## EE3301 Cheatsheet Final Exam

### Graph Theory

$$G = (V, E)$$

where $V$ is the set of nodes and $E$ is the set of edges between pairs of nodes. The graph size parameters are denoted as $n = |V|$ and $m = |E|$.

1. Eulerian trail is when it passes through every edge exactly once.
2. Eulerian cycle is if an Eulerian trail starts and ends on the same node.
3. The degree of a node is the number of edges that are adjacent to the node.
4. Hamiltonian cycle is when it visits each node exactly once, except for the node that is both the start and the end (as a cycle, it has to visit twice).
5. Hamiltonian path is when the path that visits each node exactly once
6. The neighbourhood is the set of nodes that is connected to the original node via edges, denoted as $N(v)$.
7. Simple graph is an undirected graph without loop or multiple edges.
8. Complete graph is a simple undirected graph,
    1. Every pair of distinct vertices is connected by a unique edge.
    2. The complete graph on $n$ vertices is denoted by $K_n$.
9. Bipartite graph is when $V$ are partitioned into two set of nodes $V_1$ and $V_2$ and it does not contain an odd cycle.
10. Tree is an undirected graph, connected and does not contain cycle,
    1. Two nodes have exactly one path between them.

The total number of nodes in a tree with $n$ edge s can be computed as

$$|V| = n + 1$$

The total degree in a simple graph can be computed as

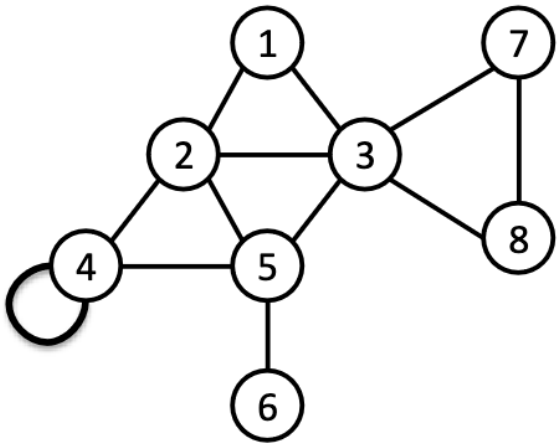
$$\sum_{v, \in V} \deg(v_i) = 2\,|E|$$

and the total number of edges can be computed as

$$|E| = \frac{n(n-1)}{2}$$

## Adjacency Matrix

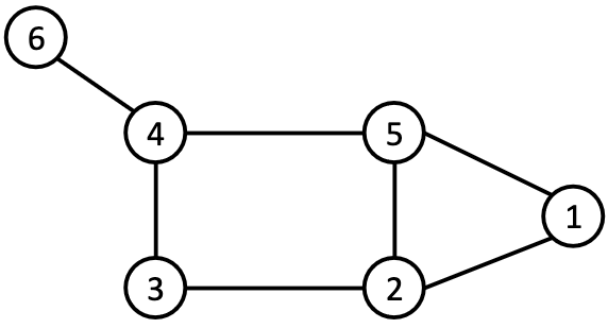
An $n \times n$ matrix, with $A_{uv} = 1$ if $(u, v)$ is an edge, otherwise $A_{uv} = 0$, as




| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 2 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 3 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 |
| 4 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| 5 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |
| 6 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 7 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 8 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |

### Incidence Matrix (Undirected Graph)

It is a $V \times E$ matrix, with $A_{v,e} = 1$ if $v_i$ and $e_i$ are incident, otherwise $A_{v,e} = 0$

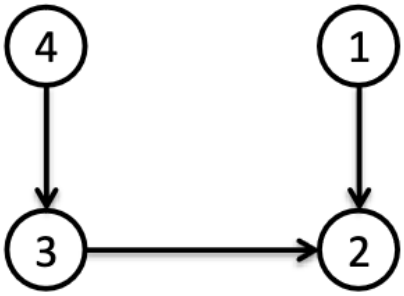



| | 1,2 | 1,5 | 2,3 | 2,5 | 3,4 | 4,5 | 4,6 |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 2 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| 3 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 5 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

