2000년 한국정보처리학회 추계 학술발표논문집 제7권 제2호

정보블록 전처리 알고리즘

송태옥 구정모 김태영
한 국 교 원 대 학 교 컴 퓨 터 교 육 학 과
{kinggem, mulgunamu, tykim}@comedu.knue.ac.kr

Information Block Preprocessing Algorithm(IBPA)

Tae-Ok Song Jung-Mo Koo Tae-Young Kim
Dept. of Computer Education, Korea National University of Education

요 약

본 논문에서는 기존의 정렬 알고리즘의 성능을 향상시키기 위하여 정보블록 전처리 알고리즘(IBPA)이라는 전처리 알고리즘을 제안한다. IBPA는 정렬될 리스트(list)에 있는 데이터에 관한 정보를 생성하고, 생성된 정보를 이용하여 각 데이터를 재배치하며, 실제적인 정렬은 기존의 정렬 알고리즘을 그대로 이용하여 이루어진다. IBPA의 성능을 측정해본 결과, 2백만개의 랜덤데이터를 정렬한 경우, $O(N^2)$의 평균시간복잡도를 갖는 정렬알고리즘의 0.003%, $O(NlogN)$의 평균시간복잡도를 갖는 정렬알고리즘의 52%, 그리고 $O(N)$의 평균시간복잡도를 갖는 정렬알고리즘의 89%정도의 비교회수만으로도 정렬할 수 있음을 보여주었다.

1. 서론

본 연구는 정렬알고리즘의 성능을 개선시키는 방법에 관한 연구로서, 기존의 정렬알고리즘은 변화시키지 않고, 다만 전처리 과정을 추가함으로써 기존의 $O(N^2)$의 평균시간복잡도를 갖는 정렬 알고리즘 뿐만 아니라 $O(NlogN)$와 $O(N)$의 평균시간복잡도를 갖는 정렬 알고리즘의 성능을 개선하고자 한다. 전처리 과정은 정보블록 전처리 알고리즘(IBPA; Information Block Preprocessing Algorithm)에 의해 수행되며, IBPA는 데이터의 분포 상태를 파악한 정보에 따라 정보블록을 생성하고, 이 정보를 이용하여 데이터를 재배치하여, 각각의 블록을 기존의 정렬 알고리즘을 적용하여 최종 정렬을 수행함으로써 정렬 효과를 향상시키는 전처리 알고리즘을 말한다.

본 연구에서는 먼저 기존의 정렬알고리즘의 종류와 성능을 살펴본 후, IBPA의 구조와 시간복잡도를 간략하게 분석하였다. IBPA의 실제적인 효과는 시간복잡도별로 몇 가지 정렬알고리즘을 선정한 후 시뮬레이션을 통하여 확인하였으며, 실험결과를 그래프로 제시하였다.

2. 정보블록 전처리 알고리즘

2.1 용어의 정의

다음은 본 연구에서 사용된 용어에 대한 정의와 그것에 대한 설명이다.

1) N : 리스트에 있는 데이터의 총 개수
2) MaxDV: 리스트에서 가장 큰 데이터의 값
3) MinDV: 리스트에서 가장 작은 데이터의 값.
4) TDR(Total Data Range): 리스트에 있는 모든 데이터가 속하는 범위를 수치화한 값.
5) DB(Data Block) : 데이터 블록. 리스트를 M개로 나누었을 때 나누어진 각 부분을 말하며, 정보블록과는 다르지만 개수는 같다.
6) IB(Information Block) : 정보블록. 데이터 블록에 관한 정보 즉, 데이터의 개수나 데이터의 범위, 그리고 데이터의 물리적 위치 등에 관한 정보가 저장되어있다.
7) BDR(Block Data Range) : 하나의 데이터 블록에 포함될 수 있는 데이터의 범위를 수치화한 값. 예를 들어 MinDV가 1이고 BDR이 50이라면 첫 번째 정보블록에 포함될 수 있는 데이터는 1에서 50까지의 값을 가지는 데이터이다. (1≤BDR)
8) M : 정보블록의 개수 (M≤N). M은 TDR을 BDR로 나눈 몫이 되며, 나머지가 있으면 1 증가시킨다. M이 N보다 큰 경우에는 M에 N을 대입시켜 극단적인 데이터로 인한 시간복잡도의 증가를 배제시켰다.
9) BDP(Blank Data Position) : 데이터가 임시기억장소로 이동하여 비어있는 위치를 의미한다. BDP는 데이터블록에서의 상대적인 위치(index)가 아니라 리스트에서의 절대적인 위치를 의미한다.

