Scheduling and Sequencing

Outline

- **Project Planning**
  - CPM
  - PERT
- **Project Planning with Resource Constraints**
- **Sequencing**
  - One Machine Problem
  - Two Machine Problem
  - Three Machine Problem

Project Planning

- **Network Representation**
  - Large projects can be seen at a glance with interaction and activities.
  - Bottlenecks can often be foreseen.
  - Cost range can be determined by interactive procedure.

Project Planning

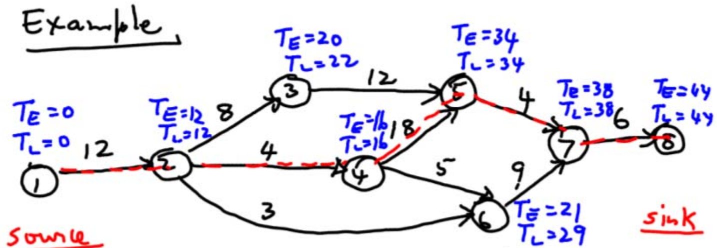
- **Critical Path Method (CPM)**
- **Program Evaluation Review Technique (PERT)**
  - These are network techniques for analyzing a system in terms of **activities** (jobs) and **events** that must be completed in a specified sequence in order to achieve a goal.

Project Planning

- **Activities**
  - Component tasks that take time and are represented by arrows.
  - Activities which require zero time are designated by dashed arrows and are sometimes called dummy activities.
- **Events**
  - Points in time that indicate that some activities have been completed and others may begin.
  - These are also known as nodes and are represented by circles.

CPM

Example,



Find all paths from source to sink

Path	Length
1 – 2 – 3 – 5 – 7 – 8	42
1 – 2 – 4 – 5 – 7 – 8	44 * ← Critical Path
1 – 2 – 4 – 6 – 7 – 8	36
1 – 2 – 6 – 7 – 8	30

Project duration = 44

CPM

- **Earliest time for an event can be determined by computing the largest sum of activity times on paths leading to the event.**
- **Latest time for an event can be computed by starting at the end of the network and working backwards. When two or more paths converge on one event, the shortest time governs.**

PERT

- **Like CPM, this is also time-oriented planning and control device.**
- **PERT develops both a measure of control tendency (mean) and a measure of dispersion (standard deviation).**

PERT

◆ **Beta distribution**

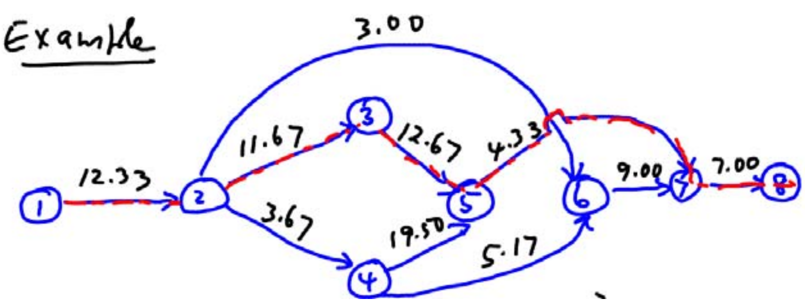
$$Mean = \frac{a + 4m + b}{6}$$

$$Variance = \left(\frac{b - a}{6}\right)^2$$

– where
– ***a*** is the most **optimistic** duration
– ***m*** is the most **likely** duration
– ***b*** is the most **pessimistic** duration

PERT

Example



PERT

Activity	a	m	b	μ	σ^2	
1-2	10	12	16	12.33	1.00	*
2-3	2	8	36	11.67	32.11	*
2-4	1	4	5	3.67	0.44	
2-6	2	3	4	3.00	0.11	
3-5	8	12	20	12.67	4.00	*
4-5	15	18	30	19.50	6.25	
4-6	3	5	8	5.17	0.69	
5-7	2	4	8	4.33	1.00	*
6-7	6	9	12	9.00	1.00	
7-8	4	6	14	7.00	2.78	*

