2000년 한국정보처리학회 추계 학술발표논문집 제7권 제2호

정보 블록 정렬 알고리즘

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

A New Sort Algorithm : Information Block Sort Algorithm(IBSA)

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

요 약

본 논문에서는 정보블록알고리즘(IBPA; Information Block Preprocessing Algorithm)을 이용한 정보블록 정렬알고리즘(IBSA; Information Block Sort Algorithm)을 제안하고 그 성능을 평가하였다. IBSA의 시간복잡도는 $O(N)$ 이며, 데이터의 분포상태에 영향을 받지 않는다. IBPA의 성능을 측정해본 결과, 2백만개의 랜덤데이터를 정렬한 경우, 중복값 허용의 경우(a)는 퀵 정렬의 32.42%, 기수정렬의 9%정도의 비교회수만으로도 정렬할 수 있음을 보여주었으며, 중복값이 없는 경우(b)는 퀵 정렬의 53.12%, 기수정렬의 12.79%정도의 비교회수만으로도 정렬할 수 있음을 보여주었다.

1. 서론

기존의 정렬 알고리즘의 성능을 향상시키기 위하여 제안된 정보블록 전처리 알고리즘(IBPA; Information Block Preprocessing Algorithm)은 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[1]. 본 논문에서는 IBPA를 이용한 새로운 정렬 알고리즘 즉, 정보블록 알고리즘(IBSA; Information Block Sort Algorithm)을 제안하였다.

IBSA는 두 가지로 나누어지는데, 하나는 일반적인 데이터를 처리하는 경우의 정렬 알고리즘(IBSA_a)이며, 나머지 하나는 중복값이 없는 데이터를 처리하는 정렬 알고리즘(IBSA_b)이다.

2장에서는 사용된 용어의 정의, IBSA에 필요한 메모리와 자료 구조 그리고 알고리즘을 서술하였다. 3장에서 IBSA의 성능을 시뮬레이션을 통하여 제시하였으며, 4장에서는 결론을 나타내었다.

2. 정보블록 정렬 알고리즘

2.1 용어의 정의

다음은 본 연구에서 사용된 용어에 대한 정의와 그것에 대한 설명이다.

1) DB(Data Block) : 데이터 블록. 리스트를 M개로 나누었을 때 나누어진 각 부분을 말하며, 정보블록과는 다르지만 개수는 같다.
2) IB(Information Block) : 정보블록.
3) BDR(Block Data Range) : 하나의 데이터 블록에 포함될 수 있는 데이터의 범위를 수치화한 값 ($1 \leq BDR$).
4) M : 정보블록의 개수 ($M \leq N$).
5) 블록순회 : 하나의 데이터가 임시기억장소로 이동되어 비어있기 시작한 때부터 다시 이곳에 다른 데이터가 들어올 때까지의 작업기간을 의미한다.

2.2 자료구조와 메모리 사용량

IBSA에서 이용되는 데이터에는 DataInfo와 CopyData라는 두 가지 자료 구조가 있다. CopyData는 데이터블록에 있는 데이터를 그대로 복사한 것으로서, 자료구조는 데이터블록과 동일하다.

DataInfo는 각각의 데이터 블록을 정렬하기 위하여 필요한 자료구조로서, 데이터블록에 대한 정보를 저장하고 있다. DataInfo의 구성요소는 <표 1>과 같다.

<표 1> TDataInfo 형의 데이터 구조

구성요소	설 명
Value(V)	데이터의 값
Data_Count(DC)	Value 값을 가진 데이터의 개수
Accumulated_ Position(AP)	누적 시작 위치. CopyData에서 이동되어야 할 데이터의 기준위치값을 의미한다.
Sent_Count(SC)	CopyData에서 데이터블록으로 데이터가 정렬되면서 이동된 'Value'값의 데이터 개수

데이터를 정렬하기 위해서 필요한 메모리는 IBPA에 필요한 메모리에 DataInfo에 할당된 메모리와 CopyData에 할당된 메모리를 더한 값이다.

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

2.3 알고리즘

IBPA를 이용하는 IBSA의 알고리즘은 (그림 1)과 같다.

** 이 논문은 2000년도 두뇌한국21 사업 핵심분야에 의해 지원되었음

Part 1. IBPA 전처리
Part 2. 정렬 : M만큼 반복
  Step 1. TDataInfo 메모리 할당 및 초기화
    (누적 시작 위치 계산)
  Step 2. TCopyData 메모리 할당 및 복사
  Step 3. 원본인 Data 배열로의 데이터 이동
  Step 4. 동적 메모리 할당 해제

(그림 1) 정보블록 정렬 알고리즘

Part.1은 [1]에서 제안된 IBPA를 그대로 이용한다.

