**Algorithm**

**Improvement**



학과 : 산업시스템공학과

학번 : 2016112582

이름 : 김유탄

N : 배열의 랜덤 숫자의 개수 K : 구간의 개수

* 개선 전 알고리즘 설명

- N개로 이루어진 랜덤 배열 생성 : O(N)

- 랜덤으로 생성된 구간의 배열 만들기 : O(K\*N)

- 랜덤으로 생성된 구간에서 최대 최소 구하기 : O(K\*N)

- 랜덤으로 생성된 구간에서 합계 구하기 : O(K\*N)

- 최종 시간 복잡도

O(N) + 3\*O(K\*N) -> O(K\*N) (최대 차수만 가져오기)

* 개선 후 알고리즘 설명

N : 배열의 랜덤 숫자의 개수 K : 구간의 개수

- N개로 이루어진 랜덤 배열 생성 : O(N)

- N개로 이루어진 구간 최소값을 위한 세그먼트 트리 만들기 : O(N)

- N개로 이루어진 구간 최대값을 위한 세그먼트 트리 만들기 : O(N)

- N개로 이루어진 구간 합을 위한 누적 배열 만들기 : O(N)

- 랜덤으로 생성된 구간에서 최대값 구하기 : O(K\*logN)

- 랜덤으로 생성된 구간에서 최소값 구하기 : O(K\*logN)

- 랜덤으로 생성된 구간에서 합계 구하기 : O(K\*1)

- 최종 시간 복잡도

3\*O(N) + 2\* O(K\*logN) + O(K) -> O(K\*logN) (최대 차수만 가져오기)

* 시간 복잡도 개선 내용

기존 알고리즘의 시간 복잡도는 O(K\*N) 이었고, 새롭게 개선한 알고리즘은 segment tree를 이용해서 최대 최소를 구하고, 누적 합계 배열을 이용해 합계를 구하는 과정에서, 배열을 N시간을 통해 만들어 두면, 구간 합계는 상수 시간 내에 구할 수 있도록 했다.

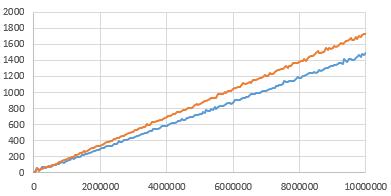
시간 복잡도를 보면 개선한 알고리즘의 성능이 더 좋다고 판단된다.

* N 고정 K 변화

1. N = 100 고정 K 변화( 0 ~ 10,000,000 )

Y축 : 시간 (ms) , X 축 : K(구간 개수) , 파란색선 : 개선 전 , 주황색선 : 개선 후

1. 시간 비교 그래프



1. 시간 비교 테이블 (ms)

|  |  |  |
| --- | --- | --- |
| K (N = 100 고정) | 개선 전 | 개선 후 |
| 1 | 0 | 0 |
| 100 | 1 | 1 |
| 10,000 | 11 | 8 |
| 1,000,000 | 240 | 282 |
| 100,000,000 | 15197 | 17822 |

1. 분석

- 개선 전 알고리즘은 O(K\*N), 개선 후 알고리즘은 O(K\*logN)의 시간 복잡도를 가진다. N이 고정 되는 경우에는 두 알고리즘은 각각 N과 K에 비례하는 시간 복잡도가 나온다. 즉 선형 시간 복잡도가 나타나고, 위 그래프에서 확인 가능하다.

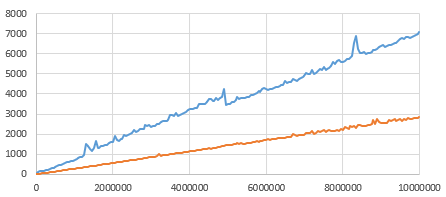
- 문제점 : 개선을 했음에도 불구하고, N = 100으로 고정했을 경우에, 개선한 알고리즘의 시간이 더 오래 걸린다.

- 문제 원인 : 개선 한 알고리즘의 시간 복잡도는 3\*O(N) + 2\* O(K\*logN) + O(K) 이고, 개선 전 알고리즘은 O(N) + 3\*O(K\*N) 이다. 개선된 알고리즘은 구간의 개수와 상관 없이 초기 O(N)의 시간 복잡도를 가진 작업이 개선 전 보다 더 많기 때문에, N이 충분히 크기 않은 경우에는 성능이 더 낮게 측정 된다.

