Sort 방법에 따른 데이터 개수와 정렬 속도 간의 상관 관계 비교 실험

2016\*\*\*\*

1. 실험 환경

OS: Windows10 Home

CPU: Intel(R) Core(TM) i3-5005U CPU @2.00GHz 2.00GHz

RAM: 4.00GB

Compiler: Visual Studio 2015 Community Release Mode

1. 구현 Sort 종류

* 아래의 정렬 알고리즘은 모두 nondecreasing order을 목표로 한다.

1. Insertion Sort
2. Selection Sort
3. Quick Sort

데이터의 크기가 3 이하일 경우 selection sort의 방식으로 남은 데이터를 처리함

1. M의 값에 따른 Quick Sort

데이터의 크기가 M 이하인 경우 Selection Sort로 남은 데이터를 처리함

1. 불완전한 Quick Sort에 대한 Insertion Sort

M개 이상인 데이터에 대해 quick sort를 실행 후, 어느 정도 정렬된 데이터에 대해 insertion sort를 사용

1. 정렬 데이터셋 구성

데이터의 종류는 총 4가지 타입으로 나누었으며, 각 타입의 데이터 개수 크기가 다른 4가지의 데이터셋을 사용하여 총 16가지의 데이터 셋을 사용했다. 타입과 크기는 다음과 같다.

1. 타입
2. Entirely Random: 0~2^15까지의 숫자 중 랜덤하게 n개만큼 배열
3. Ascending Order: 0~n-1까지 오름차순으로 정렬된 n개의 데이터 배열
4. Descending Order: N-1부터 0까지 내림차순으로 정렬되어 있는 n개의 데이터 배열

Few Swap0~n-1 로 오름차순 정렬되어 있는 n개의 데이터 중에서 0~square root(n) 사이의 값 중 임의로 선택된 두 값의 절대값이 루트 n보다 작은 경우, 두 수의 위치를 바꾼 경우의 수가 루트 n번 존재하는 n개의 데이터셋

1. 데이터 개수 크기 n 값
2. N = 2^5
3. N = 2^10
4. N = 2^15
5. N = 2^17
6. 실험 개요
7. 기본 실험 디자인

각각의 정렬 알고리즘에 대하여 주어진 데이터 셋을 정렬하는 데 걸리는 시간을 구한다. N이 2^5일 때에는 오차 범위가 미미하므로 한 번 정렬하는 데 걸린 시간을 표시하고, n이 2^10개 이상일 때에는 정렬을 총 5번 실행하여 각각의 정렬 시간의 평균값을 구한다.

1. 실험 목적

각각의 정렬 알고리즘을 직접 실행하여 실제 정렬에 걸리는 시간과 이론상의 시간복잡도를 비교해 본다. 또한, 정렬 방법 4와 5에서 최적의 결과를 얻기 위한 M의 값을 찾는다.

1. 정렬 방법에 따른 데이터 크기와 정렬 시간 간의 관계 - 실험 결과

* 위의 1~3 정렬 알고리즘으로 데이터를 정렬
* 각 정렬 알고리즘이 데이터를 정렬하는 데 걸린 상세한 실행 시간은 부록 참고.
* 아래는 각 알고리즘이 데이터를 정렬하는 데 걸린 시간의 평균값을 기록한 것임.

1. Selection Sort

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 단위: MS | Entirely Random | Ascending | Descending | Few Swap |
| N=2^5 | 0.002566 | 0.001540 | 0.002053 | 0.001540 |
| N=2^10 | 1.4935756 | 1.263874 | 1.3652796 | 1.844595 |
| N=2^15 | 991.8710564 | 908.1393068000001 | 792.9801392 | 1032.3567506 |
| N=2^17 | 12216.977539200001 | 11496.1279298 | 11748.3175772 | 11645.057616999999 |

1. Insertion Sort

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 단위: MS | Random | Ascending | Descending | Few Swap |
| N=2^5 | 0.001026 | 0.000513 | 0.000513 | 0.000513 |
| N=2^10 | 0.732315999999999 | 0.0035924000000000004 | 1.0020464 | 0.009647999999999999 |
| N=2^15 | 674.4715208 | 0.22508319999999998 | 505.26590560000005 | 0.1752014 |
| N=2^17 | 5435.7451662 | 1.240986 | 9694.2330078 | 1.4843388000000002 |

