OR/ISE 501: Operations Research

Fall 2021

Review for the Final Exam

Final Exam Format

- The Final Exam will be an individual in-class exam. The exam will be open notes and closed computer. The questions will be a mix of short answer and questions that are computationally easy enough to solve "by hand," but you can use a non-programmable calculator. The exam will be comprehensive, but based primarily on the material covered in the final Review for the Final Exam section of the class.

- Keep in mind that you will not have access to Julia during the exam, so you should be able to solve small, by-hand versions of the problems covered in class using a calculator. The exam will consist of (a) questions related to material not covered in Exams 1 and 2, mostly material not suitable for the take-home format of Exams 1 and 2; and (b) questions related to the problems in the HWs and on Exams 1 and 2 that can include (i) solving a small instance of the problem that can be solved with a calculator, (ii) justifying the posted solution approach, and (iii) possible extensions of the problems.

NL Opt: Q1

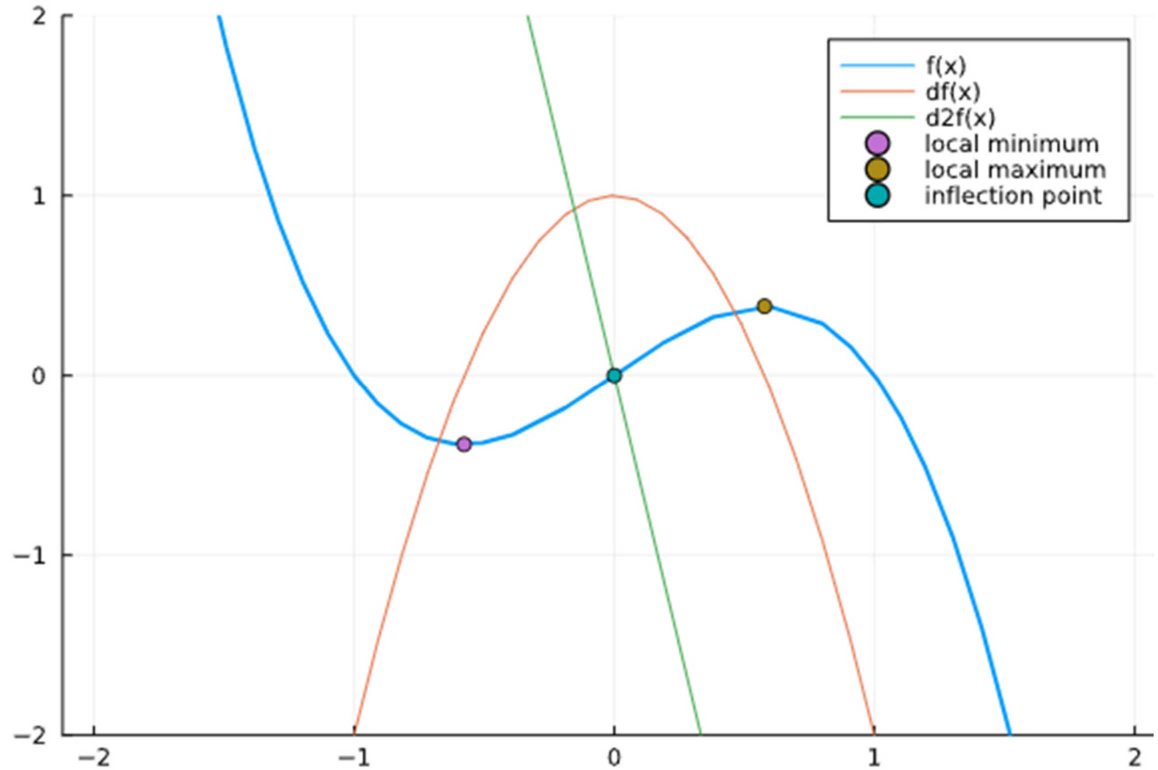
Determine the local maxima, local minima, and inflection points of $f(x) = x\left(1 - x^2\right)$.

$$y = x - x^3$$

$$\frac{dy}{dx} = 1 - 3x^2$$

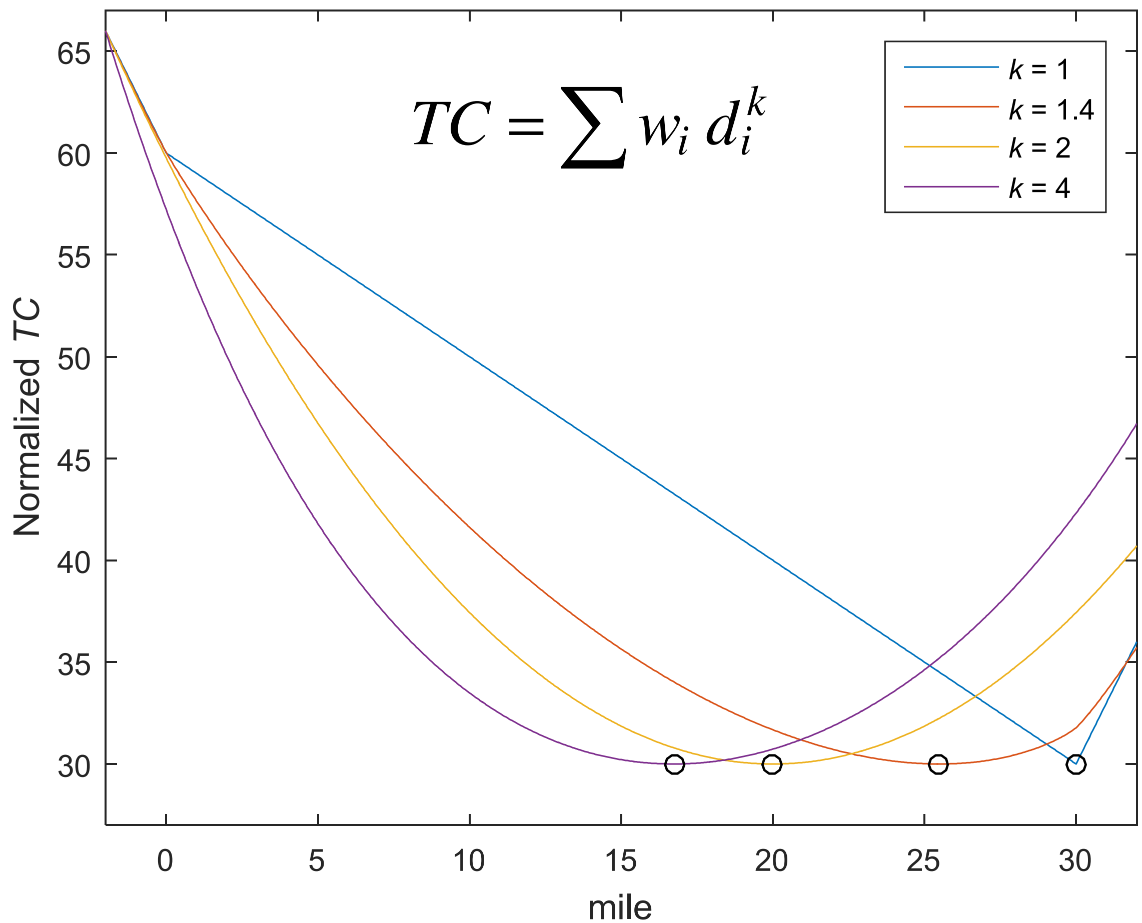
$$\frac{d^2y}{x^2} = -6x$$

$$x^* = \pm\sqrt{\frac{1}{3}}$$



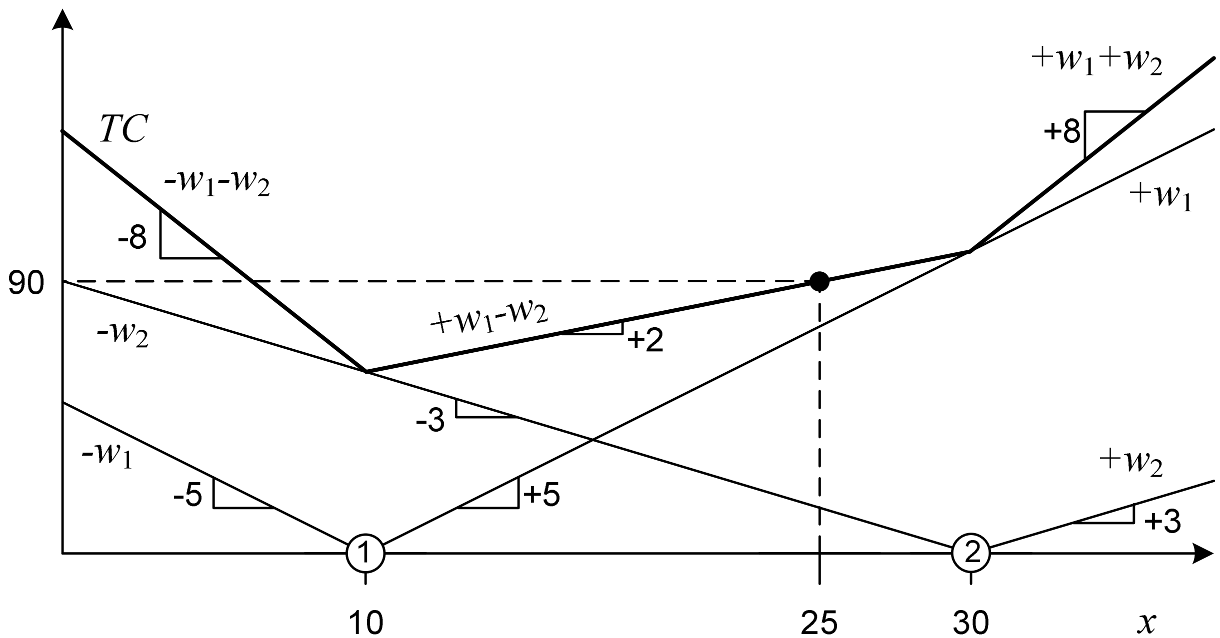
NL Opt: Q2

Determine the minimum of $f(x) = 5|x - 10| + 3|x - 30|$.



NL Opt: Q2

Determine the minimum of $f(x) = 5|x - 10| + 3|x - 30|$.



