Algorithm Homework 2B

학번 / 이름: 2014310407 / 이 준 혁

1. 구현

가장 먼저 tree 의 자식의 수(즉, 자식 노드들이 몇 개 있는지)를 입력 받은 후에, getchar 함수를 통해, 입력이 '₩n'이 아닐 때까지 입력을 받았다. 그리고 입력을 1.)공백인 경우 와 2) 숫자인 경우로 구분했다. 나머지 경우에는 오류를 출력하고 프로그램을 종료 시켰다.

우선, heap 의 역할을 할 배열을 MAX_LEN 만큼의 크기로 할당했다. 공백 때문에 최대 100 개의 문자가들어갈 수 있고, 또 heap 의 index 는 1 부터 시작하는 것이 계산하기 편해서 MAX_LEN 을 101 로설정했다. 또한 heap 의 크기를 나타내는 변수도 지정했다.

우선 임시로 숫자를 저장할 변수를 0으로 초기화 한 후, 입력으로 숫자를 받은 경우에는 기존에 저장되어 있던 값을 10 배 한 후에 숫자를 더했다. 즉 12를 입력 받으면, 처음에는 0이 저장되어 있으므로 0*10+1의 값인 1을 저장시킨다. 그 후에 2를 받으면, 1*10+2의 값인 12를 저장시킨다. 이를 통해 십진수를 잘 저장될 수 있도록 했다. 그리고 이 과정에서 입력 값이 숫자인지 판별하는 is_num 함수와 문자의 아스키코드 값을 숫자 값으로 바꿔주는 to_num 함수를 사용했다.

공백인 경우에는, 임시로 저장되어 있던 정수 값을 heap 에 넣어 주고, 다시 정수 입력을 받기 위해 임시정수 값을 초기화 시켰다.

이렇게 입력을 다 받은 후에는, 임시로 저장된 정수 값을 heap 에 마지막으로 넣어주었다. 이렇게 값을 다 저장한 후에는 heap 으로 만드는 과정이 필요했기에, Min_Heapify 함수를 구현했다. 이 함수는 heap 과 Min_Heapify 를 진행할 노드의 index, heap 의 크기와 tree 의 자식의 수를 입력으로 받았다.

그리고 입력으로 받은 노드와 그 자식들의 값을 비교해, 가장 작은 값은 찾는다. 만약 가장 작은 값이 node 라면, 아무것도 하지 않고, 아니라면 가장 작은 자식과 node 를 바꾼 후에, 바꾼 노드에 대해 다시 Min_Heapifiy 함수를 호출하는 식으로 함수를 구성했다.

이 과정은 노드의 왼쪽과 오른쪽 트리가 각각 min heap 일때만 사용이 가능하고, 이 과정을 거치고 나면 min heap 이 되므로, 이 과정을 가장 큰 index 를 가진 노드의 부모부터 root 까지 반복 시행했다.

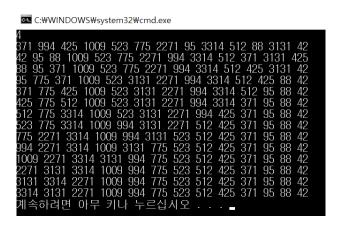
가장 큰 index 를 가진 노드의 부모부터 진행한 이유는, 이보다 큰 index 를 가지면 어차피 자식 노드가 없기 때문에, Min_Heapify 를 진행할 필요가 없었기 때문이다. 또 이렇게 root 노드까지 Min_Heapify 를 진행하면 전체 tree 가 min heap 이 된다.

이 과정에서, X_i _child 라는 함수와 X_i _parent 라는 함수를 구현했는데, 각각 tree 의 자식의 수가 X_i 일 때 i 번째 자식의 index 를 반환하는 함수와, 부모의 index 를 반환하는 함수이다.

이 과정을 거친 후 heap 을 출력했고, heapsort 를 진행했다. 가장 위에 있는 값을(가장 작은 값) heap 의가장 마지막에 있는 값과 위치를 바꾼 후, heap 의 크기를 하나 줄였다. 즉, 정렬된 값들은 heap 에서 제외되게 만든 것이다. 또한 이렇게 값을 바꾸면, 맨 위의 값을 제외한 나머지 tree 들은 다 min heap 이므로, root 노드에 대해서만 Min_Heapify 를 진행하면 다시 min heap 을 얻을 수 있었다. 이과정을 heap 의 원소가 1 개 남을 때까지 반복하고, 각 과정마다 출력했다.

이렇게 하면 매 과정마다 맨 뒤에서부터 정렬된 값들을 얻을 수 있고, 최종적으로는 내림차순으로 정렬된 값들을 얻었다.

2. 결과



내림차순으로 잘 출력됨을 확인했다.