EE360C Algorithms
University of Texas at Austin
Prof. Nur Touba
Student Name: Yonatan Teshome
Student EID: YH23572
Date: July 30, 2025

Programming Assignment #2

Report Section: UT Network

a) Minimum Cost to Connect to UT (Dijkstra)

1. **Function findMinimumStudentCost (start, dest):**
   a. **For each student s in students:**
      i. **S.minCost = infinity**
   b. **Start.minCost = 0**
   c. **H = buildHeap(students)**
   d. **While H is not empty:**
      i. **If current == dest:**
         1. **Return current.minCost**
      ii. **For each neighbor v of current with edge cost c:**
         1. **If current.minCost + c < v.mincost:**
            a. **V.minCost = current.minCost + c**
            b. **changeKey(H, v, v.minCost)**
   e. **return -1 since destination is unreachable**

RunTime: V = students E = wires O((V+E) log V)

b) Minimum Cost to Connect Entire Class to UT (Prim's)

2. **Function findMinimumClassCost():**
   a. **For each student s in students:**
      i. **S.minCost = infinity**
   b. **start = UT looking for the last student in the list**
   c. **start.minCost = 0**
   d. **visited = array[v] intiailized to false**
   e. **totalcost = 0**
   f. **heap = buildHeap(students)**
   g. **visitedCount = 0**
   h. **while heap is not empty:**
      i. **current = extractMin(heap)**
      ii. **if visited[current.name]: continue**
      iii. **visited[current.name] = true**
      iv. **visitedCount += 1**
      v. **total Cost += current.minCost**
      vi. **for each neighbor v of current with edge cost c:**
         1. **if not visited[v.name] and c < v.minCost:**
            a. **v.minCost = c**
            b. **changeKey(heap, v, c)**
   i. **if visitedCount < v:**
      i. **return -1 disconnected graph**
   j. **return total cost**

RunTime: V = students E = wires O((V+E) log V)

EE360C Algorithms
University of Texas at Austin
Prof. Nur Touba
Student Name: Yonatan Teshome
Student EID: YH23572
Date: July 30, 2025

Programming Assignment #2

c) Notes on Heap Usage

1. **buildHeap = O(V)**
2. **extractMin = O(log V)**
3. **changeKey = O(log V)**

- **Dijkstra finds the minCost to connect to UT**
- **Prim's MST finds the min total cost to connect the entire class**
- **Both algorithms rely on a minheap to effectively select the next cheapest student. Tie breakers are done using the student ID in minCost is equal**
- **Both run in O((V+E) log V)**