일반화된 정수 정렬 알고리즘

저자 문예강

초록

항상 기수 정렬 보다 효율적이고, 카운팅 정렬보다 일반화된 형태의 혼합 정렬 알고리즘을 다룬다. 이 논문에서 제안하는 정수 정렬은 최악의 경우에도 퀵과 시간복잡도가 동일하다.

목차

1)알고리즘 소개

어떤 정렬할 N개의 원소를 가진 배열이 주어진다.

이 배열을 정렬한다고 하자.

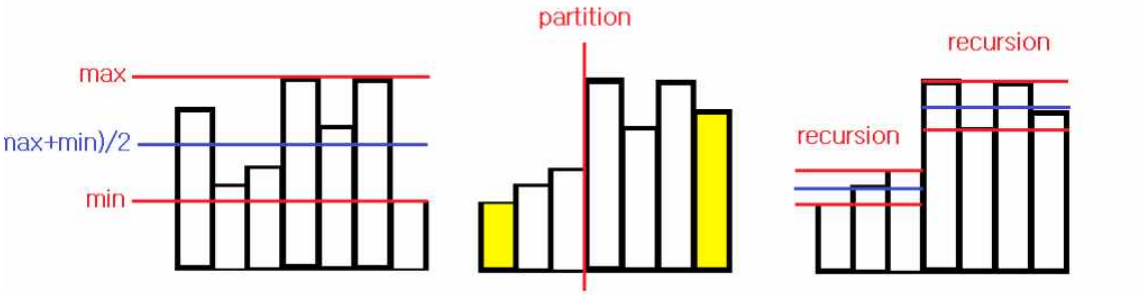
0. 배열에서 최댓값 Max와 최솟값 Min을 구한다.

1. 피벗을 (Max+Min)/2로 잡고, 퀵정렬을 수행한다.

이때, 나누어진 두 부분을 low, high라고 하고, 피벗보다 작은 값중 가장 큰값을 lowMax라고 하고, 피벗보다 큰값중 가장 작은값을 highMin이라 하자.

2. 구간 low에 대해서는, Min, lowMax를 이용해 재귀적으로 정렬하고, 구간 high에 대해서는, highLow, Max에 대해 재귀적으로 정렬한다.

이를 그림으로 나타내면 다음과 같다.



다음 조건을 만족시키는 경우, 카운팅 정렬한다.

> Max-Min+1<=N<=cntL (K는 상수)

N개의 원소를 정렬하는데에는 퀵정렬보다는 카운팅 정렬이 빠르다는 점을 이용해서 최적화하는 것이다.

해당 조건은 카운팅 정렬이 사용할수 있는 공간의 상한을 상수 cntL를 이용해 제한하기 위함이다. Min부터 Max까지의 범위를 0부터 Max-Min+1까지로 대응시켜 카운팅 정렬한다. 이때 제한조건에 의해 Max-Min+1<=cntL이므로, cntL만큼의 공간이 존재하면 문제없이 카운팅 정렬이 이루어진다.

2)알고리즘 특징

1. 시간복잡도는 O(NK)이다. (단, K는 배열에 존재하는 정수의 종류의 수)

2. 제자리 정렬이 아니다. 상수 공간인 cntL만큼을 추가로 요구한다.

3. 교환 정렬+분배 정렬이 합쳐진 혼합 정렬이다.

4. 기본적으로 정수 정렬 알고리즘이지만, 유리수를 큰 정수로 보아 정렬하는 것 또한 가능하다.

5. 최선의 경우에 가까울수록 카운팅 정렬에 가깝게 작동하고, 최악의 경우에 가까울수록 퀵정렬에 가깝게 작동한다.

6. 분포가 적은 수의 특정 구간에 집중되어 있을수록 효율적이다.

7. 수가 균일하게 분포한다면, 평균적으로 자료형의 크기보다 N이 작은 경우 O(Nlog N), 자료형의 크기보다 N이 큰 경우 O(N)알고리즘으로 작동한다.

