In [368]:
```
mod = JuMP.Model(JuMP.optimizer_with_attributes((TimeLimit=45) -> Gurobi.Optimizer(TimeLimit=120)))
set_optimizer_attribute(mod, "OutputFlag", 0)

#PARAMETERS
k = 10
n = 30
m = 3
w = players2.Points
M = 10000

#VARIABLES
@variable(mod, x[i=1:n, p=1:m], Bin)
@variable(mod, z[i=1:n,j=1:n]>=0)


#CONSTRAINTS
#epigraph constraints
@constraint(mod, [p=1:m-1, q=p+1:m, i=1:n-1, j=i+1:n], z[i,j] >= w[i] - w[j] + M*(x[i,p] + x[j,q] - 2))
@constraint(mod, [p=1:m-1, q=p+1:m, i=1:n-1, j=i+1:n], z[i,j] >= w[j] - w[i] + M*(x[i,p] + x[j,q] - 2))
@constraint(mod, [p=1:m-1, q=p+1:m, i=1:n-1, j=i+1:n], z[i,j] >= w[i] - w[j] + M*(x[i,q] + x[j,p] - 2))
@constraint(mod, [p=1:m-1, q=p+1:m, i=1:n-1, j=i+1:n], z[i,j] >= w[j] - w[i] + M*(x[i,q] + x[j,p] - 2))

#every player is assigned a group
@constraint(mod, [p=1:m], sum(x[i,p] for i=1:n)== k)
#a player is assigned to exaclty one group
@constraint(mod, [i=1:n], sum(x[i,p] for p=1:m)== 1)
@constraint(mod, [i in 1:m-1, p in i+1:m], x[i,p]==0)

@objective(mod, Min, sum(z[i,j] for i=1:n,j=1:n))

optimize!(mod)
```

```
Academic license - for non-commercial use only - expires 2022-08-19
```

```
┌ Warning: Passing optimizer attributes as keyword arguments to
│ Gurobi.Optimizer is deprecated. Use
│     MOI.set(model, MOI.RawParameter("key"), value)
│ or
│     JuMP.set_optimizer_attribute(model, "key", value)
│ instead.
└ @ Gurobi /Users/bennetthellman/.julia/packages/Gurobi/BAtIN/src/MOI_wrapper/MOI_wrapper.jl:273
```