoare[8]의 퀵 정렬, Lee[9]의 범위 피벗을 이용한 퀵 정렬, Lee[10]의 3-점 평균 퀵 정렬, Lee[11]의 빠른 계수정렬법과 Lee[12]의 가상의 기수계수 버킷 정렬법 등 많은 연구가 수행되었다. 그럼에도 불구하고 빠른 탐색법에 대해서는 순차적과 이진 탐색법이 제안된 이후 거의 연구가 진행되지 않고 있다.

제안된 방법은 사전정보를 가진 통계적 탐색법으로 수행복잡도는 평균이나 최악의 경우 모두 $O(\log_2 \log_2 n_i), n_i \simeq 0.1n$으로 보간탐색법의 평균 수행복잡도 $O(\log_2 \log_2 n)$을 10배 정도 향상시키는 효과를 나타낼 수 있다.

2장에서는 대표적인 탐색법의 문제점을 고찰해 본다. 3장에서는 블록보간법을 제안한다. 4장에서는 실험 데이터에 대해 제안된 탐색법의 효율성을 분석하여 본다.

II. 탐색법의 문제점

본 장에서는 그림 1과 같이 순차적 탐색법, 이진법, 피보나치 탐색법, 블록 탐색법과 보간법의 방법과 문제점을 고찰해 본다.

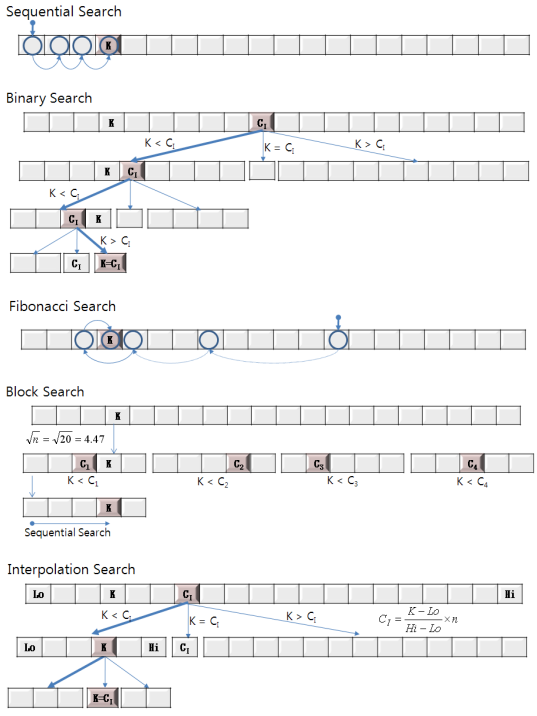


그림 1. 탐색법
Fig. 1. Searching algorithms

먼저, 순차적 탐색법(또는 선형 탐색법)은 $A[1]$부터 시작하여 $A[n]$까지 한 번에 하나씩 찾고자 하는 키 값 K와 비교하는 방법으로 평균적으로는 $O(0.5n)$, 최악의 경

우 $O(n)$의 수행 복잡도를 나타낸다. 이 방법은 알고리즘이 간단하며, 비순서화된 파일에 대해서는 유용한 방법이다. 그러나 탐색 레코드 크기가 클수록 탐색시간이 많이 소요되는 단점이 있다.

이진법은 $H=n, L=1$의 중간위치 $x_c = A[\lfloor \frac{H+L}{2} \rfloor]$을 결정하여 키 값 K와 비교한다. $K=x_c$이면 성공적으로 찾아 알고리즘이 종료된다. $K<x_c$이면 찾고자 하는 키 값 K는 $x_1, x_2, \cdots, x_{c-1}$에 존재하며, $H=\lfloor \frac{H+L}{2} \rfloor -1$로 재설정된다. $K>x_c$이면 찾고자 하는 키 값 K는 $x_{c+1}, x_{c+2}, \cdots, x_n$에 존재하므로 $L=\lfloor \frac{H+L}{2} \rfloor +1$로 재설정하고 이 부분파일 범위에서 다시 중간값을 결정하는 방법이다. 따라서 전이진트리(full binary tree)의 레벨 수 $l, (2^{l-1} < n < 2^l)$횟수 탐색으로 수행복잡도는 $O(\log_2 n)$이다. 이 방법은 탐색효율이 좋고 알고리즘이 간단하다. 또한 탐색할 레코드 수가 많아도 크기를 계속적으로 절반씩 줄이면서 탐색하므로 탐색시간을 획기적으로 감소시킬 수 있다. 단점은 반드시 레코드가 정렬되어 있어야만 하며, 특정 범위의 레벨까지 찾아가기 위해 사전 정보를 얻지 못해 시간을 단축시킬 수 없다.

피보나치 탐색법은 피보나치 수열 $F_i = F_{i-2} + F_{i-1}$, $(F_0 = 0, F_1 = 1)$에 대해 $F_i < n < F_{i+1}$인 F_i부터 내림차순으로 탐색하면서 $F_k < K$를 찾으면 F_k부터 순차적 탐색으로 K까지 도달한다.[13] 이진탐색법은 곱셈과 나눗셈 계산인데 반해 피보나치 방법은 뺄셈과 덧셈 계산으로 컴퓨터 상에서 보다 효과적으로 수행시간을 단축시킬 수 있다. 그러나 피보나치 수열을 역으로 계산하는 부가적인 오버헤드로 인해 대체적으로 이진탐색법보다 효율성이 떨어진다. 피보나치 탐색법의 수행복잡도는 $O(\log_2 n)$이다.

블록탐색법은 길이 n인 레코드를 길이 $\lfloor \sqrt{n} \rfloor$인 $\lceil \sqrt{n} \rceil$개의 블록으로 분할한다. 이 때 블록 내에서는 오름차순으로 정렬되어 있지 않아도 상관없으며, 각 블록의 최대값은 갖고 있어야만 한다. 따라서 키 값 K가 해당 블록 최대값보다 작은 블록을 선정하여, 이 블록 내에서는 순차탐색법을 찾는다. 이 알고리즘의 수행 복잡도는 $O(\sqrt{n})$이다. 이 방법은 이진탐색법보다 비효율적이며, 블록의 최대값 인덱스를 $\sqrt{n}$개 유지해야 할 부수적인 기억공간이 요구된다.[3]

보간법은 $Hi-Lo (Hi=A[n], Lo=A[1])$의 범위 중에서 $K-Lo$의 거리가 차지하는 비율에 키 값이 위치할 가능성에 대한 확률을 적용한 통계적 방법으로 $c_k = \lfloor \frac{K-L}{H-L} \times (H-L) \rfloor$, $(c_k < 1$이면 $c_k = 1)$을 결정하고 K와 $x_c = A[L + \lfloor c_k \rfloor]$를 비교하여 범위를 n'로 축소시킨다. 해당 범위 n'에 대해 다시 c_k를 결정하는 방법이다. 이 방법은 레코드들이 균일분포인 경우의 수행 복잡도는 $O(\log_2 \log_2 n)$이나 불균일 분포로 이상점(outliers)이 있는 최악의 경우 수행 복잡도는 $O(n)$이다. 이 방법은 키의 분포 상태가 균일분포일 때 매우 효율적이며, 레코드 수가 많을 때 이진탐색보다 효율성이 증대된다.[3]

Ⅲ. 블록 보간 탐색법

이진탐색법은 무정보 중간 위치를 선정하는 방법으로 범위를 축소시킨다. 보간법은 무정보 상대적 위치 선정으로 범위를 축소시킨다. 본 장에서는 길이가 n인 정렬된 리스트 $A[1,2,\cdots,n], (x_1 < x_2 < \cdots < x_n)$과 $d_k d_{k-1} \cdots d_1$의 키 값 K의 MSB d_k가 속하는 범위에 대한 사전정보를 갖고 있는 블록 시작점 $S[i], i=0,1,2,\cdots,9$를 입력으로 하여 키 값 K를 찾는 수행 복잡도 $O(\log_2 \log_2 n_i)$, $n_i \simeq 0.1n$을 제안한다. 제안된 방법은 K가 어떤 블록에 포함되어 있는지 사전정보에 기반하여 1차로 범위를 축소시키고, 해당 축소된 범위 내에서 보간탐색으로 무정보 상대적 위치를 결정하여 범위를 2차적으로 축소시킨다. 제안된 방법은 다음과 같이 수행되며, 그림 2와 같다.