PERT

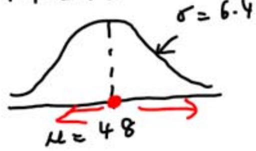
Path	Duration	
1-2-3-5-7-8	48.00 *	← critical Path
1-2-4-5-7-8	46.83	
1-2-4-6-7-8	37.17	
1-2-6-7-8	31.33	

∴ The mean project duration = 48 days

Standard deviation of the project duration

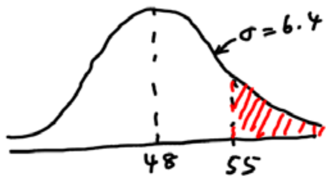
$$= \sqrt{1 + 32.11 + 4 + 1 + 2.78}$$

$$= 6.4 \text{ days}$$



PERT

Find the probability that the project will take more than 55 days



$$z = \frac{55-48}{6.4} = 1.09$$

Prob. that the project will take more than 55 days = 0.14

PERT

THE CUMULATIVE STANDARDIZED NORMAL DISTRIBUTION FUNCTION

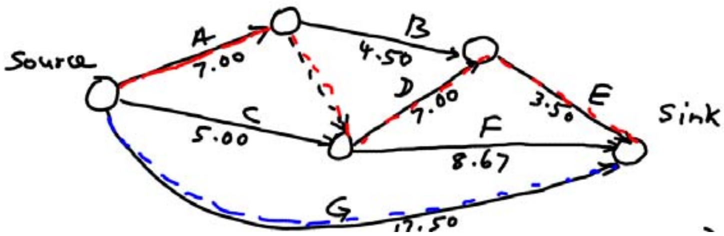
(Note: $.9^{2}8650 = .998650$)