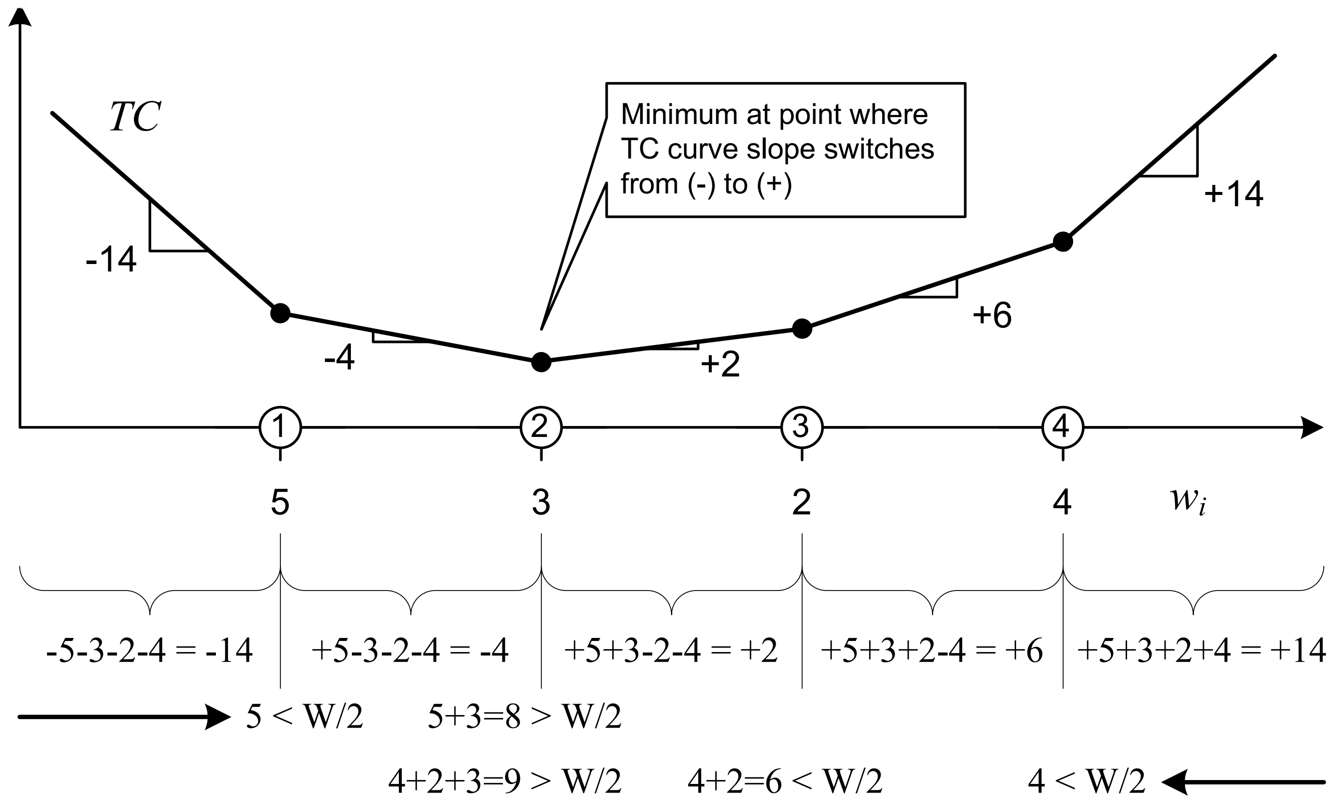
$$TC(x) = \sum w_i \, d_i = \beta_1(x - x_1) + \beta_2(x - x_2), \quad \text{where } \beta_i = \begin{cases} w_i, & \text{if } x \geq x_i \\ -w_i, & \text{if } x < x_i \end{cases}$$

$$TC(25) = w_1(25 - 10) + (-w_2)(25 - 30)$$

$$= 5(15) + (-3)(-5) = 90$$

NL Opt: Q3

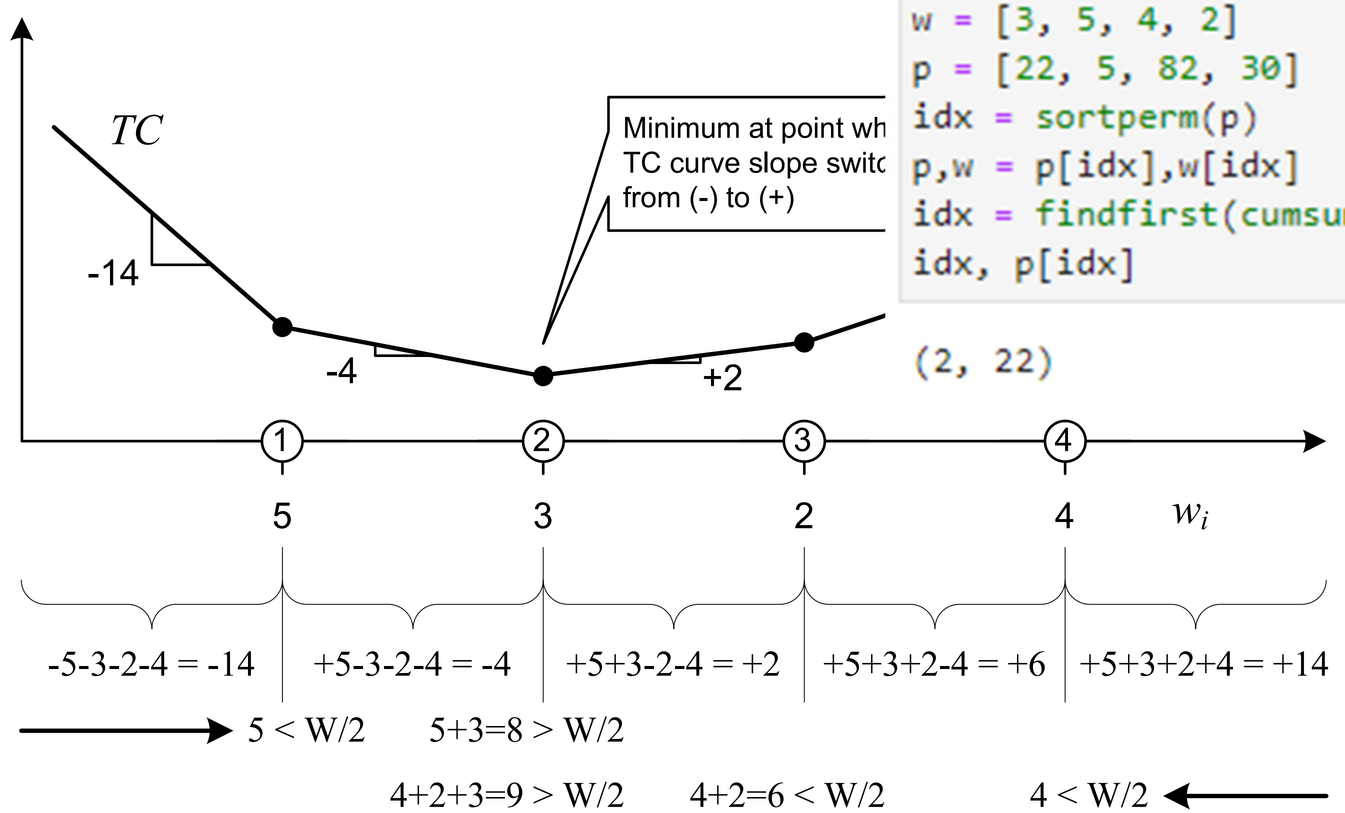
There are four stations located along a corridor, and a single testing device is to be located anywhere along the corridor so that it can receive samples from the stations. A technician must carry each sample from each station to the device. The distance in meters of the switches from the beginning of the corridor is 22, 5, 82, and 30, respectively, and the average number of daily samples per station is expected to be 3, 5, 4, and 2, respectively. Determine where the device should be located to minimize the total distance that the technician must travel.



NL Opt: Q3

Median location: For each dimension x of X:

1. Order EFs so that $|x_1| \le |x_2| \le \cdots \le |x_m|$
2. Locate x-dimension of NF at the first EFj where $\sum_{i=1}^{j} w_i \ge \frac{W}{2}$, where $W = \sum_{i=1}^{m} w_i$



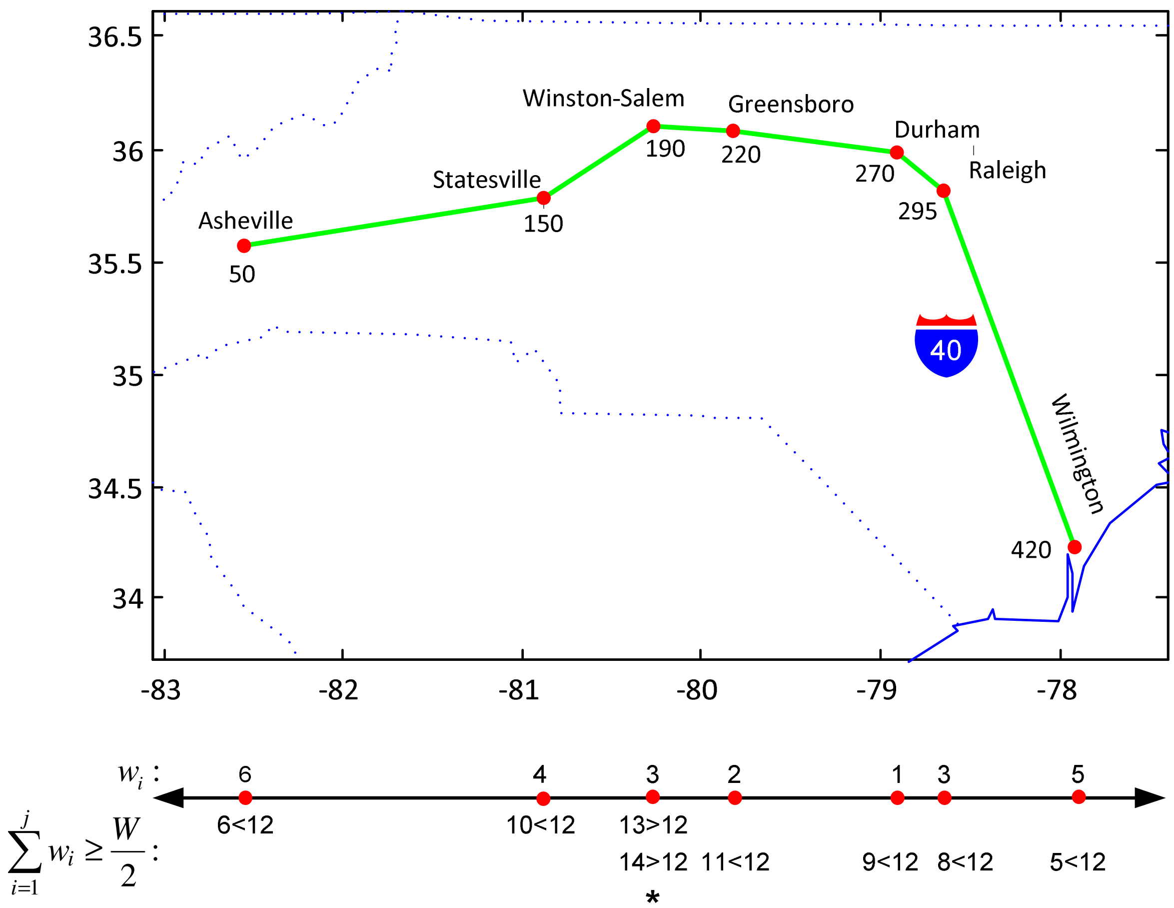

```
w = [3, 5, 4, 2]
p = [22, 5, 82, 30]
idx = sortperm(p)
p,w = p[idx],w[idx]
idx = findfirst(cumsum(w) .>= sum(w)/2)
idx, p[idx]
```

```
(2, 22)
```

