In [1]:

```
using JuMP, NamedArrays, Clp

availability =
  [ 0 0 1 1 0 0 0 0 0 1 1 0 0 0 0
    0 1 1 0 0 0 0 0 0 0 1 1 0 0 0
    0 0 0 1 1 0 1 1 1 1 0 1 1 1 1
    0 0 0 1 1 1 1 1 1 1 1 1 1 1 0
    0 0 0 0 0 0 1 1 1 1 1 0 0 0 0
    0 1 1 0 0 0 0 0 0 0 1 1 0 0 0
    0 0 0 1 1 1 1 1 1 0 0 0 0 0 0
    1 1 0 0 0 0 0 0 0 0 0 0 1 1 1
    1 1 1 0 0 0 0 0 0 0 0 1 1 0 0
    0 0 0 0 0 0 0 0 0 1 1 0 0 0 0
    0 0 0 0 0 0 1 1 1 1 1 0 0 0 0
    1 1 0 0 0 1 1 1 1 1 1 0 0 1 1
    1 1 1 0 1 1 0 0 0 0 0 0 0 1 1
    0 1 1 1 0 0 0 0 0 0 0 0 0 0 0
    1 1 0 0 1 1 0 0 0 0 0 0 0 0 0 ]

TIMES = ["10:00","10:20","10:40","11:00","11:20","11:40","lunch1","lunch2","lunch3","1:00","1:20","1
NAMES = [:Manuel,:Luca,:Jule,:Michael,:Malte,:Chris,:Spyros,:Mirjam,:Matt,:Florian,:Josep,:Joel,:Tom
times = NamedArray( availability, (NAMES,TIMES), ("NAME","TIME"))

m = Model(solver=ClpSolver())

@variable(m, x[NAMES,TIMES] >= 0)

# a candidate meet all employees
@constraint(m, a[j in TIMES], sum(x[i,j] for i in NAMES) == 1 )

# each employee meet the candidate one time
@constraint(m, b[i in NAMES], sum(x[i,j] for j in TIMES) == 1 )

@objective(m, Max, 1*sum( x[i,j]*times[i,j] for i in NAMES, j in TIMES ) )

status = solve(m)
println(status)

assignment = NamedArray( [ (getvalue(x[i,j])) for i in NAMES, j in TIMES ], (NAMES, TIMES), ("NAMES
println(assignment)
println("Total employees meeted: ", getobjectivevalue(m))
```

```
Optimal
15×15 Named Array{Float64,2}
NAMES ╲ TIMES │  10:00   10:20   10:40  …    2:00    2:20    2:40
──────────────────────────────┼──────────────────────────────────────────────────
───────────────────────────────────────────────
:Manuel        │    0.0     0.0     1.0  …    0.0     0.0     0.0
:Luca          │    0.0     0.0     0.0     0.0     0.0     0.0
:Jule          │    0.0     0.0     0.0     0.0     0.0     0.0
:Michael       │    0.0     0.0     0.0     0.0     1.0     0.0
:Malte         │    0.0     0.0     0.0     0.0     0.0     0.0
:Chris         │    0.0     0.0     0.0     0.0     0.0     0.0
:Spyros        │    0.0     0.0     0.0     0.0     0.0     0.0
:Mirjam        │    0.0     1.0     0.0     0.0     0.0     0.0
:Matt          │    0.0     0.0     0.0     1.0     0.0     0.0
:Florian       │    0.0     0.0     0.0     0.0     0.0     0.0
:Josep         │    0.0     0.0     0.0     0.0     0.0     0.0
:Joel          │    1.0     0.0     0.0     0.0     0.0     0.0
```

```
:Tom            │     0.0     0.0     0.0         0.0     0.0     1.0
:Daniel         │     0.0     0.0     0.0         0.0     0.0     0.0
:Anne           │     0.0     0.0     0.0  …      0.0     0.0     0.0
Total employees meeted: 15.0
```

In [ ]:

