

Problem 1

a.

This defines an independence system $(E, \mathcal{F}), \mathcal{F} \subseteq 2^E$

First, prove that $\emptyset \in \mathcal{F}$.

Obviously, we have $\emptyset \subseteq E = \{1, 2, \dots, n\}$, and $a(\emptyset) = \sum_{j \in \emptyset} a_j = 0 \leq b \in \mathbb{R}_+$
So, $\emptyset \in \mathcal{F}$.

Second, prove that $X \subseteq Y \in \mathcal{F} \Rightarrow X \in \mathcal{F}$

$$Y \in \mathcal{F} \Rightarrow a(Y) = \sum_{j \in Y} a_j \leq b$$

$$X \subseteq Y \Rightarrow a(X) = \sum_{j \in X} a_j = \sum_{i \in Y} a_i - \sum_{j \in Y \setminus X} a_j \leq \sum_{j \in Y} a_j \leq b$$

So, this defines an independence system $(E, \mathcal{F}), \mathcal{F} \subseteq 2^E$

b.

According to the definition, we have

$$n = 6 \Rightarrow S \subseteq E = \{1, 2, 3, 4, 5, 6\}$$

$$a = (1, 1, 1, 4, 4, 5), b = 8 \Rightarrow a(S) = \sum_{j \in S} a_j \leq 8$$

According to the definition of *rank*, we have

$$r(X) = \max_{F \in \mathcal{F}} |F \cap X| = \max_{B \in \mathcal{B}_X} |B|, \text{ and } \rho(X) = \min_{B \in \mathcal{B}_X} |B|$$

We can find that

$$\mathcal{B}_X = \{\{1, 2, 3, 4\}, \{1, 2, 3, 5\}, \{1, 2, 3, 6\}, \{4, 5\}\}$$

So that we have $r(X) = 4, \rho(X) = 2$

c.

$$O(r(X)) = O(n \log n)$$

As the greedy algorithm, order the a from small to large, use the *Best-in Greedy* to select numbers from the smallest. So the complexity is $O(n \log n)$

d.

$$O(\rho(X)) = O(n^2)$$

Need to find all the subset of S So the complexity is $O(n^2)$

e.

As b . mentioned, $B_3 = \{1, 2, 3, 6\}$, $B_4 = \{4, 5\}$. If we use *Best-in Greedy*, we may lose the B_4 , because there is a solution when the first choice is 6.

Problem 2

a.

We have the job set E like following

Job	1	2	3	4	5	6	7
Due	3	2	4	1	4	4	6
Profit	2	3	4	3	3	6	7

The purpose is to find a independent subset $S \subseteq E$

The initialization of *Best-in Greedy* is sorting the jobs, as following

Job	7	6	3	5	4	2	1
Due	6	4	4	4	1	2	3
Profit	7	6	4	3	3	3	2

Then, as *Best-in Greedy*, the process would be

Date	1	2	3	4
Job	7	6	3	5
Due	6	4	4	4
Profit	7	6	4	3

The total profit would be: $7 + 6 + 4 + 3 = 20$

b.

Suppose A and B are two independent subsets of E , so we have $A, B \subseteq E$, and $A, B \in \mathcal{F}$. Construct A and B as:

$$A = \{a_1, a_2, \dots, a_k\}, B = \{b_1, b_2, \dots, b_m\}, a_i \text{ and } b_j \text{ means the job in } A, B$$

Suppose that $k < m$, so $|A| < |B|$.

If all the due of a_i is less or equal than k , it's easy to find a b_j whose due is larger than k , because of $k < m$. So, we can add b_j to A as a_{k+1} and all the jobs can be processed,

$$b_j \in B \setminus A, A \cup \{b_j\} \in \mathcal{F}$$

If $\exists a_i, d_{a_i} > k$, we can find a $b_j \in B \setminus A$ whose due is larger than i , because of $k - i < m - i$. So, we can put a_i as $(k + 1)$ th process and set b_j as i th process, all the jobs can be processed,

$$b_j \in B \setminus A, A \cup \{b_j\} \in \mathcal{F}$$

So, (E, \mathcal{F}) is a matroid.