### Incidence Matrix (Directed Graph)

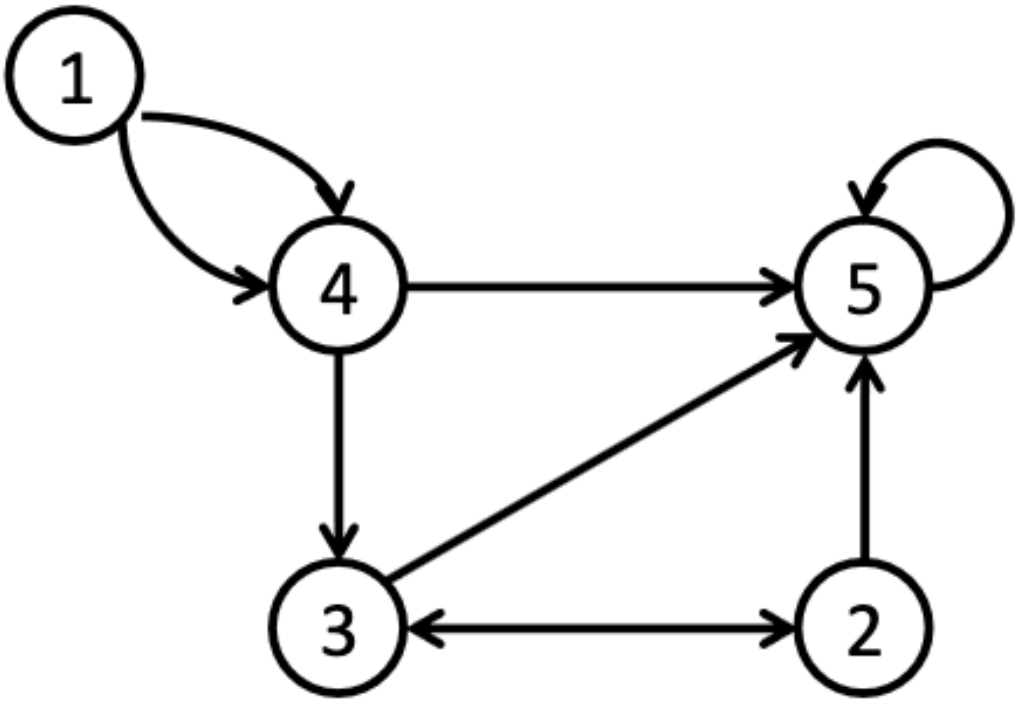
It is a $V \times E$ matrix, with

$$A_{v,e} = \begin{cases} -1, & \text{if the edge leaves the vertex} \\ 1, & \text{if the edge enters the vertex} \\ 0, & \text{otherwise} \end{cases}$$




| | 1,2 | 2,3 | 3,4 |
|---|---|---|---|
| 1 | −1 | 0 | 0 |
| 2 | 1 | 1 | 0 |
| 3 | 0 | −1 | 1 |
| 4 | 0 | 0 | −1 |

**Adjacency List**




Edge list is defined as $(1,4),(1,4),(2,3),(2,5),(3,2),(3,5),(4,3),(4,5),(5,5)$, and its adjacency list is defined as $(1,4,4),(2,3,5),(3,2,5),(4,3,5),(5,5)$.

**Isomorphic Graphs**

They are the isomorphic graphs when the same number of nodes and the same edge connectivity.

**Diameter**

It is the shortest path for each pair of nodes in the graph, and among all the shortest path, the longest length one is the diameter.

**Spanning Tree**

It is a connected subgraph that includes all the nodes of the graph in which there are no cycles, as if given a graph has $N$ nodes, then its spanning tree has $N-1$ nodes.

**Steiner Tree**

It is a tree of minimum weight that contains all terminals, and as given a subset of the nodes in the graph called terminals

## Prim's Algorithm

1. Initialize a tree with a single vertex, chosen arbitrarily from the graph.
2. Grow the tree by an edge that has the minimum weight.
    1. If $e_1$ and $e_2$ have the same weight, just pick one arbitrarily.
3. Repeat until all nodes are in the tree.
   Dijkstra's Algorithm

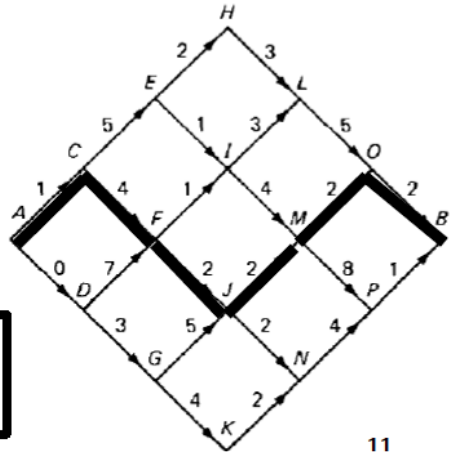
It works on both directed and undirected graph, but all edges must have non-negative weight.