	10:00	10:20	10:40	11:00	11:20	11:40	Lunch	1:00	1:20	1:40	2:00	2:20	2:40
Manuel	0	0	[✓ 1]	1	0	0	0	1	1	0	0	0	0
Luca	0	1	1	0	0	0	0	0	1	[✓ 1]	0	0	0
Jule	0	0	0	1	[✓ 1]	0	1	1	0	1	1	1	1
Michael	0	0	0	1	1	1	1	1	1	1	1	[✓ 1]	0
Malte	0	0	0	0	0	0	[✓ 1]	1	1	0	0	0	0
Chris	0	1	1	0	0	0	0	0	[✓ 1]	1	0	0	0
Spyros	0	0	0	1	1	1	[✓ 1]	0	0	0	0	0	0
Mirjam	1	[✓ 1]	0	0	0	0	0	0	0	0	1	1	1
Matt	1	1	1	0	0	0	0	0	0	1	[✓ 1]	0	0
Florian	0	0	0	0	0	0	0	[✓ 1]	1	0	0	0	0
Josep	0	0	0	0	0	0	[✓ 1]	1	1	0	0	0	0
Joel	[✓ 1]	1	0	0	0	1	1	1	1	0	0	1	1
Tom	1	1	1	0	1	1	0	0	0	0	0	1	[✓ 1]
Daniel	0	1	1	[✓ 1]	0	0	0	0	0	0	0	0	0
Anne	1	1	0	0	1	[✓ 1]	0	0	0	0	0	0	0

In [3]:

```
#question 2

using JuMP, Clp, NamedArrays

agencies1 = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
agencies2 = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
dis = [0 36.7696 26.7686 42.0043 45.5000 53.8206 35.6968 14.5344 14.3000 19.6726;
    36.7696 0 13.2575 16.6481 32.5 18.1069 21.5188 23.4361 28.5112 24.2860;
    26.7686 13.2575 0 15.8152 25.64 27.5772 27.8212 16.9 15.8686 21.792;
    42.0043 16.6481 15.8152 0 16.9 17.3442 37.9012 32.6038 29.2139 36.6083;
    45.5 32.5 25.64 16.9 0 32.6038 52.4691 41.1096 31.2 47.1239;
    53.8206 18.1069 27.5772 17.3442 32.6038 0 36.4927 41.2942 43.2921 42.3449;
    35.6968 21.5188 27.8212 37.9012 52.4691 36.4927 0 22.1 35.9562 16.0801;
    14.5344 23.4361 16.9 32.6038 41.1096 41.2942 22.1 0 15.1605 7.5802;
    14.3 28.5112 15.8685 29.2139 31.2 43.2921 35.9562 15.1605 0 22.7407;
    19.6762 24.286 21.792 36.6083 47.1239 42.3449 16.0801 7.5802 22.74007 0]

cost_per_haul = 0.5    # don't forget the return trip!

dist = NamedArray( dis, (agencies1,agencies2), ("Agencies","Agencies") )
required = Dict(zip( agencies2, [10 6 8 11 9 7 15 7 9 12] ))
present = Dict(zip( agencies1, [8 13 4 8 12 2 14 11 15 7] ))

m = Model(solver=ClpSolver())

@variable(m, x[agencies1,agencies2] >= 0)          # x[i,j] is lumber shipped from site i to mill j.

@constraint(m, sup[i in agencies1], sum(x[i,j] for j in agencies2) == present[i] )   # supply const
@constraint(m, dem[j in agencies2], sum(x[i,j] for i in agencies1) == required[j] )   # demand cons

@objective(m, Min, cost_per_haul*sum( x[i,j]*dist[i,j] for i in agencies1, j in agencies2 ) )

status = solve(m)

println(status)

# nicely formatted solution
solution = NamedArray( Int[getvalue(x[i,j]) for i in agencies1, j in agencies2], (agencies1,agencie
println( solution )
println()
println("Minimized total cost will be \$", getobjectivevalue(m))
```

```
Optimal
10×10 Named Array{Int64,2}
agencies ╲ agencies │  1   2   3   4   5   6   7   8   9  10
────────────────────┼─────────────────────────────────────────
────────────────────
1                   │  8   0   0   0   0   0   0   0   0   0
2                   │  0   6   1   0   0   5   1   0   0   0
3                   │  0   0   4   0   0   0   0   0   0   0
4                   │  0   0   0   8   0   0   0   0   0   0
5                   │  0   0   0   3   9   0   0   0   0   0
6                   │  0   0   0   0   0   2   0   0   0   0
7                   │  0   0   0   0   0   0  14   0   0   0
8                   │  0   0   0   0   0   0   0   6   0   5
9                   │  2   0   3   0   0   0   0   1   9   0
10                  │  0   0   0   0   0   0   0   0   0   7

Minimized total cost will be $152.63889999999998
```

```
Agency #2 need to move 1 car to agency #3, 5 cars to agency #6 and 1 car to agency #7
Agency #5 need to move 3 cars to agency #4
Agency #8 need to move 5 cars to agency #10
Agency #9 need to move 2 cars to agency #1, 3 cars to agency #3 and 1 car to agency #8

This movement will minimize the cost
Minimized cost is $152.63889999999998
```