Problem 3

a.

$(V(G), \mathcal{F})$ is a matroid.

First, prove that $\emptyset \in \mathcal{F}$.

Obviously, we have $\emptyset \subseteq V(G)$, and there is no edge because of no vertex.

So, $\emptyset \in \mathcal{F}$.

Second, prove that $X \subseteq Y \in \mathcal{F} \Rightarrow X \in \mathcal{F}$

If Y is independent, which means no edge inside the Y , obviously all the subsets of Y contain no edge inside.

So, this defines an independence system $(V(G), \mathcal{F}), \mathcal{F} \subseteq 2^{V(G)}$

Third, to prove that the independence system $(V(G), \mathcal{F})$ is a matroid, according to the *lemma 2*, only need to prove **for each $X \subseteq E$, all bases of X are of the same size.**

Supposed $|V(G)| = n$.

Apparently, no matter which vertices are chosen, for n is *odd*, the size of all the bases of X is equal to $\frac{n-1}{2}$, and for n is *even*, the size of all the bases of X is equal to $\frac{n}{2}$.

So, $(V(G), \mathcal{F})$ is a matroid.

b.

While $n = 5$, according to a., so that the size of bases $r(S) = \frac{n-1}{2} = 2$.

Therefore, the defining inequalities for $P_{V(G), \mathcal{F}}$:

$$S_1 = \{x_1, x_3\} \quad x(S_1) = x_1 + x_3 \leq r(S) = 2$$

$$S_2 = \{x_2, x_4\} \quad x(S_2) = x_2 + x_4 \leq r(S) = 2$$

$$S_3 = \{x_3, x_5\} \quad x(S_3) = x_3 + x_5 \leq r(S) = 2$$

$$S_4 = \{x_4, x_1\} \quad x(S_4) = x_4 + x_1 \leq r(S) = 2$$

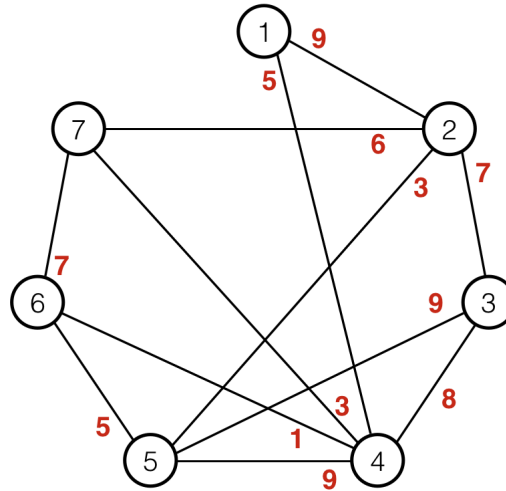
$$S_5 = \{x_5, x_2\} \quad x(S_5) = x_5 + x_2 \leq r(S) = 2$$

c.

Yes. According to the *Edmonds(1970)* The output is the linear, especially binary in this case, combination of the characteristic vectors, which are integral. So the polytope $P_{V(G), \mathcal{F}}$ is integral.

Problem 4

As the definition, the graph is following.



Use *Kruskal's Algorithm* to find the maximum-weight spanning tree.

Initialize Sort E by weight from largest to smallest.

$$E = \{(1,2), (3,5), (4,5), (3,4), (2,3), (6,7), (2,7), (1,4), (5,6), (2,5), (4,7), (4,6)\}$$

Set $T = \emptyset$

1. $T \cup \{(1,2)\}$, cycle free, $T = T \cup \{(1,2)\}$
2. $T \cup \{(3,5)\}$, cycle free, $T = T \cup \{(3,5)\}$
3. $T \cup \{(4,5)\}$, cycle free, $T = T \cup \{(4,5)\}$
4. $T \cup \{(3,4)\}$, cycle!

