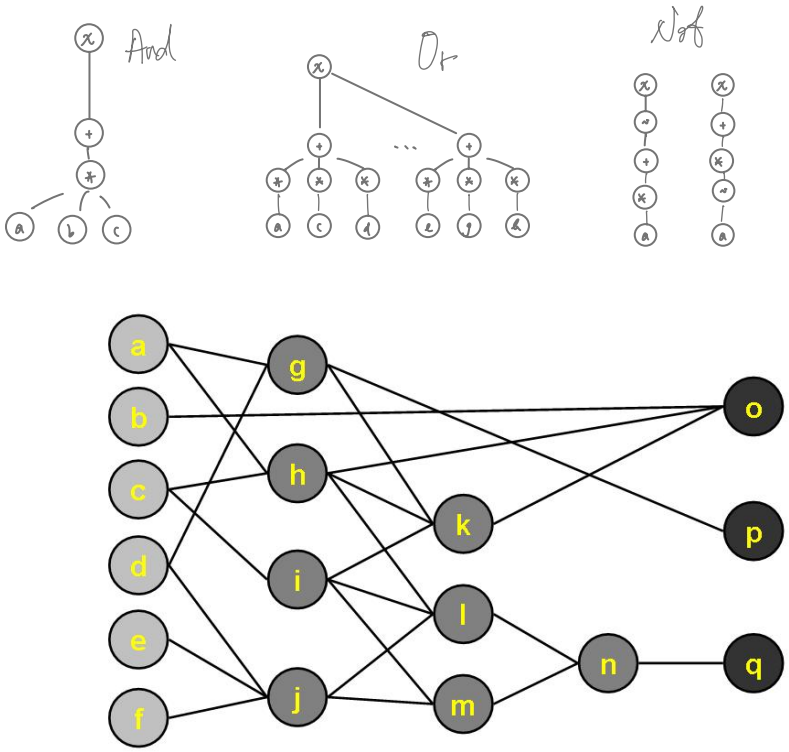
CAD Design Project 2 – Constrained Scheduling
Due: 23:55, Oct. 24, 2019

In this project, you are required to implement two versions of list scheduling algorithms for constrained scheduling problems. (1) The ML-RCS is the scheduling problem to minimize latency under resource constraints. (2) The MR-LCS is the scheduling problem to minimize resource under latency constraint. Your program would be evaluated on Linux environment according to the following requirements.

1. For simplicity, there are only 3 types of Boolean operations: *AND*, *OR*, and *NOT*.
2. Assume that every operation takes 1-cycle latency. (The PI node is not an operation.)
3. Read a BLIF file and the corresponding resource or latency constraints.
4. For ML-RCS, use option "-l", output the scheduled result under resource constraints.
5. For MR-LCS, use option "-r", output the scheduled result under latency constraint.
6. Upload your source code tarball (*.tgz) to moodle (including your Makefile)
   (NOTE: The uploaded file name should be the same with your student ID.)
7. Generate and upload one "difficult" test case with ReadMe file.

BLIF Example: sample02.blif

```
.model sample02
.inputs a b c d e f
.outputs o p q
.names a d g
1- 1
-1 1
.names a c h
11 1
.names c i
0 1
.names d e f j
1-- 1
-1- 1
--1 1
.names g h i k
1-- 1
-1- 1
--1 1
.names h i j l
111 1
.names i j m
11 1
.names l m n
11 1
.names b h k o
111 1
.names g p
0 1
.names n q
0 1
.end
```



SYNOPSIS for ML-RCS

```
%> list -l BLIF_FILE AND_CONSTRAINT OR_CONSTRAINT NOT_CONSTRAINT
```

Run-time Example:

```
%> list -l sample02.blif 2 1 1
Resource-constrained Scheduling
1: {h} {j} {i}
2: {l m} {g} {}
3: {n} {k} {p}
4: {o} {} {q}
#AND: 2
#OR: 1
#NOT: 1
END
```

Run-time Example:

```
%> list -l sample02.blif 1 0 1
Resource-constrained Scheduling
No feasible solution.
END
```