In [55]:

```
#question 3

using JuMP,Clp

tasks = 1:18
durations = [2 16 9 8 10 6 2 2 9 5 3 2 1 7 4 3 9 1]
predecessors = ( [], [1], [2], [2], [3], [4,5], [4], [6], [4,6], [4], [6], [9], [7], [2], [4,14], [8
pred_dict = Dict(zip(tasks,predecessors));   # dictionary mapping tasks --> predecessors.

# additional columns of data (maximum reduction possible )
max_reduction =  [0,  3,  1,  2,  2,  1, 1, 0,  2,  1,  1, 0, 0,  2,  2, 1,  3, 0]  # max reduction
cost_reduction = [0, 30, 26, 12, 17, 15, 8, 0, 42, 21, 18, 0, 0, 22, 12, 6, 16, 0]  # cost of reduc
bonus_amount = 30     # bonus for expediting the project ($1,000/week )


m = Model(solver=ClpSolver())

@variable(m, x[tasks] >= 0)

for i in tasks
    for j in predecessors[i]
        @constraint(m, x[i] >= x[j] + durations[j])
    end
end
@constraint(m, x[1] == 0)
@objective(m, Min, x[18] + durations[18])     # total duration is start time of last task + duratio

solve(m)
println(getvalue(x))
println("minimum duration: ", getobjectivevalue(m))
```

```
x: 1 dimensions:
[ 1] = 0.0
[ 2] = 2.0
[ 3] = 18.0
[ 4] = 18.0
[ 5] = 27.0
[ 6] = 37.0
[ 7] = 26.0
[ 8] = 43.0
[ 9] = 43.0
[10] = 26.0
[11] = 43.0
[12] = 52.0
[13] = 28.0
[14] = 18.0
[15] = 26.0
[16] = 46.0
[17] = 54.0
[18] = 63.0
minimum duration: 64.0
```

The earliest possible data is after 64 weeks' construction.

In [59]:

```
m = Model(solver=ClpSolver())

@variable(m, x[tasks] >= 0)
@variable(m, save[tasks] >=0)

for i in tasks
    for j in predecessors[i]
        @constraint(m, x[i] >= x[j] + durations[j] - save[j])
    end
end
@constraint(m, x[1] == 0)
@constraint(m, save[2] == 3)#make sure that task 2 is cmpleted early
for i in tasks
    @constraint(m, save[i] <= max_reduction[i])
end

@objective(m, Max,(63-x[18])*30-sum(save[i]*cost_reduction[i] for i in tasks))
# total duration is start time of last task + duration of last task.

solve(m)
println(getvalue(x))
println("minimmum duration: ", getvalue(x[18])+1)
println("max profit: ", getobjectivevalue(m))
```

```
x: 1 dimensions:
[ 1] = 0.0
[ 2] = 2.0
[ 3] = 15.0
[ 4] = 15.0
[ 5] = 23.0
[ 6] = 31.0
[ 7] = 23.0
[ 8] = 36.0
[ 9] = 36.0
[10] = 23.0
[11] = 36.0
[12] = 45.0
[13] = 25.0
[14] = 15.0
[15] = 23.0
[16] = 39.0
[17] = 47.0
[18] = 53.0
minimmum duration: 54.0
max profit: 87.0
```

```
The profit will be maximized to $87k
And the project will be completed after 54 weeks
```