5. $T \cup \{(2, 3)\}$, cycle free, $T = T \cup \{(2, 3)\}$
6. $T \cup \{(6, 7)\}$, cycle free, $T = T \cup \{(6, 7)\}$
7. $T \cup \{(2, 7)\}$, cycle free, $T = T \cup \{(2, 7)\}$
8. $T \cup \{(1, 4)\}$, cycle! No alone vertex, means rest edges make cycle, algorithm end.

So that the $T = \{(1, 2), (3, 5), (4, 5), (2, 3), (6, 7), (2, 7)\}$
The weight of the maximum-weight spanning tree is

$$w_{1,2} + w_{3,5} + w_{4,5} + w_{2,3} + w_{6,7} + w_{2,7} = 47$$

Problem 5

a.

$$|B| = \frac{15 \times 3 + 18 \times 5 + 12 \times 15}{1 + 2 + 4} = 45$$

b.

The maximum cardinality matching in this graph is 45.

If graph $G = (A \cup B, E)$ has a *Perfect Matching*, which has $\nu(G) = |A|$, the maximum cardinality matching will be *Perfect Matching*.

Now prove that in this case $G = (A \cup B, E)$ has a *Perfect Matching*.

According to *Theorem 3 (Frobenius(1971), Hall(1935))*, only need to prove,

$$|S| \leq |N(S)| \quad \text{for all } S \subseteq A$$

Proof by contradiction. If $\exists |S| > |N(S)|$, $S \subseteq A$. Suppose that $k = |S|$, $m = |N(S)|$, i.e. $k > m$.

Suppose that

$$k_1 + k_2 + k_3 = k > m$$

in which k_1 refers the number of vertices of degree 3, k_2 refers the number of vertices of degree 5, k_3 refers the number of vertices of degree 15.

According to the definition of B , we could get $m \geq 3k_1$, $m \geq \frac{5}{2}k_2$, $m \geq \frac{15}{4}k_3$, so we have

$$m = \left(\frac{1}{3} + \frac{2}{5} + \frac{4}{15}\right)m \geq k_1 + k_2 + k_3 = k > m$$

Which makes $m > m$, **contradiction**.

So graph $G = (A \cup B, E)$ has a *Perfect Matching*, and $\nu(G) = |A|$.

Problem 6

Use AMPL model to solve this problem.

Code :

```
#File: CuttingStock.mod
#Date: 2017.03.16
#Author: Yuan Qu

#-----CUTTING STOCK MODEL USING PATTERNS-----
param roll_width >0;

set WIDTHS;
param orders {WIDTHS} >0;

param nPAT integer >=0;
set PATTERNS := 1..nPAT;

param A{WIDTHS, PATTERNS} integer >=0;
    check{j in PATTERNS}:
        sum {i in WIDTHS} i * A[i,j] <= roll_width;

var X{PATTERNS} integer >=0;

minimize NumberOfRolls:
    sum {j in PATTERNS} X[j];

subj to OrderFill {i in WIDTHS}:
    sum {j in PATTERNS} A[i,j] * X[j] >=orders[i];

#-----KNAPSACK SUBPROBLEM FOR CUTTING STOCK-----
param dualprice {WIDTHS} default 0.0;

var C {WIDTHS} integer >=0;

minimize Reduced_Cost:
    1 - sum {i in WIDTHS} dualprice[i] * C[i];

subj to Width_Limit:
    sum {i in WIDTHS} i * C[i] <=roll_width;

#-----Reading Data and Choosing Solver-----
data CuttingStock.dat;

option solver cplex;
```

```
option solution_round 6;

#-----Defining the Two Stages-----
problem Cutting_Opt: X, NumberOfRolls, OrderFill;
option relax_integrality 1;

problem Pattern_Gen: C, Reduced_Cost, Width_Limit;
option relax_integrality 0;

#-----Initializing Pattern Set-----
let nPAT := 0;

for {i in WIDTHS} {
  let nPAT := nPAT + 1;
  let A[i,nPAT] := floor (roll_width/i);
  let {i2 in WIDTHS: i2 <> i} A[i2,nPAT] := 0;
};

#-----Two Stage Optimization Loop-----
repeat {
  solve Cutting_Opt;

  let {i in WIDTHS} dualprice[i] := OrderFill[i].dual;

  solve Pattern_Gen;

  if Reduced_Cost < -0.00001 then {
    let nPAT := nPAT + 1;
    let {i in WIDTHS} A[i,nPAT] := C[i];

    display C;
  }
  else break;
};

#-----Final Integer Program with Generated Patterns-----
option Cutting_Opt.relax_integrality 0;
solve Cutting_Opt;

#-----Displaying Results-----
display A;
display X;
```

