PICT, PUNE

31118

LP-II

Assignment 03

DOP: 4-Feb'22
DOS: 11-Feb'22.

Title: Greedy Search Algorithm.

Problem Statement:

Implement greedy search algorithm for any of the following applications:

I) Selection sort
II) Minimum spanning Tree
III) Single-Source shortest path problem
IV) Job-scheduling problem
V) Prim's min. spanning tree algo.
VI) Kruskal's min spanning tree algo.
VII) Dijkstra's min spanning tree algo.

Objectives:

1) To learn about greedy search algorithms.
2) Implement search algorithm.

Software & Hardware Requirements:

Windows-10 OS (64-bit, Home edition)
8 GB RAM, 512 GB SSD, i5-8Gen processor (4-core)
VS code latest version (Feb'22)
Python 3.8 interpreter

Theory:

Greedy Algorithm: It is an algorithmic paradigm that builds up a solution piece by piece, always choosing the next piece that offers the most obvious & immediate benefits. So the problems where

choosing locally optimal also leads to global solution.

Kruskal's algorithm:

1) MST: given a connected & undirected graph, a spanning tree of graph is a subgraph that is a tree & connects all the vertices together. Minimum spanning tree is a spanning tree of min. weight.

2) Kruskal's algorithm is a greedy algorithm to find MST in a given graph.

description & pseudocode.

1) Kruskal's algorithm initially places all the nodes of the original graph isolated from each other, to form a forest of single node trees, & then gradually merges these trees, combining at each iteration any two of all the trees with some edge of original graph.

2) Pseudo code:

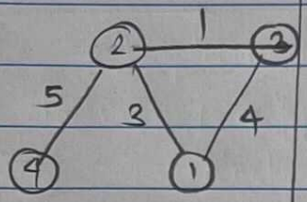
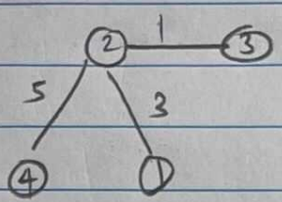
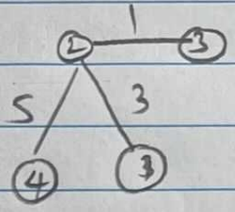
   i) Sort all the edges in non-decreasing order

   ii) Pick the smallest edge. check if it forms a cycle with spanning tree formed so far. If cycle is not formed, include this edge, else discard it.

   iii) Repeat step-ii until there are $(v-1)$ edges in the spanning tree.

PICT, PUNE

Testcase

Graph:	Expected O/p.	Actual O/p	Result
			Pass

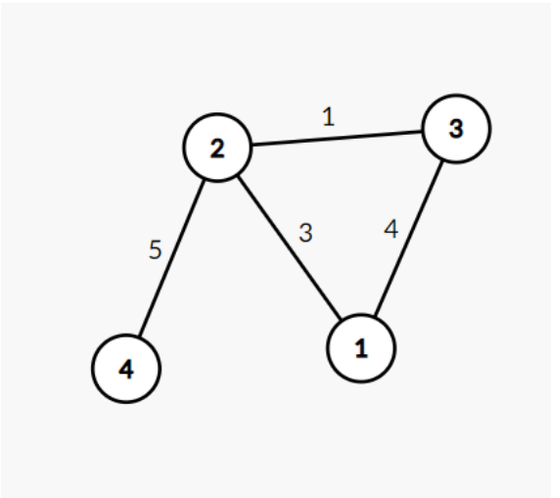
Conclusion

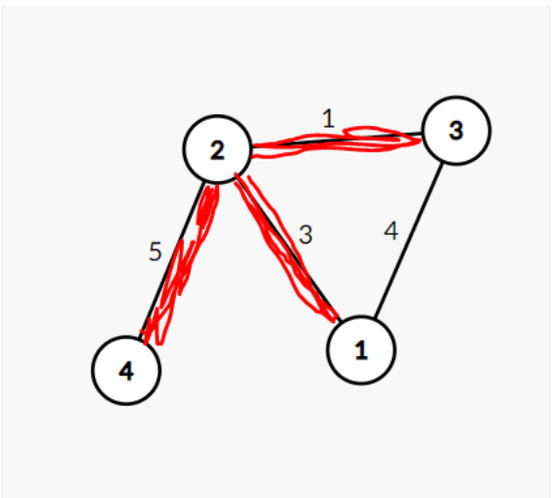
In this assignment, I learned about kruskal's greedy search algorithm & implemented the same.

```python
# Krushka's MST Algo


class DSU:
    def __init__(self, n) -> None:
        self.n = n
        self.par = [x for x in range(0, n+1)]
        self.size = [1 for x in range(0, n+1)]

    def get(self, x):
        if x == self.par[x]:
            return x
        self.par[x] = self.get(self.par[x])
        return self.par[x]

    def same_set(self, x, y):
        return self.get(x) == self.get(y)

    def unify(self, x, y):
        x = self.get(x)
        y = self.get(y)
        if (x == y):
            return 0
        if (self.size[x] < self.size[y]):
            x, y = y, x
        if (self.size[x] == self.size[y]):
            self.size[x] += 1

        self.par[y] = x
        return 1


class MST:
    # edge list representation
    def __init__(self, n) -> None:
        self.n = n
        self.edges = []

    def get_input(self):
        print("Enter u, v, wt")
        for i in range(1, self.n+1):
            u, v, wt = [int(x) for x in input().split()]
            if (u > v):
                u, v = v, u
            self.edges.append((u, v, wt))

        print("The Graph is: ")
        self.print_graph(self.edges)

    def create_mst(self):
        self.edges = sorted(
```

```
            self.edges, key=lambda tpl: (tpl[2], tpl[0], tpl[1]))

        ds = DSU(self.n)
        self.mst_edges = []
        mst_wt = 0
        for (u, v, wt) in self.edges:
            if not ds.same_set(u, v):
                ds.unify(u, v)
                mst_wt += wt
                self.mst_edges.append((u, v, wt))

        print("Weight of MST is ", mst_wt)
        print("MST edges are:")
        self.print_graph(self.mst_edges)

    def print_graph(self, edges):
        for tpl in edges:
            print(tpl)


def main():
    n = int(input("Enter number of nodes: "))
    mst = MST(n)
    mst.get_input()
    mst.create_mst()


main()
```

Testcase:






```
Shubham@Shubham-AcerSwift MINGW64 /d/College-Stuff-6th-Sem/LPII/A03 (main)
$ py 31118_A03.py
Enter number of nodes: 4
Enter u, v, wt
2 3 1
2 1 3
3 1 4
2 4 5
The Graph is:
(2, 3, 1)
(1, 2, 3)
(1, 3, 4)
(2, 4, 5)
MST edges are:
(2, 3, 1)
(1, 2, 3)
(2, 4, 5)
```