1. Quick Sort

* Pivot element 는 배열의 가장 왼쪽 값으로 지정하여 실행함.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 단위: MS | Random | Ascending | Descending | Few Swap |
| N=2^5 | 0.002566 | 0.002053 | 0.004619 | 0.002053 |
| N=2^10 | 0.7596174 | 0.7850714 | 1.3324358 | 0.6289606 |
| N=2^15 | 518.7373414 | 494.8627992 | 측정불가 | 494.69469019999997 |
| N=2^17 | 3263.5086424 | 4241.5137694 | 측정불가 | 3862.3903807999995 |

1. Quick + Selection Sort 최적화 M값 찾기

|  |  |
| --- | --- |
| N=2^17 | Random |
| M=2^5 | 2863.697754 |
| M=2^6 | 4746.457520 |
| M=2&7 | 3324.718750 |
| M=2^8 | 4788.681152 |

1. Quick + Insertion Sort 최적화 M값 찾기

|  |  |
| --- | --- |
| N=2^17 | Random |
| M=2^5 | 4619.106445 |
| M=2^4 | 2786.895996 |
| M=2^3 | 2733.338867 |
| M=2^2 | 3140.543701 |

1. 실험 정리
2. 일반적인 선택정렬의 시간복잡도는 O(n^2)이다. N이 2^5인 선택정렬의 이론적 시간복잡도는 2^10이고, N이 2^10인 선택정렬의 시간복잡도는 2^20이다. 따라서 데이터의 개수가 2^5개일 때와 2^10일 때의 선택정렬 시간은 2^10만큼 차이가 나야 한다. 실제 실험을 통해 정확하게 1024배는 아니지만 약 900배로 그 값이 1024배에 근접함을 알 수 있다.
3. 일반적인 insertion sort 역시 이론적 시간복잡도가 O(n^2)이다. 1)에서와 마찬가지로 일반적인 랜덤 배열의 데이터에서 insertion sort는 1024배에 근접하는 시간 차이를 보여준다.
4. Insertion sort에서 이미 정렬되어 있는 ascending order의 정렬 시간은 데이터의 크기가 커지더라도 정렬이 완료되는 데 걸리는 시간은 크게 늘어나지 않는다. Insertion Sort의 이론적 Best case는 O(n)이기에 실제 시간과 이론적 시간복잡도가 일치하는 것처럼 보인다. 하지만 선택 정렬의 경우 이미 정렬되어있는 데이터라 할지라도 O(n^2)만큼의 시간이 소요된다.
5. 같은 데이터에 대하여 Selection Sort와 Insertion Sort를 실행하면 Insertion Sort의 실행 속도가 더 빠름을 알 수 있다. 이는 selection sort의 경우 데이터의 모든 구간을 두 번 도는 동안에 Swap이 계속해서 발생하지만, Insertion Sort의 경우 데이터 전체를 탐색하는 구간이 적어 반복된 swap을 피할 수 있기 때문으로 추정된다. 즉, Selection Sort는 데이터의 정렬 상태와 상관 없이 평균적으로 O(n^2)의 시간복잡도가 필요한데 반해 Insertion Sort는 최악의 경우에만 O(n^2)의 시간이 걸린다.
6. Few Swap 데이터 타입에 대한 정렬 실험을 통해 데이터가 어느정도 이상 정렬되어 있는 경우에는 quick sort보다 insertion sort가 효율적임을 확인할 수 있다.
7. 위에서 insertion sort와 quick sort의 실험 결과를 통해 n이 약 2^10개까지는 quick sort 대신 insertion sort를 쓰더라도 비슷한 시간을 결과로 얻을 수 있음을 확인하였다.
8. 4번 실험을 통해 Quick + Selection Sort에서 M 값이 약 2^5 미만일 경우 어느정도 효과를 볼 수 있는 것처럼 보인다. 하지만 실제 여러 차례 실험을 진행한 결과, M 값이 데이터의 전체 정렬 속도에 유의미한 영향력을 주는 것처럼 보이지는 않는다.
9. 5번 실험 결과 quick + insertion sort 정렬 방식에서 데이터가 2^17개인 경우 M값이 2^3개 , 즉 ㄷㅔ이터를 8개 정도 남겨 놓고 퀵 정렬을 끝낸 뒤, 거의 다 정렬된 전체 데이터에 대해 insertion sort를 사용하는 것이 전체 시간을 줄이는 데 보다 효과적임을 확인하였다.
10. 실험을 마치며

컴퓨터 과학 분야에서 증명된 이론이 실제 컴파일러에서 적용되고 활용해 보는 뜻깊은 시간이었다. 본 실험을 통해 컴퓨터와 수학 간의 강한 연관성을 직접 경험을 통해 확인할 수 있었다.