2.4 예제

예제를 통하여 IBSA의 정렬과정을 살펴보자.

전처리가 종료된 리스트가 단위 정보 블록의 정렬만을 남rrushg고 있고, 이 리스트의 첫 번째 정보블록의 요소가 다음과 같다고 가정하자. 이 경우 BDR = 3, Ps = 1, Pe = 5, Rs = 1, Re = 3이다.

1) 단위 정보 블록 복사

데이터블록[1]에 있던 데이터가 동적 할당된 Copy_Data로 복사되었다.

데이터블록[1]

1	3	2	3	1
[1]	[2]	[3]	[4]	[5]

: 인덱스

⬇ 복사

Copy_Data

1	3	2	3	1
[1]	[2]	[3]	[4]	[5]

2) DataInfo 초기화

동적 할당된 DataInfo의 구성요소가 초기화된 상태는 다음과 같다.

V	DC	AP	SC
1	2	1	0
2	1	1	0
3	2	1	0

3) 누적 위치 계산

다음은 누적 시작 위치 정보가 입력된 DataInfo의 정보를 나타낸 것이다. Sent_Count의 값은 Data 배열로 이동된 데이터가 없으므로 0으로 초기화된다.

1	3	2	3	1

: CopyData

[1] [3] [4] : 누적 시작 위치

V	DC	AP	SC
1	1	1	0
2	2	3	0
3	3	4	0

4) 데이터 블록으로 이동

이동될 위치는 누적시작위치에 삽입된 개수를 더한 값이다. 데이터의 이동이 끝난 후의 DataInfo는 다음과 같다.

V	DC	AP	SC
1	1	1	2
2	2	3	1
3	3	4	2

5) 데이터 블록[1]의 정렬 완료

2.5 시간복잡도

IBPA의 경우, 데이터가 하나의 데이터 블록에 집중되는 경우와 같은 최악의 경우는 효과가 없다. 하지만 IBSA의 경우, IBPA와는 달리 데이터블록에 있는 데이터의 개수가 많을수록 메모리의 사용량은 증가하지만 시간복잡도에 커다란 영향을 끼치지는 않는다.

Part.1의 비교회수는 IBPA의 비교회수와 동일하다. 그러므로 Part.1의 비교회수는 아래와 같으므로 시간복잡도는 $O(N)$과 같다.

Part. 1의 비교회수
$\in \Theta(N)$

Part.2의 비교회수는 아래와 같이 나타낼 수 있으며, 시간복잡도는 $O(N)$과 같다.

Part. 2의 비교회수
= Step.1에서 Step.5까지의 비교회수
= $C_1M + C_2M + C_3N + C_4$
$(1 \leq M \leq N)$
$\in \Theta(N)$

그러므로 Part.1과 Part.2의 비교회수를 더하여 IBSA의 시간복잡도를 나타내면 <표 2>과 같이 나타낼 수 있다.

<표 2> IB 정렬 알고리즘의 시간복잡도

시간복잡도
$T(N) = O(N)$

중복값이 없는 데이터를 정렬하는 경우, IBSA는 중복값이 있는 데이터를 정렬하는 경우보다 더욱 간단한 알고리즘이 되어 결과적으로 시간복잡도가 낮아지게 된다.

3. 성능평가를 위한 시뮬레이터

3.1 시스템 구조와 메뉴

시스템의 구조와 메뉴는 [1]에서 제시된 것과 동일하다.

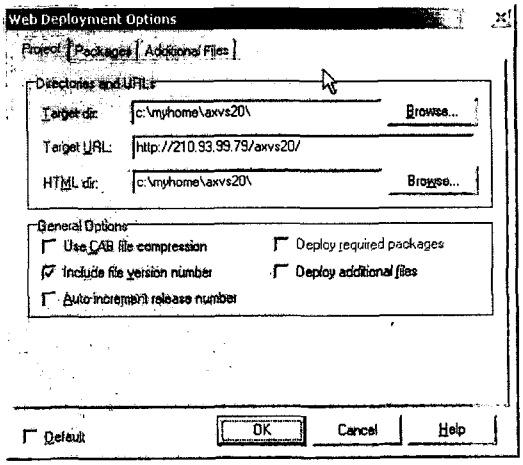
시뮬레이터는 독립형(standalone)과 웹형이 있는데, 독립형 프로그램의 경우, 웹서버로부터 프로그램을 다운로드 받아서 실행할 수 있도록 하였다. 웹형의 경우, HTML 문서를 요청하게 되면, 웹서버의 AXVS10 디렉토리에 있는 액티브엑스 컨트롤(ActiveX Control)을 전송해주도록 설정되어있다.

