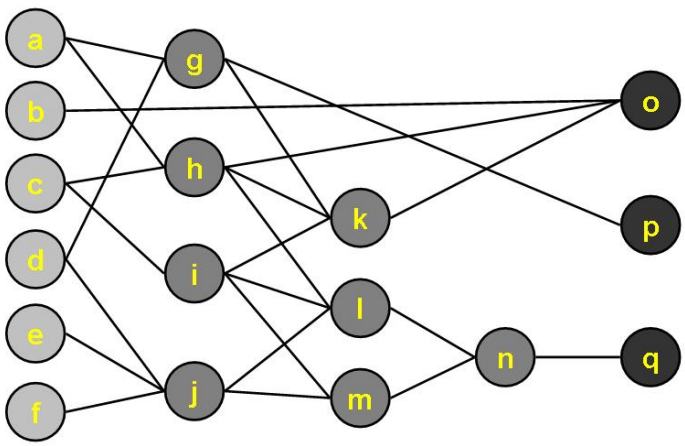
CAD Design Project 3 – Latency Constrained Scheduling
Due: 23:55, Nov. 13, 2019

In Project 2, we have implemented two versions of constrained list scheduling algorithms. In this project, you are required to further minimize the hardware resource of the latency-constrained scheduling algorithm taking operation probabilities and type distributions into account (e.g., force-directed latency-constrained scheduling, FD-LCS). 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 latency constraint.
4. Write a report with your own pseudo code and detailed explanation of your algorithm.
5. Compare and discuss the LCS results of your Project02 and Project03 in your report.
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
```



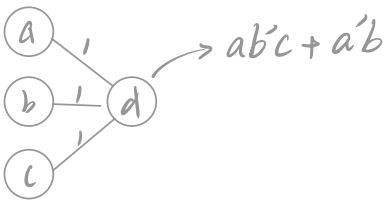
```
SYNOPSIS for LCS
%> lcs BLIF_FILE LATENCY_CONSTRAINT

Run-time Example:
%> lcs 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:
%> lcs sample02.blif 4
Latency-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:
%> lcs sample02.blif 3
Latency-constrained Scheduling
No feasible solution.
END
```

blif_names_block

assumption name cannot start with '0', '1', '-', '#', '.'

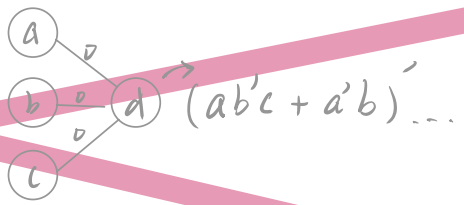
< case 1 >

```
.names  a  b  c  d
1 0 1   1
0 - 1   1
```



~~< case 2 >~~

```
.names  a  b  c  d
1 0 1   0
0 - 1   0
```



< case 3 >

```
.names  a
1
```

< case 4 >

```
.names  a
0
```

< case 5 >

```
.names  a
.names  b
```

process_names_block

	0	1	2	3
names	a	b	c	d

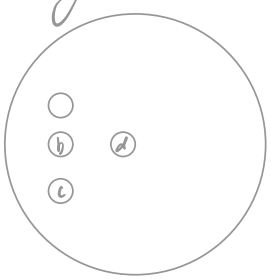
	logic
0	"1 0 1 1"
1	"0 - 1 1"

subgraph	ⓐ	ⓑ	ⓒ	ⓓ

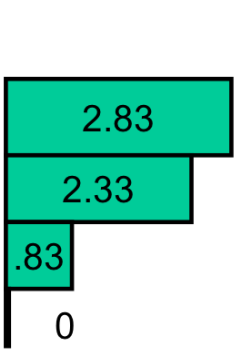
NT	0	1	2
0	a'	b'	c'
1	a	b	c

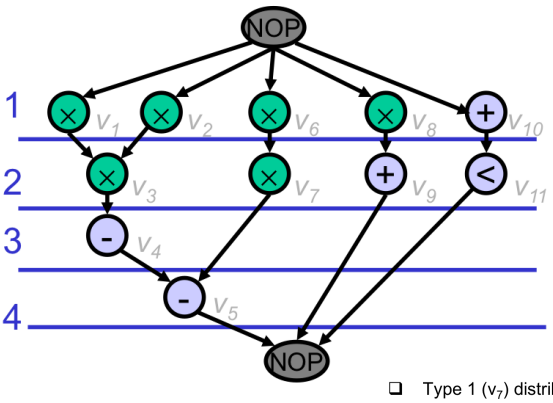
class into name logic func...

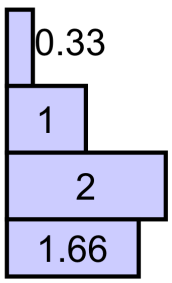
Signal_netlist



a b c + a' c + ...








