<알고리즘(1분반) HW2 보고서 >

B611197 조용우

“input.txt”라는 주어진 파일을 읽어와서 배열을 만들고, “input.txt”의 1~3번째 줄에는 오름차순으로 정렬이 되어 있는 배열을 설정하였고 4~6번째 줄에는 srand()함수를 이용하여 랜덤값을 설정하여 만들어진 정렬이 되지 않은 배열을 mergesort와 quicksort를 이용하여 start와 end 시간을 각각 측정하여 걸린 시간을 저장하여 출력하였다.

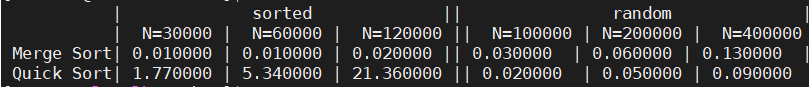
<Merge Sort>

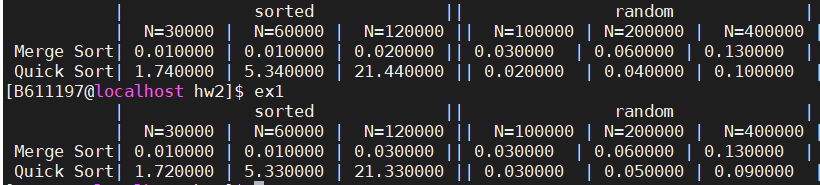
머지소트는 간단히 말해서, 각각의 배열을 하나가 될 때 까지 분할한 다음에 그것들을 정렬하면서 다시 이어 붙이는 정렬방식이다. 크게 세가지 동작으로 볼 수 있는데 분할->정렬->병합이다. mid라는 변수에 처음 인덱스와 끝 인덱스를 더한 값을 2로 나누는 것은 배열의 중간 지점을 구하려는 것이며, 조건에 충족할 때까지 (끝 인덱스가 처음 인덱스 값보다 작을 때까지) 반복하면 각각의 범위들은 원소 하나를 가르키게 된다. 이러한 나눠지는 과정을 재귀(recursion)로 실행하게 되는데, 이것은 n개의 데이터를 2/n개로 나누고 4/n…을 반복한다. 그 다음으로는 병합을 하는데 정렬해서 붙이게 된다.

<Quick Sort>

퀵소트는 피벗(pivot)을 하나 정해서, 그것과 비교하며 작은 값, 큰 값을 찾아서 정렬을 해나가는 정렬이다. 피벗을 제일 왼쪽으로 잡을 수도 중간값을 잡을 수도 있지만 나는 제일 왼쪽에 있는 [0]인덱스의 값을 피벗으로 지정하였다. 그리고 피벗보다 큰 수를 찾아서 담는 변수 i를 pivot+1의 인덱스에 지정해 놓았고, 피벗보다 작은 수를 찾아서 담는 변수 배열의 마지막 인덱스를 지정해 놓았다. while문에서 각각의 조건을 만족할 때까지 돌면서 i가 j보다 커질 때까지 반복하고 i와j의 위치를 바꾼다. 위의 mergesort와 비슷한 재귀(recursion)를 이용해 범위를 분할하고 숫자를 정렬한다.

<결과분석>





“input.txt(20000 40000 80000 100000 200000 400000)”를 읽어서 정렬한 결과이다.

1)정렬된 데이터 (3개 결과의 평균)

|  |  |  |  |
| --- | --- | --- | --- |
|  | N=30000 | N=60000 | N=120000 |
| Merge Sort | 0.01s | 0.01s | 0.025s |
| Quick Sort | 1.74s | 5.33s | 21.38s |

정렬된 데이터에서는 역시나 퀵소트가 머지소트에 느린 것을 알 수 있다. (퀵소트 최악의경우 시간복잡도 O(n^2)이기 때문이다.)

Quick Sort : N값이 60000에서 120000으로 2배증가 했을 때 (n^2)은 이론상 4배 증가한다. 5.33\*4=21.32(오차0.06)인걸로 보아 이론상 시간복잡도(O(n^2))와 일치함을 알 수 있다.

Merge Sort : N값이 60000에서 120000으로 2배증가 했을 때 실행시간은 2.5배 증가했다. 이론상 (nlogn)이라 2.~배보다 조금 더 증가해야 하지만 실제로는 더 많이 증가한 것을 볼 수 있다.

2)정렬 되어있지 않은 데이터(3개 결과의 평균)

|  |  |  |  |
| --- | --- | --- | --- |
|  | N=100000 | N=200000 | N=400000 |
| Merge Sort | 0.03s | 0.06s | 0.13s |
| Quick Sort | 0.023s | 0.047s | 0.093 |

정렬 되어있지 않은 데이터에서는 퀵소트가 머지소트에 빠른 것을 알 수 있다. 그러므로 같은 N에대하여 같은 시간복잡도를 가지고 있지만 퀵소트가 머지소트보다 빠르다는 것을 알 수 있다.

Quick Sort : N값이 100000에서 200000으로 2배증가 했을 때 시간이 2배보다 약간 증가(0.023에서0.047)했다. 또한 400000으로 4배증 가했을 때 4배보다 약간 증가(0.023에서0.093)하였다. 그러므로 이론상 시간복잡도(nlogn)와 일치한다.

Merge Sort: N값이 100000에서 200000으로 2배증가 했을 때 시간이 2배 증가(0.03에서0.06)했다. 또한 400000으로 4배증가했을 때 4배보다 약간 증가(0.03에서0.13)하였다. 그러므로 이론상 시간복잡도(nlogn)와 일치한다.

<결론>

Merge Sort(병합 정렬)은 시간복잡도가 항상 O(nlogn)으로 정렬법들 중에서 준수한 편이지만 임시배열에 원래배열들을 계속해서 옮기며 정렬을 하는 방법이기 때문에 추가적인 메모리가 필요하다는 단점이 있다.

Quick Sort(퀵 정렬)은 시간복잡도가 O(nlogn)으로 가장 빠른 정렬방법중의 하나라고 알려져 있지만 기준값(pivot)에 따라서 시간복잡도가 크게변한다. 최악의 경우 위에서 정렬된 데이터를 가지고 퀵정렬을 하는 경우 시간복잡도는 O(n^2)으로 Merge Sort에비해 상당히 느리다는 것을 알 수 있다.