www.cwthompson.com

D1 Sorting and Searching

Bubble Sort

8	3	4	6	5	7	2
8	4	6	5	7	3	2
8	6	5	7	4	3	2
8	6	7	5	4	3	2
8	7	6	5	4	3	2
8	7	6	5	4	3	2

- Sort an unsorted list in alphabetical or numerical order
- Go through the list comparing adjacent values, swapping them if not in order
- At the end of the list, repeat
- When a pass contains no swaps the list is ordered

Quick Sort

22	17	25	30	11	18	20	14	7	29
17	11	14	7	$\boxed{18}$	22	25	30	20	29
11	7	$\boxed{14}$	17	$\boxed{18}$	22	25	20	29	$\boxed{30}$
$\boxed{7}$	11	$\boxed{14}$	$\boxed{17}$	$\boxed{18}$	$\boxed{20}$	22	25	29	$\boxed{30}$
$\boxed{7}$	$\boxed{11}$	$\boxed{14}$	$\boxed{17}$	$\boxed{18}$	$\boxed{20}$	22	$\boxed{25}$	29	$\boxed{30}$
$\boxed{7}$	$\boxed{11}$	$\boxed{14}$	$\boxed{17}$	$\boxed{18}$	$\boxed{20}$	$\boxed{22}$	$\boxed{25}$	$\boxed{29}$	$\boxed{30}$

- Quick and efficient
- Choose the midpoint of the list as the first pivot
- Put all smaller values before the pivot, larger values after the pivot
- Repeat for each created sublist until all values are used as midpoints

Binary Search

1 (Berry) 2 (Connock) 3 (Curtis) 4 (Joe)
1 (Berry) 2 (Connock)
1 (Berry)
Berry found in position 1

- Searches an ordered list for a specific value
- Always select the midpoint of the sub-list
- If the target value comes before the midpoint discard the second half
- If the target value comes after the midpoint discard the first half
- Repeat until the target value is located or the sub-list contains no values
- Midpoint = $\left[\frac{F+L}{2}\right]$ where F is first position and L is last position

www.cwthompson.com

First-Fit Bin Packing

0.6 1.5 1.6 0.2 0.4 0.5 0.7 0.1 0.9 0.3

Bin 1 : 0.6, 0.2, 0.4, 0.5, 0.1 (0.2w)

Bin 2 : 1.5, 0.3 (0.2w)

Bin 3 : 1.6 (0.4w)

Bin 4 : 0.7, 0.9 (0.4w)

- Take the items in the order given
- Place each item in the first available bin that can hold it, always starting from bin 1
- Quick but rarely leads to a good solution

First-Fit Decreasing Bin Packing

1.6 1.5 0.9 0.7 0.6 0.5 0.4 0.3 0.2 0.1

Bin 1 : 1.6, 0.4 (-)

Bin 2 : 1.5, 0.5 (-)

Bin 3 : 0.9, 0.7, 0.3, 0.1 (-)

Bin 4 : 0.6, 0.2 (1.2w)

- Reorder items into descending order
- Apply first-fit bin packing
- Usually obtain a good solution and easy to do
- Does not usually lead to the optimal solution

Full-Bin Packing

0.6 1.5 1.6 0.2 0.4 0.5 0.7 0.1 0.9 0.3

Bin 1 : 1.6, 0.4 (-)

Bin 2 : 1.5, 0.5 (-)

Bin 3 : 0.9, 0.7, 0.3, 0.1 (-)

Bin 4 : 0.6, 0.2 (1.2w)

- Use observations to make combinations that lead to full bins
- Use first-fit bin packing on remaining items
- Usually leads to a good solution but is difficult to do

Lower Bound

0.6 1.5 1.6 0.2 0.4 0.5 0.7 0.1 0.9 0.3

$\sum x = 6.8$

Bin Size = 2

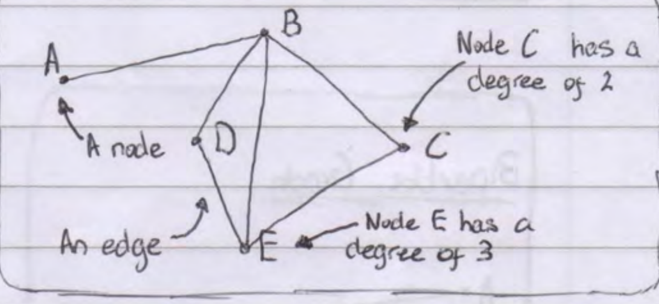
Lower Bound = $\frac{6.8}{2}$ = 3.4 → 4 bins

- Used to obtain the minimum number of bins required
- Divide the total item size by bin size and round up

D1 Graphs and Networks

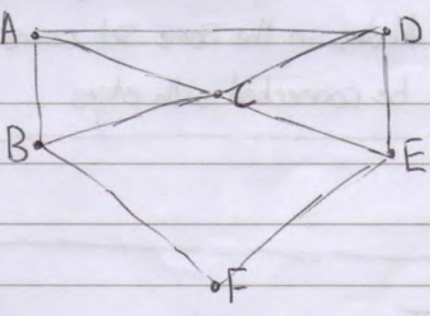
- A graph consists of points (vertices/nodes) which are connected by lines (edges/arcs)
- If a graph has a number associated with each edge (its weight) it is known as a network or weighted graph

- Nodes are written as A, B, C, etc.
- Edges are written as AB, AC, etc.
- A list of nodes is a vertex set and a list of edges is an edge set
- The degree/valency/order of a node is the number of edges incident to it



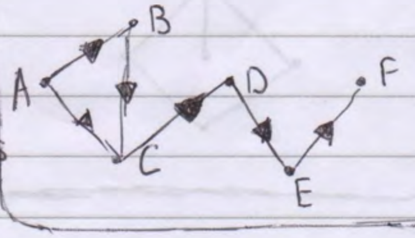
- The 'Handshaking Lemma' states the sum of degrees will be equal to twice the number of edges

- A path is a finite sequence of connected edges in which no node appears twice
- A walk is a path in which you can return to nodes
- A cycle is a closed path, starting and ending at the same node

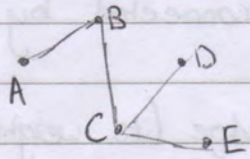


Path Example : A B F E C
Walk Example : A B C D A C E
Cycle Example : A B F E C A

- A subgraph of a graph is part of the original graph
- Two nodes are connected if there is a path between them and a graph is connected if all nodes are
- A loop is an edge that starts and finishes at the same node
- A simple graph contains no loops and has no more than one edge connecting two nodes
- A graph with directed edges is a digraph



Tree

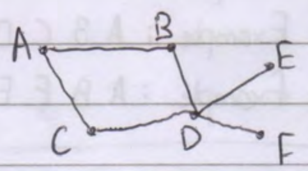


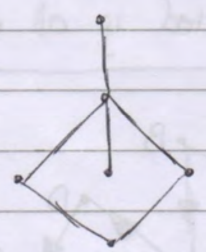
- A connected graph with no cycles

Bipartite Graph

- Consists of two sets of nodes
- Nodes in the same set cannot be connected with edges

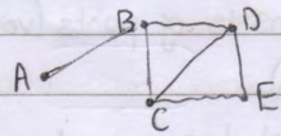
Isomorphic Graphs

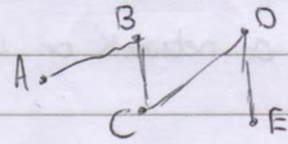




- Graphs that shows the same information in different ways
- Same number of nodes and edges connected in the same way

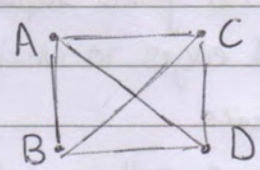
Spanning Tree





- A subgraph including all nodes and is a tree
- Graph 2 is a spanning tree of graph 1

Complete Graph



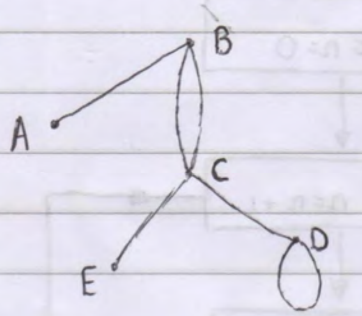
- All nodes are directly connected to other nodes
- Denoted by K_n where n is the number of nodes

www.cwthompson.com

D1 Matrices

Adjacency Matrix

- Records the number of direct links between nodes

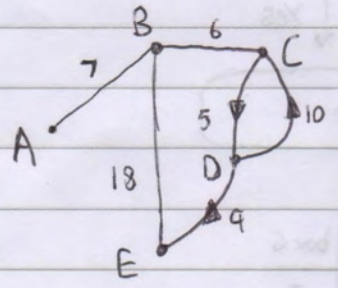


	A	B	C	D	E
A	0	1	0	0	0
B	1	0	2	0	0
C	0	2	0	1	1
D	0	0	1	2	0
E	0	0	1	0	0

- 0 means no connections, 1 means one connection, 2 means two connections, etc.
- A loop can be travelled in either direction so counts as two connections