1. N = 1000 고정 K 변화( 0 ~ 10,000,000 )

Y축 : 시간 (ms) , X 축 : K(구간 개수) , 파란색선 : 개선 전 , 주황색선 : 개선 후

1. 시간 비교 그래프



1. 시간 비교 테이블 (ms)

|  |  |  |
| --- | --- | --- |
| K (N = 1000 고정) | 개선 전 | 개선 후 |
| 1 | 0 | 1 |
| 100 | 2 | 1 |
| 10,000 | 50 | 12 |
| 1,000,000 | 919 | 426 |
| 100,000,000 | 80764 | 29741 |

1. 분석

- 개선 전 알고리즘은 O(K\*N), 개선 후 알고리즘은 O(K\*logN)의 시간 복잡도를 가진다. N이 고정 되는 경우에는 두 알고리즘은 각각 N과 K에 비례하는 시간 복잡도가 나온다. 즉 선형 시간 복잡도가 나타나고, 위 그래프에서 확인 가능하다.

- A번과 다르게 N이 충분히 큰 경우에는 개선 한 알고리즘의 성능이 높게 평가된다. 그 이유로는 N이 커지면, 랜덤하게 생성되는 구간에 대한 각각의 연산(최대, 최소, 합계)의 시간이 영향을 받는다. 구간 관련 연산에서 개선 전에는 O(K\*N), 개선 후에는 O(K\*logN)으 시간 복잡도를 갖고, N에 대한 가중치는 개선 전의 알고리즘이 더 높다..

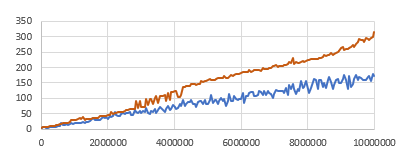
즉 N이 증가함에 따라 알고리즘 시간에 대한 영향이 개선 전알고리즘에서 더 크게 받기 때문에, N이 일정 값보다 크다면 개선된 알고리즘의 성능이 더 좋다.

* K 고정 N 변화

1. K = 10 고정 K 변화( 2 ~ 10,000,000 )

Y축 : 시간 (ms) , X 축 : N(구간 개수) , 파란색선 : 개선 전 , 주황색선 : 개선 후

1. 시간 비교 그래프



1. 시간 비교 테이블 (ms)

|  |  |  |
| --- | --- | --- |
| N (K = 10 고정) | 개선 전 | 개선 후 |
| 2 | 0 | 0 |
| 10 | 0 | 0 |
| 100 | 1 | 0 |
| 1,000 | 0 | 1 |
| 10,000 | 3 | 1 |
| 100,000 | 9 | 9 |
| 1,000,000 | 31 | 41 |
| 10,000,000 | 250 | 425 |

1. 분석

- 개선 전 알고리즘은 O(N) + 3\*O(K\*N), 개선 후 알고리즘은 3\*O(N) + 2\* O(K\*logN) + O(K) 의 시간 복잡도를 가진다. K가 고정 되는 경우에는 두 알고리즘은 모두 N에 비례하는 시간 복잡도를 갖게 된다. 이는 위 그래프에서 확인 가능 하다.

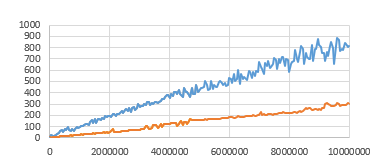
- 문제점 : 개선을 했음에도 불구하고, N = 100으로 고정했을 경우에, 개선한 알고리즘의 시간이 더 오래 걸린다.

- 문제 원인 : 개선 한 알고리즘의 시간 복잡도는 3\*O(N) + 2\* (K\*logN) + O(K) 이고, 개선 전 알고리즘은 O(N) + 3\*O(K\*N) 이다. 개선 된 알고리즘은 N이 충분히 큰 경우에는 구간 관련 연산에 대한 시간을 줄여주지만, N이 충분히 크지 않다면, 구간 관련 연산 외에 다른 곳에서, 더 오래 걸리기 때문에, 개선 전 보다 성능이 더 떨어지는 모습을 보인다.