Entry = $P(Z < Z_{1-\alpha}) = 1 - \alpha$

$Z_{1-\alpha}$	.00	.01	.02	.03	.04	.05	.06	.07	.08	.09
.0	.5000	.5040	.5080	.5120	.5160	.5199	.5239	.5279	.5319	.5359
.1	.5398	.5438	.5478	.5517	.5557	.5596	.5636	.5675	.5714	.5753
.2	.5793	.5832	.5871	.5910	.5948	.5987	.6026	.6064	.6103	.6141
.3	.6179	.6217	.6255	.6293	.6331	.6368	.6406	.6443	.6480	.6517
.4	.6554	.6591	.6628	.6664	.6700	.6736	.6772	.6808	.6844	.6879
.5	.6915	.6950	.6985	.7019	.7054	.7088	.7123	.7157	.7190	.7224
.6	.7257	.7291	.7324	.7357	.7389	.7422	.7454	.7486	.7517	.7549
.7	.7580	.7611	.7642	.7673	.7703	.7734	.7764	.7794	.7823	.7852
.8	.7881	.7910	.7939	.7967	.7995	.8023	.8051	.8078	.8106	.8133
.9	.8159	.8186	.8212	.8238	.8264	.8289	.8315	.8340	.8365	.8389
1.0	.8413	.8438	.8461	.8485	.8508	.8531	.8554	.8577	.8599	.8661
1.1	.8643	.8665	.8686	.8708	.8729	.8749	.8770	.8790	.8810	.8830
1.2	.8849	.8869	.8888	.8907	.8925	.8944	.8962	.8980	.8997	.90147
1.3	.90320	.90490	.90658	.90824	.90988	.91149	.91309	.91466	.91621	.91774
1.4	.91924	.92073	.92220	.92364	.92507	.92647	.92785	.92922	.93056	.93189
1.5	.93319	.93448	.93574	.93699	.93822	.93943	.94062	.94179	.94295	.94408
1.6	.94520	.94630	.94738	.94845	.94950	.95053	.95154	.95254	.95352	.95449
1.7	.95543	.95637	.95728	.95818	.95907	.95994	.96080	.96164	.96246	.96327
1.8	.96407	.96485	.96562	.96638	.96712	.96784	.96856	.96926	.96995	.97062
1.9	.97128	.97193	.97257	.97320	.97381	.97441	.97500	.97558	.97615	.97670
2.0	.97725	.97778	.97831	.97882	.97932	.97982	.97030	.98077	.98124	.98169
2.1	.98214	.98257	.98300	.98341	.98382	.98422	.98461	.98500	.98537	.98574
2.2	.98610	.98645	.98679	.98713	.98745	.98778	.98809	.98840	.98870	.98899
2.3	.98928	.98956	.98983	$.9^{2}0097$	$.9^{2}0358$	$.9^{2}0613$	$.9^{2}0863$	$.9^{2}1106$	$.9^{2}1344$	$.9^{2}1576$
2.4	$.9^{2}1802$	$.9^{2}2024$	$.9^{2}2240$	$.9^{2}2451$	$.9^{2}2656$	$.9^{2}2857$	$.9^{2}3053$	$.9^{2}3244$	$.9^{2}3431$	$.9^{2}3613$
2.5	$.9^{2}3790$	$.9^{2}3963$	$.9^{2}4132$	$.9^{2}4297$	$.9^{2}4457$	$.9^{2}4614$	$.9^{2}4766$	$.9^{2}4915$	$.9^{2}5060$	$.9^{2}5201$
2.6	$.9^{2}5339$	$.9^{2}5473$	$.9^{2}5604$	$.9^{2}5731$	$.9^{2}5855$	$.9^{2}5975$	$.9^{2}6093$	$.9^{2}6207$	$.9^{2}6319$	$.9^{2}6427$
2.7	$.9^{2}6533$	$.9^{2}6636$	$.9^{2}6736$	$.9^{2}6833$	$.9^{2}6928$	$.9^{2}7020$	$.9^{2}7110$	$.9^{2}7197$	$.9^{2}7282$	$.9^{2}7365$
2.8	$.9^{2}7445$	$.9^{2}7523$	$.9^{2}7599$	$.9^{2}7673$	$.9^{2}7744$	$.9^{2}7814$	$.9^{2}7882$	$.9^{2}7948$	$.9^{2}8012$	$.9^{2}8074$
2.9	$.9^{2}8134$	$.9^{2}8193$	$.9^{2}8250$	$.9^{2}8305$	$.9^{2}8359$	$.9^{2}8411$	$.9^{2}8462$	$.9^{2}8511$	$.9^{2}8559$	$.9^{2}8605$
−3.0	$.9^{2}8650$	$.9^{2}8684$	$.9^{2}8736$	$.9^{2}8777$	$.9^{2}8817$	$.9^{2}8856$	$.9^{2}8893$	$.9^{2}8930$	$.9^{2}8965$	$.9^{2}8999$

PERT

Example.



PERT

Activity	a	m	b	μ	σ^2
A	4	6	14	7.00	100/36
B	3	4	8	4.50	25/36
C	4	5	6	5.00	4/36
D	7	7	7	7.00	0
E	3	3	6	3.50	9/36
F	6	8	14	8.67	64/36
G	13	18	20	17.50	49/36

PERT

Of all paths, there are two paths that are the longest.

$ADE = 7.00 + 7.00 + 3.50 = 17.50$ days

$G = 17.50$ days

Variances

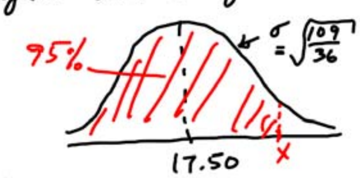
$ADE = \frac{100}{36} + 0 + \frac{9}{36} = \frac{109}{36}$

$G = \frac{49}{36}$

This is the critical path because $\frac{109}{36} > \frac{49}{36}$.

PERT

Find the numbers of days required to finish the project so that we are 95% confident that the project will be finished.



For 95%, $z = 1.64$

$$1.64 = \frac{X - 17.5}{\sqrt{109/36}}$$

$\Rightarrow X = \underline{\underline{20.35 \text{ days}}}$

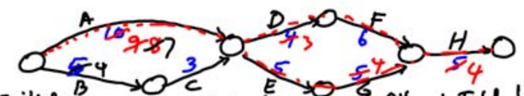
Optimal Crashing Schedule

Example

Job	Predecessor	Normal Time	Crash Time	Cost of Crashing Per day
A	—	10	7	4
B	—	5	4	2
C	B	3	2	2
D	A, C	4	3	3
E	A, C	5	3	3
F	D	6	3	5
G	E	5	2	1
H	F, G	5	4	4

Given that there is an overhead of $5/day of project duration, determine the optimal crashing schedule.