Distance Matrix

- Records the weight on each edge



	A	B	C	D	E
A	-	7	-	-	-
B	7	-	6	-	18
C	-	6	-	5	-
D	-	-	10	-	9
E	-	18	-	-	-

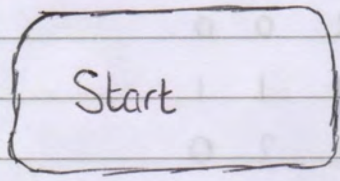
- In a directed network the matrix will not be symmetrical
- If there is no edge it is represented by -

www.cwthompson.com

D1 Algorithm Flow Charts

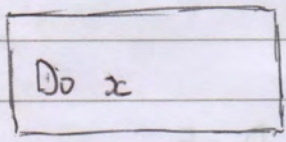
- Flow charts are used to express the steps in an algorithm
- The order is represented by arrows

Start / Stop



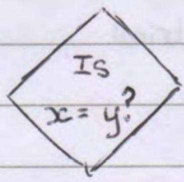
- Placed at the Start and end of the flow chart

Sequence

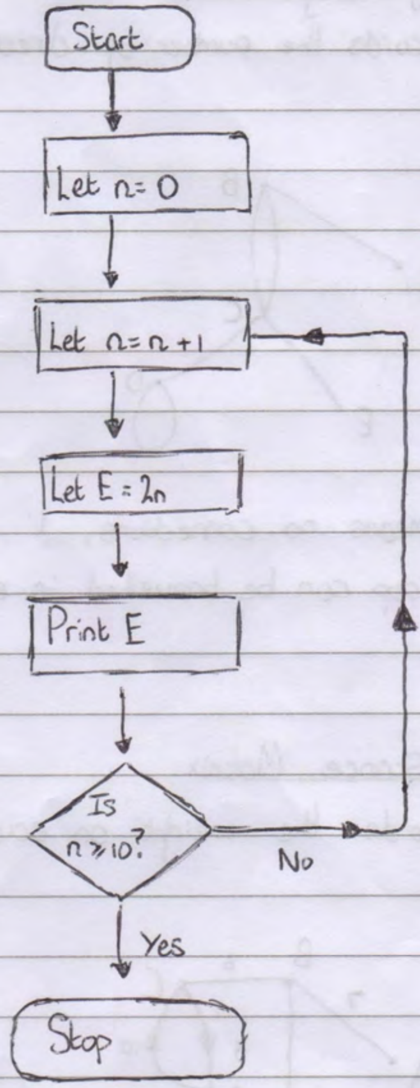


- Explain the steps of the algorithm
- The most common box

Selection / Decision



- Used for repetition or selection in the algorithm, different paths are followed
- Usually a yes/no question with each selecting a different path



n	E	box 6
0	–	–
1	2	no
2	4	no
3	6	no
4	8	no
5	10	no
6	12	no
7	14	no
8	16	no
9	18	no
10	20	yes

Outputs 2, 4, 6, 8, 10, 12
14, 16, 18, 20

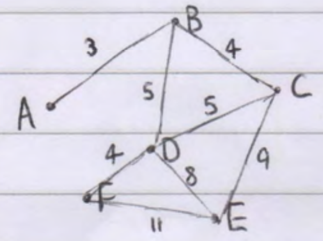
www.cwthompson.com

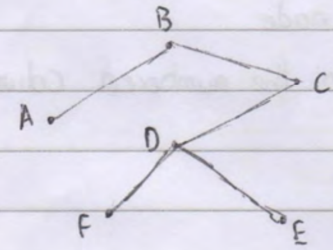
D1 Minimum Spanning Tree

- A minimum spanning tree is a spanning tree such that the total lengths of its arcs is as small as possible