2.2 정보블록의 구조와 메모리 사용량

정보블록(IB)은 모두 M개의 단위 정보블록으로 구성되는데, 단위정보블록의 구성요소는 <표 1>과 같다.

<표 1> 단위 정보블록의 요소

요소	설명
Range_start(Rs)	데이터블록(DB)에 속할 수 있는 가장 작은 값
Range_end(Re)	DB에 속할 수 있는 가장 큰 값
Number_data(Nd)	DB에 속하는 데이터의 개수
Pointer_start(Ps)	리스트상에서 DB가 시작되는 곳의 위치
Pointer_end(Pe)	리스트상에서 DB가 끝나는 곳의 위치
Now_Pointer(Np)	이동여부를 검사해야할 데이터의 위치

IB의 구조에 따라 전처리에 필요한 실제 메모리는 리스트의 크기에 IB의 크기를 더한 값이 된다.

필요한 메모리 = 리스트의 크기 + 정보블록의 크기
정보블록의 크기 = M * 단위정보블록의 크기
단위정보블록의 크기 = 6 × sizeof(요소의 데이터형)

2.3 알고리즘

IBPA는 크게 2부분으로 구성되어있다. 즉, IB를 구성하는 부분과 알맞은 DB로 데이터를 재배치시키는 부분이다.

** 이 논문은 2000년도 두뇌한국21 사업 핵심분야에 의해 지원되었음

(그림 1)과 같이 IBPA를 나타낼 수 있다.

```
Part 1. IB 초기화 및 재구성
 Step 1. TDR, BDR, M 설정 및 IB 할당
 Step 2. IB의 초기화
 Step 3. 데이터의 분포 파악
 Step 4. IB의 정보 재구성
                ↓
Part 2. 데이터 재배치
 Step 5. 데이터 이동
  for I ←1  To  M ···①
  do for j ←i번째 Np to I번째 Pe···②
    do Temp ←data[j]
       BDP ←j
       while 블록순회가 계속인 동안···③
       do ▷ 해당 블록 찾기 ···④
          해당블록←((Temp-MinDV) div BDR)+1
          ▷ 데이터 이동
          For k ←해당 블록의 Np
              to  해당 블록의 Pe···⑤
          do if BDP = k ···⑥
             then data[k] ← Temp
                  해당 블록의 Np 1증가
                  ⑤와 ③ 루프 탈출, ②로 복귀
             if data[k]가 해당 블록에 속하는가?
             then 해당 블록의 Np 1증가···⑦
             else Temp와 data[k]의 교환···⑧
                  해당 블록의 Np 1증가
                  ⑤ 루프 탈출, ③으로 복귀
```

(그림 1) 정보블록 전처리 알고리즘

2.4 시간 복잡도

Part.1의 비교회수는 다음과 같다.

Part. 1의 비교회수
= Step.1에서 Step.4까지의 비교회수
= $C_1N + C_2M + C_3N + C_4M$ $1 \leq M \leq N$)
$\in \Theta(N)$

Part.2의 비교회수는 다음과 같다.

Part 2의 비교회수
$= (C_5 + C_6I_1 + C_7G_1 + C_8E_1)$
$+ \cdots$
$+ (C_5 + C_6I_m + C_7G_m + C_8E_m)$
$= C_5M + (C_6 \sum_{i=1}^{m} I_i + C_7 \sum_{i=1}^{m} G_i + C_8 \sum_{i=1}^{m} E_i)$
$= C_5M + (C_9N)$
$\in \Theta(N)$ (I+G+E=N)

Part.2는 데이터를 재배치하는 부분으로서 IBPA의 핵심적인 부분이다. Im + Gm + Em은 N으로서, Part.2의 시간복잡도가 $O(N)$이라는 것을 의미한다. 가장 많은 비교가 소요되는 Em이 N만큼 발생한 경우 C_8N만큼 비교되며, 가장 적은 비교가 소요되는 Gm의 경우 C_7N만큼 비교된다.

Part.3은 기존의 정렬 알고리즘을 이용하여 각각의 DB를 정렬하는 부분인데, $O(N)$과 $O(NlogN)$ 그리고 $O(N^2)$의 평균시간복잡도를 갖는 정렬 알고리즘의 세 가지 경우로 나누어 기술하였다.