1. Construct a table with all nodes and its current distance from the starting node.
2. Initially, the value of distance should be $\infty$, except for the starting node.
3. Pick the shortest distance by using $\min(X, Y)$.
4. Underline the determined minimal value of the distance
5. Repeat until all nodes are assigned into the finished set
6. State the shortest path and draw it

### Dynamic Programming (DP)

## Bellman's Principle of Optimality

$P_O = B$, $P_P = B$;
$P_L = O$, $P_M = O$, $P_N = P$;
$P_H = L$, $P_I = M$, $P_J = M$, $P_K = N$;
$P_E = I$, $P_F = J$, $P_G = J$ or $K$;
$P_C = F$, $P_D = G$;
$P_A = C$.




so the shortest path is:
$A \rightarrow C \rightarrow F \rightarrow J \rightarrow M \rightarrow O \rightarrow B$

$S_X$ is defined as the minimal distance from $X$ to $B$, and it is called the optimal value function which is a rule that assign value to various subproblems associated with the different values of $X$. Also, to find the boundary condition from the backward, and to compute all optimal value function $S_x$ with recurrence relations.

$$S_x = \min \begin{bmatrix} d_a + a \\ d_b + b \end{bmatrix}$$

where $d$ is the distance from $x$ to $i$, i.e.

$$S_A = \min \begin{bmatrix} 1 + S_c \\ 0 + S_D \end{bmatrix}$$

Then, the boundary conditions will be

$$S_o = 2 \text{ and } S_p = 1$$

Next, to find $P_x$, that is the node after node $X$ on the shortest path from $X$ to $Y$, and it is called optimal policy function, i.e.

$$P_M = O \rightarrow P_O = B$$

**Additions and Comparisons**

For $N$ stages problem, which $N$ is observed by the sum of number of nodes in the lines $CB$ and $DB$ excluding $B$.

If it is using dynamic programming, then it requires

| | |
|---|---|
| Addition | $\frac{N^2}{2+N}$ |
| Comparison | $\frac{N^2}{4}$ |

If it using brute force, then it requires

| | |
|---|---|
| Addition | $(N-1)\binom{N}{N/2}$ |
| Comparison | $\binom{N}{N/2} - 1$ |

**Bellman's Principle of Optimality with the Coordinate System**

$$S(x,y) = \min \begin{bmatrix} a_u(x,y) + S(x+1, y+1) \\ a_d(x,y) + S(x+1, y-1) \end{bmatrix}$$

where $a_u(x,y)$ is the distance of the edge between $(x,y)$ and $(x+1, y+1)$, $a_d(x,y)$ is the distance of the edge between $(x,y)$ and $(x+1, y-1)$, and if there is no link, $a_u$ or $a_d$ will equal to $\infty$.
Therefore, first to find the boundary condition from backward, $S_n = (x_n, y_n)$, then use recurrence relation to compute all the optimal value function.

**Bellman's Principle of Optimality with Forward Dynamic Programming**

$$S(x,y) = \min \begin{bmatrix} a_u(x-1,y-1) + S(x-1,y-1) \\ a_d(x-1,y+1) + S(x-1,y+1) \end{bmatrix}$$

where $a_u(x,y)$ is the distance of the edge between $(x,y)$ and $(x+1,y+1)$, $a_d(x,y)$ is the distance of the edge between $(x,y)$ and $(x+1,y-1)$, and if there is no link, $a_u$ or $a_d$ will equal to $\infty$.

**Equipment Replacement (ER)**

$$S(p) = \min \begin{bmatrix} \text{Buy:} & p - t(x) + c(0) + S(1,k+1) \\ \text{Keep:} & c(x) + S(x+1,k+1) \end{bmatrix}$$

1. $y$ is the age of the machine at the beginning of year 1
2. $c(i)$ is the cost of operating for one year a machine which is of age $i$ at the start of the year
3. $p$ is the price of a new machine ($y = 0$)
4. $t(i)$ is the trade-in value received when a machine which is of age $i$ at start of a year is traded for a new machine at the start of the year
5. $s(i)$ is the salvage value received for a machine that has just turned age $i$ at the end of year $N$
6. $S(x,k)$ is the minimum cost of owning a machine from year $k$ through $N$, starting year $k$ with a machine just turned age $x$

Therefore, the boundary condition is

$$S(x, N+1) = -s(x)$$

and the equally acceptable boundary condition is

$$S(x,N) = \min \begin{bmatrix} p - t(x) + c(0) - s(1) \\ c(x) - s(x+1) \end{bmatrix}$$

To use $P(i,j)$ to determine whether buy or keep, i.e. $P(1,5) = \text{keep}$.