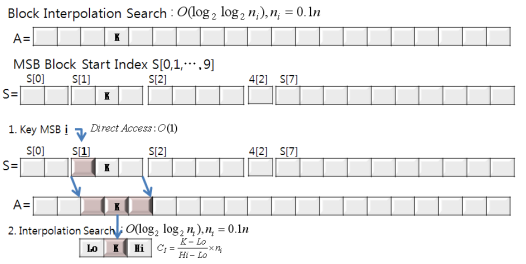


그림 2. 블록 보간 탐색법
Fig. 2 Block Interpolation Search

Step 1.키 값 범위 결정 /* 수행복잡도 $O(1)$ */

키 값 K의 MSB $d_k = i$인 $S[i], S[i+1]$을 탐색하여 $A[S[i]] \le K \le A[S[i+1]-1]$범위를 결정한다.

Step 2.키 값 탐색 /* 수행복잡도 $O(\log_2\log_2 n_i), (n_i = S[i+1]-S[i] \simeq 0.1n)$*/

$A[S[i]] \le K \le A[S[i+1]-1]$에 대해 보간법으로 $c_k = \left\lfloor \frac{K-L}{H-L} \times (S[i+1]-S[i]) \right\rfloor, (c_k < 1$이면 1로 설정), $H = A[S[i+1]-1], L = A[S[i]]$을 결정하여 K와 $A[L+c_k-1]$을 비교한다.

(1) $K = A[L+c_k-1]$이면 알고리즘을 종료한다.

(2) $K < A[L+c_k-1]$이면 $H = A[L+c_k-2]$로, $K > A[L+c_k-1]$이면 $L = A[L+c_k]$로 설정하고 $K = A[L+c_k-1]$가 될 때까지 Step 2를 반복 수행한다.

제안된 방법은 키 값 K가 속한 범위를 $d_k = i$인 $S[i]$ 블록으로 직접 찾아 수행복잡도는 $O(1)$이다. 다음으로 $A[S[i]] \le K \le A[S[i+1]-1]$에 대해 보간탐색법을 적용하기 때문에 수행복잡도는 $O(\log_2\log_2 n_i), (n_i = S[i+1]-S[i] \simeq 0.1n)$이다. 단, 제안된 알고리즘은 추가적인 메모리 $S[i], i = 0,1,2,\cdots,9$가 요구되는 반면에 보간탐색법의 평균 수행 복잡도 $O(\log_2\log_2 n)$에 비해 10배 빠른 효율성을 나타낸다.

보간 탐색법은 사전정보없이 키 값이 전체 레코드 값 범위 중에서 특정 거리에 존재한다는 확률 기반 값에 대해 킷값을 확률기반 값과 비교하는 방법이다.[3,4] 반면에, 제안된 방법은 키 값이 어떤 블록에 포함되어 있는지에 대한 사전정보에 기반하여 1차로 탐색 범위를 축소시키고, 해당 축소된 범위 내에서 보간탐색으로 무정보 상대적 위치를 결정하여 범위를 2차적으로 축소시키는 '사전정보기반 축소된 범위에 대한 무정보 위치 탐색법'으로 명확한 차이점이 있다. 따라서 기존의 보간탐색법의 전체 탐색범위를 크게 축소시켜 빠른 탐색을 할 수 있는 시간과 메모리 절감 효과를 얻는 장점이 있다.

Ⅳ. 적용 결과 및 분석

본 장에서는 그림 3의 6개 데이터에 대해 블록 보간탐색법을 적용하고, 다른 탐색법들과 성능을 비교하여 본다.