Kruskal's Algorithm

- Sort the arcs into ascending order of weight
- Start with the arc of least weight
- Select the next arc of least weight and add it to the tree if it does not form a cycle
- Repeat until all nodes are connected





AB(3), BC(4), DF(4), BD(5), CD(5), DE(8), CE(9), EF(11)

Select AB

Add BC

Add DF

Add CD

Reject BD

Add DE

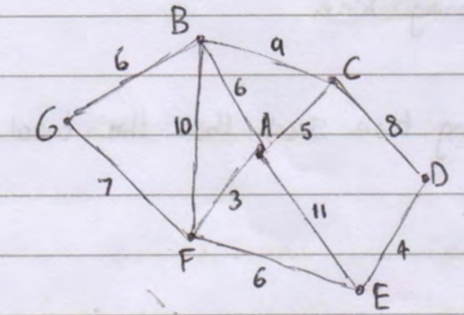
AB, BC, DF, CD, DE

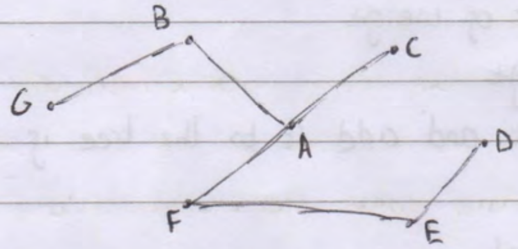
Weight = 24

Prim's Algorithm

- Choose any node to start with
- Select the arc of least weight that connects a node in the tree to a node not in the tree
- Repeat until all nodes are connected

www.cwthompson.com





Start with A
Add AF (3)
Add AC (5)
Add EF (6)
Add DE (4)
Add AB (6)
Add BG (6)

AF, AC, EF, DE, AB, BG
Weight = 30

Prim's Algorithm on a Distance Matrix

- Start with any node
- Delete the row in the matrix for the chosen node
- Number the column in the matrix for the chosen node
- Put a ring around the lowest undeleted value in the numbered columns
- The ringed value becomes the next arc
- Repeat until all rows are deleted

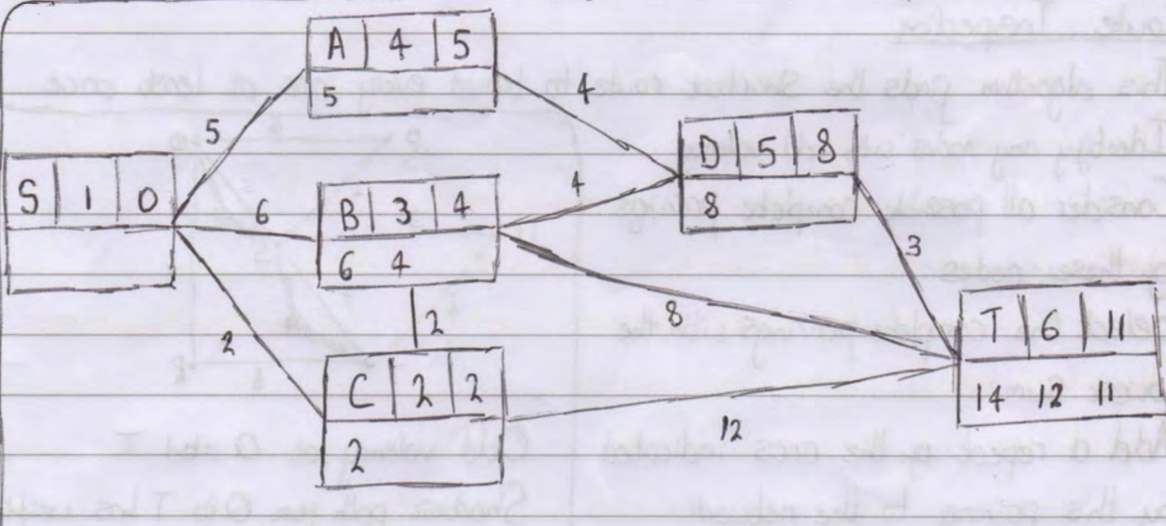
	1	4	2	3	5
	A	B	C	D	E
A	~~-~~	~~27~~	~~12~~	~~23~~	~~74~~
B	~~27~~	~~-~~	~~47~~	~~15~~	~~71~~
C	~~12~~	~~47~~	~~-~~	~~28~~	~~87~~
D	~~23~~	~~15~~	~~28~~	~~-~~	~~75~~
E	~~74~~	~~71~~	~~87~~	~~75~~	~~-~~