Data:

#File: CuttingStock.dat

```
#Date: 2017.03.16  
#Author: Yuan Qu
```

```
param roll_width := 24 ;
```

```
param: WIDTHS: orders :=  
      3    12300  
      5    9800  
     12    3600  
      7    5750  
      8    15250  
      9    6700 ;
```

Result:

In this part, A means the Patterns, in which $A[i,j]$ refers in Pattern i , the cut number of Width j . X means the optimal solution, in which $X[i]$ refers the number of Pattern i processed.

The processing result during the iteration is not copied.

```
A [*,*] (tr)  
:   3   5   7   8   9  12   :=  
1   8   0   0   0   0   0  
2   0   4   0   0   0   0  
3   0   0   0   0   0   2  
4   0   0   3   0   0   0  
5   0   0   0   3   0   0  
6   0   0   0   0   2   0  
7   0   1   0   0   2   0  
8   0   2   2   0   0   0  
9   1   4   0   0   0   0  
10  0   3   0   0   1   0  
11  2   0   0   0   2   0  
;
```

```
X [*] :=  
1   869  
2    0  
3  1800  
4    0  
5  5084  
6    0  
7    0  
8  2875  
9    0  
10  1350  
11  2675
```


;

Problem 7

Use AMPL model to solve this problem.

Code :

```
#File: P7.mod
#Date: 2017.03.16
#Author: Yuan Qu

param total >0;
set ID;

param profit {ID} > 0;
param cost {ID} > 0;
param f_min {ID} >= 0;
param f_max {j in ID} >= f_min[j];

var Invest {j in ID} integer >= f_min[j], <= f_max[j];

maximize Total_Profit: sum {j in ID} profit[j] *
    Invest[j];

subject to Total_Cost: sum {j in ID} cost[j] * Invest[j]
    <= total ;

data P7.dat;

option solver cplex;
solve;

display Total_Profit;
display Invest;
```

Data:

```
#File: P7.dat
#Date: 2017.03.16
#Author: Yuan Qu

param total := 100 ;
```

```
set ID := A B C D E F G H I J K L;
param: profit cost f_min f_max :=
  A 30 16 0 1
  B 25 15 0 1
  C 5 2 0 1
  D 9 4 0 1
  E 11 4 0 1
  F 7 3 0 1
  G 14 9 0 1
  H 19 12 0 1
  I 40 29 0 1
  J 29 20 0 1
  K 30 18 0 1
  L 34 25 0 1;
```

Result: it means the max total profit is 172 by investing $\{A, B, C, D, E, G, H, J, K\}$

```
CPLEX 12.7.0.0: optimal integer solution; objective 172
5 MIP simplex iterations
0 branch-and-bound nodes
No basis.
Total_Profit = 172
```

```
Invest [*] :=
A 1
B 1
C 1
D 1
E 1
F 0
G 1
H 1
I 0
J 1
K 1
L 0
;
```

Problem 8

Use AMPL model to solve this problem.

Code:

In this part, couldn't find a concise way to organize the model, so the code

is simple and redundant.