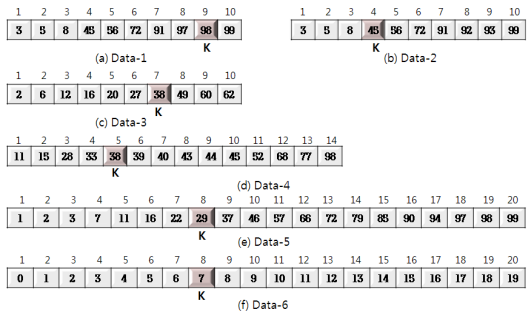
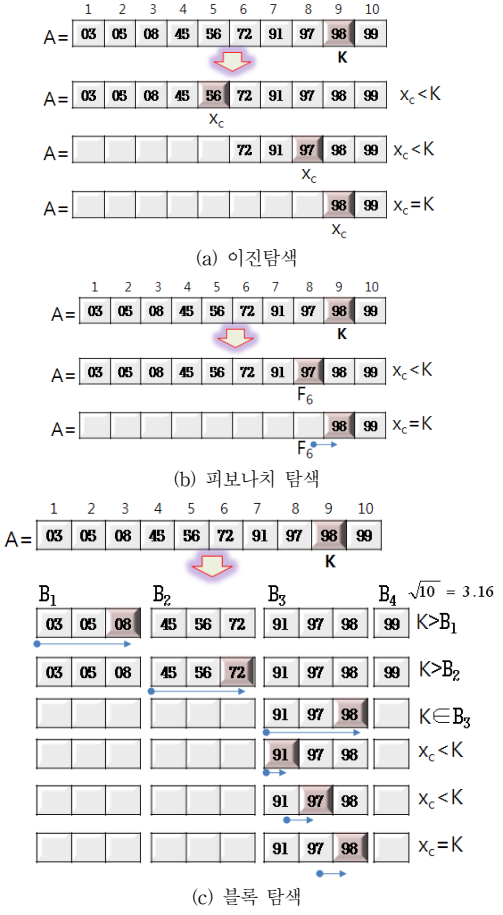


그림 3. 실험 데이터
Fig. 3. Experimental Data

그림 3의 (a) Data-1의 $A[8] = 97$을 이진탐색, 피보나치 탐색, 블록탐색, 보간탐색과 블록 보간탐색으로 수행한 결과는 그림 4와 같다. 순차탐색은 $A[1]$부터 $A[8]$까지 8회를 수행하므로 생략되었다. 이진탐색은 2회, 피보나치 탐색은 5회, 블록탐색은 2회, 보간탐색은 3회가 수행되었으며, 블록 보간탐색은 2회를 수행하였다.



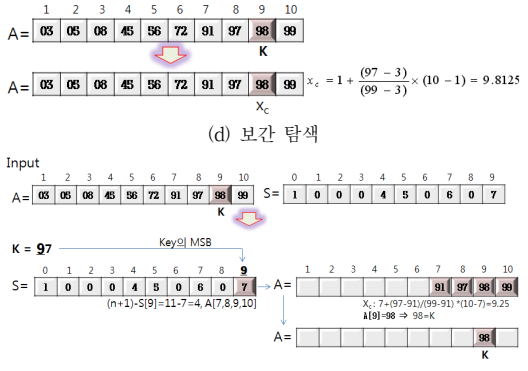
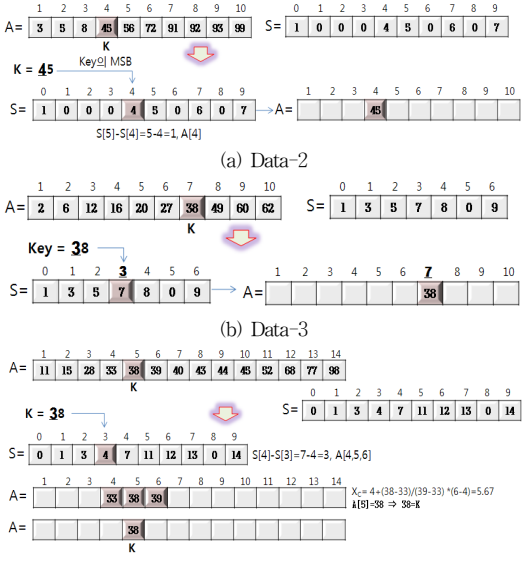


그림 4. Data-1의 탐색방법 비교
Fig. 4. Compare of searching algorithms for Data-1

그림 3의 (b)~ (f)의 5개 데이터에 대해 블록 보간탐색법을 수행한 결과는 그림 5에 제시되어 있다.