SYNOPSIS for MR-LCS

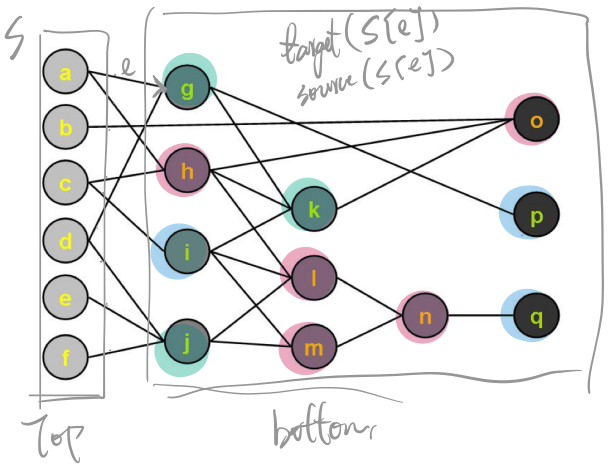
```
%> list -r BLIF_FILE LATENCY_CONSTRAINT
```

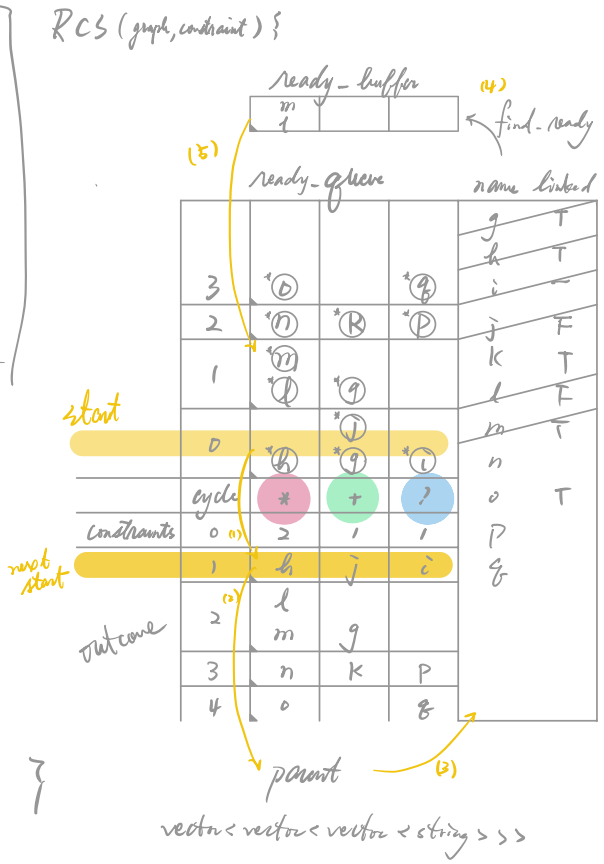
Run-time Example:

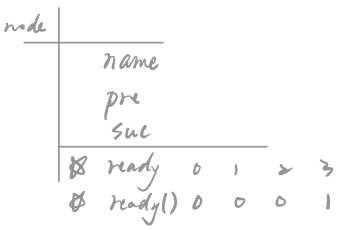
```
%> list -r sample02.blif 5
Latency-constrained Scheduling
1: {h} {j} {i}
2: {m} {} {}
3: {l} {g} {}
4: {n} {k} {p}
5: {o} {} {q}
#AND: 1
#OR: 1
#NOT: 1
END
```

Run-time Example:

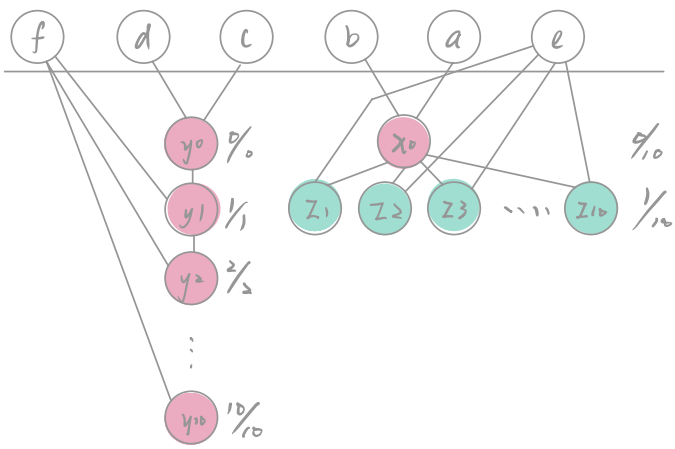
```
%> list -r sample02.blif 3
Latency-constrained Scheduling
No feasible solution.
END
```