**Resource Allocation (RA)**

Given $X$ units of a resource and the resource must be distributed among $N$ activities, also given $N$ data tables $r_i(x)$ representing the return realised from an allocation of $x$ units of resource to activity $i$. To maximise the total return $\sum_{i=1}^{N} r_i(x_i)$ with the constraints $\sum_{i=1}^{N} x_i = X$, and $x_k$ is the allocation to activity $k$ and $f_k(x)$ must be computed for $x = 0, \ldots, X$

$$f_k(x) = \max_{x_k=0,1,\ldots,x} [r_k(x_k) + f_{k+1}(x - x_k)]$$

where $f_k(x)$ is the maximum return obtainable from activities $k$ through $N$, given $x$ units of resource remaining to be allocated, and its boundary condition is

$$f_0(j, \_) = d_{1j}$$

and

$$f_N(x) = r_N(x)$$

**Dijkstra's Algorithm with DP**

$$f_i(j) = \begin{bmatrix} f_{i-1}(j) & \text{if } j \in N_i(1), \\ \min\{f_{i-1}(j), f_{i-1}(k_i) + d_{k_i,j}\} & \text{if } j \notin N_i(1); \end{bmatrix}$$

where $f_i(j)$ is the length of the shortest path from node $1$ to node $j$ when we are restricted to use paths such that all nodes preceding node $j$ belong to $N_i(1)$, and its boundary condition is

$$\begin{gathered} N_1(1) = \{1\} \\ k_1 = 1 \\ f_1(j) = d_{1j} \end{gathered}$$

**Secretary Problem**

$$\begin{aligned} v(m) &= \frac{m}{m+1} v(m+1) + \frac{1}{m+1} u(m+1) \\ u(m) &= \max\left[\frac{m}{N}, v(m)\right] \end{aligned}$$

where $v(m)$ is the probability to choose the best among $N$ candidates if the $m\text{th}$ candidate is not selected after the $m\text{th}$ interview; $u(m)$ is the probability to choose the best among $N$ candidates under optimal policy if the $m\text{th}$ candidate is the best among the first $m$ candidate after $m\text{th}$ interview, and its boundary condition is:

$$u(N) = 1$$
$$v(N) = 0$$

and to substitute $u(m)$ into $v(m)$,

$$v(m) = \frac{m}{m+1}v(m+1) + \frac{1}{m+1}\max\left[\frac{m+1}{N}, v(m+1)\right]$$

and its boundary condition is

$$v(N) = 0$$
$$v(N-1) = \frac{1}{N}$$

Therefore, $P(m)$ is defined as

$$P(m) = \begin{cases} \text{Accept} & \text{if } m/s > v(m) \\ \text{Reject} & \text{otherwise} \end{cases}$$

The critical value $m^*$ for large $N$ in the secretary problem can be computed as

$$m^* = \left\lceil \frac{N}{e} \right\rceil$$

**Traveler Salesperson Problem (TSP)**

Let $N_j = \{2, 3, \ldots, j-1, j+1, \ldots N\}$ and $S$ be a subset of $N_j$ containing $i$ members. Its optimal value function $f_i(j, S)$ is defined as the length of the shortest path from city $1$ to city $j$ via the set of $i$ intermediate cities $S$ (the stage variable $i$ indicates the number of cities in $S$)

The recurrence relation is defined as

$$f_i(j, S) = \min_{k \in S}[f_{i-1}(k, S - (k)) + d_{kj}] \qquad (i = 1, 2, \ldots, N-2;\ j \neq 1;\ S \subseteq N_j)$$

and its boundary condition is defined as

$$f_0(j, \_) = d_{1j}$$

Therefore, the answer is

$$\min_{j=2,3,\ldots,N}[f_{N-2}(j, N_j) + d_{j_1}]$$

## Linear Programming (LP)

$$\begin{aligned} \text{Maximize} \quad & c^T x \\ \text{Subject to} \quad & Ax \leq b \\ \text{and} \quad & x \geq 0 \end{aligned}$$

