자료구조 프로젝트

메모리 할당자 시뮬레이터 설계 기술서

20231765 박종승

1. 전체 자료구조

메모리 할당자 시뮬레이터를 제작하기 위해선 수많은 메모리 삽입과 탐색을 가장 우선으로 고려해야 한다. 성능 평가 요소 중 실제 시뮬레이터 작동 시간이 중요한 요소이므로, 여러 자료구조 중 시간 복잡도에서 유리한 트리 구조를 선택하였다. 여러 트리 구조 중 AVL 트리와 RB 트리를 비교해본 결과, 데이터의 분포를 알 수 없는 상황에서 최악의 경우 AVL 트리는 균형 유지를 위한 많은 회전 연산에 많은 시간이 소모될 것이라고 예상하였다. 반면, RB 트리의 경우 적은 회전으로도 균형을 유지할 수 있으며, 색상 정보만 저장하면 되기에 수많은 데이터들을 간단한 연산(할당, 제거)으로 처리하기에 더 유익하다고 판단하여 RB 트리를 선택했다.

2. 전체 알고리즘

메모리 할당 시 순서대로 할당하는 것이 아닌, 현재 빈 공간들 중 할당할 메모리의 크기와 가장 유사한 공간에 할당하는 'best fit' 방식을 통해 낭비되는 메모리의 양을 줄인다. 이를 위해 먼저 자료구조 클래스인 RB 트리의 클래스를 상단부에 구현한다. 그 후 메모리 할당 및 해제를 관리하는 클래스인 Allocator 클래스를 구현해 준다. 메모리 블록은 청크 단위로 관리하며, 할당된 청크는 'used_chunks' 딕셔너리에 저장되고, 자유 청크는 레드-블랙 트리의 'free_chunks' 에 저장한다.

Allocator 클래스가 초기화될 때 기본 청크 크기(16KB)와 기본 변수들을 설정해 준다. 그 후 메모리를 할당받아 오는데 입력 파일을 열고 요청에 따라 할당 혹은 해제를 처리한다. 할당 처리가 필요할 경우 우선 RB 트리 속 get_best_fit 메서드를 호출하여 적절한 크기의 자유 블록을 찾아준다. 만약 적절한 블록이 존재하지 않으면 OS에서 새로운 청크를 요청한다. 이 과정을 통해 블록을 할당하고 남은 블록은 다시 자유 청크 트리에 삽입해 준다. 해제가 필요한 경우 (used_chunks) 딕셔너리에서 해당 블록의 시작 위치와 크기를 찾아온다. 찾은 블록을 해제한 후 메모리의 효율적인 사용을 위해서 앞뒤 인접한 자유 블록과 병합 과정을 거친다. 이 과정에서 RB 트리는 새로운 노드를 삽입하고 제거하며 트리 균형을 위해 회전 연산을 진행한다.

3.시간복잡도 분석

우선 bestfit을 찾기 위한 RB 트리에서 진행되는 삽입 연산과 삭제 연산에서는 노드의 위치를 찾기 위한 시간 O(log n), 노드를 삽입 혹은 삭제 후 트리의 균형 유지를 위한 회전 작업 및 색상 변경으로 O(log n)의 시간이 소모되어 전체 O(log n)의 시간이 소모된다. 그리고 bestfit 블록을 찾기 위해 트리의 높이만큼 탐색이 진행되므로 탐색 연산의 시간 복잡도 또한 O(log n)이다.

다음으로 Allocator 클래스에선 메모리 할당의 경우 트리에서 'get_best_fit' 메서드를 탐색할 경우 O(log n) 시간이 소요되며 이후 OS에서 새로운 청크를 받는다면 이 땐 작업시간은 상수 시간 O(1)이 소요된다. 따라서 삽입의 전체 시간 복잡도는 O(log n)이다. 해제 또한 동일한 과정으로 O(log n)이란 시간 복잡도를 가진다.

따라서 전체 시간 복잡도를 계산해보면 각 요청당 O(log n)의 시간 복잡도를 가지므로, 모든 요청에 대한 시간 복잡도는 m * O(log n) 즉 **O(m log n)** 이란 시간 복잡도를 가진다. (m은 입력 파일의 라인 수 (메모리 할당 및 해제 요청의 수) , n은 레드-블랙 트리에 저장된 자유 청크의 수이다.)