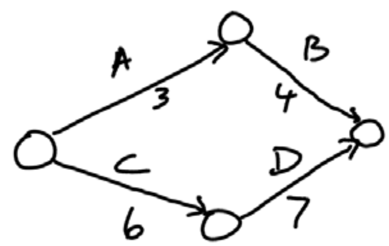
Optimal Crashing Schedule



Iteration	Critical Path(s)	Job crashed (New time)	Added cost due to crash	Proj. Duration	OH savings	Total Cost	Act. Now at min.
0	A-D-F-H A-E-G-H	—	–	25	—	125	—
1	"	H (4)	4	24	5	124	H
2	"	A (9)	4	23	5	123	H
3	All Paths	A (8)	4	22	5	122	H
4	"	D (3), G (4)	3+1	21	5	121	H, D
5	"	A (7), B (4)	4+2	20	5	122	H, D, A, B
6	"	F (5) G (3)	5+1	19	5	123	"
7	"	F (4) G (2)	"	18	5	124	"
8	"	F (3) E (4)	5+3	17	5	127	G H D A B F

Project Planning with Resource Constraints

Example



Activity	Res. Req.
A	2
B	2
C	3
D	2

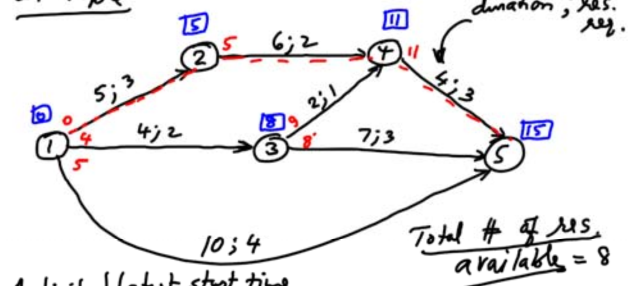
Total # of resources available = 4

Lang's Algorithm

- (Note that this is for allocation of units for a single resource type)
- Order the activities according to the latest start times.
- If there is a tie, break it in the following order:
  - » (a) Activity with the least float is scheduled first.
  - » (b) Activity with the longest duration is scheduled first.
  - » (c) Activity with the largest resource requirement first.
  - » (d) Alphabetically or in numerical order.

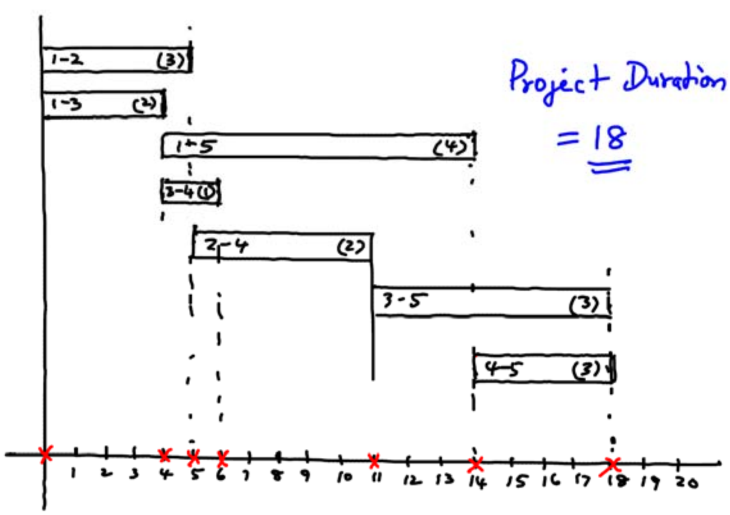
Lang's Algorithm

Example



Activity	Latest start time	
✓1 – 2	0	
✓1 – 3	4	
✓2 – 4	5	Float is 0
✓1 – 5	5	Float is 5
✓3 – 5	8	
3 – 4	9	
4 – 5	11	