## Integer Linear Programming (ILP)

$$\begin{aligned} \text{Maximize} \quad & c^T x \\ \text{Subject to} \quad & Ax \leq b \\ \text{and} \quad & x \geq 0 \\ \text{and} \quad & x \text{ integer} \end{aligned}$$

## Geometric Approach

1. Let $x$ and $y$ be 0 and find the axis intersection
2. Draw the line
3. Find the intersection of two lines

## Examples

### Bob's Bakery Problem (LP)

To bake a dozen bagels Bob needs 5 cups of flour, 2 eggs, and one cup of sugar.
To bake a dozen muffins Bob needs 4 cups of flour, 4 eggs and two cups of sugar.
Bob can sell bagels in $10/dozen and muffins in $12/dozen.
Bob has 50 cups of flour, 30 eggs and 20 cups of sugar.

How many bagels and muffins should Bob bake in order to maximize his revenue?

| | Bagels | Muffins | Avail. |
|---|---|---|---|
| Flour | 5 | 4 | 50 |
| Eggs | 2 | 4 | 30 |
| Sugar | 1 | 2 | 20 |

$$\begin{aligned} \max \quad & 10x_1 + 12x_2 \\ \text{s.t.} \quad & 5x_1 + 4x_2 \leq 50 \\ & 2x_1 + 4x_2 \leq 30 \\ & x_1 + 2x_2 \leq 20 \\ & x_1 \geq 0, x_2 \geq 0 \end{aligned}$$

### Washing Machine Production Problem (LP)

Beacon Co. currently sells two models of washing machines: Arkel and Kallex. At the start of every production cycle, Beacon must decide how many units of each washing machine to produce, given its available resources. In the coming production cycle, Beacon faces key resource constraints. In particular, it has only 3,132 hours of labour, 1,440 feet of rubber hosing, and 200 drums available.

Selling each Arkel unit earns the company a profit of $350 while selling each Kallex unit earns the company a profit of $300. At the same time, manufacturing each Arkel unit requires 18 hours of labour, 6 feet of rubber hosing, and 1 drum, while manufacturing each Kallex unit requires 12 hours of labour, 8 feet of rubber hosing, and 1 drum. Details of the relevant facts are summarised in the table "Summary of Production of Washing Machines".

Based on these facts and the assumption that 100% of production will be sold, Beacon must decide how many units of each washing machine to produce in the coming production run to maximise profits.

| | Arkel | Kallex | Total available |
|---|---|---|---|
| Labour hours | 18 hours per unit | 12 hours per unit | 3132 hours |
| Rubber hosing | 6 feet per unit | 8 feet per unit | 1440 feet |
| Drums | 1 per unit | 1 per unit | 200 drums |
| Profits | $350 per unit | $300 per unit | |

$$
\begin{aligned}
\max \quad & 350x_1 + 300x_2 \\
\text{s.t.} \quad & x_1 + x_2 \leq 200 \\
& 18x_1 + 12x_2 \leq 3132 \\
& 6x_1 + 8x_2 \leq 1440 \\
& x_1 \geq 0, x_2 \geq 0
\end{aligned}
$$

### Car Production Problem (LP)

Suppose a particular Ford plant can build Escorts at the rate of one per minute, Explorer at the rate one every 2 minutes, and Lincoln Navigators at the rate of one every 3 minutes. The vehicles get 25, 15, and 10 miles per gallon, respectively, and Congress mandates that the average fuel economy of vehicles produced be at least 18 miles per gallon. Ford loses $1000 on each Escort, but makes a profit of $5000 on each Explorer and $15000 on each Navigator. What is the maximum profit this Ford plant can make in one 8-hour day?