NL Opt: Q3

Median location: For each dimension x of X:

1. Order EFs so that $|x_1| \le |x_2| \le \cdots \le |x_m|$
2. Locate x-dimension of NF at the first EFj where $\sum_{i=1}^{j} w_i \ge \frac{W}{2}$, where $W = \sum_{i=1}^{m} w_i$



NL Opt: Q4

Determine the minimum value of the function $x - x^3$ between -1.5 and $\sqrt{1/3}$ with a tolerance of 0.25 using Golden Section Search.

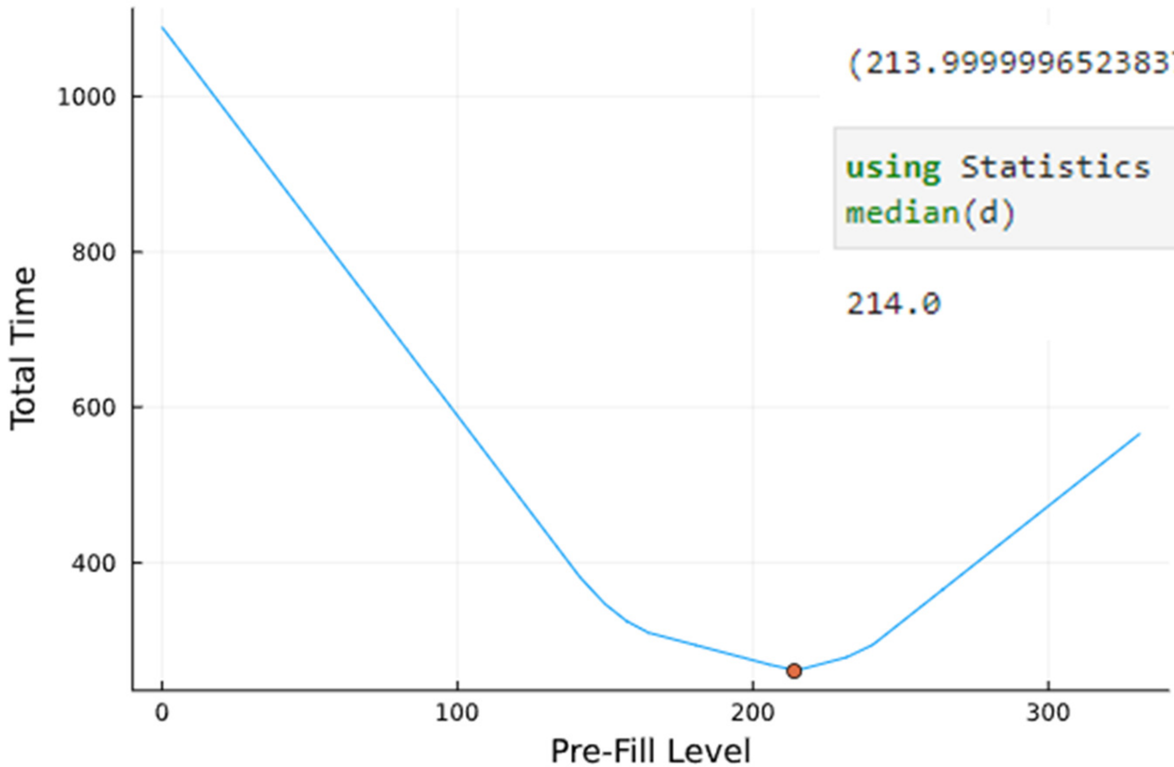
```
function mygoldensearch(n,LB,UB)
    x1 = LB + ((3 - sqrt(5))/2)*(UB - LB)
    x2 = LB + ((sqrt(5) - 1)/2)*(UB - LB)
    println(n,":",LB,",",x1,",",x2,",",UB)
    if abs(x1 - x2) <= 0.25
        return xminest = (x1+x2)/2
    elseif f(x1) < f(x2)
        UB = x2
    else
        LB = x1
    end
    return (LB,UB)
end
```

```
f(x) = x - x^3
LB,UB = -1.5,sqrt(1/3)
done,n,res = false,0,Inf
while !done
    n += 1
    res = mygoldensearch(n,LB,UB)
    if length(res) == 2
        LB,UB = res
    else
        println("Estimated minimum = ", res)
        done = true
    end
end
```

```
1:-1.5,-0.7065228037083066,-0.2161269271020676,0.5773502691896257
2:-1.5,-1.0096041233937614,-0.7065228037083062,-0.2161269271020676
3:-1.0096041233937614,-0.7065228037083064,-0.5192082467875225,-0.2161269271020676
Estimated minimum = -0.6128655252479145
```

NL Opt: Q5

submerge a part. The following demand data is available regarding the cubic volume of solution that was needed to test recent parts: 237, 214, 161, 146, and 331. Determine the cubic volume of solution that should be pre-filled in the tank to minimize the time required to submerge the part, given that the tank can be filled or drained at the same rate.




```
(213.99999965238376, 261.00000034761626)
```

```julia
using Statistics
median(d)
```

```
214.0
```

NL Opt: Q6

A new vaccine fulfillment center has been established that has the capacity to serve two customers. The dispensing and sterile packaging equipment can only fill one common-size container per customer because the set-up time to change the container size is very time-consuming. A single 235-unit container will be used for each customer's demand, any demand beyond the container capacity will be lost. The revenue received per unit of demand is $5 and the cost to provide each unit is $1. Once opened, each container must be used over a short period of time, and since the customers are dispersed geographically, it is not possible for them to share the vaccine. Determine the total profit given demands of 247 and 214 units per customer.

```
d = [247, 214]
p = 5                                   # Unit price
c = 1                                   # Unit cost
fπ(q,di) = p*minimum([q, di]) - c*q     # Profit (fπ ⇒ f\pi<TAB>)
fΠ(q,d) = sum([fπ(q,di) for di in d])   # Total profit (fΠ ⇒ f\Pi<TAB>)
fΠ(235,d)
```

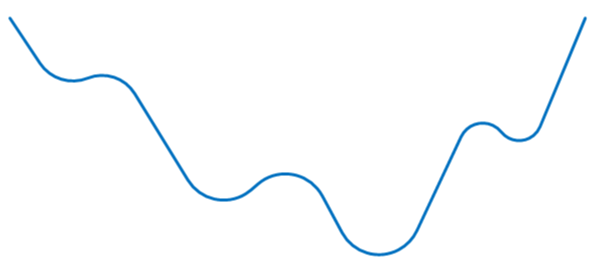
```
1775
```

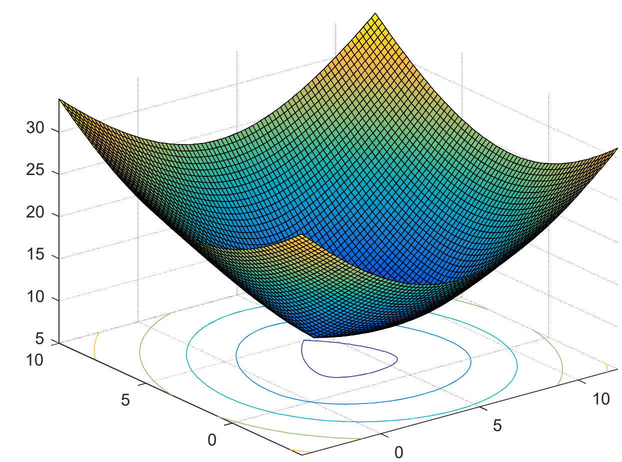
NL Opt: Q7-9

Explain how Brent's method is able to converge to the optimum solution faster than interval search.

Under what circumstances is using a loss function for regression that does not allow full information recovery a good feature?

In HW2-Q2, why was it necessary to use several different starting points to find your solution?