3)다른 정렬과의 비교

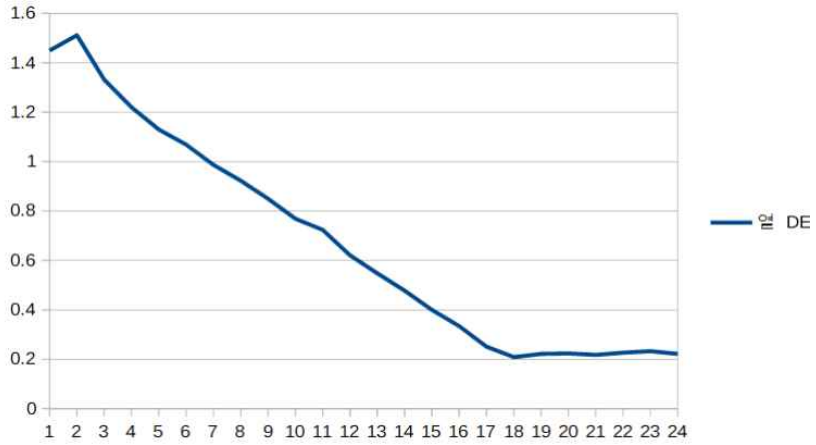
1. 카운팅 정렬
카운팅 정렬이 배열의 범위에 영향을 받는 반면, 이 정렬은 숫자의 가짓수에 영향을 받는다.
카운팅 정렬이 배열의 범위의 길이에 해당하는 추가 공간을 요구하는 반면, 이 정렬은 특정 상수만큼의 추가 공간을 요구한다.
{1,10^15}같은 배열을 정렬할 때 이 정렬이 카운팅 정렬보다 유리하다.

2. 기수 정렬
기수 정렬이 자릿수에 영향을 받는 반면, 이 정렬은 숫자의 가짓수에 영향을 받는다. 기수 정렬보다 상수가 작다.
마찬가지로 {1,10^15} 같은 배열을 정렬할 때 이 정렬이 카운팅 정렬 보다 유리하다.

3. 퀵 정렬
K=N인 경우, 퀵과 이 정렬은 동일한 성능을 보인다. 다만 약간의 오버헤드 때문에 퀵이 조금 더 빠르다. 숫자의 가짓수가 적은 경우, 퀵보다 훨씬 빠른 정렬 속도를 보인다.
{0,1,2,1,0} 같은 배열을 정렬할 때 이 정렬이 퀵 정렬 보다 유리하다. K<=N이기 때문이다.
cntL이 5일 때, {5,3,4,2,1}같은 배열을 정렬 할때에도 퀵정렬 보다 유리하다. 왜냐하면 N<=cntL이기 때문에 카운팅 정렬과 속도가 동일하기 때문이다.

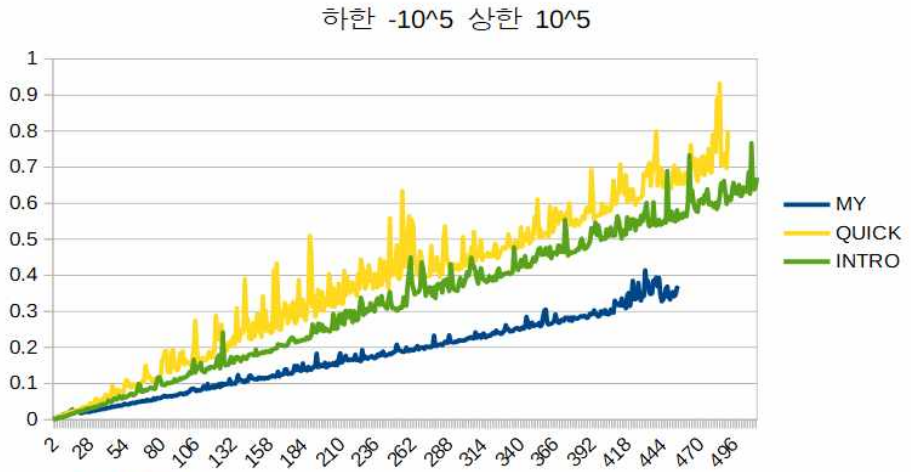
4)알고리즘 분석
1. 최악의 경우
이 알고리즘의 시간복잡도는 NK이지만, 실제로 K=N이 되는 경우는 드물다. 왜냐하면 이 케이스는 피벗을 나쁘게 잡아 한번의 퀵으로 하나의 원소 밖에 정렬이 이루어지지 않는 경우인데, 이 경우가 발생하려면 수의 범위가 2^N가 되기 때문이다. 이는 N이 불과10000개인 배열에서도 최댓값이 2^10000이 됨을 의미하는데, 일상적인 경우 거의 다루지 않는 숫자이다.
자료형이 32비트로 정해진 경우, N=K인 최악의 경우가 발생하는 경우는, N이 32정도로 매우 작은 수에서만 발생한다. 그리고 숫자가 등비가 2보다 큰 등비 수열에 가깝게 놓여있어야 최악의 경우가 발생한다.

