

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ**

**ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**(ДГТУ)**

Факультет Информатика и вычислительная техника

Кафедра Кибербезопасность информационных систем

**ПРАКТИЧЕСКАЯ РАБОТА № 4**

на тему «Алгоритмы построения остова минимальной стоимости»

Выполнила обучающаяся гр. ВКБ32

Попова Алина Петровна

Проверил

Савельев Василий Александрович

Ростов-на-Дону

2022

Найти объем четырехмерного конуса, ограниченного поверхностями

|  |
| --- |
| class Graph():  INF = 999999  def \_\_init\_\_(self, num\_vertices):  self.V = num\_vertices  self.graph = [[0 for column in range(num\_vertices)] for row in range(num\_vertices)]    # pretty print of the minimum spanning tree  # prints the MST stored in the list var `parent`  def printMST(self, parent):  print("Edge Weight")  for i in range(1, self.V):  print(f"{parent[i]} - {i} {self.graph[i][parent[i]]}")    # finds the vertex with the minimum distance value  # takes a key and the current set of nodes in the MST  def minKey(self, key, mstSet):  min = self.INF  for v in range(self.V):  if key[v] < min and mstSet[v] == False:  min = key[v]  min\_index = v  return min\_index    # prim's algo, graph is represented as an v by v adjacency matrix  def prims(self):  # used to pick minimum weight edge  key = [self.INF for \_ in range(self.V)]  # used to store MST  parent = [None for \_ in range(self.V)]  # pick a random vertex, ie 0  key[0] = 0  # create list for t/f if a node is connected to the MST  mstSet = [False for \_ in range(self.V)]  # set the first node to the root (ie have a parent of -1)  parent[0] = -1    for \_ in range(self.V):  # 1) pick the minimum distance vertex from the current key  # from the set of points not yet in the MST  u = self.minKey(key, mstSet)  # 2) add the new vertex to the MST  mstSet[u] = True    # loop through the vertices to update the ones that are still  # not in the MST  for v in range(self.V):  # if the edge from the newly added vertex (v) exists,  # the vertex hasn't been added to the MST, and  # the new vertex's distance to the graph is greater than the distance  # stored in the initial graph, update the "key" value to the  # distance initially given and update the parent of  # of the vertex (v) to the newly added vertex (u)  if self.graph[u][v] > 0 and mstSet[v] == False and key[v] > self.graph[u][v]:  key[v] = self.graph[u][v]  parent[v] = u    self.printMST(parent)    g = Graph(5)  g.graph = [ [0, 2, 0, 6, 0],  [2, 0, 3, 8, 5],  [0, 3, 0, 0, 7],  [6, 8, 0, 0, 9],  [0, 5, 7, 9, 0]]    g.prims() |