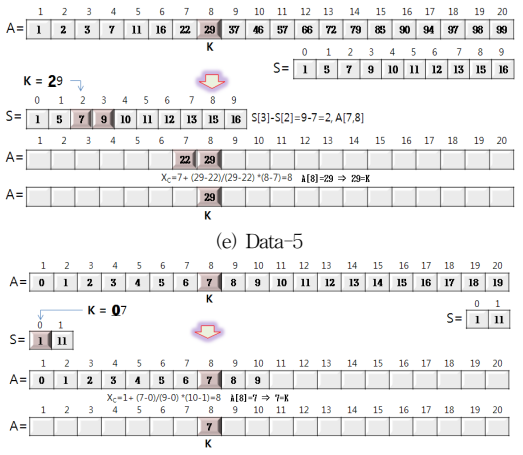


그림 5. 블록 보간 탐색법
Fig. 5. Block interpolation search

그림 3의 6개 데이터에 대해 탐색법들의 수행횟수를 비교한 결과는 표 1에 제시되어 있으며, 블록 보간탐색법이 수행횟수가 가장 적음을 알 수 있다. 균등분포인 경우 보간탐색이 추가적인 메모리가 불필요하여 블록보간탐색법 보다 효율성이 좋으나 비균등 분포일 경우 블록보간탐색법이 보다 효율적으로 $O(n)$을 $O(\log_2 \log_2 n_i)$, $n_i \simeq 0.1n$을 향상시킬 수 있다. 또한, $n > 100$인 경우 블록탐색법의 블록 개수 $\sqrt{n} > 10$에 비해 블록 보간탐색법은 최대 10개로 보다 효율적이다.

$n = 500, [1,500]$ 데이터에 대해 $K = A[329] = 329$를 찾고자 한다고 가정하여 보자. 순차탐색은 329회, 블록탐색은 $\sqrt{500} = 22.36$으로 22개 길이를 가진 23개 블록에 대해 B_{14}인 $A[308]$에서 순차적으로 22회 탐색하면 $A[329]$를 찾는다. 이진탐색은 $500 \rightarrow 250 : A[251-500] \rightarrow 376 : A[251-375] \rightarrow 313 : A[314-375] \rightarrow 344\, A[314-343] \rightarrow 329$로 5회를 수행한다. 반면에 보간탐색은

표 1. 실험 데이터의 알고리즘 수행횟수
Table 1. The number of execution for Algorithms

데이터	n	K	수행횟수						블록 수	
			순차탐색	이진탐색	피보나치탐색	블록탐색	보간탐색	블록보간탐색	블록탐색	블록보간탐색
Data-1	10	A[9]=98	9	3	2	6	**1**	**1**	4	10
Data-2	10	A[4]=45	4	3	4	3	**1**	**1**	4	10
Data-3	10	A[7]=38	7	3	4	4	2	**1**	4	7
Data-4	14	A[5]=38	5	4	3	4	**1**	2	4	10
Data-5	20	A[8]=29	8	4	2	6	3	**1**	5	10
Data-6	20	A[8]=7	8	4	2	6	**1**	**1**	5	2

$1+\frac{329-1+1}{500-1+1}\times 500-1=329$로 1회 만에 찾으며, 블록 보간탐색은 $S[0]=1,\ S[1]=100,\ S[2]=200,\ S[3]=300, S[4]=400, S[5]=500$에 대해 329의 MSB $d_3=3$은 $S[3]\le d_k=3<S[4]$으로 $A[300-399]$에 대해 $300+\frac{329-300+1}{399-300+1}\times 100-1=329$로 1회 만에 찾는다.

V. 결론

본 논문은 키 값 $K, d_k=i, (0\le i\le 9)$가 속하는 블록 $S[i]\le K<S[i+1]$에 대한 사전정보를 갖고 있는 상태에서 해당 블록 내에서는 통계적인 보간탐색법을 적용한 알고리즘을 제안하였다. 제안된 방법은 사전정보를 가진 통계적 방법으로 기존의 무정보 탐색법과 통계적 탐색법의 특징을 결합시킨 방법이다.

제안된 알고리즘은 보간탐색법의 불균일 분포인 최악의 경우 $O(n)$의 단점을 개선하기 위해 키 값의 MSB $0\le d_k\le 9$에 대한 시작점 $S[i]$를 정렬을 수행한 시점에서 준비하여 추후 탐색시 활용하는 방법을 적용하였다.