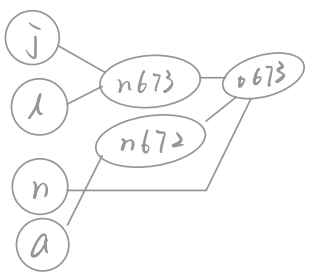


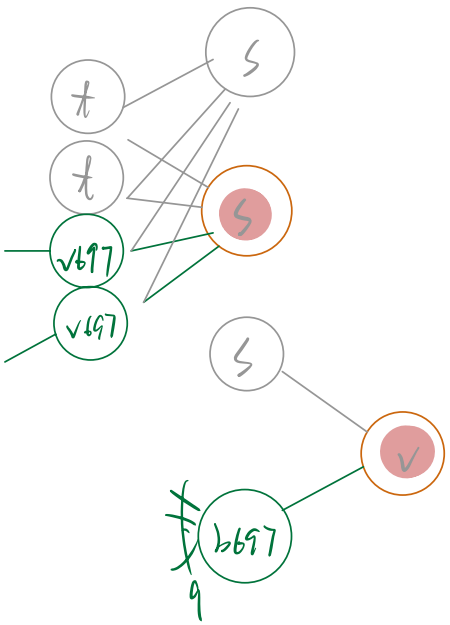

```
Run-time Example:
%> list -l sample02.blif 2 1 1
Resource-constrained Scheduling
1: {h} {j} {i}
2: {l m} {g} {}
3: {n} {k} {p}
4: {o} {} {q}
#AND: 2
#OR: 1
#NOT: 1
END
```




```
Run-time Example:
%> list -l sample02.blif 1 0 1
Resource-constrained Scheduling
No feasible solution.
END
```