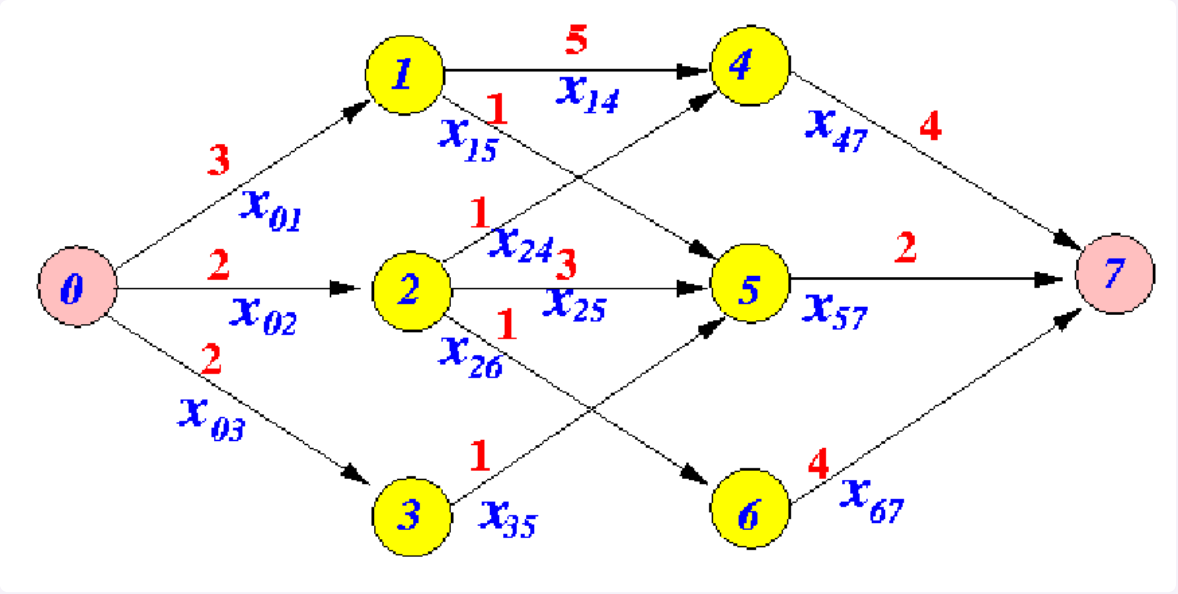
$$
\begin{aligned}
\max \quad & -1000x_1 + 5000x_2 + 15000x_3 \\
\text{s.t.} \quad & x_1 + 2x_2 + 3x_3 \leq 480 \\
& 7x_1 - 3x_2 - x_3 \geq 0 \\
& x_1, x_2, x_3 \geq 0
\end{aligned}
$$

### Max Flow Problem (LP)

Maximize the traffic flow from node 0 to node 7, subject to the capacity and flow conservation constraints.

Capacity constraints: Traffic on a link cannot be more than the link capacity limitation.

Flow conservation constraints: Total traffic enters an internal node must be equal to total traffic that exits that internal node. All nodes excluding Nodes 0 and 7 are internal nodes.




$$
\begin{aligned}
X_{01} \leq 3, \quad & X_{25} \leq 3 \\
X_{02} \leq 2, \quad & X_{26} \leq 1 \\
X_{03} < 2, \quad & X_{35} \leq 1 \\
X_{14} \leq 5, \quad & X_{47} \leq 4 \\
X_{15} \leq 1, \quad & X_{57} \leq 2 \\
X_{24} \leq 1, \quad & X_{67} \leq 4
\end{aligned}
$$

$$
\begin{aligned}
Node_1 \quad & X_{01} = X_{14} + X_{15} \\
Node_2 \quad & X_{02} = X_{24} + X_{25} + X_{26} \\
Node_3 \quad & X_{03} = X_{35} \\
Node_4 \quad & X_{14} + X_{24} = X_{47} \\
Node_5 \quad & X_{15} + X_{25} + X_{35} = X_{57} \\
Node_6 \quad & X_{26} = X_{67}
\end{aligned}
$$

$$
\max \quad \text{Flow} = X_{01} + X_{02} + X_{03}
$$

**Max Flow Min Cut Theorem**

In a flow network, the maximum amount of flow passing from the source (start node) to the sink (end node) is equal to the total weight (capacity) of the edges in the minimum cut, i.e. the smallest total weight (capacity) of the edges which if removed would disconnect the source from the sink.

$$
\begin{array}{ll}
\text{Cut 1:} & (0,1),(0,2),(0,3) \\
\text{Cut 2:} & (1,4),(2,4),(1,5),(2,5),(2,6),(3,5) \\
\text{Cut 3:} & (1,4),(2,4),(1,5),(2,5),(2,6),(0,3) \\
\text{Cut 4:} & (4,7),(5,7),(6,7) \\
\text{Cut 5:} & (4,7),(5,7),(2,6)
\end{array}
$$

$G = (V, E)$ is a directed graph. $V$ is the set of vertices and $E$ is the set of edges.

$s \in V$ is the source (start node). $t \in V$ is the sink (end node). $c_{uv}$ is the capacity of edge $(u, v) \in E$.

An s-t cut $C(S, T)$ is a partition of $V$ such that $s \in S$ and $t \in T$, as it is a division of the vertices of the network into two parts, with the source in one part and the sink in the other.