By the way, find that the result will be same if there is no multiple limitation of *own power plants* and *rent power plants*, so there is no limitation in code.

```
#File: P8.mod
#Date: 2017.03.17
#Author: Yuan Qu

#The demand for 1-7 days
param d1 = 1230;
param d2 = 1190;
param d3 = 845;
param d4 = 935;
param d5 = 915;
param d6 = 1625;
param d7 = 1510;

#The own plants for 1-7 days
var o1 integer >= 0, <= 500;
var o2 integer >= 0, <= 500;
var o3 integer >= 0, <= 500;
var o4 integer >= 0, <= 500;
var o5 integer >= 0, <= 500;
var o6 integer >= 0, <= 500;
var o7 integer >= 0, <= 500;

#The rent plants for 1-7 days
var r1 integer >= 0, <= 600;
var r2 integer >= 0, <= 600;
var r3 integer >= 0, <= 600;
var r4 integer >= 0, <= 600;
var r5 integer >= 0, <= 600;
var r6 integer >= 0, <= 600;
var r7 integer >= 0, <= 600;

#The purchase(buy) plants for 1-7 days
var b1 integer >= 0, <= 500;
var b2 integer >= 0, <= 500;
var b3 integer >= 0, <= 500;
var b4 integer >= 0, <= 500;
var b5 integer >= 0, <= 500;
var b6 integer >= 0, <= 500;
var b7 integer >= 0, <= 500;

#Store of electricity for 1-7 days, cumulative
var s1 integer >= 0;
```

```
var s2 integer >= 0;
var s3 integer >= 0;
var s4 integer >= 0;
var s5 integer >= 0;
var s6 integer >= 0;
var s7 integer >= 0;

#Objective Function
minimize Cost: 20*(o1+o2+o3+o4+o5+o6+o7) +
    25*(r1+r2+r3+r4+r5+r6+r7) + 45*(b1+b2+b3+b4+b5+b6+b7)
    + 2*(s1+s2+s3+s4+s5+s6+s7);

#Constraints of the defination of store
subject to Store1: s1 = o1 + r1 + b1 - d1;
subject to Store2: s2 = o2 + r2 + b2 + s1 - d2;
subject to Store3: s3 = o3 + r3 + b3 + s2 - d3;
subject to Store4: s4 = o4 + r4 + b4 + s3 - d4;
subject to Store5: s5 = o5 + r5 + b5 + s4 - d5;
subject to Store6: s6 = o6 + r6 + b6 + s5 - d6;
subject to Store7: s7 = o7 + r7 + b7 + s6 - d7;

#Constraints that fill the demand
subject to Demand1: o1 + r1 + b1 >= d1;
subject to Demand2: o2 + r2 + b2 + s1 >= d2;
subject to Demand3: o3 + r3 + b3 + s2 >= d3;
subject to Demand4: o4 + r4 + b4 + s3 >= d4;
subject to Demand5: o5 + r5 + b5 + s4 >= d5;
subject to Demand6: o6 + r6 + b6 + s5 >= d6;
subject to Demand7: o7 + r7 + b7 + s6 >= d7;

#Use cplex to colve integer program
option solver cplex;
solve;

display Cost;

display o1,o2,o3,o4,o5,o6,o7;
display r1,r2,r3,r4,r5,r6,r7;
display b1,b2,b3,b4,b5,b6,b7;
display s1,s2,s3,s4,s5,s6,s7;
```

Result:

CPLEX 12.7.0.0: optimal **integer** solution; objective
202470
7 MIP simplex iterations