부록

Selection Sort

2^5 root, 0.001540

2^5 내림차순, 0.002053

2^5 오름차순, 0.001540

2^5 Entirely Random, 0.002566

2^10 Entirely Random,

1.214198

0.945801

1.077690

3.147369

1.082822

Everage : 1.4935756

2^10 Root

1.012516

2.074810

4.259955

0.930919

0.944775

Everage : 1.844595

2^10 내림차순

1.758688

1.146457

1.673499

1.192644

1.055110

Everage : 1.3652796

2^10 오름차순

1.198802

0.920655

0.791846

2.299073

1.108994

Everage : 1.263874

2^15 랜덤,

791.737610

763.721313

1036.006104

854.934326

1512.955933

Everage : 991.8710564

2^15 root,

789.124512

927.949280

1137.093872

1018.589111

1289.026978

Everage : 1032.3567506

2^15 오름차순

746.646606

998.084290

1052.086182

784.709534

959.169922

Average : 908.1393068000001

2^15 내림차순

645.470581

590.695740

681.125610

822.641968

1224.966797

Everage : 792.9801392

2^17 Entirely Random

10754.116211

12933.951172

12421.023438

10141.002930

14834.793945

Everage : 12216.977539200001

2^17 오름차순

11057.259766

10177.330078

12259.686523

11529.572266

12456.791016

Everage : 11496.1279298

2^17 내림차순

10309.402344

12525.227539

10119.295898

15976.194336

9811.467773

Everage : 11748.3175772

2^17 root

11666.134766

10747.325195

12846.983398

12422.957031

10541.887695

Everage : 11645.057616999999

IS

2^5 Entirely Random : 0.001026

2^5 내림차순: 0.000513

2^5 오름차순 : 0.000513

2^5 Few Swap : 0.000513

2^10 Entirely Random :

0.273015

1.757661

0.481368

0.660983

0.488553

Everage : 0.7323159999999999

2^10 오름차순

0.003592

0.002053

0.002566

0.007185

0.002566

Everage : 0.0035924000000000004

2^10 내림차순

1.113613

1.532886

1.305031

0.616849

0.441853

Everage : 1.0020464

2^10 Few Swap :

0.008211

0.009751

0.015396

0.006671

0.008211

Everage : 0.009647999999999999

2^15 Entirely Random

922.854858

426.677063

580.661987

637.694946

826.468750

Everage : 674.4715208

2^15 오름차순

0.133941

0.193471

0.180128

0.236579

0.381297

Everage : 0.22508319999999998

2^15 내림차순

359.579529

410.511169

517.863525

494.499695

743.875610

Everage : 505.26590560000005

2^15 few swap

0.205274

0.141639

0.270962

0.127270

0.130862

Everage : 0.1752014

2^17 Entirely random

4819.166016

3796.591553

8743.718750

4234.103516

5585.145996

Everage : 5435.7451662

2^17 오름차순

0.154982

3.796549

1.229080

0.625060

0.399259

Everage : 1.240986

2&17 내림차순

10237.650391

8511.423828

9839.684570

9182.071289

10700.334961

Everage : 9694.2330078

2^17 Few swap

0.963250

0.964276

1.852601

1.954212

1.687355

Everage : 1.4843388000000002

Quick Sort

2^5 Entirely Random : 0.002566

2^5 오름차순 : 0.002053

2^5 내림차순 : 0.004619

2^5 Few swap : 0.002053

2^10 Entirely Random

0.505488

1.006357

0.709736

0.870876

0.705630

Everage : 0.7596174

2^10 오름차순

0.720000

1.172629

1.012002

0.715894

0.304832

Everage : 0.7850714

2^10 내림차순

1.396892

1.732515

1.015082

1.553414

0.964276

Everage : 1.3324358

2^10 Few Swap

0.367954

0.539358

1.661696

0.228881

0.346914

Everage : 0.6289606

2^15 entirely random

301.526428

671.490784

716.303162

621.822144

282.544189

Everage : 518.7373414

2^15 오름차순

541.110291

700.867065

365.362122

339.268646

527.705872

Everage : 494.8627992

2^15 Few Swap

518.815430

727.550171

279.364502

445.665955

502.077393

Everage : 494.69469019999997

2^17 Entirely Random

2602.458008

3732.330078

3289.478271

3893.521484

2799.755371

Everage : 3263.5086424

2^17 오름차순

4421.308594

4367.841797

3586.558105

3562.769531

5269.090820

Everage : 4241.5137694

2&17 few swap

3127.146973

2736.192139

4671.774902

4353.842285

4422.995605

Everage : 3862.3903807999995