**알고리즘 설계와 분석 실험 보고서**

**20211590 정은교**

1. **개발 환경**

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

위는 과제2의 experiment를 진행한 H/W의 정보이고, 서버는 교내 cspro 서버에서 진행되었다. 과제2의 개발은 vscode에서 cspro 서버에 원격으로 연결하여 진행하였다. Test case은 각 sort 함수에 대하여 여러 개(10, 100, 1000, 10000, 100000)의 random-order test case , 1개의 decreasing-order test case로 나누어 진행하였다. Decreasing-order test case의 입력은 1000개의 input size로 진행되었다.

1. **알고리즘 성능 비교**
2. Selection sort
   * Random order test case

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Input size | 10 | 100 | 1000 | 10000 | 100000 | Average |
| Time(sec) | 0.000004 | 0.000086 | 0.006808 | 0.377813 | 25.625831 | 5.2021084 |

1. Quick sort
   * Random order test case

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Input size | 10 | 100 | 1000 | 10000 | 100000 | Average |
| Time(sec) | 0.000004 | 0.000030 | 0.000399 | 0.004945 | 0.036456 | 0.0083668 |

1. Merge sort
   * Random order test case

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Input size | 10 | 100 | 1000 | 10000 | 100000 | Average |
| Time(sec) | 0.000013 | 0.000109 | 0.001224 | 0.012689 | 0.089393 | 0.0206856 |

1. My sort

Input size가 15이하일 때는 insertion sort를, 아닌 경우 quick sort를 진행한다. Quick Sort는 최악의 경우 시간 복잡도가 O(n^2)일 수 있다. 이미 정렬된 데이터나 거의 정렬된 데이터에 대해서는 성능이 떨어질 수 있다. 이 경우에는 Insertion sort와 같은 알고리즘이 Quick Sort보다 나을 수 있다. 이들은 최악의 경우에 O(nlogn)의 성능을 보장하기 때문이다.

초반에 Introspective Sort의 아이디어를 적용하려고 했으나, 초기 설계가 오히려 Quick Sort보다 시간이 더 걸리는 현상을 발견했다. 따라서, Heap Sort와 Randomly Partition을 제외하고 Median-of-Three Partition을 적용하여도 여전히 Quick Sort보다 시간이 많이 소요되었다. 최종적으로, Insertion Sort를 Quick Sort에만 적용하여 성능을 개선하였다.

* + Random order test case

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Input size | 10 | 100 | 1000 | 10000 | 100000 | Average |
| Time(sec) | 0.000004 | 0.000027 | 0.000344 | 0.004983 | 0.033431 | 0.0077578 |

Blue: selection sort

Green: Quick sort

Grey: Merge sort

Yellow: My sort

라인, 평행, 도표, 그래프이(가) 표시된 사진

자동 생성된 설명

위는 random order input의 경우 my sort, quick sort, merge sort, selection sort 순으로 시간 소요가 적었다. (그림에서green(quick sort) 그래프가 잘 보이지 않는 이유는 yellow(my sort)와 비슷한 값을 가지기 때문이다. 하지만, my sort가 더 빠르다.)

1. **Decreasing order(Worst-case)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sorting type | Selection | Quick | Merge | My sort |
| Time(sec) | 0.000077 | 0.000104 | 0.000173 | 0.000021 |

worst case에 해당하는 100~1까지 decreasing order의 input을 4개의 sorting algorithm으로 정렬하였을 때 My sort, selection, quick, merge sort 순으로 시간 소요가 적었다.