사용자가 설정할 수 있는 변수 역시 정렬방법, 구간값, 데이터의 범위, 데이터의 개수, 데이터의 종류, 애니메이션 속도로서, 모두 6가지이다.

3.2 구현 화면

1) ActiveX Control을 위한 설정화면

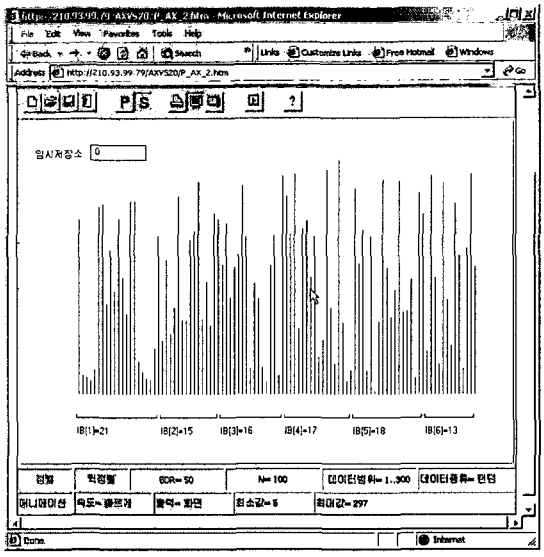
서버에 OCX를 설치하기 위해서는 (그림 2)에서 보는 것과 같이 환경설정이 필요하다. 즉, OCX가 설치될 로컬경로, 그리고 웹서버에서의 디렉토리를 설정한다.



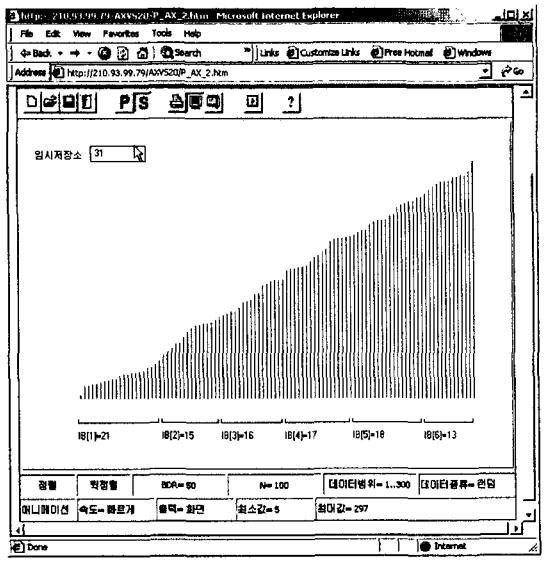
(그림 2) OCX 환경설정

2) 정렬 화면

(그림 3)에서는 정렬전의 데이터를, (그림 4)에서는 정렬후의 데이터를 보여주고 있다. 상태바를 보면, 현재 설정된 옵션들을 알 수 있는데, 알고리즘은 퀵정렬을 이용한 정보블록 정렬이며, BDR, N, TDR, MinDV, MaxDV의 값을 알 수 있다.



(그림 3) 정렬전 데이터



(그림 4) 정렬후 데이터

4. IB 정렬 알고리즘의 성능 평가

4.1 실험 조건

IBSA의 성능을 측정하기 위한 실험조건은 다음과 같다.

1) 정렬알고리즘 : 퀵정렬과 기수정렬을 실험대상으로 선정하였다. 기수정렬은 기수교환정렬(radix exchange sort)로서, 32비트의 키(key)가 사용되었다. 그리고 중복값이 없는 경우도 실험 알고리즘에 포함시켰다.
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.
4) 비교회수 : 'if'와 'while' 그리고 'for'와 같은 명령문에서도 실제로는 비교가 발생하므로 정확한 비교회수를 측정하기 위해 이러한 명령문 역시 비교회수에 누적되었다.