Lang's Algorithm



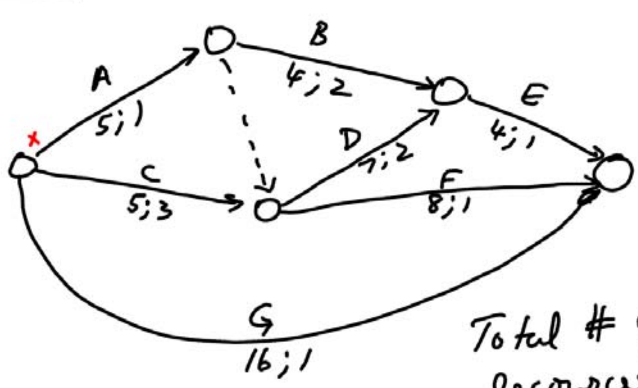
Project Duration = 18

Brook's Algorithm

- (Note that this is for allocation of units for a single resource type)
- Calculate **ACTIM** value for every activity. Then schedule activities in decreasing order of ACTIM value.
- If there is a tie, break it in the following order:
  - » (a) Activity with the longest duration is scheduled first.
  - » (b) Activity with the largest resource requirement first.
  - » (c) Alphabetically or in numerical order.

Brook's Algorithm

Example



Total # of resources available = 3

Brook's Algorithm

ACTIM for an activity is the duration of the longest path from the beginning of that activity to the sink node.

Activity	ACTIM
A	16
B	8
C	16
D	11
E	4
F	8
G	16

Brook's Algorithm

Activity	G	C	A	D	F	B	E
ACTIM	16	16	16	11	8	8	4
Duration	16	5	5	7	8	4	4
Res. Req.	1	3	1	2	1	2	1
TEARL	0	0	0	21	21	5	28
TSTART	0	16	0	21	21	5	28
TFIN	16	21	5	28	29	9	32

Iteration #	1	2	3	4	5	6
TNOW	0	5	9	16	21	28
Res. Available	~~3~~ ~~2~~ 1	~~1~~ 0	2	~~3~~	~~3~~ ~~1~~ 0	~~2~~ 1
Act Allowed	~~G~~, C, ~~A~~	C, ~~B~~	C	~~C~~	~~D~~, ~~F~~	~~E~~

Project Duration = 32 days

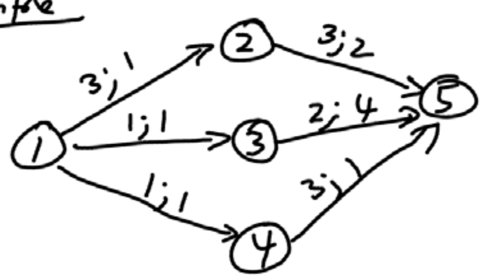
Gleeson’s Algorithm

– (Note that this is for allocation of units for a single resource type)

– This is similar to Brook’s algorithm except that here **ACTRES** is used instead of ACTIM.

Gleeson’s Algorithm

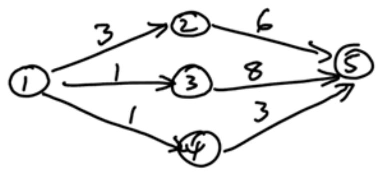
Example



Total # of resources available = 5

Gleeson's Algorithm

Modify Network



Activity	ACTRES
1 – 2	9
2 - 5	6
1 - 3	9
3 – 5	8
1 – 4	4
4 - 5	3

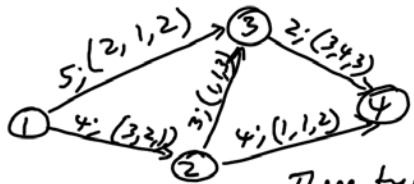
Gleeson's Algorithm

Activity	1-2	1-3	3-5	2-5	1-4	4-5
ACTRES	9	9	8	6	4	3
Duration	3	1	2	3	1	3
Res. Req.	1	1	4	2	1	1
TEARL	0	0	1	3	0	1
TSTART	0	0	1	3	0	3
TFIN	3	1	3	6	1	6