1) $O(N^2)$의 정렬 알고리즘의 경우

Part 3의 비교회수
= 블록1의 비교회수 + ··· + 블록M의 비교회수
$= A_1^2 + A_2^2 + \cdots + A_m^2$
$= pN$ $(1 \leq p \leq N)$
$\in \Theta(N^2)$

최악이 아닌 경우는 Part.3의 비교회수가 가장 많은 데이터를 가진 DB의 데이터 개수에 비례하게 되므로, 최악의 경우가 아닌 경우도 함께 고려하여 변수 p를 사용하여 pN으로 나타낼 수 있다.

2) $O(NlogN)$의 정렬 알고리즘의 경우

Part 3의 비교회수
= 블록 1의 비교회수 + ··· + 블록 M의 비교회수
$= A_1 \log A_1 + A_2 \log A_2 + \cdots + A_m \log A_m$
= qNlogN $(0<q \leq 1)$
$\in \Theta(NlogN)$

이 경우 역시 비교회수는 가장 많은 데이터를 가진 DB의 데이터개수에 비례하게 되므로 1보다 같거나 작은 변수 q를 사용하여 qNlogN으로 나타낼 수 있다.

3) $O(N)$의 정렬 알고리즘의 경우

Part 3의 비교회수
= 블록 1의 비교회수 + ··· + 블록 M의 비교회수
$= A_1 + A_2 + \cdots + A_m$
= N
$\in \Theta(N)$

기존의 정렬알고리즘에 IBPA를 적용했을 때 최상의 경우는 데이터가 이미 정렬된 경우이고, 최악의 경우는 데이터가 하나의 데이터 블록에 위치해있는 경우를 말한다. 그러므로 Part.1과 Part.2 그리고 part.3의 시간복잡도를 정리하면 <표 2>와 같이 정리하여 나타낼 수 있다.

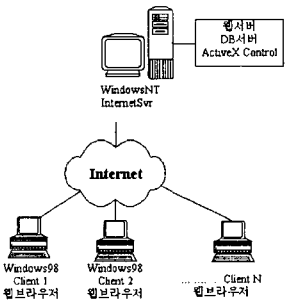
<표 2> 시간 복잡도

종류	최상(best)	평균(average)	최악(worst)
$O(N^2)$	$O(N)$	$O(pN)$	$O(N^2)$
$O(NlogN)$	$O(N)$	$O(qNlogN)$	$O(NlogN)$
$O(N)$	$O(N)$	$O(N)$	$O(N)$

$(0<q \leq 1,\ 1 \leq p \leq N)$

3. 성능평가를 위한 시뮬레이터

3.1 시스템 구조



(그림 2) 시스템의 전체구조

시스템의 전체 구조를 이용된 S/W와 H/W를 중심으로 살펴보면 (그림 2)와 같이 나타낼 수 있다. 시뮬레이터는 독립형(standalone)과 웹형이 있는데, 웹서버는 웹형의 시뮬레이터를 웹브라우저로 다운로드 받는데 필요하다.

3.2 메뉴

메뉴의 구조는 <표 3>과 같다. 사용자가 설정할 수 있는 변수는 정렬방법, 구간값, 데이터의 범위, 데이터의 개수, 데이터의 종류, 애니메이션 속도로서, 모두 6가지이다.

<표 3> 메뉴

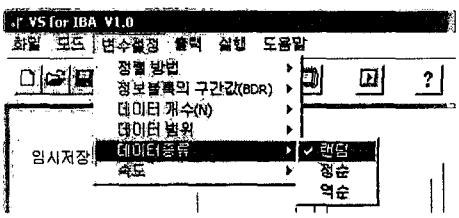
메뉴	하위메뉴	최하위메뉴
화일	새로만들기	·
	저장하기	·
	불러오기	·
	종료	·
모드	**에니메이션**	**전처리알고리즘, 정렬알고리즘**
	성능측정	전처리알고리즘, 정렬알고리즘
변수 설정	정렬방법	퀵, 기수, 삽입정렬
	BDR(구간값)	1, 10, **50**, 100
	데이터 개수	10, 50, **100**
	데이터 범위	1..100, 1..200, **1..300**
	데이터 종류	**랜덤**, 정순, 역순
	속도	**빠르게**, 보통, 느리게
출력	프린터	·
	화면	·
	화일(DBF)	·
실행	실행	·

*** 굵은 글씨는 디폴트값임**

3.3 구현 화면

1) 사용자 변수 설정 화면

(그림 3)은 메뉴에서 데이터의 종류를 랜덤으로 설정하고 있는 화면이다.