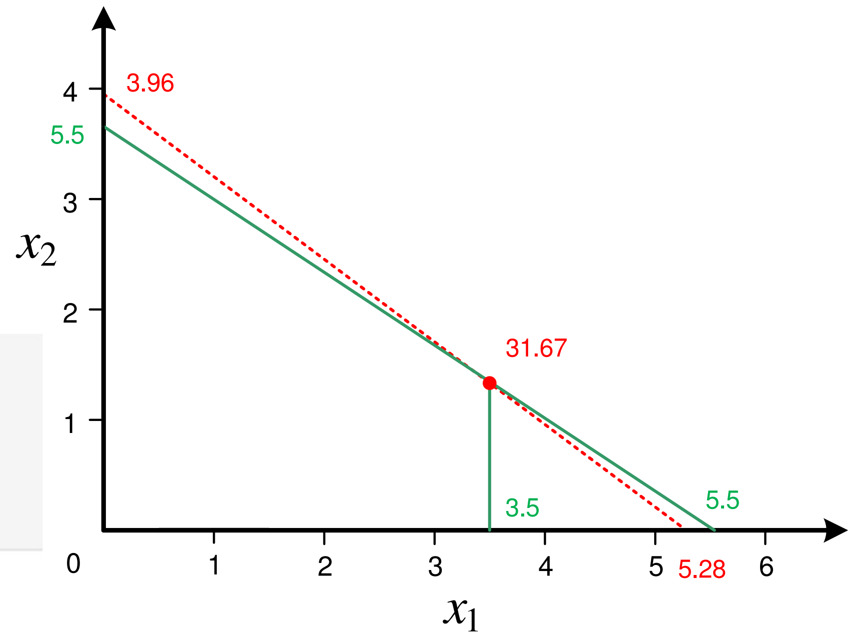




LP: Q10

Draw the feasible region of the LP below along with the line of the objective passing through the optimal point.

$$\begin{aligned} \max\ & 6x_1 + 8x_2 \\ \text{s.t.}\ & 2x_1 + 3x_2 \le 11 \\ & 2x_1 \le 7 \\ & x_1, x_2 \ge 0 \\ & x_1, x_2 \text{ integer} \end{aligned}$$




```
@show x11 = 11/2
@show x21 = 11/3
@show x12 = 7/2
x212 = (11 - 2(7/2))/3
x112 = (11 - 3x212)/2
f(x1,x2) = 6x1 + 8x2
@show f(0,0), f(0,x21), f(x112,x212), f(x12,0)
```

```
x11 = 11 / 2 = 5.5
x21 = 11 / 3 = 3.6666666666666665
x12 = 7 / 2 = 3.5
(f(0, 0), f(0, x21), f(x112, x212), f(x12, 0)) = (0, 29.333333333333332, 31.6666
21.0)
(0, 29.333333333333332, 31.666666666666664, 21.0)
```

```
@show x10 = f(x112,x212)/6
x20 = f(x112,x212)/8
```

```
x10 = f(x112, x212) / 6 = 5.277777777777778
3.9583333333333333
```

LP: Q11-13

In Ex 4 of LP-3, why were continuous decision variables used to represent the number of employees even though each employee is a discrete person?

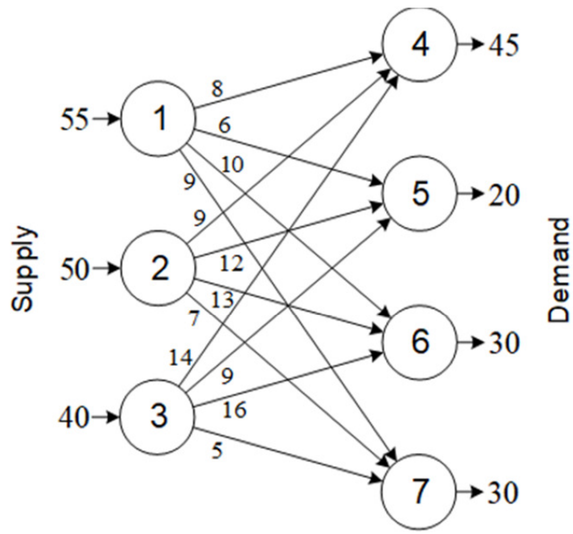
Explain why the simplex method always terminates.

Given an LP with the decision variables x_1, and x_2 representing the production in tons of products 1 and 2, respectively, determine a constraint that can represent the requirement that product 1 must represent at least a quarter of total production.

$$x_1 \geq \frac{x_1 + x_2}{4}$$

Net Mod: Q14(a)

(a) Determine the solution returned by the following greedy procedure.




```
function greedytrans(C,s,d)
    s, d = copy(s), copy(d)
    F = zeros(size(C))
    while !all(d .== 0)
        idxs = findall(s .!= 0)    # Available supply
        idxd = findall(d .!= 0)    # Unmet demand
        idx = argmin(C[idxs,idxd])
        # Min of avail supply and unmet demand
        f = min(s[idxs[idx[1]]],d[idxd[idx[2]]])
        F[idxs[idx[1]],idxd[idx[2]]] = f
        s[idxs[idx[1]]] -= f
        d[idxd[idx[2]]] -= f
    end
    TC = sum(C.*F)
    return (F=F, TC=TC)
end
```

```
res = greedytrans(C,s,d)
Matrix([res.F s; d' res.TC])
```

```
4×5 Matrix{Float64}:
 35.0  20.0   0.0   0.0    55.0
 10.0   0.0  30.0   0.0    50.0
  0.0   0.0   0.0  30.0    40.0
 45.0  20.0  30.0  30.0  1030.0
```

Trans	4	5	6	7	Supply	
1	8	6	10	9	55	-20 = 35 -35 = 0
2	9	12	13	7	50	-10 = 40 -30 = 10
3	14	9	16	5	40	-30 = 10
Demand	~~45~~	~~20~~	~~30~~	~~30~~		
	~~10~~ 0	0	0	0		

$$TC = 5(30) + 6(20) + 8(35) + 9(10) + 13(30) = 1{,}030 \quad (\text{vs } 970 \text{ optimal})$$

Net Mod: Q14(b)

(b) Explain why the supply at node 3 can increase without affecting the solution.

Trans	4	5	6	7	Supply
1	8	6	10	9	55 -20 = 35 -35 = 0
2	9	12	13	7	50 -10 = 40 -30 = 10
3	14	9	16	5	40 -30 = 10
Demand	~~45~~	~~20~~	~~30~~	~~30~~	
	~~10~~ 0	0	0	0	

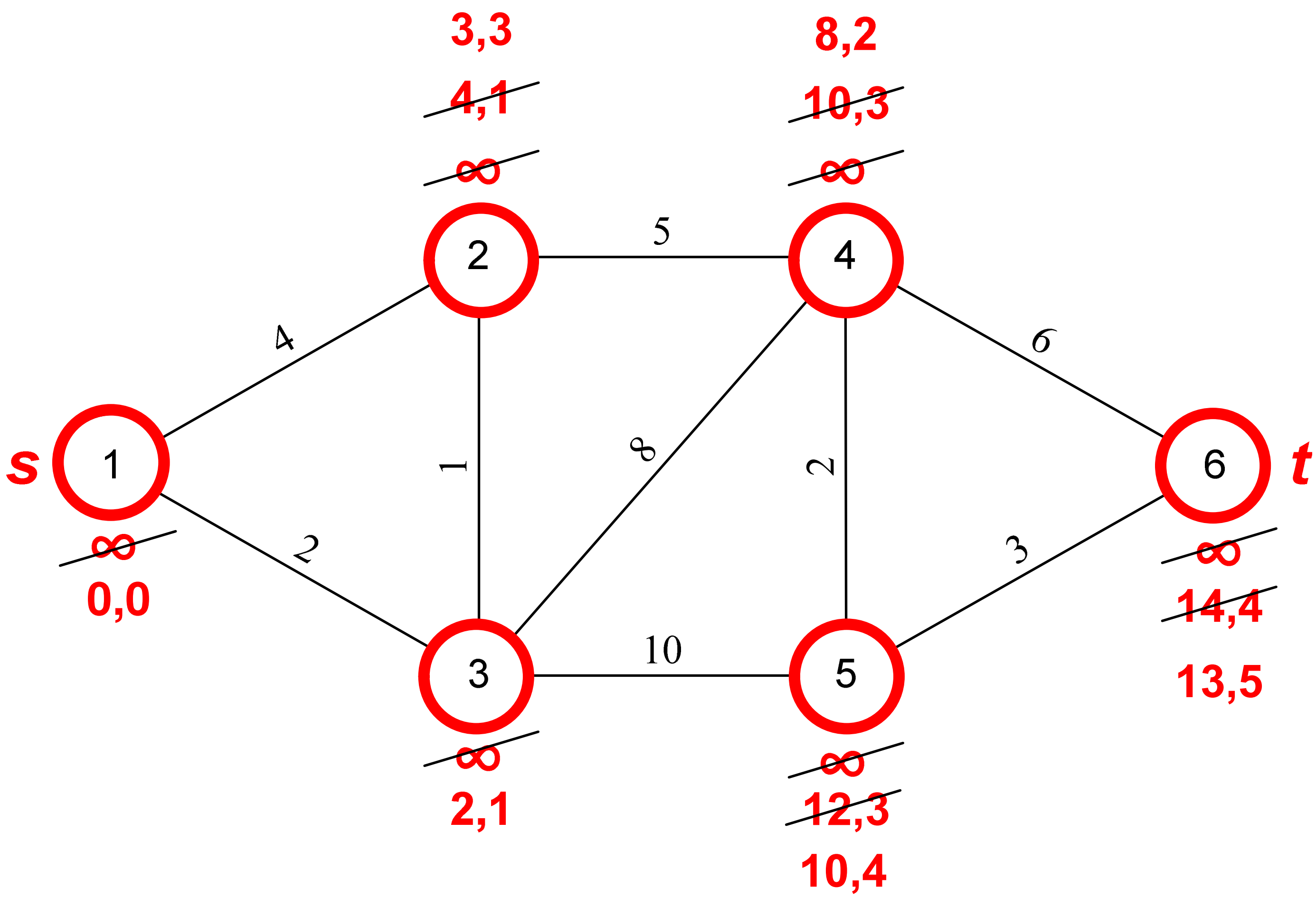
Net Mod: Q14(c)

(c) Explain why copies of the supply and demand arrays, `s,d = copy(s),copy(d)`, are made inside of the procedure, but a copy of `C` is not made.

```
C = [8 6 10 9; 9 12 13 7; 14 9 16 5]
d = [45, 20, 30, 30]
s = [55, 50, 40]
function greedytrans(C,s,d)
    s, d = copy(s), copy(d)
    F = zeros(size(C))
    while !all(d .== 0)
        idxs = findall(s .!= 0)   # Available supply
        idxd = findall(d .!= 0)   # Unmet demand
        idx = argmin(C[idxs,idxd])
        # Min of avail supply and unmet demand
        f = min(s[idxs[idx[1]]],d[idxd[idx[2]]])
        F[idxs[idx[1]],idxd[idx[2]]] = f
        s[idxs[idx[1]]] -= f
        d[idxd[idx[2]]] -= f
    end
    TC = sum(C.*F)
    return (F=F, TC=TC)
end
res = greedytrans(C,s,d)
Matrix([res.F s; d' res.TC])
```

Net Mod: Q15

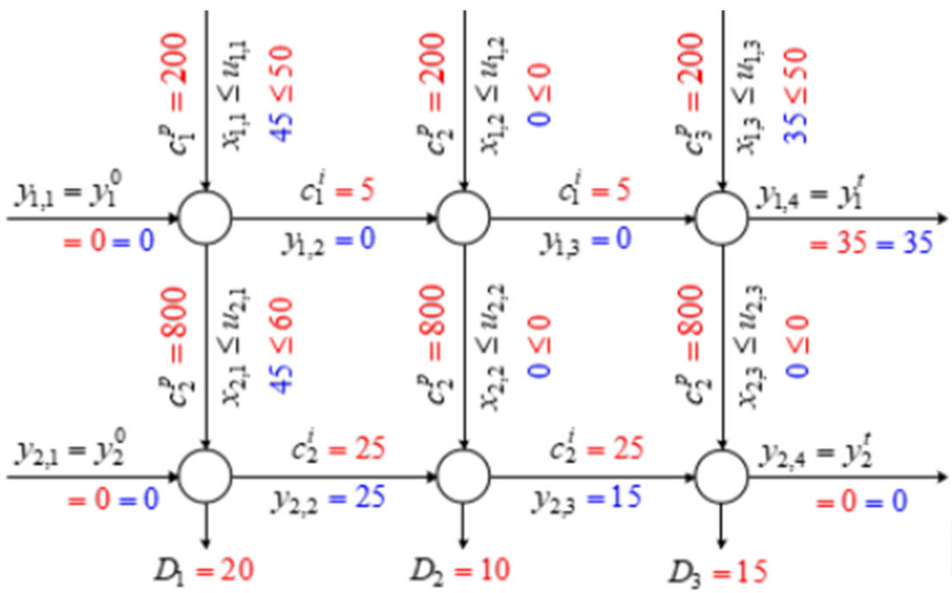
The cost of each arc is shown in the network below. Use Dijkstra's procedure to determine the least cost path from node 1 to node 6.