gate_cnt = 685





outputs s t v

violation

.names t v697 s

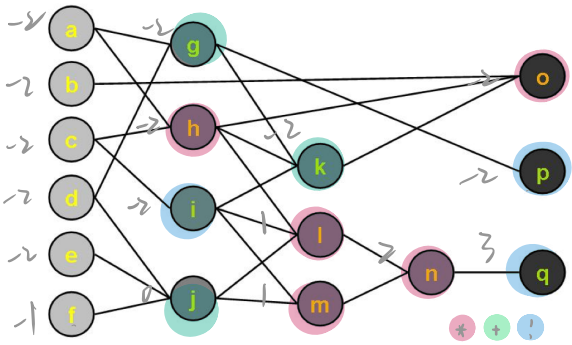
.names s b709 v

all_nodes

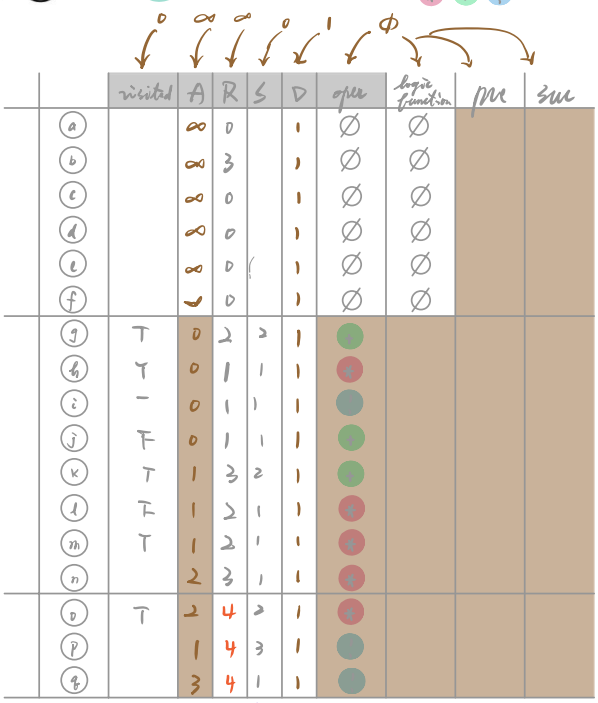
0	
1	t v697 s
2	s t v

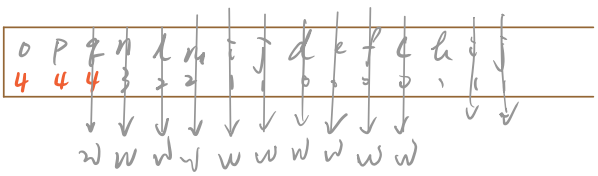
input
inner
output

.names t v697 s



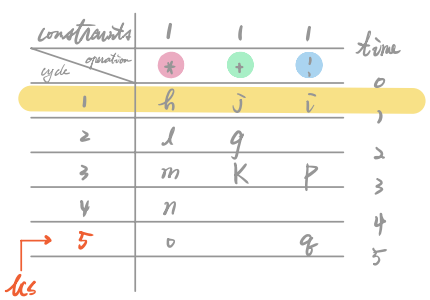

```
Run-time Example:
%> list -r sample02.blif 5
Latency-constrained Scheduling
1: {h} {j} {i}
2: {m} {} {}
3: {l} {g} {}
4: {n} {k} {p}
5: {o} {} {q}
#AND: 1
#OR: 1
#NOT: 1
END

Run-time Example:
%> list -r sample02.blif 3
Latency-constrained Scheduling
No feasible solution.
END
```





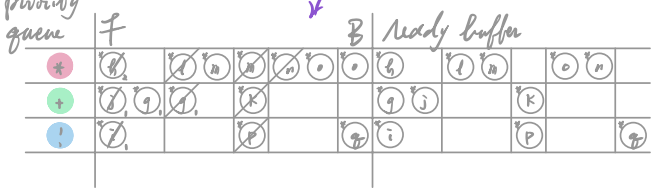
(1) (2) (3) check if feasible

LCS

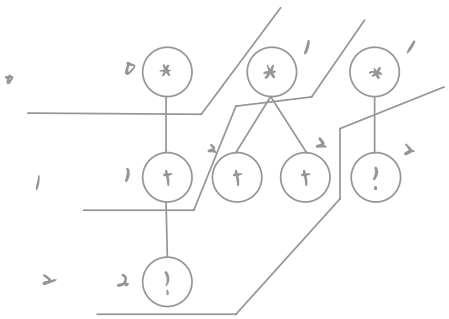


if constraints too strict,
++constraints, clear schedule, queue, clear visit

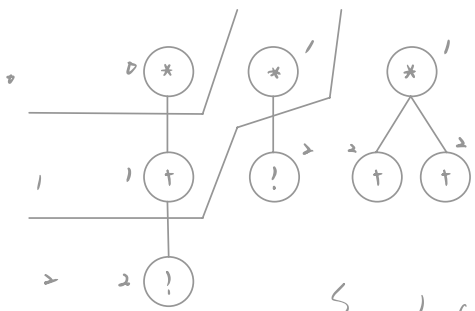
(6) (5) find ready (4)



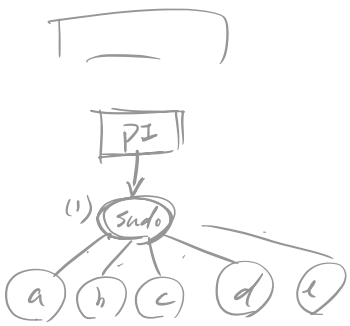
while (!all scheduled)