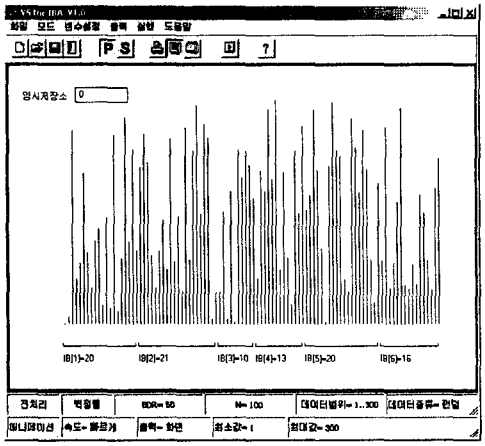


(그림 3) 사용자 변수 설정 화면

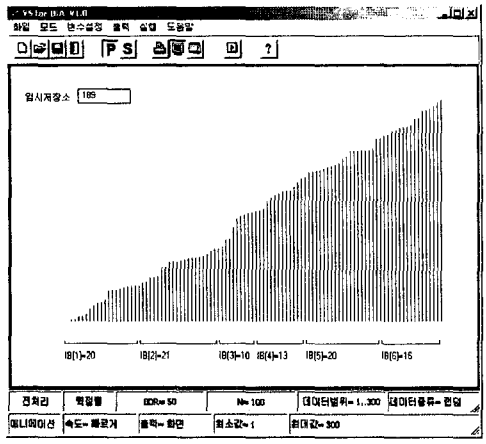
2) 전처리 전후의 데이터 분포

파일메뉴에서 '새로만들기'를 선택한 후, 설정된 변수의 값에 따라 초기화면이 (그림 4)와 같이 구성된다.

전처리가 완료된 데이터의 상태는 (그림 5)에서 확인할 수 있다.



(그림 4) 초기 데이터 분포



(그림 5) 전처리 후의 데이터 분포

4. 성능 평가

4.1 실험 조건

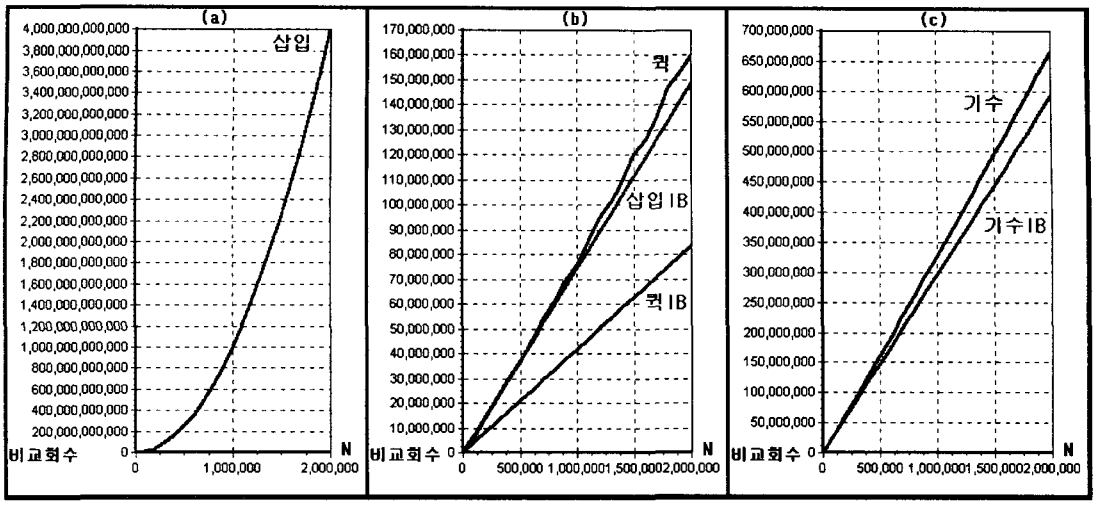
IBPA의 성능 개선 정도를 측정하기 위한 실험조건은 다음과 같다.

1) 정렬알고리즘 : 삽입정렬과 퀵정렬[2][3] 그리고 기수정렬[3]을 실험대상으로 선정하였다. 기수정렬은 기수교환정렬(radix exchange sort)로서, 32비트의 키(key)가 사용되었다.

2) 데이터의 종류 : 비교적 고른 분포를 보이는 난수발생기(random generator)로 생성된 랜덤 데이터로 실험하였다.

3) 데이터의 개수 : BDR이 50으로 설정된 상태에서 1,000개를 측정단위로 하여 2백만개 데이터까지 실험하였다. 그러나 삽입정렬은 $O(N^2)$의 평균시간복잡도를 가지므로 데이터의 개수가 10만개이상일 때는 측정단위를 10만개로 설정하여 실험하였다.

