**Homework 4C report**

**2014313303 홍태하(개발환경 : Linux)**

1. **알고리즘**

Prim’s algorithm은 Kruskal’s algorithm과 같이 minimum spanning tree를 찾는 대표적인 알고리즘이다. Minimum spanning tree란 어떤 그래프의 vertex가 모두 연결되면서도 그들의 weight의 합이 최소인 tree를 말한다.

이번 과제에서는 각 vertex에 key값을 주어 그 vertex와 이어진 weight중에 가장 작은 것들을 저장하고 그 값들을 기준으로 priority queue(heap을 사용.)를 이용하여 key값이 가장 작은 vertex를 쉽게 뽑아 내었다. key값이 가장 작은 vertex u를 뽑아낸 후 u와 이어진 vertex v를 찾아서 현재 key값이 u와의 weight보다 크다면 weight로 key값을 갱신해주고 heap을 재구성해준다. 이 과정을 heap이 비워질 때까지 반복하면 minimum spanning tree가 구성된다. 이번 과제의 result는 MST의 weight들의 합인데 heap에서 key값이 가장 작은 vertex가 뽑힐 때 그 vertex의 key값들을 더해주면 된다.

1. **코드 설명.**
2. **minHeapify 함수.**

minHeapify 함수는 minimum heap을 만들어주는 함수이다. parameter로는 heap포인터 M과 Index가 들어오는데 Index위치에서 heapify를 진행하라는 의미이다.

Heap은 position함수에 이진 트리의 형태로 구성되어 있다고 생각하였다. 또 position함수가 저장하고 있는 값은 vertex의 이름이다. 이진 트리이므로 Index의 왼쪽 자식 노드의 index는 index\*2+1이 되고 오른쪽 자식 노드의 index는 index\*2+2가 된다. 자식 노드들은 먼저 heap안에 존재하는지 체크한 후 key값이 가장 작은 것을 찾는다. Index와 두 자식 노드 3개를 비교하여 가장 작은 key값을 가진 vertex의 이름을 smallest에 저장하고 smallest가 Index가 아니면 smallest와 Index의 위치를 바꿔준 후 smallest의 위치에서 다시 minHeapify를 진행한다.

1. **extractMin 함수.**

extractMin 함수는 heap에서 root값을 뽑아 return해주는 함수이다. Root는 position[0]에 있으므로 position[0]의 값을 vertex name으로 가지는 vertex를 리턴해주면 된다. Root가 뽑혔으므로 heap을 재구성해주어야 한다. Heap의 마지막 자리를 root와 바꿔주고 root의 자리에서 minHeapify를 실행한다. Heap에 존재하는 vertex 수를 의미하는 remainVertexNumber는 하나 줄여준다.

1. **decreaseKey 함수.**

decreaseKey 함수는 parameter로 heap 포인터 M과 V, Key가 들어오는데 vertex V의 key값을 Key로 바꿔주는 일을 한다.

1. **addVertexToHeap 함수.**

addVertexToHeap 함수는 heap이 생성되었을 때 그 값들을 채워주는 함수이다. vertexList배열을 할당하고 Graph의 vertexList에 저장되어 있는 vertex들을 가리키게 한다. 따라서 Graph의 vertexList와 MinHeap의 vertexList는 같은 vertex를 가리키고 있게 된다. 또 position함수에는 vertex name이 작은 것부터 순서대로 채워준다. position함수는 나중에 minHeapify를 이용하여 minimum heap으로 만들 것이다. 마지막으로 remainVertexNumber를 증가시켜준다.

1. **primAlgorithm 함수.**

primAlgorithm 함수는 Prim’s Algorithm을 구현하는 함수이다. 먼저 heap을 생성하고 addVertexToHeap을 이용하여 heap structure의 값들을 채워준다. 처음 시작하는 vertex의 key값은 decreaseKey를 이용하여 0으로 바꿔준다. 0번째 vertex가 position의 루트 자리에 있으므로 minHeapify를 할 필요는 없다.

이제 Heap이 empty가 될 때까지 while문을 돌면서 다음 작업을 수행한다. 먼저 extractMin을 이용해서 가장 작은 key값을 가지고 있는 vertex를 뽑아 u에 저장한다. u의 key값들을 모두 더한 것이 result가 된다. 그 후에는 u와 연결된 vertex v를 찾고 v가 heap안에 있는지 본 후 v의 key값보다 u와 v 사이의 weight가 더 작으면 v의 key값을 weight로 바꿔준다. U와 연결된 모든 v에 같은 작업을 반복한 후 key값들이 바뀌었기 때문에 heap을 minHeapify를 이용해 다시 구성한다. Heap이 empty가 될 때까지 반복하면 MST의 모든 weight들의 합이 구해진다.

1. **예시**

**5 5**

**0 1 2**

**0 3 10**

**0 4 4**

**1 4 7**

**2 4 1**

의 인풋이 들어왔다고 하자. 먼저 0부터 4까지의 vertex가 Graph와 Heap에 저장된다. minHeap의배열 position에는 0부터 4까지 순서대로 들어간다. 모두의 key값은 INT\_MAX인 상태이다. Vertex 0에서 시작할 것이기 때문에 vertex 0의 key값을 0으로 만들어준다.

while문에 들어가면 u로 vertex 0이 뽑히고 result에는 0이 더해진다. 그 후 u와 연결된 vertex 1, 3, 4를 찾고 각각의 key값은 2, 10, 4로 바뀐다. 그 후 heap을 다시 구성하면 2의 key값을 가지고 있는 vertex 1이 root자리에 오게 된다. while문을 다시 돌면 u로 vertex 1이 뽑히게 되고 이전과 같은 작업을 반복한다. Heap이 empty가 될 때까지 반복하면 result는 17이 된다.