Start with A
Add AC
Add AD
Add DB
Add BE

AC, AD, DB, BE
Weight = 121

D1 Dijkstra's Algorithm

- Dijkstra's algorithm is used to find the shortest path between two nodes
- First label the start node, S, with the final label, 0
- Record a working value at every node directly connected to the node which just received a label
- If there is already a working value it is only replaced if the new value is smaller
- The node with the smallest working value gets the final label equal to its working value.
- Repeat until the destination node is reached
- Trace back to the start node for the shortest path



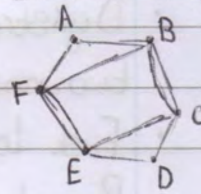
- Starting at S
- New Working Values: A(5), B(6), C(2)
- C selected
- New Working Values: B(4), T(14)
- B selected
- ...
- T selected
- Length of shortest route is therefore 11
- Working back to S through DT, BD, CB, SC
- Shortest path is S C B D T with length 11

Vertex	Order of Labelling	Final Value
Working Values		

D1 Route Inspection

- If all valencies in a graph are even, the graph is Eulerian
- If two valencies are odd the graph is Semi-Eulerian

Node	A	B	C	D	E	F
Valency	2	4	4	2	4	4



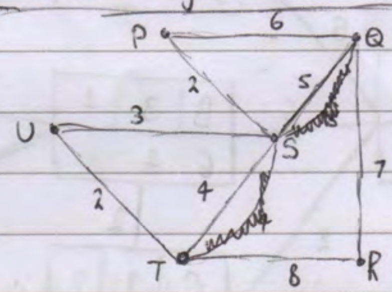
All valencies even, graph is Eulerian

- A graph is traversable if it can be traversed by every arc once without taking pen from paper. It must be Eulerian.
- Semi-Eulerian graphs are semi-traversable
- A graph with more than two odd valencies is not traversable

Route Inspection

- This algorithm finds the shortest route to travel every arc at least once
- Identify any nodes with odd valency
- Consider all possible complete pairings of these nodes
- Select the complete pairings with the lowest sum
- Add a repeat of the arcs indicated by this pairing to the network

- The length of the shortest path on a Eulerian graph is equal to the weight of the network



Odd valency at Q and T

Shortest path from Q to T has weight 9

Weight of network is 37

$\therefore$ Length $= 37 + 9$

$= 46$

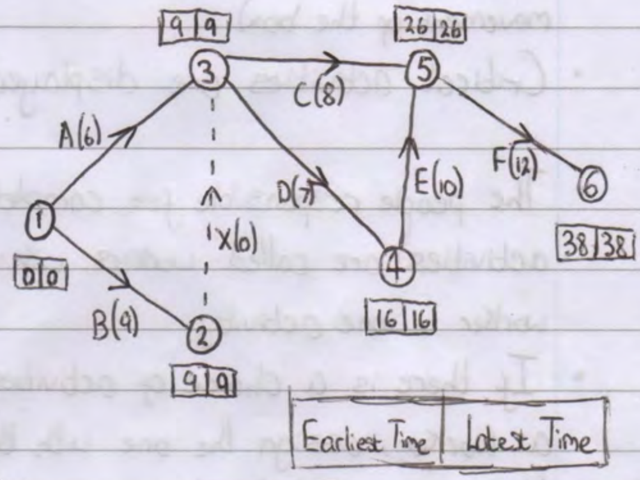
Possible Route: SQPSQRTSUTS

www.cwthompson.com

Critical Path Analysis

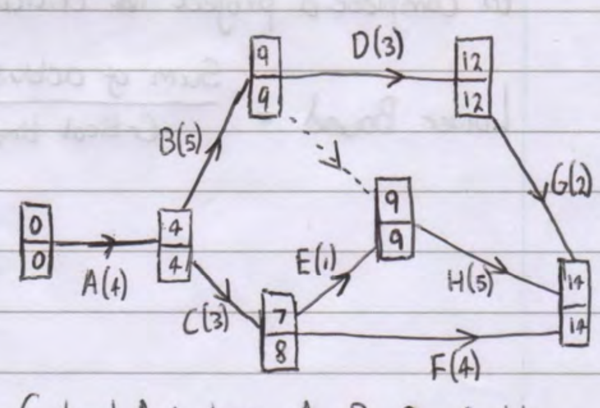
- Precedence tables (dependence tables) are used to show the order in which activities must be completed.
- They do not have to include weight of activities.

