**Vogel’s Approximation Method**

from collections import defaultdict

costs  = {'X': {'A': 19, 'B': 30, 'C': 50, 'D': 10 },

          'Y': {'A': 70, 'B': 30, 'C': 40, 'D': 60 },

          'Z': {'A': 40, 'B': 8 , 'C': 70, 'D': 20 }}

demand = {'A': 5, 'B': 8, 'C': 7, 'D': 14 }

cols = sorted(demand.keys())

supply = {'X': 7, 'Y': 9, 'Z': 18 }

res = dict((k, defaultdict(int)) for k in costs)

g = {}

for x in supply:

    g[x] = sorted(costs[x].keys(), key=lambda g: costs[x][g])

for x in demand:

    g[x] = sorted(costs.keys(), key=lambda g: costs[g][x])

while g:

    d = {}

    for x in demand:

        d[x] = (costs[g[x][1]][x] - costs[g[x][0]][x]) if len(g[x]) > 1 else costs[g[x][0]][x]

    s = {}

    for x in supply:

        s[x] = (costs[x][g[x][1]] - costs[x][g[x][0]]) if len(g[x]) > 1 else costs[x][g[x][0]]

    f = max(d, key=lambda n: d[n])

    t = max(s, key=lambda n: s[n])

    t, f = (f, g[f][0]) if d[f] > s[t] else (g[t][0], t)

    v = min(supply[f], demand[t])

    res[f][t] += v

    demand[t] -= v

    if demand[t] == 0:

        for k, n in supply.items():

            if n != 0:

                g[k].remove(t)

        del g[t]

        del demand[t]

    supply[f] -= v

    if supply[f] == 0:

        for k, n in demand.items():

            if n != 0:

                g[k].remove(f)

        del g[f]

        del supply[f]

for n in cols:

    print("\t", n,end="")

print()

cost = 0

for g in sorted(costs):

    print(g, "\t",end="")

    for n in cols:

        y = res[g][n]

        if y != 0:

            print(y,end="")

        cost += y \* costs[g][n]

        print("\t",end="")

    print()

print("\n\nTotal Cost = ", cost)

**OUTPUT:**

A B C D

X 5 2

Y 7 2

Z 8 10

Total Cost = 779