In [37]:

```
#question 4


A = [0 1;5/3 -1;-1 0;0 -1];
b = [500; 0; 0; 0]

using JuMP, Clp, LinearAlgebra
# If you haven't yet installed the linear algebra package, do the standard thing:
# using Pkg, Pkg.add("LinearAlgebra")

m = Model(solver=ClpSolver())
@variable(m, ra >= 0)           # radius
@variable(m, xa[1:2])           # coordinates of center
for i = 1:size(A,1)
    @constraint(m, A[i,:]'*xa + ra*norm(A[i,:]) <= b[i])
end
@objective(m, Max, ra)      # maximize radius

status = solve(m)
center = getvalue(xa)
radius = getvalue(ra)

println(status)
println("The coordinates of the Chebyshev center are: ", center)
println("The largest possible radius is: ", radius)
```

```
Optimal
The coordinates of the Chebyshev center are: [108.452, 391.548]
The largest possible radius is: 108.45240525773497
```

In [38]:

```
# Matrices A, b defining the inequalities
A = [2/3 1;-5/3 1;3 -1;0 -1];
b = [700; 0; 1500; 0]

using JuMP, Clp, LinearAlgebra
# If you haven't yet installed the linear algebra package, do the standard thing:
# using Pkg, Pkg.add("LinearAlgebra")

m = Model(solver=ClpSolver())
@variable(m, rb >= 0)          # radius
@variable(m, xb[1:2])          # coordinates of center
for i = 1:size(A,1)
    @constraint(m, A[i,:]'*xb + rb*norm(A[i,:]) <= b[i])
end
@objective(m, Max, rb)      # maximize radius

status = solve(m)
center = getvalue(xb)
radius = getvalue(rb)

println(status)
println("The coordinates of the Chebyshev center are: ", center)
println("The largest possible radius is: ", radius)
```

```
Optimal
The coordinates of the Chebyshev center are: [355.092, 201.05]
The largest possible radius is: 201.0495106921289
```

The museums should be located two different centers:
The first one's center: [108.452, 391.548], and radius is 108.45240525773497
The second one's center: [355.092, 201.05], and radius is 201.0495106921289

In [ ]:

```
```

I would formulate the problem by find the max distance between two museum center so that I can get the largest area to construct the museum.(following code that I formulated might be a way to compute)

In [ ]:

```
A = [0 1;2/3 1;3 -1;0 -1;-1 0];
b = [500;700;1500;0;0];

using JuMP, NLopt

m = Model(solver = NLoptSolver(algorithm = :LD_MMA))
@variable(m, r1>=0)
@variable(m, r2>=0)
@variable(m, x1[1:2])
@variable(m, x2[1:2])
for i = 1:size(A,1)
    @constraint(m,A[i,:]'*x1 + r1*norm(A[i,:])<=b[i])
end
for i = 1:size(A,1)
    @constraint(m,A[i,:]'*x2 + r2*norm(A[i,:])<=b[i])
end

@NLconstraint(m,(x1[1]-x2[1])^2 + (x1[2]-x2[2])^2 >= (r1 + r2)^2)
@objective(m,Max,r1+r2)

status = solve(m)
println(status)
println("The coordinates of the first new center are: ", getvalue(x1))
println("The radius of first new cycle: ", getvalue(r1))
println("The coordinates of the second new center are: ", getvalue(x2))
println("The radius of second new cycle: ", getvalue(r2))
```

This problem will not be linear problem since the constrains in the new model is not linear and linear solvers cannot solver this problem.