Activity	Depend On	Days
A (train workers)	-	6
B (purchase material)	-	9
C (produce product 1)	A, B	8
D (produce product 2)	A, B	7
E (test product 2)	D	10
F (assemble products)	C, E	12



- An activity on arc network can be used to represent a precedence table
- The nodes represent events
- The arcs represent activities
- The weights represent duration
- The source node (usually numbered 0 or 1) is the first node
- The sink node is the last node and the end of the project
- There can only be one activity between events so dummy activities can be used which have a duration of 0
- The early event time is the earliest time we can arrive at the event
- The late event time is the latest time we can leave the event without extending the project time

- Critical activities are activities where increasing the duration causes an equivalent increase in project duration
- A critical path is the longest network path
- Early event time = Late event time for critical events
- Total float of an activity is the maximum delay without affecting project duration
- Total float = Late Event Time - Duration - Early Event Time



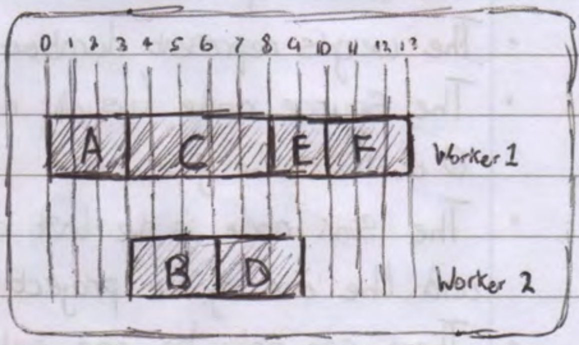
Critical Activities: A, B, D, G, H

Critical Paths: ABDG and ABH

www.cwthompson.com

Cascade/Gantt Charts

- Gantt charts are used to display the possible start and finish times for all activities
- The number scale shows elapsed time, so the first period of time is between 0 and 1
- The total float of activities is shown by the dotted boxes (which is the range of movement of the box)
- Critical activities are displayed in a line along the top



- The people responsible for completing activities are called workers, with one worker to one activity
- If there is a choice of activities for a worker, assign the one with the lowest 'latest finish time' value
- A scheduling diagram (right) can be used to show which activities are assigned to which workers
- The process of assigning workers to activities is known as Scheduling

- The lower bound for number of workers to complete a project in critical time is:

$$\text{Lower Bound} = \frac{\text{Sum of activity times}}{\text{Critical time}}$$

Total Duration = 19

Critical Time = 13

$19/13 = 1.4615...$

$\therefore$ Lower Bound = 2

Linear Programming

- Decision variables are the numbers that can be changed
- The objective function is the aim of the problem, and is usually to maximise ~~or~~ minimise a value.
- Constraints prevent variables from being negative(sometimes) or infinite
- Each constraint gives one inequality equation
- Values for decision variables that satisfy each constraint give a feasible solution
- When plotted on a graph the area containing all feasible solutions is the feasible region
- The optimal solution is the feasible solution that meets the objective

Curtis is making chocolate cakes and fruit cakes, Each fruit cakes requires 1 egg, 250g flour, 200g sugar. Each chocolate cake requires 2 eggs, 250g flour, 300g sugar. Curtis has 36 eggs, 7kg flour, 6kg sugar.
He sells fruit cakes for £3.50 and chocolate cakes for £5, and wants to maximise money.

