**Homework 2B report**

**2014313303 홍태하(개발환경 : Linux)**

1. **알고리즘**

인풋을 받아서 X-nary tree의 형태를 파악하고 배열에 각 노드의 값들을 저장한다. 그 후 힙소트를 이용하여 정렬을 하는데 먼저 마지막 리프 노드의 부모 노드에서부터 루트노드까지 Min\_Heapify함수를 적용시킨다. Min\_Heapify는 해당 노드의 자식 노드들과 값을 비교하여 가장 작은 값을 찾아 해당 노드와 자리를 바꾸어 주는데 리프 노드에 가거나 바꿀 필요가 없을 때까지 진행한다. 그렇게하면 minimum heap이 만들어진다.

Minimum heap을 만들었으면 루트 노드와 마지막 리프 노드와 자리를 바꾸고 heap size를 하나 줄여서 다시 Min\_Heapify를 한다. 이 과정을 끝까지 진행하면 내림차순으로 정렬된다.

1. **코드 설명.**
2. **인풋 받기.**

먼저 X라는 이름의 변수로 트리의 형태를 정하는 인풋을 받는다(X-nary tree). 그 후 input\_c를 이용해 하나씩 인풋을 받는데 input\_i라는 변수에 int 형으로 바꾸어 저장한다. 인풋을 앞에서부터 하나씩 받기 때문에 ‘ ‘이 들어오기 전까지 input\_i에 10을 곱하면서 자리수를 늘려가면서 받는다. ‘ ‘이 들어오면 하나의 값이 다 들어온 것이므로 tree라는 배열에 input\_i를 저장한다. 그 과정을 반복하다가 ‘\n’이 들어오면 인풋 받는 것을 종료한다.

1. **Print\_Tree 함수**

Tree 노드의 값들을 전부 출력해주는 함수이다.

1. **Parent 함수**

X-nary tree에서 어떤 노드의 부모 노드의 인덱스를 return하는 함수이다.

1. **Make\_Child\_Index 함수**

X-nary tree에서 어떤 노드의 자식 노드의 인덱스를 child\_arr라는 배열에 저장해주는 함수이다.

1. **Min\_Heapify 함수**

어떤 노드에 대해서 그 노드의 자식 노드와 자신을 비교하여 자식 노드에 가장 작은 값이 있으면 그 노드와 자신의 위치를 바꿔준다. 이 과정을 자신이 가장 작은 값이 되거나 리프 노드까지 반복한다.

이 코드에서는 Make\_Child\_Index함수를 통해서 child\_arr배열에 자식노드들의 인덱스를 저장하고 자신과 자식 노드들을 비교하여 가장 작은 값을 가진 인덱스를 smallest라는 변수에 저장하였다. 그리고 smallest가 자신의 인덱스가 아니면 자신과 smallest의 값을 바꿔주고 Min\_Heapify를 다시 실행하여 이 과정을 반복한다.

1. **Build\_Min\_Heap 함수**

마지막 리프 노드의 부모노드에서부터 Min\_Heapify를 실행시켜 minimum heap을 만드는 함수이다. 어차피 리프 노드에서는 Min\_Heapify를 해도 의미가 없으므로 마지막 리프 노드의 부모노드에서부터 실행한다.

1. **HeapSort 함수**

먼저 Build\_Min\_Heap함수를 실행시켜 minimum heap을 만들어 놓은 후 루트 노드와 마지막 리프 노드의 자리를 바꾼다. 그 후 heap size를 하나 줄이고 Min\_Heapify를 루트노드에서 실행시키면 heap size가 하나 줄어든 minimum heap이 다시 생성된다. Minimum heap에서 루트 노드가 전체 노드의 값 중 가장 작은 값이기 때문에 이 과정을 반복하면 내림차순의 정렬된 배열이 생성된다.

1. **예시**

**4**

**12 3 5 15 1 4 10 20**

인풋이 이렇게 들어왔다고 하면 X에 4가 저장되고 tree배열에 12 3 5 15 1 4 10 20이 저장된다. Index는 7이 된다. X가 4이므로 child배열은 4칸짜리 배열이 된다. HeapSort를 하면 Build\_Min\_Heap을 실행하고 minimum tree가 생성되는데 마지막 리프 노드인 20의 부모노드는 3이다. 3부터 Min\_Heapify를 하면 child\_arr배열에 3의 자식 노드인 4 10 20의 인덱스인 5 6 7이 차례대로 저장된다. 이제 3과 자식 노드들의 값을 비교하는데 3이 제일 작다. 따라서 아무것도 바뀌지 않고 함수가 끝나게 된다. 다시 Build\_Min\_Heap으로 돌아와서 12에서 Min\_Heapify를 하면 12의 자식 노드의 인덱스인 1 2 3 4가 child\_arr배열에 저장되고 그 값들과 비교하면 인덱스가 4인 1이 제일 작다. 따라서 12와 1을 바꾸고 바꾼 자리에서 다시 Min\_Heapify를 한다. 하지만 바꾼 자리가 리프 노드였기 때문에 함수가 종료된다. 이렇게 minimum heap이 만들어지고 HeapSort로 돌아가서 루트 노드와 마지막 리프 노드의 값을 바꾸고 heap size를 줄이면서 Min\_Heapify를 하면서 매번 출력을 해주면 아웃풋이 이렇게 된다.

**1 3 5 15 12 4 10 20**

**3 4 5 15 12 20 10 1**

**4 10 5 15 12 20 3 1**

**5 10 20 15 12 4 3 1**

**10 12 20 15 5 4 3 1**

**12 15 20 10 5 4 3 1**

**15 20 12 10 5 4 3 1**

**20 15 12 10 5 4 3 1**