The cut-set $X_c$ of a cut $C$ is a the set of edges that connect the source part of the cut to the sink part (edges directed from a node in $S$ to a node in $T$).

$$X_c := \{(u, v) \in E : u \in S, v \in T\}$$

The capacity of an s-t cut, denoted $c(S, T)$, is the total capacity of its edges,

$$c(S, T) := \sum_{(u,v) \in X_c} c_{uv}$$

## The Transportation Problem LP Formulation

$$
\begin{aligned}
\min \quad & \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} \\
\text{s.t.} \quad & \sum_{j=1}^{n} x_{ij} \le a_i && \text{for } i = 1, 2, \ldots, m \\
& \sum_{i=1}^{m} x_{ij} \ge b_j && \text{for } j = 1, 2, \ldots, n \\
& x_{ij} \ge 0 && \text{for } i = 1, 2, \ldots, m \text{ and } j = 1, 2, \ldots, n
\end{aligned}
$$

### The Transportation Problem

Given $m = 3$ and $n = 2$; $a_1 = 45$, $a_2 = 60$ and $a_3 = 35$; $b_1 = 50$ and $b_2 = 60$; and finally, $c_{11} = 3$, $c_{12} = 2$, $c_{21} = 1$, $c_{22} = 5$, $c_{31} = 5$, and $c_{32} = 4$. Then, substitution of these values into the above formulation leads to the following explicit problem

$$
\begin{aligned}
\min \quad & 3x_{11} + 2x_{12} + x_{21} + 5x_{22} + 5x_{31} + 4x_{32} \\
\text{s.t.} \quad & x_{11} + x_{12} \le 45 \\
& x_{21} + x_{22} \le 60 \\
& x_{31} + x_{32} \le 35 \\
& x_{11} + x_{21} + x_{31} \ge 50 \\
& x_{12} + x_{22} + x_{32} \ge 60 \\
& x_{ij} \ge 0
\end{aligned}
$$

### Traffic Routing Problem

All the links in this network are assume to be bidirectional, and the traffic demand (total in both directions) between node $i$ and $j$ is denoted $D_{ij}$. The bandwidth capacity of link $\{ij\}$ is denoted $B_{ij}$. Traffic between nodes 1 and 2 is the sum of the direct traffic between 1 and 2 denoted $X_{12}$ and the traffic between 1 and 2 denoted $X_{12}$ and the traffic between 1 and 2 that is routed through Node 3, denoted $X_{132}$. As a results, we must satisfy the constraint $X_{12} + X_{132} = D_{12}$.

$$
\begin{aligned}
\min \quad & P = X_{12} + X_{13} + X_{23} + 2X_{132} + 2X_{213} + 2X_{123} \\
\text{s.t.} \quad & X_{12} + X_{132} \le D_{12} && X_{12} \ge 0 \\
& X_{13} + X_{123} = D_{13} && X_{13} \ge 0 \\
& X_{23} + X_{213} = D_{23} && X_{23} \ge 0 \\
& X_{13} + X_{132} + X_{213} \le B_{13} && X_{123} > 0 \\
& X_{12} + X_{123} + X_{213} \le B_{12} && X_{132} \ge 0 \\
& X_{23} + X_{132} + X_{123} \le B_{23} && X_{213} \ge 0
\end{aligned}
$$

**Least Squares**

Let $(x_i, y_i)$, $i = 1, 2, 3, \ldots, n$ is a set of given data points where $x_i$, is an independent variable and $y_i$ is a dependent variable.

$f(x, \mathbf{p})$ is the model function, where $x$ represents an independent variable and $p$ is a vector of parameters that we need to tune so that $f(x, \mathbf{p})$ is a best-fit predictor of $y$ which is the dependent variable corresponding to the independent variable $x$.

Therefore, the fit of a model to a data point is measured by its residual,

$$r_i = y_i - f(x_i, \mathbf{p})$$

The least-squares method finds the optimal parameter values by minimising the sum, $S$, of the squared residuals,

$$S = \sum_{i=1}^{n} (r_i)^2 = \sum_{i=1}^{n} (y_i - f(x_i, \mathbf{p}))^2$$

Least Squares for Simple Linear Regression

$\bar{x}$ is the average of all the $x_i$ as

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$$

$\bar{y}$ is the average of all the $y_i$ as

$$\bar{y} = \frac{\sum_{i=1}^{n} y_i}{n}$$

Let $\hat{a}$ and $\hat{b}$ denote the optimal solution for $a$ and $b$, respectively.

