Beer Distribution Problem for the Pulp Modeler

MSDS 650 -- Sean O'Malley

- Integer programming problems are used to maximize or optimize results.
- In integer problems some or all of the variables are restricted to integers.

In [50]:
```
# Import PuLP modeler supply nodes

from pulp import *
```

In [51]:
```
# Create list of all supply nodes

Warehouses = ["A", "B"]
```

In [52]:
```
# Create dictionary for number of units of supply for each supply node

supply = {"A" : 1000,
         "B" : 4000}
```

In [53]:
```
# Create a list of all demand nodes

Bars = ["1","2","3","4","5"]
```

In [54]:
```
# Creates a dictionary for number of units of demand for each demand node

demand = {"1" : 500,
          "2" : 900,
          "3" : 1800,
          "4" : 200,
          "5" : 700}
```

In [55]:
```
# Creates a list of costs of each transportation path

costs = [    # bars
         #1,2,3,4,5
         [2,4,5,2,1], #Path A    Warehouses
         [3,1,3,2,3]  #Path B
        ]
```

In [56]:
```
# Turn cost data into a dictionary with makeDict function

costs = makeDict([Warehouses, Bars], costs, 0)
```

In [57]:
```
# Create Problem variable  to contain the problem data

prob = LpProblem("Beer Distribution Problem", LpMinimize)
```

In [58]:
```
# Create a list of tuples (finite ordered list of elements) containing all the possible routes of tra
nsport

routes = [(w,b) for w in Warehouses for b in Bars]
```

In [59]:
```
# Create dictionary called 'Vars to contain the referenced variables (routes)

vars = LpVariable.dicts("Route",(Warehouses,Bars),0,None,LpInteger)
```

In [60]:
```
# Create objective function to add to 'Problem'

prob += lpSum([vars[w][b]*costs[w][b] for (w,b) in routes]), "Sum_of_Transporting_Costs"
```

In [61]:
```
# Supply maximum constraints to Problem for each supply node (warehouse)

for w in Warehouses:
    prob += lpSum([vars[w][b] for b in Bars]) <= supply[w], "Sum_of_Products_out_of_Warehouse_%s"%w
```

In [62]:
```
# Demand minimum constraints are added to prob for each demand node (bar)

for b in Bars:
    prob += lpSum([vars[w][b] for b in Bars]) <= supply[w], "Sum_of_Products_into_Bar%s"%b
```

In [63]:
```
# Write problem to a .lp file

prob.writeLP("/Users/SeanOMalley1/Desktop/MSDS_650/Week_8/BeerDistributionProblem.lp")
```

Beer Distribution Problem Minimize OBJ: __dummy Subject To Sum_of_Products_into_Bar1: Route_B_1 + Route_B_2 + Route_B_3 + Route_B_4

- Route_B_5 <= 4000 Sum_of_Products_into_Bar2: Route_B_1 + Route_B_2 + Route_B_3 + Route_B_4
- Route_B_5 <= 4000 Sum_of_Products_into_Bar3: Route_B_1 + Route_B_2 + Route_B_3 + Route_B_4
- Route_B_5 <= 4000 Sum_of_Products_into_Bar4: Route_B_1 + Route_B_2 + Route_B_3 + Route_B_4
- Route_B_5 <= 4000 Sum_of_Products_into_Bar5: Route_B_1 + Route_B_2 + Route_B_3 + Route_B_4
- Route_B_5 <= 4000 Bounds 0 <= Route_B_1 0 <= Route_B_2 0 <= Route_B_3 0 <= Route_B_4 0 <= Route_B_5 __dummy = 0 Generals Route_B_1 Route_B_2 Route_B_3 Route_B_4 Route_B_5 End

In [64]:
```
# Problem is solved using PuLp's choice of Solver

prob.solve()
```

Out[64]: 1

In [65]:
```
# Status of the solution is printed to screen

print ("Status:", LpStatus[prob.status])
```

```
Status: Optimal
```

In [66]:
```
# Print each bariable with its resolved optimum value
for v in prob.variables():
    print (v.name, "=", v.varValue)
```

```
Route_A_1 = 0.0
Route_A_2 = 0.0
Route_A_3 = 0.0
Route_A_4 = 0.0
Route_A_5 = 0.0
Route_B_1 = 0.0
Route_B_2 = 0.0
Route_B_3 = 0.0
Route_B_4 = 0.0
Route_B_5 = 0.0
```

In [68]:
```
# Lastly, print optimized objective function value

print ("Total Cost of Transportation = ",value(prob.objective))
```

```
Total Cost of Transportation =  0.0
```

Conclusion

We are minimising the transportation cost for a brewery operation. The brewery transports cases of beer from its warehouses to several bars in the optimal fashion given our integer programming results

In [ ]:
```
```