Let f be fruit cakes made
Let c be chocolate cakes made

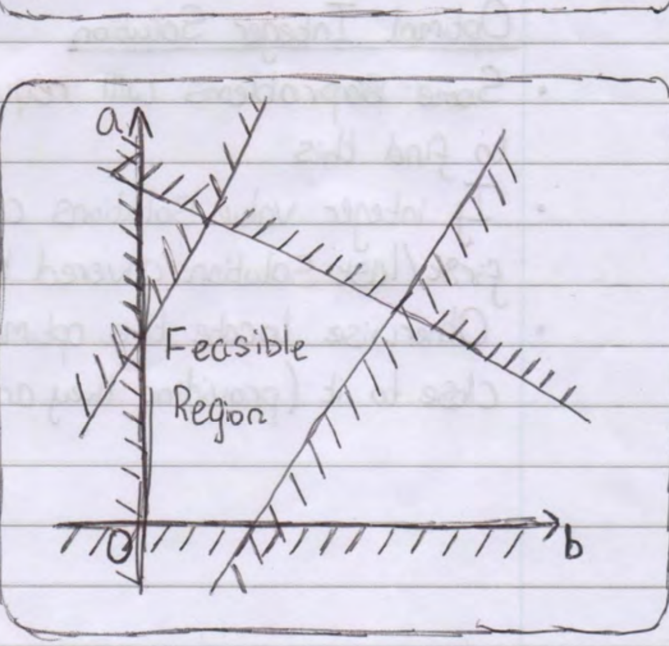
Maximise $P = 3.5f + 5c$

eggs: $f + 2c \leq 36$
flour: $250f + 250c \leq 7000$
$\therefore f + c \leq 28$
Sugar: $200f + 300c \leq 6000$
$\therefore 2f + 3c \leq 60$

$f, c \geq 0$

- To formulate a linear programming problem you must:
  - Define decision variables
  - State the objective
  - Write constraints as inequalities

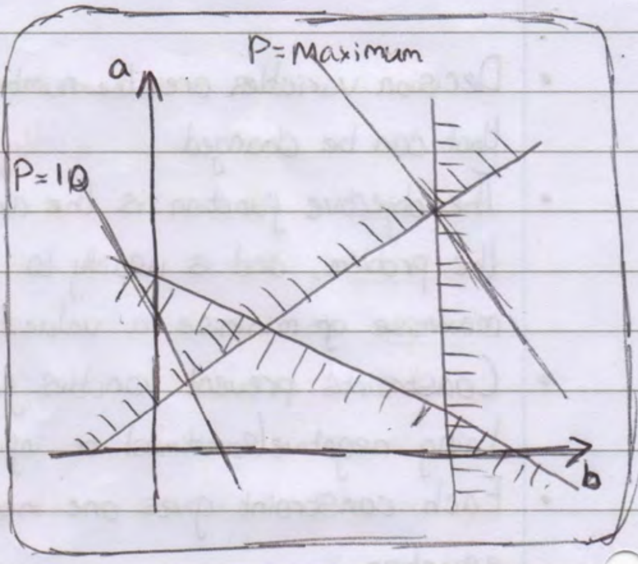
- A problem can be expressed graphically by drawing all inequalities, and shading the sides which don't match
- The unshaded area is the feasible region



www.cwthompson.com

Objective Line Method

- Identifies optimal point in feasible region
- Draw an objective line for any value (of the objective function)
- All objective lines will be parallel
- For a maximum point find the last point ~~by~~ covered by an objective line as it leaves the feasible region
- For a minimum point find the first point covered by an objective line as it enters the feasible region



Vertex Testing Method

- The optimal point will be at a vertex
- Evaluate the objective function at each pair of vertex coordinates
- Select the vertex that gives the optimal value

Vertex	Coord	Value $(2x+y)$
A	$x=4,\ y=11$	19
B	$x=4.5,\ y=6$	15
C	$x=9,\ y=10.5$	28.5
D	$x=16,\ y=20$	52

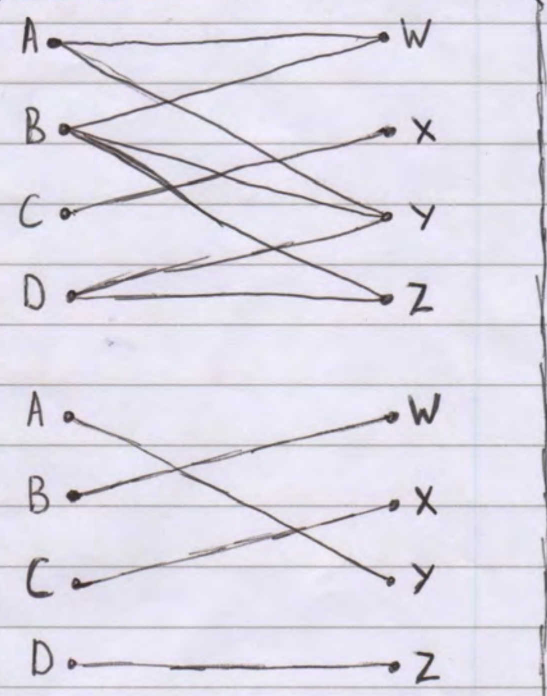
Minimum value at vertex B (15)

Optimal Integer Solution

- Some ~~pr~~problems will require integer solutions, there are two ways to find this
- If integer value solutions can be plotted accurately, select the first/last solution covered by an objective line
- Otherwise locate the optimal non-integer solution and test integer solutions close to it (providing they are in the feasible region).