The analytic solution can be defined as

$$\hat{a} = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n} (x_i - \bar{x})^2}$$
$$\hat{b} = \bar{y} - \hat{a}\bar{x}$$

## Least Squares Excel Steps

1. Input $x$ and $y$ data
2. Select all data and go to Insert section
    1. Scatter plot
    2. Quick Layout
        1. Change to have $f(x)$ function plot
3. Go to Data section and use Data Analysis
    1. Input $y$ and $x$ range
    2. Toggle on Residuals
4. We need $Residual^2$, so to use Excel formula `sum=…^2`
5. Finally use `sum=(…)` to find the

## 0,1 Knapsack Problem

### ILP Formulation

$$\max_{x_i} \sum_{i=1}^{N} v_i x_i$$

and it is subjects to

$$\sum_{i=1}^{N} w_i x_i \leq W$$

All the $w_i$ are positive integers.

$$x_i = 0 \text{ or } 1 \qquad i = 1, 2, 3, \ldots, N$$

### LP Formulation

Using the above formula and constraint with different variable's constraints,

$$1 \geq x_i \geq 0, \qquad i = 1, 2, 3, \ldots N$$

### DP Formulation

Optimal value function is that $S(k, w)$ equals to the best (maximum) value of all possible solution of weight less or equal to $w$ consisting only of 0 or 1 items of types $k$, $k+1$, …, $N$ and $w$ should starts from $0, 1, \ldots, W$.

Recurrence relation is defined as

$$S(k,w) = \max_{x_k=0,1(\text{weight}\leq w)} [x_k v_k + S(k+1, w - x_k w_k)]$$

Boundary conditions is defined as

$$\begin{aligned} S(N+1,w) &= 0, && w \geq 0 \\ S(k,w) &= -\infty, && w < 0 \text{ for all } k \end{aligned}$$

**ILP Excel Steps**

1. Input all the weight such as $w$, $v$, $W$ and $x$ (which the initial values should be 1)
2. Create $\max$ function using `SUMPRODUCT(…,…)`
3. Create subject function using `SUMPRODUCT(…)`
4. Add a constraints about all $x$ is binary
5. Select solving method be Simplex LP

**LP Excel Steps**

1. Input all the weight such as $w$, $v$, $W$ and $x$ (which the initial values should be 1)
2. Create $\max$ function using `SUMPRODUCT(…,…)`
3. Create subject function using `SUMPRODUCT(…)`
4. Add a constraints about $0 \leq x \leq 1$
5. Select solving method be Simplex LP

**Bounded Knapsack Problem (BKP)**

$$\max_{x_i} \sum_{i=1}^{N} v_i x_i$$

and it is subjects to

$$\sum_{i=1}^{N} w_i x_i \leq W$$

where its constraints is

$$c_i \geq x_i \geq 0, \qquad i = 1, 2, 3, \ldots, N$$

and

$$x_i \text{ integer}, \qquad i = 1, 2, 3, \ldots, N$$

**BKP with min**

$$\min_{x_i} \sum_{i=1}^{N} c_i x_i$$

Subjects to:
$\sum_{i=1}^{N} w_i x_i \geq W$

where its constraints is

$$\begin{aligned} c_i \geq x_i \geq 0, \qquad & i = 1, 2, 3, \ldots, N \\ x_i \text{ integer}, \qquad & i = 1, 2, 3, \ldots, N \end{aligned}$$

**Unbounded Knapsack Problem (UKP)**

$$\max_{x_i} \sum_{i=1}^{N} v_i x_i$$

and it is subjects to

$$\sum_{i=1}^{N} w_i x_i \leq W$$

where its constraints are

$$\begin{aligned} x_i \geq 0, \qquad & i = 1, 2, 3, \ldots, N \\ x_i \text{ integer}, \qquad & i = 1, 2, 3, \ldots, N \end{aligned}$$

**UKP with min**

$$\min_{x_i} \sum_{i=1}^{N} c_i x_i$$

and it is subjects to

$$\sum_{i=1}^{N} w_i x_i \geq W$$

where its constraints is

$$\begin{aligned} x_i \geq 0, \qquad & i = 1, 2, 3, \ldots, N \\ x_i \text{ integer}, \qquad & i = 1, 2, 3, \ldots, N \end{aligned}$$