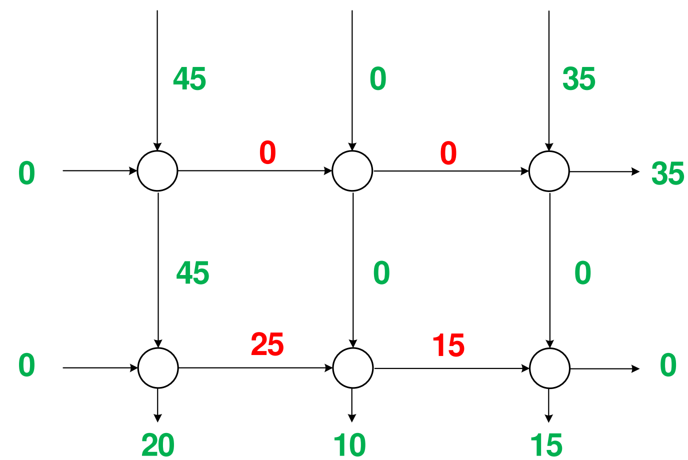


Path: $1 \leftarrow 3 \leftarrow 2 \leftarrow 4 \leftarrow 5 \leftarrow 6:\quad 13$

Net Mod: Q16

A single product is produced in two stages. Determine the total production and inventory cost over the three month planning horizon. The forecasted demand is 20, 10, and 15 tons per month. Inventory costs are \$5 and \$25 per ton per month, respectively, and production costs are \$200 and \$800 per ton, respectively. Each month, 45, 0, 35 and 45, 0, 0 tons, respectively, are produced at each stage. Initially, no inventory is in storage at both stages, and 35 and 0 tons are in storage for stage 1 and 2, respectively, at the end of three months.






```
TCp, TCi = sum(sum(cp.*X°)), sum(sum(ci.*Y°[:,2:end]))
```

```
(52000.0, 1175.0)
```

```
TC = TCp + TCi
```

```
53175.0
```

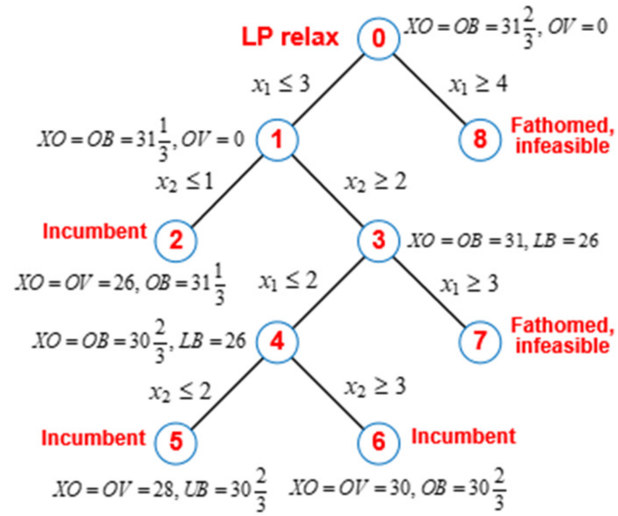
Net Mod: Q17-18

In a multi-period production-inventory flow network, why is it incorrect to include the cost of carrying both the initial and final inventory in the total inventory cost calculation?

Since the network model is run for each period as part of a rolling horizon, including both the initial and the final inventory cost in the total cost would double count the inventory cost.

Explain how it is possible to force an arc to be used in a shortest path determination.

MIP: Branch and Bound



XO: Objective value of LP relaxation with branch constraints.
OV: ILP objective value.
OB: Best objective bound.

$$\begin{aligned} \text{Maximize} \quad & 6x_1 + 8x_2 \\ \text{subject to} \quad & 2x_1 + 3x_2 \le 11 \\ & 2x_1 \le 7 \\ & x_1, x_2 \ge 0, \text{ integer} \end{aligned}$$

Branch and Bound for ILP

1. (Initial Relaxation) Solve LP relaxation of problem.
   a. If relaxation solution is feasible (all integer variables), stop; otherwise, use *XO* as *OB*.
   b. If heuristic solution available, use as *OV*; otherwise, use $\pm\infty$ as *OV* for min/max.
2. (Branching) Pick non-integer variable, add left ($\le$) and right ($\ge$) branch nodes from current node.
3. (Node Selection) If all branch nodes are incumbent or fathomed, then stop; otherwise, select node not incumbent or fathomed and solve LP with branch constraints.
   a. If feasible: If *XO* improves *OV*, then use *XO* as *OV* and node becomes incumbent; otherwise, node fathomed. Repeat node selection.
   b. If LP feasible but not integer feasible: If *XO* improves *OB*, then use *XO* as *OB*; then branch (step 2) from current node.
   c. If not LP feasible, then node fathomed. Repeat node selection.

MIP: Q19

Determine the relative and absolute gap associated with an ILP objective value of 26 and a best objective bound of 31.00.

$$\text{Relative gap: } \frac{|OV - OB|}{OV} = \frac{|26 - 31.00|}{26} = 0.1923$$

$$\text{Absolute gap: } |OV - OB| = |26 - 31.00| = 5.00$$

MIP: Q20

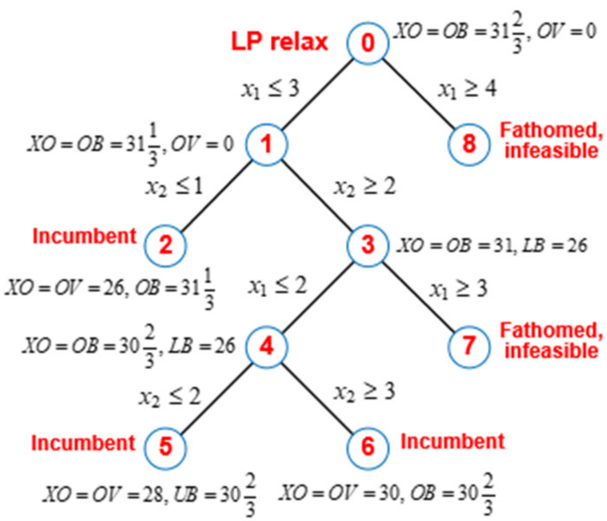
For the ILP shown below, the following table represents the LP relaxation solutions found by adding branch constraints. Each row represents the solution found after adding the less-than-or-equal and greater-than-or-equal branch constraints on the two decision variables, with the right-hand-side value of each constraint listed.

(a) Determine a sequence of nodes for a branch-and-bound tree that solves the ILP, listing, for each node, its predecessor node in the tree and the objective of the LP relaxation (XO), ILP objective value (OV), and the best objective bound (OB). (Note: not all rows need to be included in the tree.)

(b) Determine what the solution would be (i.e., the ILP objective and decision variable values) if the brand-and-bound procedure stopped after its relative gap (Gap) was within 10%.

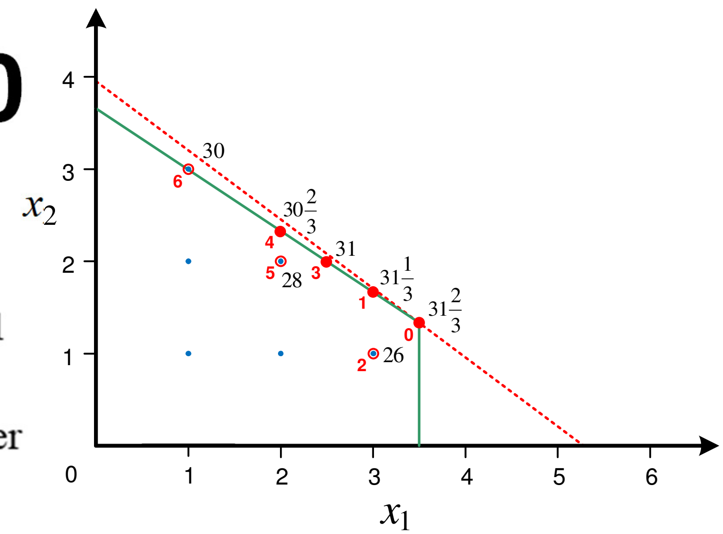
$$\begin{aligned} \text{Maximize} \quad & 6x_1 + 8x_2 \\ \text{subject to} \quad & 2x_1 + 3x_2 \le 11 \\ & 2x_1 \le 7 \\ & x_1, x_2 \ge 0, \text{ integer} \end{aligned}$$

	Node	Pred	$x_1 \le$	$x_1 \ge$	$x_2 \le$	$x_2 \ge$	x_1	x_2	*XO*	*OV*	*OB*	*Gap*
1			2			2	2.00	2.33	30.67			
2			3		1		3.00	1.00	26.00			
3			2			3	1.00	3.00	30.00			
4			1				1.00	3.00	30.00			
5				4					Infeas			
6						3	1.00	3.00	30.00			
7			3	3		2			Infeas			
8							3.50	1.33	31.67			
9			3			2	2.50	2.00	31.00			
10				1			3.50	1.33	31.67			
11			2		2	2	2.00	2.00	28.00			
12			2	3					Infeas			
13			3				3.00	1.67	31.33			
14			1			3	1.00	3.00	30.00			
15			2		1		2.00	1.00	20.00			
16					3		3.50	1.33	31.67			