```
1.  forall_node(v, G)
2.      for(l = A[v] to R[v])
3.          stack<node> pre, suc;
4.          forall_in_edges(e,v)
5.              pre.push(G.source(e))
6.          while(!pre.empty())
7.              p = pre.top();
8.              pre.pop();
9.              if(l-A[p]<S[p])
10.                 for(i = A[p] to l-1)
11.                     psF(v,l) += 1/(l-A[p]) * q(*,i)
12.                 for(i = A[p] to R[p])
13.                     psF(v,l) -= (1/S[p]) * q(*,i))
14.                 forall_in_edges(e,p)
                        pre.push(G.source(e))
```

算 pF

- Type 1 (v_7) distribution:
  - q(1) = 2.8; q(2) = 2.3; q(3) = 0.8; q(4) = 0
- Assign v_6 to step 2:
  - Time frame of v_7 is reduced
  - 1 * (0.8) – 0.5 * (2.3 + 0.8) = -0.75
- Type 2 (v_9) distribution:
  - q(1) = 0.3; q(2) = 1; q(3) = 2; q(4) = 1.6
- Assign v_8 to step 2:
  - Time frame of v_9 is reduced
  - 0.5 * (2 + 1.6) – 0.3 * (1 + 2 + 1.6) = 0.3

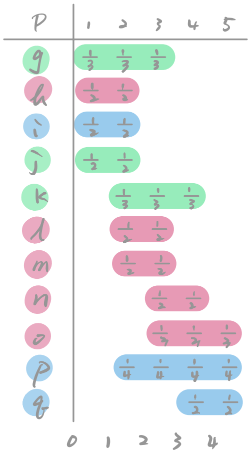
- x 的祖先 Arrive time 和 require time 皆至少少一
- 若 p 對 x 的 predecessor force 為 0，則 p 的祖先加 S 對 x 的 predecessor force 皆為 0
- 若非 x 的祖先則對其沒有 predecessor force

selfF[v, l] = q[Ru[v], l] – bF[v]

while block 有 > 0 的

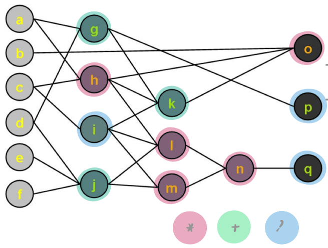
block == 0 不用做

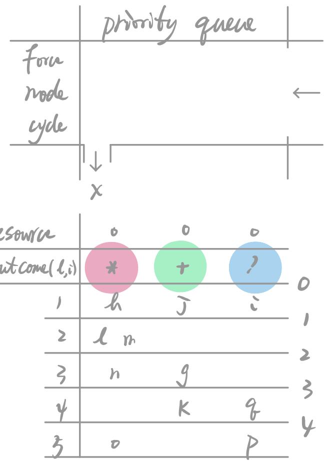
Array	A	R	S
g	0	2	2
h	0	2	2
i	0	1	1
j	0	1	1
k	1	3	2
l	1	2	1
m	1	2	1
n	2	3	1
o	2	4	2
p	1	4	3
q	3	4	1



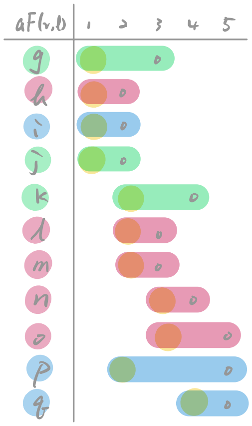
(1)

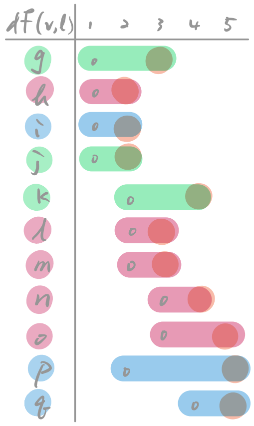
q(r,l)	1	2	3	4	5
*	0.5	1.5	1.83	0.83	0.33
+	0.83	1.17	0.67	0.33	0.00
?	0.5	0.75	0.25	0.75	0.75





Array	base_Force
g	(0.83 + 1.17 + 0.67) / 3
h	(0.5 + 1.5) / 2
i	(0.5 + 1.5) / 2
j	
k	
l	⋮
m	
n	
o	
p	
q	



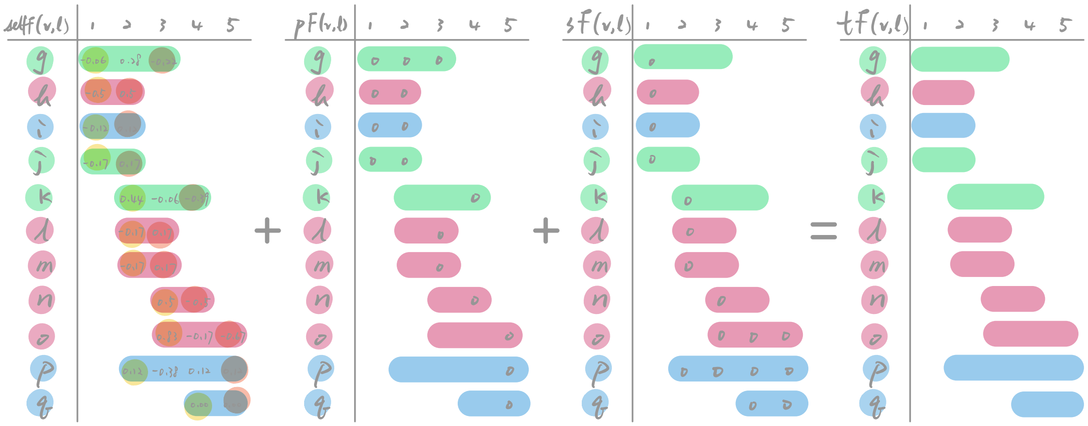