constraints of (?) will be 2



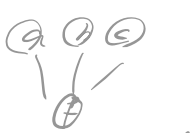
this



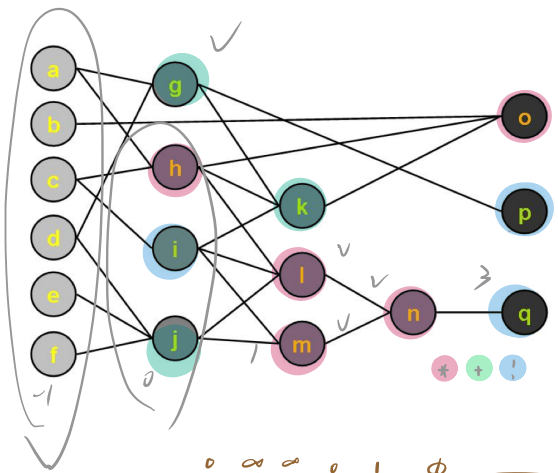
{a, b, c, d, e}

new node("a")

sudo.suc.push_back(new node("a"))

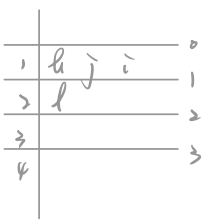


.names abc f



	visited	A	R	S	D	oper	logic function	pre	succ
a		∞	0		1	∅	∅		
b		∞	3		1	∅	∅		
c		∞	0		1	∅	∅		
d		∞	0		1	∅	∅		
e		∞	0		1	∅	∅		
f		✓	0		1	∅	∅		
g	T	0	2	2	1	*			
h	T	0	1	1	1	*			
i	-	0	1	1	1	*			
j	F	0	1	1	1	*			
k		1	3	2	1	*			
l		1	2	1	1	*			
m		1	2	1	1	*			
n		2	3	1	1	*			
o	-	2	4	2	1	*			
p		1	4	3	1	*			
q		3	4	1	1	*			

```
Run-time Example:
%> list -r sample02.blif 5
Latency-constrained Scheduling
1: {h} {j} {i}
2: {m} {} {}
3: {l} {g} {}
4: {n} {k} {p}
5: {o} {} {q}
#AND: 1
#OR: 1
#NOT: 1
END

Run-time Example:
%> list -r sample02.blif 3
Latency-constrained Scheduling
No feasible solution.
END
```

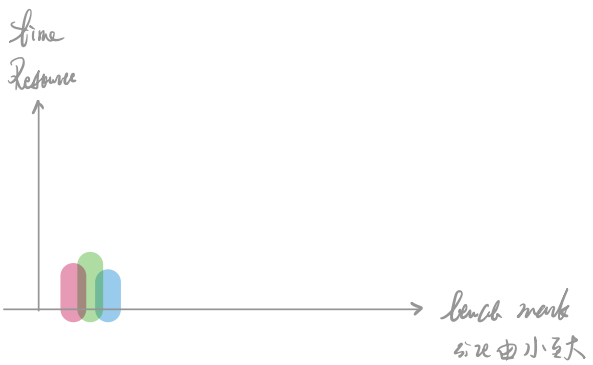


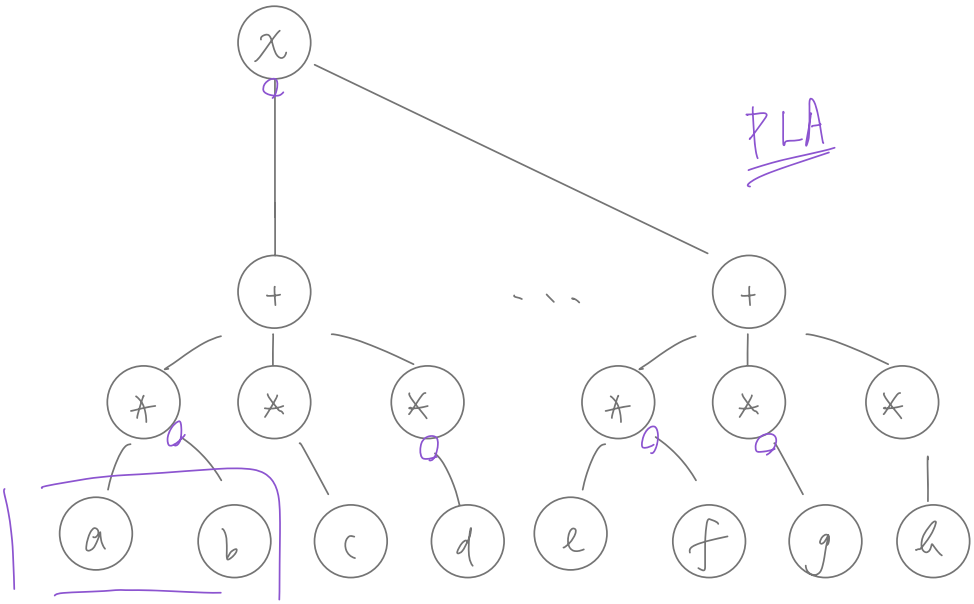
- MR-LCS
  - 演算法說明
  - Pseudo code
  - Time complexity 分析
  - worst case

