|  |  |
| --- | --- |
| **Name** | Manish Shashikant Jadhav |
| **UID** | 2023301005 |
| **Subject** | Design and Analysis of Algorithms (DAA) |
| **Experiment No.** | 6 |
| **Aim** | To implement Greedy Approach ( Prim's Algorithm and Dijkstra's Algorithm) |
| **Code:** | #include <stdio.h>  #include <stdlib.h>  #include <limits.h>  void primsAlgorithm(int \*\**graph*, int *vertices*) {      int parent[*vertices*];      int key[*vertices*];      int mstSet[*vertices*];      for (int i = 0; i < *vertices*; i++) {          key[i] = INT\_MAX;          mstSet[i] = 0;      }      key[0] = 0;      parent[0] = -1;      for (int count = 0; count < *vertices* - 1; count++) {          int minKey = INT\_MAX, minIndex;          for (int v = 0; v < *vertices*; v++) {              if (mstSet[v] == 0 && key[v] < minKey) {                  minKey = key[v];                  minIndex = v;              }          }          mstSet[minIndex] = 1;          for (int v = 0; v < *vertices*; v++) {              if (*graph*[minIndex][v] && mstSet[v] == 0 && *graph*[minIndex][v] < key[v]) {                  parent[v] = minIndex;                  key[v] = *graph*[minIndex][v];              }          }      }      printf("Edge \tWeight\n");      for (int i = 1; i < *vertices*; i++) {          printf("%d - %d \t%d \n", parent[i], i, *graph*[i][parent[i]]);      }  }  void dijkstraAlgorithm(int \*\**graph*, int *vertices*, int *src*) {      int dist[*vertices*];      int visited[*vertices*];      for (int i = 0; i < *vertices*; i++) {          dist[i] = INT\_MAX;          visited[i] = 0;      }      dist[*src*] = 0;      for (int count = 0; count < *vertices* - 1; count++) {          int minDist = INT\_MAX, minIndex;          for (int v = 0; v < *vertices*; v++) {              if (!visited[v] && dist[v] <= minDist) {                  minDist = dist[v];                  minIndex = v;              }          }          visited[minIndex] = 1;          for (int v = 0; v < *vertices*; v++) {              if (!visited[v] && *graph*[minIndex][v] && dist[minIndex] != INT\_MAX &&                  dist[minIndex] + *graph*[minIndex][v] < dist[v]) {                  dist[v] = dist[minIndex] + *graph*[minIndex][v];              }          }      }      printf("Vertex \tDistance from Source\n");      for (int i = 0; i < *vertices*; i++) {          printf("%d \t%d\n", i, dist[i]);      }  }  int main() {      int choice, vertices, src;        printf("Enter the number of vertices: ");      scanf("%d", &vertices);      int \*\*graph = (int \*\*)malloc(vertices \* sizeof(int \*));      for (int i = 0; i < vertices; i++) {          graph[i] = (int \*)malloc(vertices \* sizeof(int));      }      printf("Enter the adjacency matrix:\n");      for (int i = 0; i < vertices; i++) {          for (int j = 0; j < vertices; j++) {              scanf("%d", &graph[i][j]);          }      }      do {          printf("\nChoose an option:\n");          printf("1. Prim's Algorithm\n");          printf("2. Dijkstra's Algorithm\n");          printf("3. Exit\n");          scanf("%d", &choice);          switch (choice) {              case 1:                  primsAlgorithm(graph, vertices);                  break;              case 2:                  printf("Enter the source vertex for Dijkstra's Algorithm: ");                  scanf("%d", &*src*);                  if (src >= 0 && src < vertices) {                      dijkstraAlgorithm(graph, vertices, src);                  } else {                      printf("Invalid source vertex.\n");                  }                  break;              case 3:                  printf("Execution Completed\n");                  break;              default:                  printf("Invalid choice. Please enter again.\n");          }      } while (choice != 3);      for (int i = 0; i < vertices; i++) {          free(graph[i]);      }      free(graph);      return 0;  } |
| **Output** |  |
| **Pseudo Code** |  |
| **Conclusion** | Hence, by completing this experiment I came to know about implementation of Prims and Dijkestra algorithm. |