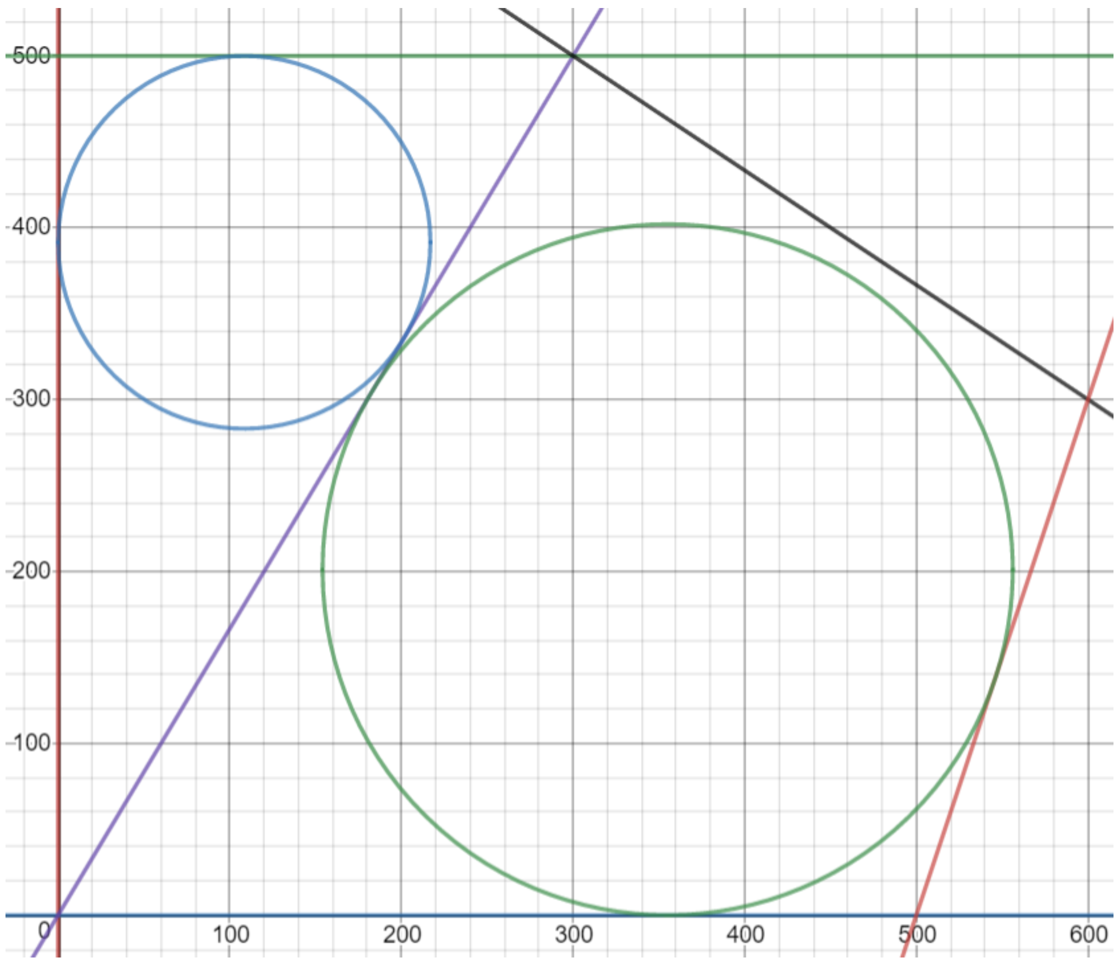
In [50]:

```
slope1 = (getvalue(xa[2])-getvalue(xb[2]))/(getvalue(xa[1])-getvalue(xb[1]))
slope2 = 5/3
slope1 * slope2
```

Out[50]:

```
-1.2872920039070335
```

I think the two museums will have a larger area if there is no limitation.
This is because the segment between two centers in previous problem is not vertical to the red line which means the red line is not the best line to divide space. Moreover, the two radius added together is also larger than the distance between two centers and this also indicates there is still much space to be occupied.
Therefore, two museums would have a larger total area if there area no limitations.



500
400
300
200
100
0
100
200
300
400
500
600