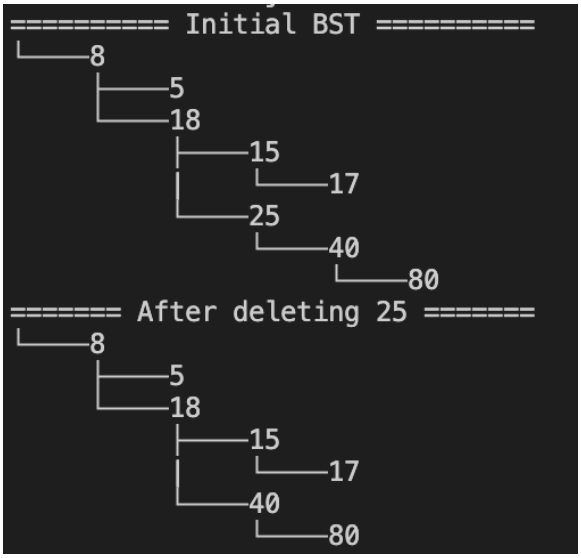
CFDS Homework 6

2022-24790 박성우 (cfds037)

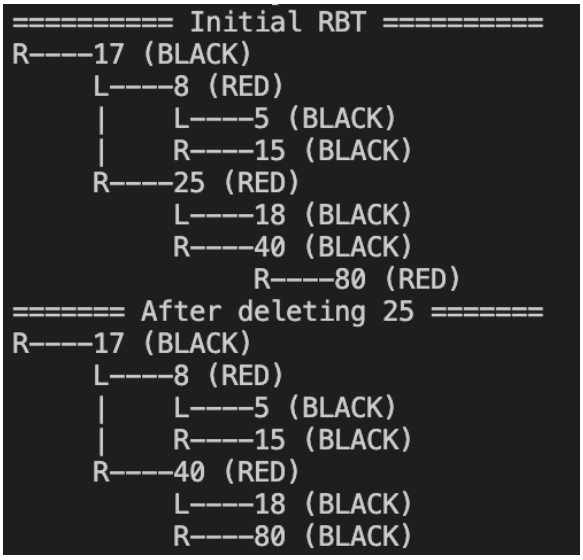
Problem 1.

- `bst` 디렉토리의 `bst.cpp` 파일을 실행한 결과는 아래와 같다.



Problem 2.

- `rbt` 디렉토리의 `rbt.cpp` 파일을 실행한 결과는 아래와 같다.



Problem 3.2

INSERT

- BST, RBT 각각에 대해서 100개와 10000개의 경우에 대해서 비교하였으며, 각각에 대해서 ascending sorted, unsorted, descending sorted의 경우에 대해서 비교하였다.
  - 100개의 경우 소요시간

단위: ms	BST	RBT
Ascending sorted	0.124	0.054
Unsorted	0.049	0.062
Descending sorted	0.160	0.052

  - 10000개의 경우 소요시간

단위: ms	BST	RBT
Ascending sorted	878.853	3.087
Unsorted	3.278	3.643
Descending sorted	912.511	2.434

  - 이를 통해 볼 때 특히 sorted된 경우에서 BST와 RBT의 성능 차이가 큰 것을 확인할 수 있다. 이는 BST는 sorted되지 않은 경우에 대해서 평균적으로 $O\left(\log N\right)$의 시간복잡도를 갖지만 최악의 경우 $O\left(N\right)$의 시간복잡도를 갖는 반면, RBT는 모든 경우에서 $O\left(\log N\right)$의 시간복잡도를 갖기 때문인 것으로 파악할 수 있다.

ERASE

- Insert와 마찬가지로 BST, RBT 각각에 대해서 100개와 10000개의 경우에 대해서 비교하였으며, 각각에 대해서 ascending sorted, unsorted, descending sorted의 경우에 대해서 비교하였다.
  - 100개의 경우 소요시간

단위: ms	BST	RBT
Ascending sorted	0.009	0.036
Unsorted	0.021	0.037
Descending sorted	0.197	0.068

  - 10000개의 경우 소요시간

단위: ms	BST	RBT
Ascending sorted	0.199	3.245
Unsorted	1.155	2.670
Descending sorted	1462.131	2.803

  - Ascending sorted와 descending sorted에서 성능 차이가 심하게 나는 이유는 Erase를 1부터 100(또는 10000)까지 순차적으로 진행했기 때문인 것으로 생각된다.
  - 그러나 역시 BST의 경우 unsorted의 경우와 sorted의 경우에서 큰 차이가 나는 반면 RBT에서는 어떤 경우든 관계 없이 안정적인 속도를 보이는 것을 알 수 있다.

FIND

- 다른 경우들과 마찬가지로 BST, RBT 각각에 대해서 100개와 10000개의 경우에 대해서 비교하였으며, 각각에 대해서 ascending sorted, unsorted, descending sorted의 경우에 대해서 비교하였다.
  - 100개의 경우 소요시간

단위: ms	BST	RBT
Ascending sorted	0.055	0.009
Unsorted	0.021	0.009
Descending sorted	0.055	0.010

  - 10000개의 경우 소요시간

단위: ms	BST	RBT
Ascending sorted	778.242	1.052
Unsorted	1.324	1.126
Descending sorted	757.421	1.042

  - Find의 경우 Insert의 경우와 비슷한 성능 분포를 보이는 것으로 확인되었다. 다만 insert와는 달리 삽입 및 수정 과정이 필요하지 않아 insert보다는 전반적으로 빠른 속도를 보였다는 것을 확인할 수 있다.
  - 또한, BST의 경우 unsorted의 경우와 sorted의 경우에서 각각 $O\left(\log N\right)$과 $O\left(N\right)$으로 큰 차이가 나는 반면 RBT에서는 어떤 경우든 관계 없이 안정적인 속도를 보이는 것을 알 수 있다.

SIZE

- 다른 경우들과 마찬가지로 BST, RBT 각각에 대해서 100개와 10000개의 경우에 대해서 비교하였으며, 각각에 대해서 ascending sorted, unsorted, descending sorted의 경우에 대해서 비교하였다.
  - 100개의 경우 소요시간

단위: ms	BST	RBT
Ascending sorted	0.000	0.000
Unsorted	0.000	0.001
Descending sorted	0.000	0.000

  - 10000개의 경우 소요시간

단위: ms	BST	RBT
Ascending sorted	0.000	0.000
Unsorted	0.000	0.000
Descending sorted	0.000	0.000

  - SIZE에서는 모든 경우에 대해서 큰 차이가 나지 않았을 뿐 아니라, 걸리는 시간이 거의 0초에 가까웠다. 이는 코드 구현상 insert를 수행할 때 size를 추가하고, erase를 수행할 때 size를 줄이는 방식으로 구현했기 때문인 것으로 생각된다.