Iter. #	1	2	3
TNOW	0	1	3
Res. Avail.	~~5~~ ~~4~~ ~~3~~ 2	~~4~~ 0	~~4~~ ~~3~~ 2
Act. Allowed	~~1-2~~, ~~1-3~~, ~~1-4~~	~~3-5~~, 4-5	~~2-5~~, ~~4-5~~

Project Duration = 6

Allocation of Multiple Resources



Three types of resources are available

Max. (5, 4, 4)

Activity	ACTIM
1-2	9
1-3	7
2-3	5
2-4	4
3-4	2

Allocation of Multiple Resources

Time	Activity	Duration	Start	Finish	R1	R2	R3	Allowable Act.
0	–	–	–	–	5	4	4	~~1-2~~, ~~1-3~~
0	1-2	4	0	4	2	2	3	
0	1-3	5	0	5	0	1	1	
4	1-2	–	–	–	3	3	2	2-3, ~~2-4~~
4	2-4	4	4	8	2	2	0	
5	1-3	–	–	–	4	3	2	2-3
8	2-4	–	–	–	5	4	4	~~2-3~~
8	2-3	3	8	11	4	3	1	
11	2-3	–	–	–	5	4	4	~~3-4~~
11	3-4	2	11	13	2	0	1	
13	3-4	–	–	–	5	4	4	

$\boxed{\text{Project Duration} = 13}$

Resource Allocation vs. Resource Balancing

- **Resource Allocation Problem**
  - Given the number of resource units, what is the minimum time required to complete the project?
- **Resource Balancing Problem**
  - Given the time required for a project, what is the minimum number of resource units required to meet that time?

Sequencing

- Basics
  - Processing time, t_i
  - Due date, d_i
  - Lateness (positive or negative)
    - Deviation between completion time and due date, L_i
  - Tardiness
    - Measure of positive lateness, $T_i = \max\{0, L_i\}$

Sequencing

- **Flow time** – Span between the point at which a task is available for processing and point at which it is completed, F_i
- **Completion time** - Span between the beginning of work on the first job and when job i is finished, C_i
- If all jobs are available at $t = 0$, then $C_i = F_i$

Sequencing

- **Makespan** - Span of time when we start working on the first job on the first machine till we finish working on the last job on the last machine.

Sequencing *n* jobs on one machine

Makespan

$$M_s = \sum_{i=1}^{n} t_i$$

where M_s = makespan for n tasks in sequence S.

If all tasks are available at $t = 0$

$$F_{i,s} = C_{i,s}$$

where $F_{i,s}$ = flow time for task i in sequence S

$C_{i,s}$ = completion time " " " "

Sequencing *n* jobs on one machine

Mean flow time in sequence S

$$\overline{F_s} = \frac{1}{n} \sum_{i=1}^{n} F_{i,s}$$

$$L_{i,s} = C_{i,s} - d_i$$

$$T_{i,s} = \text{Max}\{0, L_{i,s}\}$$

$$= \text{Max}\{0, C_{is} - d_i\}$$

Mean lateness in sequence S

$$\overline{L_s} = \frac{1}{n} \sum_{i=1}^{n} L_{i,s}$$

Mean Tardiness in sequence S

$$\overline{T_s} = \frac{1}{n} \sum_{i=1}^{n} T_{i,s}$$

Sequencing *n* jobs on one machine

No. of tardy jobs, N_T

$$N_T = \sum_{i=1}^{n} \delta_i$$

$$\text{where } \delta_i = 1 \text{ if } T_i > 0$$
$$= 0 \text{ otherwise}$$

Also,

$$T_{Max} = \max\{0, L_{max}\}$$
$$L_{max} = \max\{L_{i,s}\} \text{ for all } i \text{ in } n$$

Sequencing *n* jobs on one machine

Example

All tasks are available at $t=0$

Task i	Proc. time t_i	Due date d_i	Flowtime F_i	Lateness L_i
1	5	15	5	-10
2	8	10	13	3
3	6	15	19	4
4	3	25	22	-3
5	10	20	32	12
6	14	40	46	6
7	7	45	53	8
8	3	50	56	6