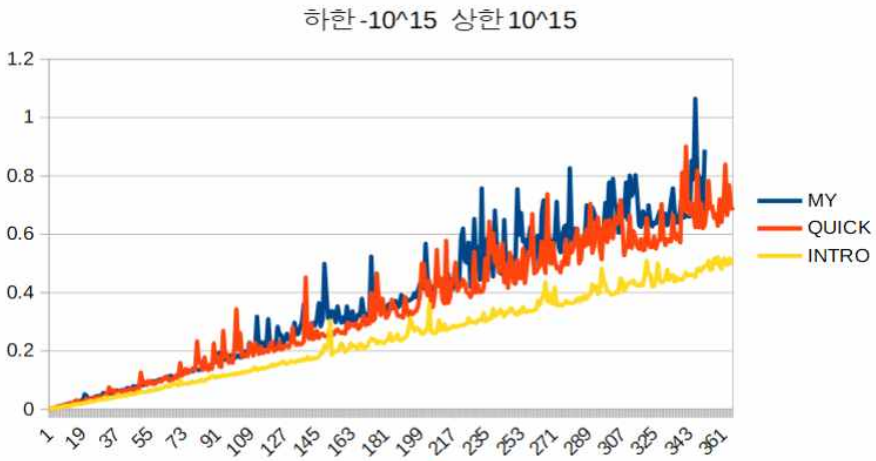
2. 추가 할당 공간
cntL의 길이가 길수록 대체로 수행시간이 감소한다. 아래 그래프는 N=10000000일때의 수행시간이다. 그래프의 가로축은 x=$\log_2 cntL$이다. 세로축은 소요시간(초)이다.



3. 퍼포먼스

아래 실험 결과는 x축 단위는 10000개이고, y축은 초단위다. -O2 옵션을 달아놓은 상태로 실행한 결과이다. 데이터는 메르센 트위스터를 사용하여 N개의 원소를 임의로 생성하여 이용하였다. 정렬에 사용한 cntL은 1000이다.





상대적으로 가짓수가 적은 -10^5~10^5범위의 수를 정렬하는데에는 시간이 인트로 정렬(STL sort)보다 빠름을 알 수 있다.

4. 추가 최적화 방법
알고리즘 상 Min과 Max값을 구하게 되어있는데, Min==Max인 경우 곧바로 리턴하는 것 만으로도 추가적인 최적화가 가능하다. 이 경우 모든 같은 수에 대해 처리가 일어나지 않으므로, 1이 N개 들어있는 배열을 정렬하는데 상수시간이 걸린다.

4)의의
이 정렬은 기수정렬과 카운팅 정렬을 일반화했다고 볼수 있다.
최대 최소를 이용해 재귀하는 부분을 lowMax와 HighMin이 아닌 (Max+Min)/2으로 잡고, 맨 정렬함수 최초 호출시 Max=자료형의 최댓값, Min=자료형의 최솟값으로 잡는다면 기수정렬과 동일하게 작동한다.
cntL을 배열의 최댓값으로 잡는다면, 카운팅 정렬과 동일하게 작동한다.
cntL=0로 잡는다면 퀵정렬과 동일하게 작동한다.
또한 단순한 구현으로 수의 가짓수에 맞춰 자동적으로 최적화할 수 있다는 점에서 의의가 있다. 일반적인 통계자료의 분포는 특정 구간에 몰린 숫자가 많으므로, 분포가 특정 구간에 집중된 경우 효율을 발휘하는 이 정렬이 의미가 있음을 알 수 있다.