1. K = 100 고정 K 변화( 2 ~ 10,000,000 )

Y축 : 시간 (ms) , X 축 : K(구간 개수) , 파란색선 : 개선 전 , 주황색선 : 개선 후

1. 시간 비교 그래프



1. 시간 비교 테이블 (ms)

|  |  |  |
| --- | --- | --- |
| N (K = 100 고정) | 개선 전 | 개선 후 |
| 2 | 0 | 0 |
| 10 | 1 | 0 |
| 100 | 0 | 0 |
| 1,000 | 2 | 1 |
| 10,000 | 8 | 2 |
| 100,000 | 32 | 9 |
| 1,000,000 | 194 | 40 |
| 10,000,000 | 910 | 411 |

1. 분석

- 개선 전 알고리즘은 O(N) + 3\*O(K\*N), 개선 후 알고리즘은 3\*O(N) + 2\* O(K\*logN) + O(K) 의 시간 복잡도를 가진다. K가 고정 되는 경우에는 두 알고리즘은 모두 N에 비례하는 시간 복잡도를 갖게 된다. 이는 위 그래프에서 확인 가능 하다.

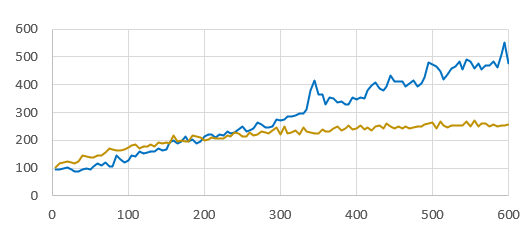
- A번과 다르게 K가 충분히 큰 경우에는 개선 한 알고리즘의 성능이 높게 평가된다. 그 이유로는 K가 커지면, 랜덤하게 생성되는 구간에 대한 각각의 연산(최대, 최소, 합계)의 시간이 영향을 받는다. 구간 관련 연산에서 개선 전에는 O(K\*N), 개선 후에는 O(K\*logN)으 시간 복잡도를 갖고, K에 대한 가중치는 개선 전의 알고리즘이 더 높다..

즉 K가 증가함에 따라 알고리즘 시간에 대한 영향이 개선 전알고리즘에서 더 크게 받기 때문에, K가 일정 값보다 크다면 개선된 알고리즘의 성능이 더 좋다.

* N과 K의 관계

K가 N에 비해 매우 큰 경우 K = 1,000,000 고정, N ( 2 ~ 600 )

Y축 : 시간 (ms) , X 축 : K(구간 개수) , 파란색선 : 개선 전 , 주황색선 : 개선 후



- 개선 전 알고리즘은 O(N) + 3\*O(K\*N), 개선 후 알고리즘은 3\*O(N) + 2\* O(K\*logN) + O(K) 의 시간 복잡도를 가진다.

위의 분석들의 모든 시간 그래프는 선형 그래프였다. logN의 시간 복잡도에 대한 효율을 알아보기 위해 구간의 개수에 대한 값이 K를 배열의 크기인 N보다 매우 크게 하여 O(K\*N)과 O(K\*logN)에 대한 비교를 진행 하였다.(이렇게 하면 나머지 O(N)은 영향력이 작고, O(K)는 K가 고정인 경우 상수 시간이 되기 때문에 무시할 수 있게 된다. )

그 결과로 개선 후 알고리즘은 O(logN)의 형태를 가지고, 개선 전 알고리즘은 O(N)의 형태가 나온다.

K = 1000000일때 N이 대략 200보다 작을 때는, 개선 전의 알고리즘의 효율이 더 높고, N이 200보다 커지는 경우에는 개선된 알고리즘의 성능이 더 높다는 것을 알 수 있다.

* 프로젝트 평가

알고리즘의 시간 복잡도는 전체적인 시간에 대한 그래프를 보여준다. 하지만 이는 어느 알고리즘이 더 좋고 나쁜지를 판별하는 기준이 아니라는 점을 알 수 있다.

시간 복잡도를 개선 하더라도, 변수들의 범위에 따라서 실제 측정 시간 성능의 평가는 달라질 수 있기 때문에 이 점을 고려해야 한다.

시간 복잡도를 측정하는 과정에서 시간을 직접 측정 하기 보다는, 내가 원하는 부분에 대해서 연산이 얼마나 이루어졌는가에 대한 기준으로 평가하는 방식도 좋을 것 같다. 구간과 배열 값을 랜덤으로 선정하기 때문에, 측정 시간 범위가 충분하지 않다면, 어떤 시간 복잡도를 가진지 판단하기 어렵다.(랜덤으로 생성되는 구간의 크기에 따라 시간이 달라지기 때문->최악, 최선 발생)