0 branch-and-bound nodes
No basis.
Cost = 202470

o1 = 500
o2 = 500
o3 = 500
o4 = 500
o5 = 500
o6 = 500
o7 = 500

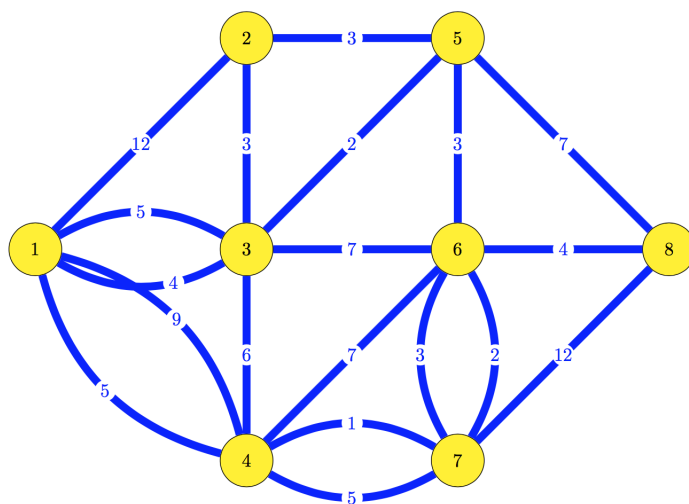
r1 = 600
r2 = 600
r3 = 600
r4 = 600
r5 = 600
r6 = 600
r7 = 600

b1 = 130
b2 = 90
b3 = 0
b4 = 0
b5 = 0
b6 = 0
b7 = 330

s1 = 0
s2 = 0
s3 = 255
s4 = 420
s5 = 605
s6 = 80
s7 = 0

Problem 9

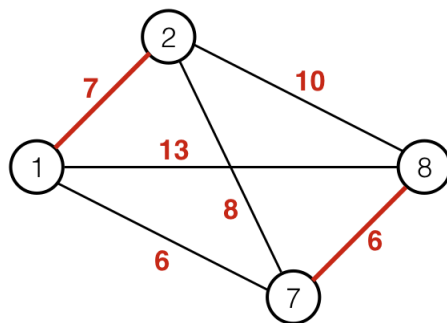
Problem 10



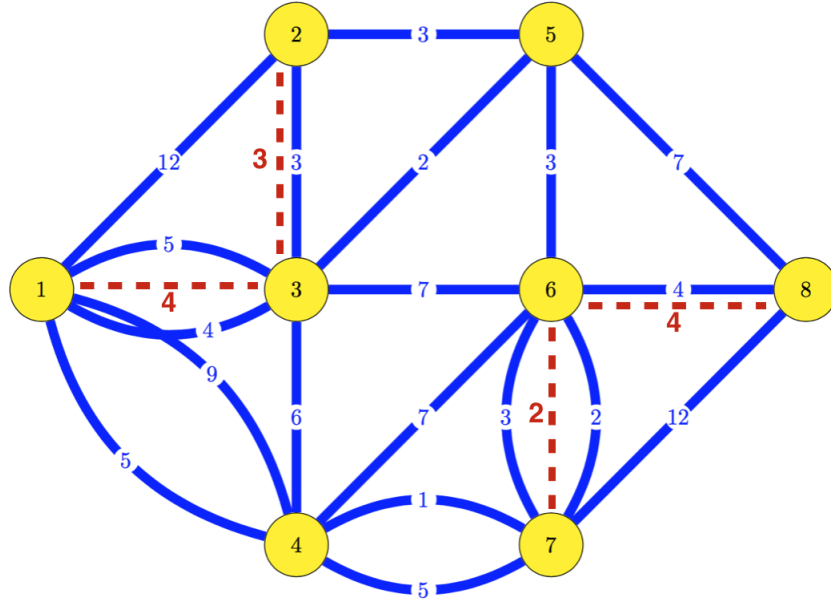
a.

Because the vertices V_1, V_2, V_7, V_8 have odd degree, so there is no *Eulerian path* in this Graph G .

So, to get the optimal solution of *Chinese Postman Problem*, we need to consider the K_4 , the The complete graph on 4 vertices of V_1, V_2, V_7, V_8 .



According to that, we choose path $V_1 \rightarrow V_2$ and $V_7 \rightarrow V_8$ to add on (means repeat) the original graph.