4.2 실험 결과

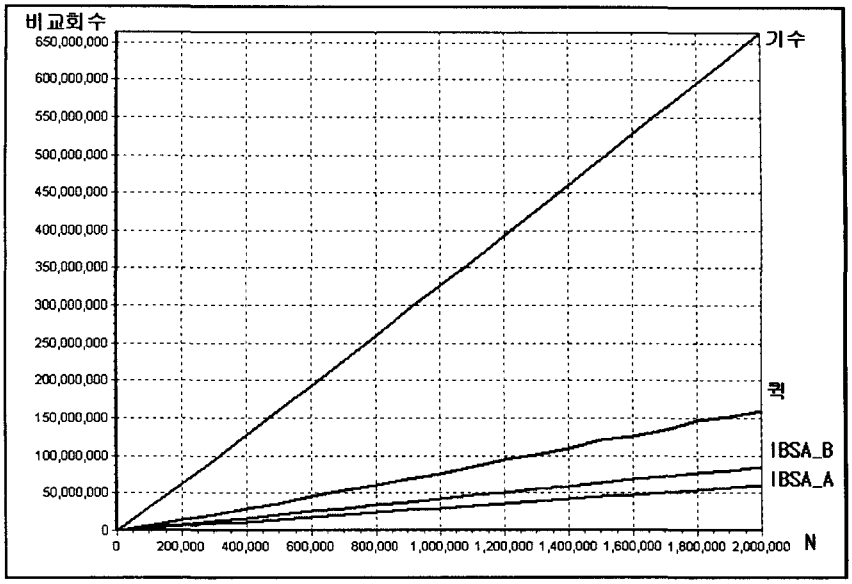
정렬알고리즘의 실험결과를 <표 3>과 (그림 5)에 나타내었다.

이 데이터를 보면, 2백만개의 랜덤데이터를 정렬한 경우, IBPA의 성능 측정 결과를 다음과 같이 분석할 수 있다.

중복값이 없는 IBSA_a의 경우, 퀵 정렬의 32.42%, 기수정렬의 9%정도의 비교회수만으로도 정렬할 수 있음을 보여주었다.

중복값이 있는 IBSA_b의 경우, 퀵 정렬의 53.12%, 기수정렬의 12.79%정도의 비교회수만으로도 정렬할 수 있음을 보여주었다.

따라서, IBSA_a의 효과가 IBSA_b보다 약 39.54%정도 우수한 성능을 보여주었음을 알 수 있다.



(그림 5) IBPA의 성능 측정-2

<표 3> IBPA의 성능 측정-1

알고리즘 / N	퀵	기수	IBSA_a	IBSA-b
100,000	6,508,913	30,804,945	2,993,318	4,247,838
200,000	14,078,821	62,686,712	5,984,862	8,495,930
300,000	21,997,105	94,883,326	8,977,247	12,743,847
400,000	29,511,842	127,587,293	11,971,008	16,991,828
500,000	36,702,715	160,523,742	14,964,004	21,239,816
600,000	45,190,842	193,119,302	17,956,770	25,487,874
700,000	53,055,451	226,330,705	20,949,949	29,735,793
800,000	60,435,795	259,571,928	23,942,249	33,983,805
900,000	69,097,174	293,029,894	26,936,878	38,231,714
1,000,000	75,664,428	326,516,669	29,928,390	42,479,846
1,100,000	85,303,696	359,473,505	32,920,334	46,727,858
1,200,000	94,548,009	392,899,692	35,913,453	50,975,893
1,300,000	101,028,857	426,631,143	38,905,803	55,223,855
1,400,000	109,118,130	460,308,863	41,900,532	59,471,748
1,500,000	120,064,074	494,297,381	44,894,092	63,719,804
1,600,000	125,846,906	528,005,463	47,884,618	67,967,834
1,700,000	134,453,913	561,869,233	50,876,546	72,215,942
1,800,000	147,149,809	596,019,506	53,870,985	76,463,869
1,900,000	152,537,415	629,975,649	56,862,818	80,711,846
2,000,000	159,946,280	664,097,438	59,858,646	84,959,886

5. 결론

IBPA를 이용한 새로운 정렬 알고리즘인 IBSA의 알고리즘과 성능을 분석해보았다.

IBSA_a는 중복값이 없는 데이터를 정렬하는 알고리즘이며, IBSA_b는 중복값의 존재가 불확실한 상태의 데이터를 정렬하는 알고리즘이다. IBSA_a와 IBSA_b는 일반적인 정렬알고리즘보다 빠르다는 점에서 긍정적인 평가를 받을 수 있지만, IBPA가 데이터의 분포상태에 따라서 성능이 약간 불안정해지는 문제점을 보완했다는 점에서 좀 더 나은 평가를 받을 수 있을 것이다.

앞으로 연구해야할 과제라고 할 수 있는 것은 [1]에서 제시한 것 이외에 랜덤 데이터의 생성 방법과 같은 실험 데이터의 조건을 보완하는 것이다.

참고문헌

[1] 송태옥, 구정모, 김태영. 정보블록 전처리 알고리즘. 한국정보처리학회 추계학술논문집, 2000.

[2] 송태옥, 정보블록알고리즘, 마이크로소프트웨어 1998.12월호~1999.2월호.

[3] Glenn J. Brooksheer, Computer Science, The Benjamin/Cummings Publishing Company, 1994.

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

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