MIP: Q20

$$
\begin{aligned}
\text{Maximize} \quad & 6x_1 + 8x_2 \\
\text{subject to} \quad & 2x_1 + 3x_2 \le 11 \\
& 2x_1 \le 7 \\
& x_1, x_2 \ge 0, \text{integer}
\end{aligned}
$$



	Node	Pred	$x_1 \le$	$x_1 \ge$	$x_2 \le$	$x_2 \ge$	x_1	x_2	*XO*	*OV*	*OB*	*Gap*
1	4	3	2			2	2.00	2.33	30.67	26	30.67	17.95
2	2	1	3		1		3.00	1.00	26.00	26	31.33	20.51
3	6	4	2			3	1.00	3.00	30.00	30	30.67	2.22
4			1				1.00	3.00	30.00			
5	8	0		4					Infeas			
6						3	1.00	3.00	30.00			
7	7	3	3	3		2			Infeas			
8	0						3.50	1.33	31.67	0	31.67	Inf
9	3	1	3			2	2.50	2.00	31.00	26	31.00	19.23
10				1			3.50	1.33	31.67			
11	5	4	2		2	2	2.00	2.00	28.00	28	30.67	9.52
12			2	3					Infeas			
13	1	0	3				3.00	1.67	31.33	0	31.33	Inf
14			1			3	1.00	3.00	30.00			
15			2		1		2.00	1.00	20.00			
16					3		3.50	1.33	31.67			

MIP: Q21

Given the LP feasible solution to an ILP of $x_1 = 3.62$, $x_2 = 4.00$, and $x_3 = 7.12$, determine, as part of a branch and bound procedure, which variable would be selected using (a) the most infeasible branching rule and (b) the least infeasible branching rule.

```
# Determine non-integer variables
x₁,x₂,x₃ = 3.62,4.00,7.12
idx = findall([!isapprox(x,round(x)) for x in [x₁,x₂,x₃]])
println("Non-integer variables: ", idx)
```

```
Non-integer variables: [1, 3]
```

```
# (a) Most infeasible non-integer variable has the largest minimal fractional component
(ceil(x₁)-x₁, x₁-floor(x₁), ceil(x₃)-x₃, x₃-floor(x₃))
```

```
(0.3799999999999999, 0.6200000000000001, 0.8799999999999999, 0.1200000000000001)
```

```
(min(ceil(x₁)-x₁, x₁-floor(x₁)), min(ceil(x₃)-x₃, x₃-floor(x₃)))
```

```
(0.3799999999999999, 0.1200000000000001)
```

```
println("Most infeasible variable: ",
    idx[argmax([min(ceil(x₁)-x₁, x₁-floor(x₁)), min(ceil(x₃)-x₃, x₃-floor(x₃))])])
```

```
Most infeasible variable: 1
```

```
# (b) Least infeasible non-integer variable has the largest maximal fractional component
(max(ceil(x₁)-x₁, x₁-floor(x₁)), max(ceil(x₃)-x₃, x₃-floor(x₃)))
```

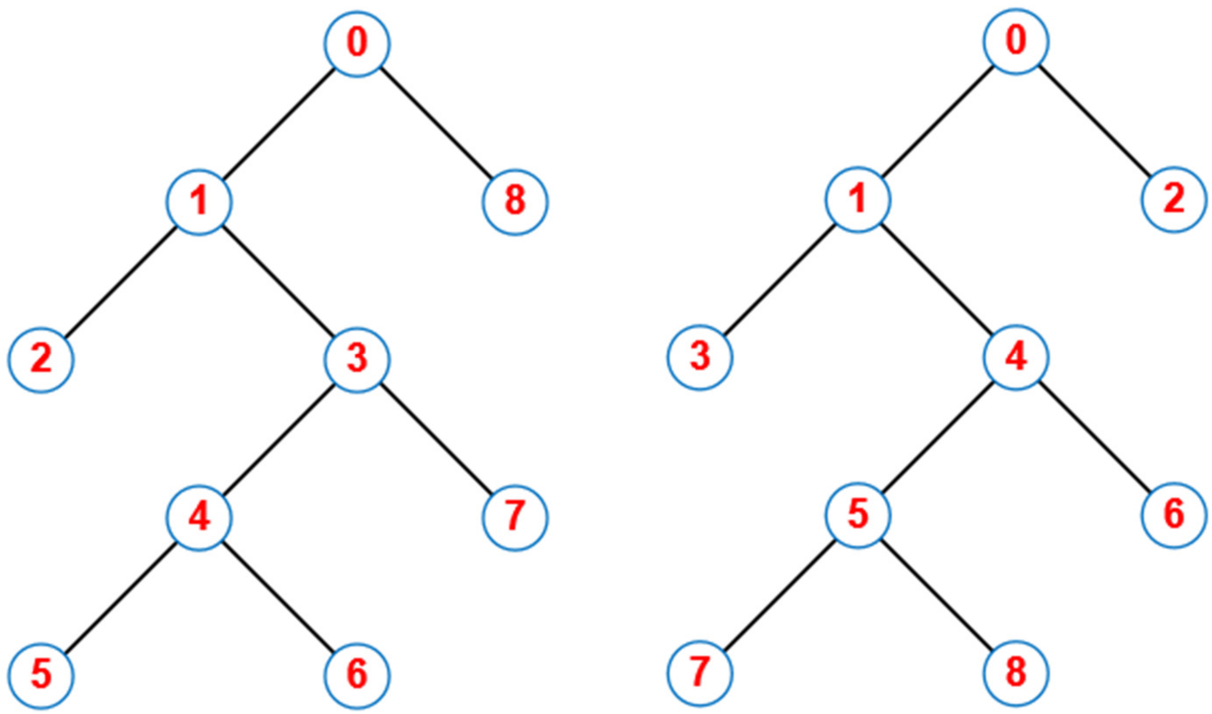
```
(0.6200000000000001, 0.8799999999999999)
```

```
println("Least infeasible variable: ",
    idx[argmax([max(ceil(x₁)-x₁, x₁-floor(x₁)), max(ceil(x₃)-x₃, x₃-floor(x₃))])])
```

```
Least infeasible variable: 3
```

MIP: Q22

In the branch and bound graphs below, the top node is labeled 0 and corresponds to the initial relaxation. (a) In the left graph, find a labeling of the remaining nodes assuming that node selection is done depth first. (b) In the right graph, find a labeling of the remaining nodes assuming that node selection is done breadth first.



MIP: Q23

The decision variables x_1, x_2, and x_3 represent whether an activities 1–3, respectively, occur in a BIP, with 1 indicating occurrence.

(a) Add a constraint to the BIP to ensure that activity 1 occurs only when activity 2 or 3 occurs.

```
x₁ >= x₂ && x₁ >= x₃     # If Y or Z then X
```

(b) Add a constraint to ensure the occurrence of at least one of the activities.

```
x₁ + x₂ + x₃ >= 1    # Select at least one (covering constraint)
```

MIP: Q24

The decision variable *x* represents the amount of a bulk commodity in a MILP. Add any additional variables and constraints to the MILP to ensure that, if any of the commodity is used, the amount must be between five and twenty.

```
@variable(model, y, Bin )           # Add binary decision variable
@constraint(model, x >= 5y )        # Switch on LB only if y = 1
@constraint(model, x <= 20y )       # Force y = 1 if x > 0 (where 20 is max x UB)
```

MIP: Q25

A single product is produced in a two-stage production process. Stage one has a capacity to produce 40 tons of the product per month, and there is a fixed cost of $6000 per month and a variable cost of $200 for each ton of product produced. For stage two, capacity is 80 tons per month, and there is a fixed cost of $18,000 and a variable cost of $800. The fixed costs are only incurred when any amount of production occurs at that stage for that month. Assuming that the annual inventory carrying rate is 0.4 $/$-yr, determine the inventory cost for each stage.

```
u = [40, 80]                  # capacity of each stage (ton)
cp = [200, 800]               # variable production cost for each stage ($/ton)
ck = [6000, 18000]            # fixed production cost for each stage ($/month)
h = 0.4/12                    # monthly inventory carrying rate (1/month)
ci = cumsum(cp + ck./u)*h     # inventory cost for each stage ($/ton)
```

```
2-element Vector{Float64}:
 11.666666666666666
 45.833333333333336
```

MIP: Q26

Explain how an implementation of the following mathematical programming formulation was used in the solution of one of the questions in HW 5.

$$
\begin{aligned}
\text{Minimize} \quad & \sum_{k \in N} y_k \\
\text{subject to} \quad & \sum_{k \in N} x_{ik} = 1, && i \in M \\
& x_{ik} + x_{jk} \leq 1, && (i,j) \in W; k \in N \\
& x_{ik} \leq y_k, && i \in M, k \in N \\
& y_i \in \{0,1\}, && i \in M \\
& x_{ij} \in \{0,1\}, && i \in M; j \in M
\end{aligned}
$$

```
# Problem 3b, graph coloring to determine minimum number of different frequencies needed
M = 1:size(W,1)
@variable(model, y[1:length(M)], Bin )
@variable(model, X[1:length(M),1:length(M)], Bin )
@objective(model, Min, sum(y[i] for i ∈ M ))
@constraint(model, [i ∈ M], sum(X[i,k] for k ∈ M) == 1 )
@constraint(model, [(i,j) ∈ zip(b,e), k ∈ M], X[i,k] + X[j,k] <= 1 )
@constraint(model, [i ∈ M, k ∈ M], X[i,k] <= y[k] )
```

MIP: Q27-30

27. Explain how the use of a Gomory cut can speed up the solution of an ILP.

28. Why does the per-unit cost of carrying inventory usually increase at each stage in a multi-stage production-inventory flow network, while it remains constant across each period in a stage?

It increases at each stage in the network because each stage usually adds value to the product, while the decrease in value associated with carrying inventory for a period is usually the same for each period.

29. In the solution for Ex 1 of MIP-4,

(a) Why was production cost separated into variable and fixed components in the model?

So that the economies of scale associated with production at each stage could be included in the model (an alternate approach would be to use a piecewise linear approximation to the concave production costs).

(b) How was the fact that only 30 tons of product can be stored at each stage included in the model?

The storage constraint is included in the model as an upper bound on the amount of inventory carried across an arc.