Matchings

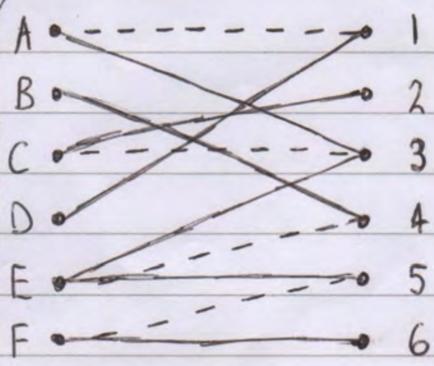
- A bipartite graph has two sets of nodes, and arcs connect nodes of different sets but not the same set.
- A matching is the 1-to-1 pairing of nodes from one set to nodes of the other set. Nodes are paired.
- A complete matching is where both sets have n nodes and the matching has n arcs
- A maximal matching is a matching with the maximum number of arcs, and may not always be a complete matching



Complete Matching

- The maximum matching algorithm (matching improvement algorithm) is used to find an improved matching from an initial matching
- An alternating path starts at an unmatched node in one set and ends at an unmatched node in the other set, using arcs that are alternately in or not in the initial matching

- An alternate path is searched for
- If found, the status of each arc is changed
- The new matching is listed
- This is repeated until the matching is complete



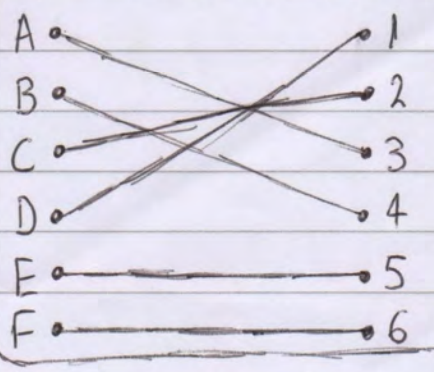
— Possible Matching

--- Initial Matching

B, D, 2, 6 no initial matching

B - 4 = E - 5 = F - 6 c.s. B=4 - E=5 - F=6

D - 1 = A - 3 = C - 2 c.s. D=1 - A=3 - C=2



www.cwthompson.com

Definitions

Bipartite Graph — a graph consisting of two sets of nodes where a node in one set can only connect to nodes in the other

Complete Matching — a matching where every node has an edge incident to it

Critical Activity — an activity where increasing its duration results in an increase of the overall project duration

Critical Path — a path from source node to sink node only following critical activities

Cycle — a closed path

Digraph — a graph with directions associated with some edges

Early Event Time — the earliest time at which all dependent events can be completed

Eulerian Graph — a graph where all valencies (of nodes) are even

Graph — consists of points (nodes) connected by lines (edges)

Isomorphic Graphs — different graphs showing the same information

Late Event Time — the latest time at which any of the dependent events can be completed without delaying the project

Loop — an edge starting and finishing at the same node

Matching — a 1 to 1 pairing of some or all nodes on a bipartite graph

Maximal Matching — a matching including as many edges as possible

Minimum Spanning Tree — a spanning tree such that the total length of its edges is as short as possible

Path — a finite sequence of edges, such that the end node of one edge in the sequence is the start node of the next, and in which no nodes appears more than once

Semi-Eulerian Graph — a graph where only two valencies (of nodes) are odd

Simple Graph — a graph with no loops and no more than one edge connecting any pair of nodes

Spanning Tree — a subgraph containing all nodes of the original graph, and is a tree

Subgraph — part of a graph

www.cwthompson.com

Total Float	the amount of time an activity may be delayed without affecting the project duration
Tree	a connected graph with no cycles
Walk	a path which can include the same node more than once
Weighted Graph	a graph with values associated with some of the edges
Valency	number of edges incident to a node