- FD-LCS
  - 演算法說明
  - Pseudo code
  - Time complexity 分析

- 實驗數據比較
  - bench mark
  - worst case

- 結論





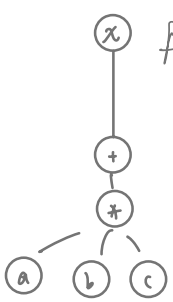
PLA

e f g h x

out = (vec<(vec<>, vec<>>), vec<(vec<>, vec<>>))

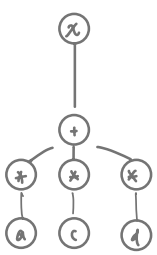
or = vec<(vec<>, vec<>>)

and = (vec<>, vec<>>)

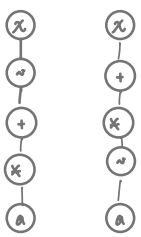


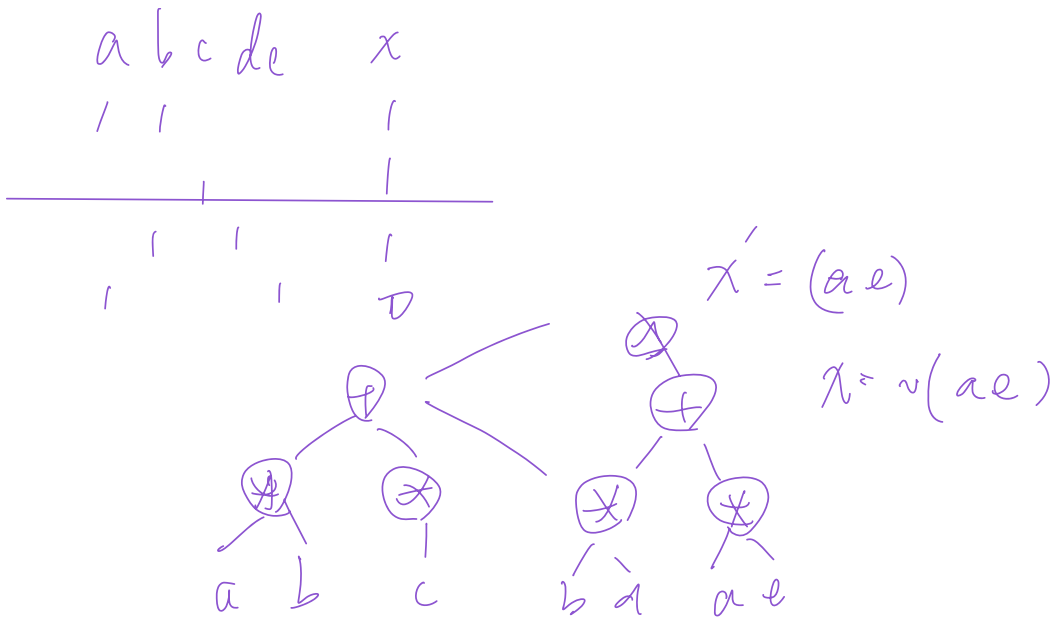
And

Or



Not




a b c d e x
D
x' = (a e)
x = v(a e)