30. In the solution for Ex 2 of MIP-4, why was production cost not included in the objective function?

Comb Opt: Q31

A couple is renovating three different rental properties that have just become vacant. They would like to finish the renovation as soon as possible and have received quotes for the work at each property to be completed next week from three different contractors. The first contractor has quoted 4, 2, and 5 thousand dollars for properties 1–3, respectively; the second contractor, 3, 4, and 4 thousand dollars; and the third contractor, 5, 5, and 8 thousand dollars. Determine which contractor should work on which property.

```
using Combinatorics
C = [4 2 5; 3 4 4; 5 5 8]
n = size(C,1)
TC°, α° = Inf, []
for α in permutations(1:n)
    TC = sum([C[i,j] for (i,j) in enumerate(α)])
    if TC < TC°
        TC°, α° = TC, α
    end
    println(α,": ",TC)
end
```

```
[1, 2, 3]: 16
[1, 3, 2]: 13
[2, 1, 3]: 13
[2, 3, 1]: 11
[3, 1, 2]: 13
[3, 2, 1]: 14
```

```
for i in 1:n println("Assign contractor ", string(i), " to property ", string(α°[i])) end
println("Total cost = \$", string(TC°), ",000")
```

```
Assign contractor 1 to property 2
Assign contractor 2 to property 3
Assign contractor 3 to property 1
Total cost = $11,000
```

Comb Opt: Q32

Using the *firstfit* procedure, assign samples of sizes 3, 6, 2, 5, 4, 4, 2, 4, 4, 2, 4, and 5 to containment devices in the given sequence. Each device has a size capacity of eight.

```
function firstfit(v,V)
    # Put first object into first bin
    B = [[1]]                                  # Array of arrays
    b = [v[1]]                                 # Array of scalars
    # Add remaining objects to bins
    for i = 2:length(v)
        idx = findfirst(b .+ v[i] .<= V)       # Find first bin object fits into
        if ~isnothing(idx)                     # Found existing bin
            b[idx] += v[i]
            push!(B[idx],i)
        else                                   # Create new bin for object
            push!(b,v[i])
            push!(B,[i])
        end
    end
    return B
end
v = [3, 6, 2, 5, 4, 4, 2, 4, 4, 2, 4, 5]
V = 8
firstfit(v,V)
```

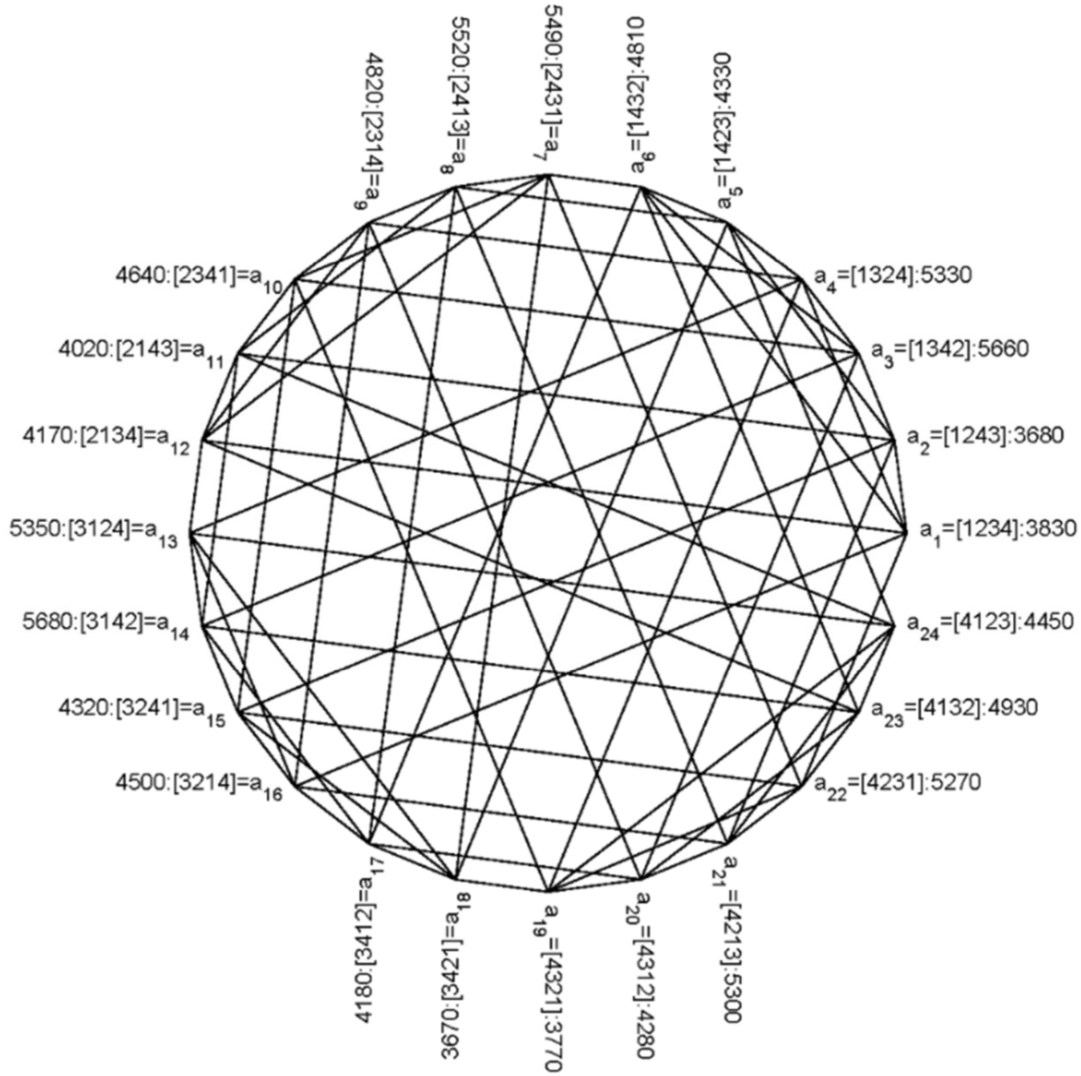
```
7-element Vector{Vector{Int64}}:
 [1, 3, 7]
 [2, 10]
 [4]
 [5, 6]
 [8, 9]
 [11]
 [12]
```

Comb Opt: Q33

Starting with an initial assignment array of [2, 3, 1, 4], use the following pairwise interchange graph to determine the final assignment that would be found by the *SDPI* procedure:

```
sdpi([2,3,1,4],W,D).TC
```

```
4820.0: [2, 3, 1, 4]
4170.0: [2, 1, 3, 4]
3830.0: [1, 2, 3, 4]
3680.0: [1, 2, 4, 3]
3680.0: [1, 2, 4, 3]
```



Comb Opt: Q34

Estimate the expected total cost associated with randomly assigning twenty resources to sites in a layout problem if the cost c_{ijkl} of assigning resource i to site k if resource j is assigned to site l is uniformly distributed between 0 and 1.

```
n = 20
TCexp = (0.5)*n^2
```

```
200.0
```

Comb Opt: Q35

A hospital has four identical operating rooms available. Three different types of operations will be performed that each takes 50, 30, and 150 minutes, respectively, on average, and there are ten, fifteen, and six operations of each type, respectively, planned for tomorrow. Determine a lower bound on the maximum number of hours that it will take to complete all of the operations, assuming that it takes fifteen minutes to switch between each operation in the operating room no matter the type of operation or the type of the preceding operation.

```
toper = [50, 30, 150]              # Operation type time (min)
t = toper .+ 15                    # Operation and switch time (min)
f = [10, 15, 6]                    # Operation frequency (oper/day)
m = 4                              # Number of identical operating rooms
@show t .* f
@show sum(t .* f)
LB = (sum(t .* f)/m)/60            # LB on max time (hr)
```

```
t .* f = [650, 675, 990]
sum(t .* f) = 2315

9.645833333333334
```

Comb Opt: Q36

Given two identical resources and four independent tasks with times of 1, 2, 3, and 4, respectively, that have initially been allocated to resources 1, 1, 2, and 2, respectively, explain why, in this instance, the new allocation that is found by using the *relocate* procedure minimizes the maximum time required to complete all of the tasks.

```
m = 2                           # Number of resources
n = 4                           # Number of tasks
t = [1:n;]                      # Time of each task
displaytasktimes([1,1,2,2],t)   # Inital allocation, makespan = 7
```

```
3: [1, 2]
7: [3, 4]
```

```
displaytasktimes([1,1,1,2],t)   # Task 2 relocated to resource 1, makespan = 6
```

```
6: [1, 2, 3]
4: [4]
```

```
displaytasktimes([2,1,1,2],t)   # Task 1 relocated to resource 2, makespan = 5
```

```
5: [2, 3]
5: [1, 4]
```

```
LB = sum(t)/m                   # Makespan found by relocate equals LB => optimal
```

```
5.0
```

ML: Q37

Determine the cube per order for the following item master and order dataset inline CSV files, where any missing SKU data should be dropped and any missing QTY data should be imputed with a representative value:

```
SKU,LENGTH,WIDTH,DEPTH,CUBE,WEIGHT,UOM
1,5,3,2,30,1.25,EA
2,3,2,4,24,4.75,EA
3,8,6,5,180,9.65,CA
4,4,4,3,32,6.35,EA
```

```
ORDER,SKU,QTY
1,1,
1,2,3
1,3,2
1,4,6
2,1,4
2,3,1
3,1,2
3,,
4,2,2
5,,1
```

ML: Q37

```
ORDER,SKU,QTY
1,1,
1,2,3
1,3,2
1,4,6
2,1,4
2,3,1
3,1,2
3,,
4,2,2
5,,1
```

	ORDER	SKU	QTY
	Int64	Int64?	Int64?
1	1	1	*missing*
2	1	2	3
3	1	3	2
4	1	4	6
5	2	1	4
6	2	3	1
7	3	1	2
8	3	*missing*	*missing*
9	4	2	2
10	5	*missing*	1

```
dropmissing!(df0,:SKU)
```

8 rows × 3 columns

	ORDER	SKU	QTY
	Int64	Int64	Int64?
1	1	1	*missing*
2	1	2	3
3	1	3	2
4	1	4	6
5	2	1	4
6	2	3	1
7	3	1	2
8	4	2	2

```
using Statistics
idx = findall(ismissing.(df0.QTY))
for i in idx
    df0[i,:QTY] = Int(round(mean(skipmissing(
                        df0.QTY[df0.SKU.==df0[i,:SKU]]))))
end
df0
```