제안된 블록 보간 탐색법은 수행복잡도가 $O(\log_2\log_2 n_i)\, n_i\simeq 0.1n$로 보간탐색법의 평균 수행복잡도 $O(\log_2\log_2 n)$을 10배 정도 향상시킨 효과를 나타내며, 이진탐색법의 $O(\log_2 n)$보다 효율적임을 알 수 있다.

References

[1] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, "Introduction to Algorithms, MIT Press, ISBN: 0-262-03384-4, 2005.

[2] R. Sedgewick, "Algorithms in C, Parts 1-4: Fundamentals, Data Structures, Sorting, Searching", 3rd Ed., Addison-Wesley, ISBN: 978-0201314526, 1998.

[3] G. H. Gonnet L. D. Rogers, and J. A. George, "An Algorithmic and Complexity Analysis of Interpolation Search," Acta Informatica, Vol. 13, No. 1, pp. 39-52, Jan. 1980, doi:10.1007/BF00288534.

[4] A. Arne and C. Mattsson, "Dynamic Interpolation Search in O(loglog*n*) Time," Automata, Languages and Programming, edited by Andrzej Lingas, Rolf Karlsson, and Svante Carlsson, pp. 5 - 27, Lecture Notes in Computer Science,1993, doi:10.1007/3-540-56939-1_58.

[5] D. Knuth, "The Art of Computer Programming **3** (3rd ed.), Section 6.1: Sequential Searching," Addison-Wesley, pp. 396 - 408, ISBN: 0-201-89685-0, 1997.

[6] S. R. Shorofsky. R. W. Peters, E. J. Rashba, and M. R. Gold, "Comparison of Step-Down and Binary Search Algorithms for Determination of Defibrillation Threshold in Humans," Pacing Clin Electrophysiol, Vol. 27, No. 2, pp. 218-220, Feb. 2004, doi:10.1111/j.1540- 8159.2004.00413.x.

[7] A. R. Chadha, R. Misal, and T. Mokashi, "Modified Binary Search Algorithm," International Journal of Applied Information Systems, Vol. 7, No. 2, pp. 37-40, Apr. 2014, arXiv:1406. 1677

[8] C. A. R. Hoare, "Quicksort", Communications of the ACM, Vol. 4, No. 7, pp. 321-327, Jul. 1961, doi:10.1145/366622.366644

[9] S. U. Lee, "Quicksort Using Range Pivot," Journal of KSCI, Vol. 17, No. 4, pp. 139-145, Apr. 2012, uci: G704-001619.2012.17.4.010.

[10] S. U. Lee, "3-Points Average Pivot Quicksort," Journal od IIBC, Vol. 14, No. 6, pp. 295-391, Dec. 2014, doi:10.7236/JIIBC.2014.14.6.

[11] S. U. Lee, "Proposal of Fast Counting Sort," Journal od IIBC, Vol. 15, No. 5, pp. 61-68, Oct. 2015, doi:10.7236/JIIBC.2015.15.5.61.

[12] S. U. Lee, "Virtual Radix Counting Bucket sort," Journal od IIBC, Vol. 15, No. 6, pp. 95-102, Dec. 2015, doi:10.7236/JIIBC.2015.15.6.95.

[13] D. E. Ferguson, "Fibonaccian Searching," Communications of the ACM, Vol. 3, No. 12, pp. 648, Dec. 1960, doi:10.1145/367487.367496.

저자 소개

이 상 운(정회원)

- 1987년: 한국항공대학교 항공전자공학과 (학사)
- 1997년: 경상대학교 컴퓨터과학과 (석사)
- 2001년 : 경상대학교 컴퓨터과학과 (박사)
- 2003년 : 강원도립대학 컴퓨터응용과 전임강사
- 2004년 ~ 2007.2 : 국립 원주대학 여성교양과 조교수
- 2007.3 ~ 2015.3 : 강릉원주대학교 멀티미디어공학과 부교수
- 2015.4 ~ 현재 : 강릉원주대학교 멀티미디어공학과 정교수

<관심분야 :소프트웨어 프로젝트 관리, 개발 방법론, 분석과 설계 방법론, 시험 및 품질보증, 소프트웨어 신뢰성, 그래프 알고리즘>

- e-mail : sulee@gwnu.ac.kr