**문제 1. 최소 최대값 동시 찾기 문제**

1. Minimum() 함수와 Maximum() 함수를 활용한 최소 최대값 찾기 (출력)

텍스트, 폰트, 스크린샷, 블랙이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

수업시간에 배운 Minimum 함수 알고리즘에서 부호를 반대 방향으로 수정해 Maximum 함수를 작성했습니다.

먼저, 배열의 크기를 1000으로 define한 후, srand(time(NULL)을 통해 시드를 초기화한 후, 반복문을 돌려 1~100000 사이의 숫자 중 랜덤으로 1000개를 생성해 A[SIZE] 배열에 담았습니다. (여기서 rand() % 100000 + 1; 라고 표현한 이유는 단순 rand() % 100000을 하면 0부터 값이 나올 수 있기 때문에 이를 방지하고자 +1하여 1부터 숫자가 나올 수 있도록 설정했습니다.)

Minimum함수와 Maximum함수 호출을 통해 최소 최대 값을 각각 구했는데, 기존 Minimum함수가 배열의 0번째 index 값(A[0])을 일단 가장 작은 값이라 가정하고 비교횟수인 1~n-1까지 A[i]값과 min값을 비교해 만약 A[i]값이 min보다 더 작은 경우 A[i]값으로 min값을 업데이트했습니다. 이와 같은 로직으로, Maximum함수 또한 A[0]값을 임의로 max값으로 설정한 후, 반복문을 통해 A[k]값과 max값을 비교해, max보다 A[k]값이 더 큰 경우에 A[k]값으로 max값이 업데이트 되도록 설정했습니다. 각 함수마다 min과 max를 각각 반환돼 이를 출력하고 마무리했습니다.

1. **텍스트, 폰트, 스크린샷, 블랙이(가) 표시된 사진

   AI가 생성한 콘텐츠는 부정확할 수 있습니다.**FindMinMax() 함수를 활용한 최소 최댓값 찾기 (출력)

FindMinMax()함수의 경우에는 배열에서 두 요소를 함께 확인해 최대 최소를 한 함수 내에서 함께 찾을 수 있게끔 설계했습니다. 랜덤 시드를 초기화하고, 1~100000범위내에서 1000개의 랜덤값을 생성한 것은 위와 같지만, FindMinMax내부 로직은 다릅니다. 일단 FindMinMax내부에서 min과 max변수를 활용하기 위해 메인함수에서 선언해준 뒤, FindMinMax함수 내에서 Minimum과 Maximum값에 쓰레기값이 들어가지 않도록 하기 위해 A[0]으로 설정해주었습니다. 1부터 n-1까지 i번째 값과 i+1번째 값을 비교해 둘 중 더 작은 값은 Smaller에, 더 큰 값은 Larger에 저장한 후, 다음 조건문에서 smaller와 minimum을, 또한 maximum과 larger값을 비교합니다. 기존 최솟값보다 저장된 Smaller값이 더 작은 경우 이를 Minimum값으로 업데이트하고, 기존 최댓값보다 Larger값이 더 큰 경우 이를 최댓값으로 업데이트해줍니다. i번째와 i+1번째를 비교하기 때문에 i+=2 마다 반복될 수 있도록 설정해주었는데, 만약 n이 홀수 개인 경우에는 마지막 값도 비교하기 위해 A[n-1]과 minimum, A[n-1]과 Maximum을 비교한 후, A[n-1]이 minimum보다 더 작거나, Maximum보다 더 큰 경우 값을 업데이트해줍니다. 이후 이를 최소값과 최대값으로 출력했습니다.

**문제 2. 다단계 합병정렬 (테이블 완성 및 총 블록 개수 구하기)**

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

**문제 3. 알고리즘의 점근적 수행시간 구하기 ( 빅세타 활용 )**

1. 수행시간 T(n)을 점화식으로 표현

텍스트, 폰트, 스크린샷, 친필이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

1. 답을 구하는 과정 풀이

텍스트, 친필, 폰트, 잉크이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

**문제 4. 이진탐색트리 그리기 (T1 생성, T6 삭제한 후 모습)**

1. **트리 T1 생성하는 과정 그리기**

그림, 스케치, 패턴이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

루트 노드를 먼저 넣은 후, 현재 노드의 값과 새로 삽입하려는 값을 비교해, 만약 삽입하려는 값이 더 작은 경우 왼쪽 서브 트리로, 더 큰 경우 오른쪽 서브 트리에 값을 넣도록 하였습니다. NULL위치를 찾을 때까지 (자식노드가 더 이상 없는 경우가 나올 때까지) 위와 같은 방법으로 계속 이동해 그 값을 삽입할 수 있도록 그렸습니다. (트리구조로 데이터를 빠르게 탐색할 수 있도록 하는 이진탐색 트리의 기본 규칙을 생각했을 때, 왼쪽 서브트리에는 현재 노드보다 작은 값들이, 오른쪽 서브트리에는 현재 노드보다 큰 값들이 있어야 하기 때문에 이와 같이 그렸습니다.)

1. 그림, 스케치, 패턴이(가) 표시된 사진

   AI가 생성한 콘텐츠는 부정확할 수 있습니다.**생성된 T1에서 6을 삭제한 후의 모습 그리기**

6을 삭제할 때 오른쪽 서브 트리에서 가장 작은 값인 8과 교체해주었습니다. 삭제하더라도, 이진탐색트리의 구조(왼쪽 서브트리에는 자신보다 작은 값, 오른쪽 서브트리에는 자신보다 큰 값)를 유지해주어야 하므로 단순히 6을 삭제해주지 않고, 6을 삭제한 후 오른쪽 서브트리에서 가장 작은 값인 8과 교체해주었습니다. 그 후, 8의 자식 노드인 9를 8의 부모노드인 10에 연결해 (b)와 같이 이진탐색트리 구조를 유지한 채 결과가 나오도록 그렸습니다.