```
In [25]: import sys
         import numpy as np

         from gamspy import (
             Container,Set,Alias,Parameter,Variable,Equation,Model,Problem,Sense,Opti
             Domain,Number,Sum,Product,Smax,Smin,Ord,Card,SpecialValues,
         )

         m = Container()
```

```
In [26]: # YOU NEED TO UPDATE NODES and ARCS here
         i = Set(m,'i',records= ['store', 'trash'] + [f't{i}' for i in range(1, 11)]

         z = Alias(m,'z',i)
         j = Alias(m,'j',i)
         k = Alias(m,'k',i)

         t = Set(m,'t',domain=i, records = [f't{i}' for i in range(1, 11)])

         h = Set(m,'h',domain=i, records = [f'h{i}' for i in range(1, 11)])

         d = Parameter(m,'d', domain=t, domain_forwarding = True,
             records=[ ('t1', 50), ('t2', 60), ('t3', 80), ('t4', 70),
                 ('t5', 50), ('t6', 60), ('t7', 90), ('t8', 80), ('t9', 50), ('t10',
         alpha = Parameter(m,'alpha', records=200)
         beta = Parameter(m,'beta', records=75)
         gamma = Parameter(m,'gamma', records=25)
         p = Parameter(m,'p', records=4)
         q = Parameter(m,'q', records=2)

         arcs = Set(m, 'arcs', domain=[i, i])
         arcs['store', t] = True
         arcs[t, h].where[Ord(t) == Ord(h)] = True
         arcs[h, t].where[Ord(h) == Ord(t) - p] = True
         arcs[h, t].where[Ord(h) == Ord(t) - q] = True
         arcs[h, 'trash'] = True
         arcs['store', 'trash'] = True
         display(arcs.pivot())
```

```
/home/samjenkins2001/CS524/venv/lib/python3.10/site-packages/gams/transfer/s
yms/_mixins/pivot.py:121: FutureWarning: Downcasting object dtype arrays on
.fillna, .ffill, .bfill is deprecated and will change in a future version. C
all result.infer_objects(copy=False) instead. To opt-in to the future behavi
or, set `pd.set_option('future.no_silent_downcasting', True)`
  df.fillna(fill_value, inplace=True)
```

	trash	t1	t2	t3	t4	t5	t6	t7	t8	t9	...	h1	
store	True	True	True	True	True	True	True	True	True	True	...	False	Fal
t1	False	False	False	False	False	False	False	False	False	False	...	True	Fal
t2	False	False	False	False	False	False	False	False	False	False	...	False	Tr
t3	False	False	False	False	False	False	False	False	False	False	...	False	Fal
t4	False	False	False	False	False	False	False	False	False	False	...	False	Fal
t5	False	False	False	False	False	False	False	False	False	False	...	False	Fal
t6	False	False	False	False	False	False	False	False	False	False	...	False	Fal
t7	False	False	False	False	False	False	False	False	False	False	...	False	Fal
t8	False	False	False	False	False	False	False	False	False	False	...	False	Fal
t9	False	False	False	False	False	False	False	False	False	False	...	False	Fal
t10	False	False	False	False	False	False	False	False	False	False	...	False	Fal
h1	True	False	False	True	False	True	False	False	False	False	...	False	Fal
h2	True	False	False	False	True	False	True	False	False	False	...	False	Fal
h3	True	False	False	False	False	True	False	True	False	False	...	False	Fal
h4	True	False	False	False	False	False	True	False	True	False	...	False	Fal
h5	True	False	False	False	False	False	False	True	False	True	...	False	Fal
h6	True	False	False	False	False	False	False	False	True	False	...	False	Fal
h7	True	False	False	False	False	False	False	False	False	True	...	False	Fal
h8	True	False	False	False	False	False	False	False	False	False	...	False	Fal
h9	True	False	False	False	False	False	False	False	False	False	...	False	Fal
h10	True	False	False	False	False	False	False	False	False	False	...	False	Fal

21 rows × 21 columns

In [27]:
```
# put your code here
b = Parameter(m,"b",domain=i)
for a in range(1, 11):
    b[f't{a}'] = -d[f't{a}']
    b[f'h{a}'] = d[f't{a}']
b['store'] = 5000
b['trash'] = -5000

c = Parameter(m, 'c', domain=[i,i])
c['store', t] = alpha
for y in range(1, 9):
    c[f'h{y}', f't{y+2}'] = beta
for y in range(1, 7):
    c[f'h{y}', f't{y+4}'] = gamma

x = Variable(m, "x", "positive", domain=[i,i])
```

```
balance = Equation(m, 'balance', domain=i)
balance[z] = Sum(arcs[z,j], x[z,j]) - Sum(arcs[k,z], x[k,z]) == b[z]

Malfoy = Model(m,
    name="Malfoy",
    equations=m.getEquations(),
    problem=Problem.LP,
    sense=Sense.MIN,
    objective=Sum(arcs[i,z], c[i,z]*x[i,z]),
)

Malfoy.solve()
```

Out[27]:

	Solver Status	Model Status	Objective	Num of Equations	Num of Variables	Model Type	Solver	Solve Tim
0	Normal	OptimalGlobal	66750	23	46	LP	CPLEX	0.00

In [28]:

```
Cost = Parameter(m,'Cost')
Cost[:] = Malfoy.objective_value

NumEqu= Parameter(m,'NumEqu')
NumEqu[:] = Malfoy.num_equations

NumBought= Parameter(m,'NumBought')
# following may need update if your variables are different
NumBought[:] = Sum(t, x.l['store',t])

print(f"Cost = {Cost.toValue()}")
print(f"NumEqu = {NumEqu.toValue()}")
print(f"NumBought = {NumBought.toValue()}")
```

```
Cost = 66750.0
NumEqu = 23.0
NumBought = 260.0
```

In [ ]: