graph

March 24, 2023

1 To find the maximum sum, we can execute the code in one go.

[3]:
```python
lines = open("G81.txt", "r").readlines()

# This maintains a record of the grouping to which each vertex belongs - known␣
  ↪as a vertex grouping dictionary.
vert = {}

# dictionary of vertices to neighbours
neigh = {}

# array of edges
edges = [] # weight, node1, node2

# unnecessary variable that we forgot to remove
i = 0
```

[4]:
```python
for line in lines[1:]:
    split = line.strip().split()

    # initially leave the grouping of each vertex as None
    vert[split[0]] = None
    vert[split[1]] = None

    for i in 0, 1:
        # add each vertex to the neighbours dictionary
        if split[i] not in neigh:
            neigh[split[i]] = []

        # add the neighbour and the weight of the edge
        # this adds [neighbour, weight] to the array
        neigh[split[i]].append([split[1 - i], int(split[2])])

    # add [weight, vertex1, vertex2] to the edge array
    edges.append([int(split[2]), split[0], split[1]])
```

Initially, our approach was to traverse a list of edges in reverse-sorted order and repeatedly divide the edges along this list until it's no longer possible to divide them further.

[5]:
```
# reverse sorting out edges
edges.sort()
edges.reverse()
```

During the edge-cutting process, if neither vertex is assigned to a grouping, this function determines the position of the left vertex. We initially planned to introduce randomness into this process and created a separate function for it. However, we later devised a more efficient approach and decided not to implement this function.

[6]:
```
def put_zeros():
    return 0
```

[7]:
```
# sum of edges that we cut (variable name is a misnomer)
sum_vert = 0

# Our algorithm involves examining each edge in decreasing order of weight and␣
 ↪analyzing the vertices connected by the edge.
# If both vertices are unassigned to any group, they are allocated to different␣
 ↪groups based on the outcome of the "put_zeros" function.
# If one vertex has no grouping, and the other does, we assign the unassigned␣
 ↪vertex to the opposite group of the assigned vertex.
# If both vertices are already grouped, we do not take any action.

for edge in edges:
    n1 = vert[edge[1]]
    n2 = vert[edge[2]]



    if n1 is None:
        if n2 is None:
            vert[edge[1]] = put_zeros()
            vert[edge[2]] = 1 - vert[edge[1]]
        else:
            vert[edge[1]] = 1 - vert[edge[2]]
    elif n2 is None:
        vert[edge[2]] = 1 - vert[edge[1]]

    # if the vertices are in different groups, add the weight of this edge
    # to our sum
    if vert[edge[1]] != vert[edge[2]]:
        sum_vert += edge[0]

print(sum_vert)
```

```
6210
```