$\bar{F}_s = 30.75$ $\bar{L}_s = 3.25$

Sequencing *n* jobs on one machine

Use Shortest Processing time (SPT) Rule

Task i	t_i	d_i	F_i	L_i
4	3	25	3	-22
8	3	50	6	-44
1	5	15	11	-4
3	6	15	17	2
7	7	45	24	-21
2	8	10	32	22
5	10	20	42	22
6	14	40	56	16

$\bar{F}_S = 23.875$ $\bar{L}_S = -3.625$

Sequencing *n* jobs on one machine

- SPT minimizes mean flow time
- SPT minimizes mean lateness

Sequencing *n* jobs on one machine

Minimize Maximum Lateness

Use EDD (earliest due date) Rule

Task i	t_i	d_i	F_i	L_i
2	8	10	8	-2
1	5	15	13	-2
3	6	15	19	4
5	10	20	29	9
4	3	25	32	7
6	14	40	46	6
7	7	45	53	8
8	3	50	56	6

Sequencing *n* jobs on one machine

- **Hodgson's Algorithm – Minimize the number of tardy jobs**
  - If EDD rule results in no tardy jobs or only one tardy job, it will be optimum! If there are two or more tardy jobs, the we proceed with the following steps.
  - **Step 1**. Identify the first tardy job in the EDD sequence. If none, go to step 3.
  - **Step 2**. Let the first tardy job be in the i^{th} position. Find the job with the longest processing time in the first i jobs. Remove it and set it aside. Revise the flow times. Go to step 1.
  - **Step 3**. Place all the jobs that were set aside at the end of the current sequence. STOP.

Sequencing *n* jobs on one machine

Example

Task	$\boxed{2}$	1	3	5	4	6	7	8
t_i	8	5	6	10	3	14	7	3
C_i	8	13	19	29	32	46	53	56
d_i	10	15	15	20	25	40	45	50
L_i	-2	-2	4	9	7	6	8	6

Set aside task 2

Task	1	3	$\boxed{5}$	4	6	7	8
t_i	5	6	10	3	14	7	3
C_i	5	11	21	24	38	45	48
d_i	15	15	20	25	40	45	50
L_i	-10	-4	1	-1	-2	0	-2

Set aside task 5

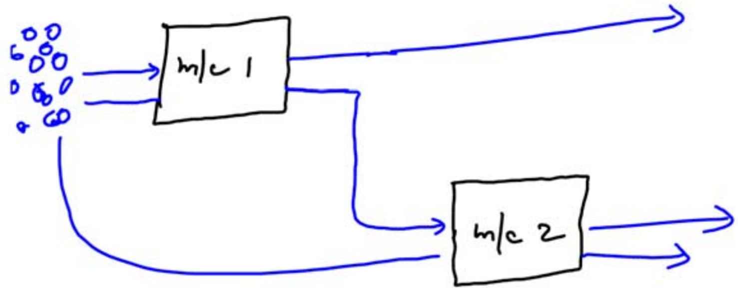
Sequencing *n* jobs on one machine

Task	1	3	4	6	7	8	2	5
t_i	5	6	3	14	7	3	8	10
C_i	5	11	14	28	35	38	46	56
d_i	15	15	25	40	45	50	10	20
L_i	-10	-4	-11	-12	-10	-12	36	36

only 2 tardy job!

Sequencing *n* jobs on two machines

An example of two machine case



Sequencing *n* jobs on two machines

- **Johnson's Algorithm – Minimize the makespan**
  - **Step 1**. Create a list of processing times of all jobs on machine 1 (M_1) and machine 2 (M_2).
  - **Step 2**. Identify the shortest processing time in this list. Break ties arbitrarily.
  - **Step 3**. If the shortest processing time is on M_1, then assign the corresponding job to the next available position starting at the beginning of the sequence. Go to step 4. If it is on M_2, then assign the corresponding job to the next available position starting from the end of the sequence. Go to step 4. .
  - **Step 4**. Remove the assigned job from the list. Repeat steps 2 and 3 until all jobs are assigned.