8 rows × 3 columns

	ORDER	SKU	QTY
	Int64	Int64	Int64?
1	1	1	3
2	1	2	3
3	1	3	2
4	1	4	6
5	2	1	4
6	2	3	1
7	3	1	2
8	4	2	2

ML: Q37

	SKU	LENGTH	WIDTH	DEPTH	CUBE	WEIGHT	UOM
	Int64	Int64	Int64	Int64	Int64	Float64	String
1	1	5	3	2	30	1.25	EA
2	2	3	2	4	24	4.75	EA
3	3	8	6	5	180	9.65	CA
4	4	4	4	3	32	6.35	EA

```
gdf = groupby(df,:ORDER)
[sum(i.QTY.*i.CUBE) for i in gdf]
```

```
4-element Vector{Int64}:
 714
 300
  60
  48
```

```
Cube_per_Order = mean([sum(i.QTY.*i.CUBE) for i in gdf])
```

```
280.5
```

	ORDER	SKU	QTY
	Int64	Int64	Int64?
1	1	1	3
2	1	2	3
3	1	3	2
4	1	4	6
5	2	1	4
6	2	3	1
7	3	1	2
8	4	2	2

```
df = innerjoin(dfO,dfIM,on=:SKU)
```

8 rows × 9 columns

	ORDER	SKU	QTY	LENGTH	WIDTH	DEPTH	CUBE	WEIGHT	UOM
	Int64	Int64	Int64?	Int64	Int64	Int64	Int64	Float64	String
1	1	1	3	5	3	2	30	1.25	EA
2	1	2	3	3	2	4	24	4.75	EA
3	1	3	2	8	6	5	180	9.65	CA
4	1	4	6	4	4	3	32	6.35	EA
5	2	1	4	5	3	2	30	1.25	EA
6	2	3	1	8	6	5	180	9.65	CA
7	3	1	2	5	3	2	30	1.25	EA
8	4	2	2	3	2	4	24	4.75	EA

ML: Q38

Demand is being estimated by a model that minimizes the *L1* loss function. Given the demand data in the array d given below along with the following partition of the data into training and test sets, determine the root mean squared error of the model:

```
using MLJ
d = [237, 214, 161, 146, 331, 159, 423, 332, 139, 327, 152, 98, 116]
train, test = partition(eachindex(d), 0.7, shuffle=true, rng=123)
```

```
([1, 11, 6, 12, 7, 4, 13, 8, 9], [10, 2, 3, 5])
```

```
using Optim
fd̂(α) = α[1]
fL1(α) = sum(abs.(fd̂(α) .- d))
α1° = optimize(α -> fL1(α), minimum(d), maximum(d)).minimizer
```

```
161.00000039485334
```

```
mypredict(d) = median(d)
ŷ = mypredict(d)
```

```
161.0
```

```
ŷ, y = mypredict(d[train]), mypredict(d[test])
```

```
(152.0, 270.5)
```

```
res = ŷ .- y
```

```
-118.5
```

```
rms = sqrt(mean(res.^2))
```

```
118.5
```

ML: Q39

Five email messages are available to train a spam filter. Two features have been identified as being the most useful for detecting spam. Two of the emails are spam and have values of 4 and 6 for feature one and 8 and 3 for feature two, respectively. The three other messages are not spam and have values of 1, 5, and 8 and 1, 6, and 4 for features one and two, respectively. Determine the information gain associated with using 7 as the first threshold for feature two using the *CART* procedure in creating a decision tree for spam detection.

```
impurity(t) = 1 - sum(([sum(t .== i) for i in unique(t)]./length(t)).^2)
function infogain(s,f,t)
    is = f .<= s
    IG = impurity(t) - impurity(t[is])*length(f[is])/length(f) -
        impurity(t[.!is])*length(f[.!is])/length(f)
end
t = [0,1,0,1,0] .>= .5
f2 = float([1,8,6,3,4])
s = 7
is = f2 .<= s
impurity(t), 1 - (3/5)^2 - (2/5)^2
```

```
(0.48, 0.48)
```

```
impurity(t[is]), 1 - (3/4)^2 - (1/4)^2
```

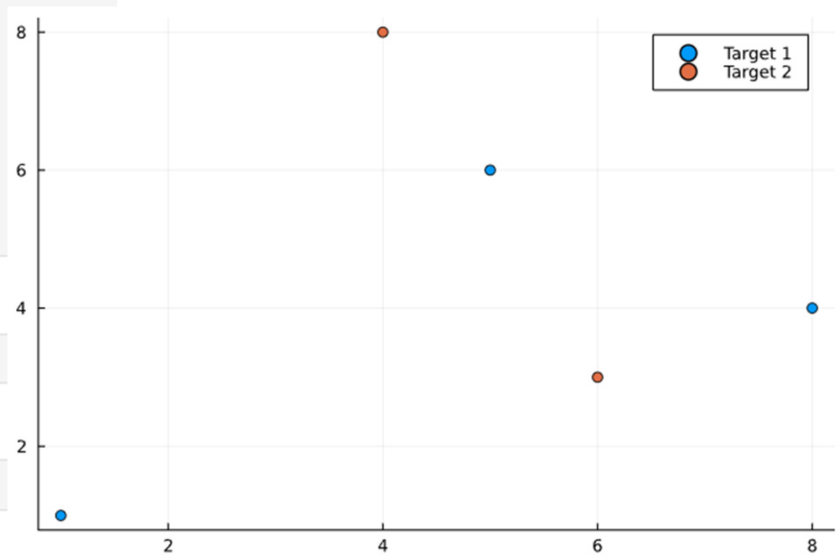
```
(0.375, 0.375)
```

```
impurity(t[.!is]), 1 - (1/1)^2
```

```
(0.0, 0.0)
```

```
infogain(s,f2,t), impurity(t) - impurity(t[is])*(4/5) - impurity(t[.!is])*(1/5)
```

```
(0.18, 0.17999999999999994)
```



ML: Q40

With respect to Ex 2 of ML-3, the table contains the feature values for three passengers. Use the decision tree to determine the predicted probability that each passenger would survive.

Row	Pclass Int64	Sex Int64	Age Float64	Sibsp Int64	Parch Int64	Fare Float64
1	3	2	21.0	0	0	7.8292
2	3	1	8.0	8	2	69.55
3	3	1	18.0	0	1	14.4542

```
Feature 2, Threshold 1.5
L-> Feature 1, Threshold 2.5
    L-> Feature 6, Threshold 29.35625
        L-> Feature 3, Threshold 22.5
            L-> 2 : 12/12
            R-> 2 : 26/33
        R-> 2 : 61/61
    R-> Feature 6, Threshold 22.737499999999997
        L-> Feature 4, Threshold 0.5
            L-> 2 : 32/47
            R-> 1 : 11/20
        R-> Feature 6, Threshold 31.331249999999997
            L-> 1 : 9/9
            R-> 1 : 4/6
R-> Feature 6, Threshold 26.26875
    L-> Feature 3, Threshold 11.5
        L-> Feature 4, Threshold 2.0
            L-> 2 : 7/7
            R-> 1 : 1/1
        R-> Feature 3, Threshold 32.5
            L-> 1 : 130/150
            R-> 1 : 65/67
    R-> Feature 4, Threshold 2.5
        L-> Feature 3, Threshold 36.5
            L-> 2 : 20/33
            R-> 1 : 28/38
        R-> Feature 3, Threshold 3.5
            L-> 1 : 4/5
            R-> 1 : 8/8
```

```
predict(mach,df)
```

```
3-element MLJBase.UnivariateFiniteVector{OrderedFactor{2},
 UnivariateFinite{OrderedFactor{2}}(0=>0.867, 1=>0.133)
 UnivariateFinite{OrderedFactor{2}}(0=>0.667, 1=>0.333)
 UnivariateFinite{OrderedFactor{2}}(0=>0.319, 1=>0.681)
```

```
println("1:", 1 - 130/150, "\n2:", 1 - 4/6,"\n3:", 32/47)
```

```
1:0.13333333333333333
2:0.33333333333333337
3:0.6808510638297872
```

General: Q41

The following are descriptions of a problem. Based on the models and solution procedures discussed in class, recommend both a model and solution procedure that would be best suited for solving the problem. Include a justification for both your model and solution procedure recommendations.

a. *Problem*: A single generator will be used to power three different refrigeration units located on an industrial site. Where should the generator be placed so that the cost of running power lines to the units is minimized?

   Model: Nonlinear optimization to minimize total distance of power lines

   Solution procedure: Nelder-Mead since 2-D

General: Q41

b. *Problem*: A student would like to minimize their weekly expenditure for food while still maintaining a healthy diet. They have determined the recommended nutritional requirements along with the cost and nutritional information of several of their favorite foods. How should they determine what foods to purchase each week?

Model: Linear programming, since objective and all constraints are linear and assuming all food quantities can be approximated as continuous decision variables

Solution procedure: Revised simplex method (or any other LP procedure would be fine)

General: Q41

c. *Problem*: A developer would like to create a software package to detect 4000 known computer viruses using 8000 substrings of 20 or more consecutive bytes from the viruses that are not found in normal code and can be used to detect viruses in code. In order to operate efficiently, only a small number of substrings, much less than 8000, can be used.

Model: Set covering, where the 8000 substrings are the sets and the 4000 viruses are the objects to be covered (can cover with 150 substrings, making virus checking every efficient)

Solution procedure: MILP, since easy to find optimal solutions to the set covering problem even for large instances.

General: Q41

d. *Problem*: Two hundred and forty-five boxes with different heights need to be stacked floor-to-ceiling in a room, and the area required for storage should be minimized.

Model: Bin packing problem

Solution procedure: MILP since not “too many” boxes (otherwise, the *multifirstfit* heuristic procedure)

General: Q41

e. *Problem*: NASA would like to determine the scaled sound pressure level, in decibels, of different size airfoils at various wind tunnel speeds and angles of attack.

   Model: Regression (not enough info given to determine the type of regression model needed)

   Solution procedure: Any regression procedure

General: Q41

f. *Problem*: Several different commodities are available for purchase. There is a limited capacity available for shipping the commodities, and there are limited funds available for the purchase of the commodities. Subject to a minimum order quantity, any amount of each commodity can be purchased.

Model: MILP (not LP because of MOQ)

Solution procedure: MILP with semi-continuous variables used to handle the MOQ