In this case, it's easy to find a *Eulerian path* that:

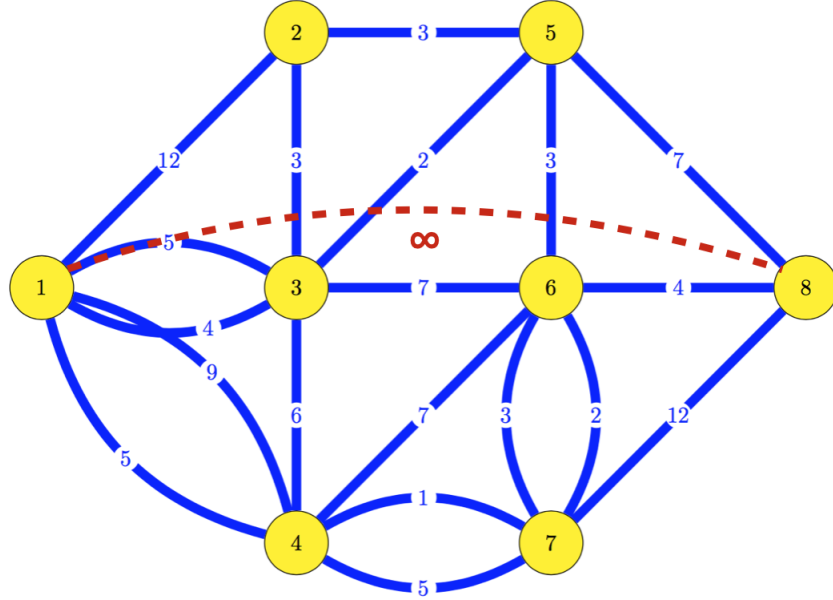
$V_1 \rightarrow V_2 \rightarrow V_5 \rightarrow V_8 \rightarrow V_6 \rightarrow V_5 \rightarrow V_3 \rightarrow V_6 \rightarrow V_8 \rightarrow V_7 \rightarrow V_6 \rightarrow V_7$
 $\rightarrow V_6 \rightarrow V_4 \rightarrow V_7 \rightarrow V_4 \rightarrow V_3 \rightarrow V_2 \rightarrow V_3 \rightarrow V_1 \rightarrow V_3 \rightarrow V_1 \rightarrow V_4 \rightarrow V_1$

Total time:

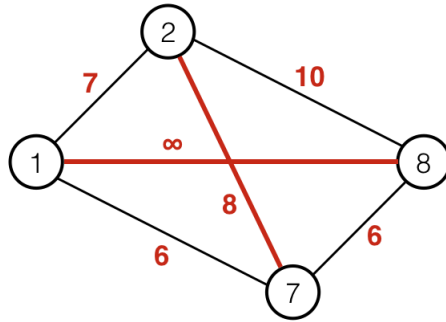
$$\sum_{E \in G} w_E + w_{V_1 \rightarrow V_2} + w_{V_7 \rightarrow V_8} = 100 + 7 + 6 = 113$$

b.

To end in area 8, consider a *Eulerian path* that the last step is $V_8 \rightarrow V_1$, which means V_1 and V_2 are neighbors there is a edge between them. Because we don't need to go back area 1, so set $w_{(V_1, V_8)} = \infty$. Now G looks like:



Construct a *Eulerian path* from Graph G , same with $a.$, construct a K_4 , the complete graph on 4 vertices of V_1, V_2, V_7, V_8 .



Because we have to choose $V_1 \rightarrow V_8$ as the last step, so we have to choose $V_2 \rightarrow V_7$ as the another route.

So in this case, the *Eulerian path* is:

$V_1 \rightarrow V_2 \rightarrow V_3 \rightarrow V_1 \rightarrow V_3 \rightarrow V_4 \rightarrow V_1 \rightarrow V_4 \rightarrow V_6 \rightarrow V_7 \rightarrow V_4 \rightarrow V_7$
 $\rightarrow V_6 \rightarrow V_3 \rightarrow V_5 \rightarrow V_2 \rightarrow V_5 \rightarrow V_6 \rightarrow V_5 \rightarrow V_8 \rightarrow V_6 \rightarrow V_7 \rightarrow V_8 \rightarrow V_1$

The optimal path is eliminate the last step to V_1 from the *Eulerian path*. And the total time is:

$$\sum_{E \in G} w_E + w_{V_2 \rightarrow V_7} = 100 + 8 = 108$$