Sequencing *n* jobs on two machines

◆Example

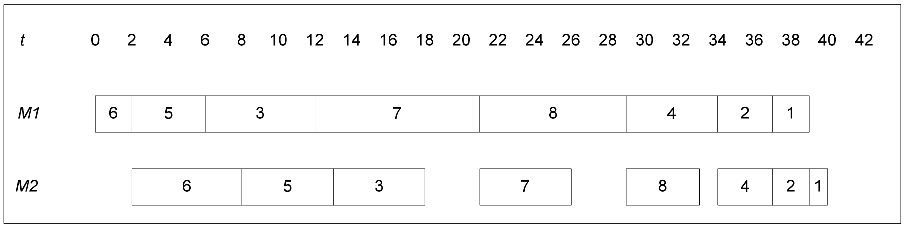
Job	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
First Machine	2	3	6	5	4	2	9	8
Second Machine	1	2	5	3	5	6	5	4

Sequencing *n* jobs on two machines

Iteration	1								1
Iteration	2	6							1
Iteration	3	6						2	1
Iteration	4	6					4	2	1
Iteration	5	6	5				4	2	1
Iteration	6	6	5			8	4	2	1
Iteration	7	6	5		7	8	4	2	1
Iteration	8	6	5	3	7	8	4	2	1

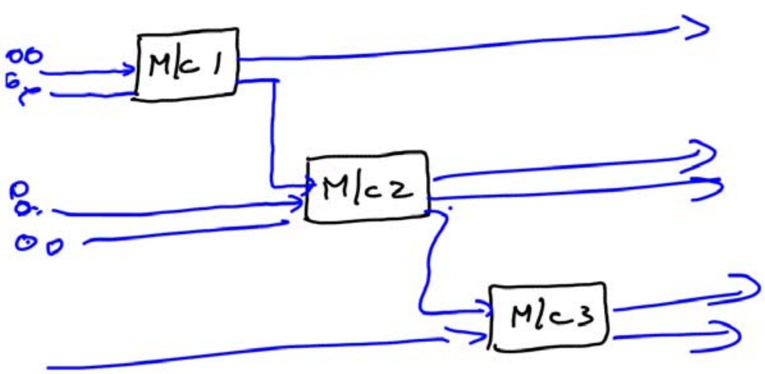
Sequencing *n* jobs on two machines

- Thus the optimal sequence is **6-5-3-7-8-4-2-1**.
- The makespan is 40 (see Gantt chart below).



Sequencing *n* jobs on three machines

3 m/c case



Sequencing *n* jobs on three machines

Example

Job	t_{i1} M/c 1	t_{i2} M/c 2	t_{i3} M/c 3
1	5	3	9
2	7	2	5
3	$\boxed{4}$	3	7
4	8	$\boxed{4}$	3
5	6	2	$\boxed{2}$
6	7	0	8

Sequencing *n* jobs on three machines

Convert this into a 2-m/c problem as follows:

M/c 1′ = M/c 1 + M/c 2

M/c 2′ = M/c 2 + M/c 3

Sequencing *n* jobs on three machines

Job	M/c 1'	M/c 2'
1	8	12
2	9	7
3	7	10
4	12	7
5	8	4
6	7	8

6	3	1	2	4	5

Use this sequence in the 3 machine problem.

Make span = 41

Sequencing *n* jobs on three machines

Condition for Optimality

The solution to the three m/c problem will be optimal using the above method if

either $\min. t_{i_1} \geq \max t_{i_2}$

or $\min. t_{i_3} \geq \max t_{i_2}$

In our example, the 1st of the above conditions is met!

Hence, the sequence 6, 3, 1, 2, 4, 5 is, in fact, optimal for the 3 m/c problem.

Sequencing *n* jobs on three machines

Note:

If for some reason, the above conditions of optimality are not met, the procedure does not guarantee an optimal solution. It is, however, still a a very good heuristic solution!