4) 비교회수 : 'if'와 'while' 그리고 'for'와 같은 명령문에서도 실제로는 비교가 발생하므로 정확한 비교회수를 측정하기 위해 이러한 명령문 역시 비교회수에 누적되



(그림 6) IBPA의 성능 측정

었다.

4.2 실험 결과

실험결과를 (그림 6)에 그래프로 나타내었다.

(그림 6)을 보면, 특이한 사실이 두 가지 있다.

첫째, IBPA가 적용된 경우 선형 시간(linear time)알고리즘의 경우처럼 비교회수의 출력선이 직선이 된다는 것이다. 이 사실은 시간복잡도의 기술부분에서 언급한 변수 p는 N과 차이가 있음을, 변수 q는 1보다 작은 수를 의미하는 것으로 해석될 수 있다. 물론 모든 실험데이터가 하나의 데이터블록에 속하는 최악의 경우에는 변수 p와 q는 각각 N과 1의 값을 가지게 되어 전처리가 무의미하게 된다.

둘째, 실험데이터가 랜덤데이터이므로 데이터의 상태에 따라 시간복잡도가 달라지는 점을 감안해야하지만, 삽입정렬에 IBPA를 적용했을 때 삽입IB 정렬알고리즘은 퀵 정렬과 대등한 정렬효과를 나타낸다는 것이다.

이 두 가지 사실에서 IBPA를 적용했을 때 정렬알고리즘의 시간복잡도는 $O(NlogN)$ 이하의 시간복잡도로 변화되어 향상된 정렬 효과를 보여준다고 할 수 있다.

결론적으로 2백만개의 랜덤데이터를 정렬한 경우를 기준으로 IBPA의 성능을 평가한다면, $O(N^2)$의 평균시간복잡도를 갖는 정렬알고리즘의 0.003%, $O(NlogN)$의 평균시간복잡도를 갖는 정렬알고리즘의 52%, 그리고 $O(N)$의 평균시간복잡도를 갖는 정렬알고리즘의 89%의 비교회수만으로도 정렬을 할 수 있음을 보여주었다. 따라서, IBPA의 효과는 $O(N^2)$의 평균시간복잡도를 갖는 정렬알고리즘의 경우에 가장 효과가 크다는 사실을 알 수 있다.

5. 결론

IBPA는 정렬 효과를 향상시키기 위한 새로운 전처리 알고리즘으로서, 성능 측정 결과 IBPA는 기존의 정렬알고리즘이 가지는 평균시간복잡도를 개선시키는 효과가 있음을 알 수 있었다.

그리고 본 연구에 이용된 알고리즘 이외의 다른 정렬 알고리즘의 경우에서도 즉, 버블정렬과 힙정렬, 그리고 선택정렬과 쉘정렬에 있어서도 알고리즘 개선 효과를 확인할 수 있었다. 또한 이미 정렬된 데이터를 이용하여 실험한 경우에 있어서도 마찬가지 효과가 있었다.

그러나 플래그(flag)를 사용하는 알고리즘의 경우, 이미 정렬된 데이터를 정렬할 때에는, 플래그(flag)를 사용하는 정렬알고리즘이 $O(N)$의 시간복잡도를 가지게 되므로 전처리 알고리즘의 효과가 없었다.

앞으로 연구해야할 과제라고 할 수 있는 것은 크게 세 가지로 볼 수 있다. 첫째, IBPA를 이용하여 기존의 정렬알고리즘보다 빠른 새로운 정렬알고리즘을 개발하는 것이다. 둘째, IBPA의 성능을 좌우하는 요소가 블록데이터범위(BDR)와 정보블록의 개수(M), 그리고 변수 p와 q이기 때문에, IBPA의 성능 향상을 위해서 이 네 가지 요소의 최적화에 대한 연구가 필수적이다. 셋째, 본 연구에서 사용된 데이터블록의 개념은 병렬처리시스템(parallel processing system)에서 유용할 것으로 생각되므로 이에 적용하는 것이다.

참고문헌

[1] 송태욱, 정보블록알고리즘, 마이크로소프트웨어 1998.12월호~1999.2월호.

[2] Herbert Schildt, Schildt's Expert C++, McGraw-Hill, 1996.

[3] Robert Sedgewick, Algorithms in C, Addison Wesley, 1998.

[4] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Introduction to Algorithms, The MIT Press, 1989.