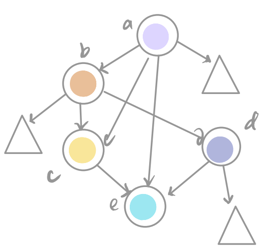
(10)

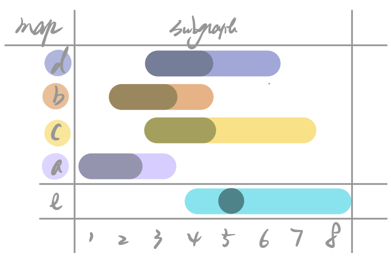
(9) 改 A、R、S

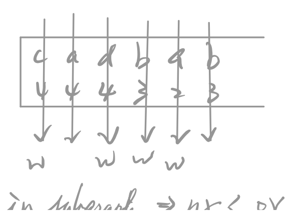
(8)

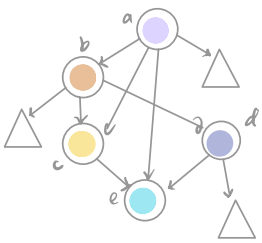
smallest_force

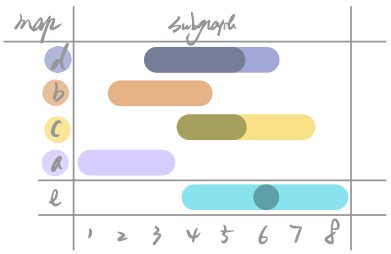


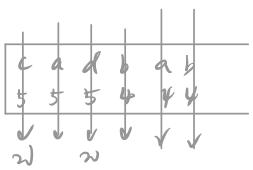




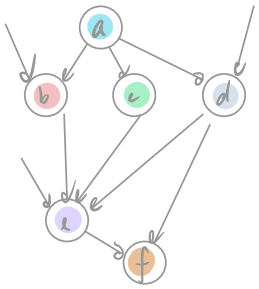


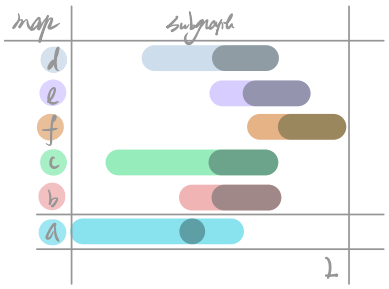


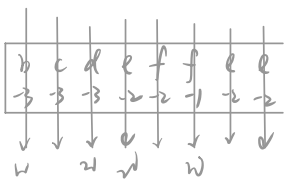




not in = => nR < R

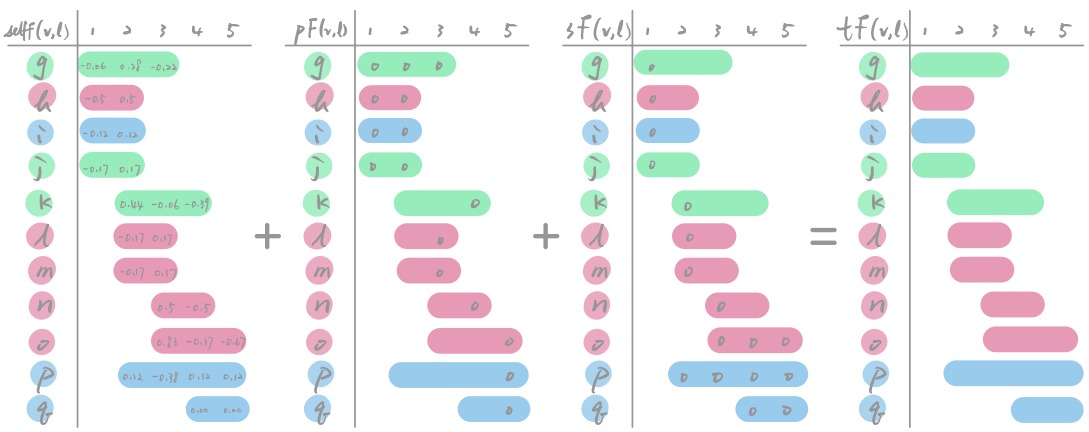






in subgraph => nA > oA

not in = = nA > A



$$0.83 - \frac{0.83 + 1.17 + 0.67}{3} \qquad 1 * 0.83 - \frac{1}{3} *$$

$$1.17$$

Force = constant * displacement

$$-\frac{1}{5} - \frac{1}{33}$$

Schedule = $q * p$

$$selfF[v,l] = \sum_{l \in \{A[v], \ldots, R[v]\}} q[Ru[v], l] * p[v,l]$$

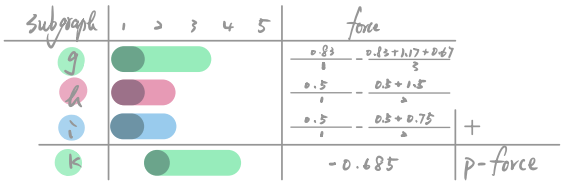
$$= q[Ru[v], l] - p[v,l] \sum_{l \in A[v], \ldots, R[v]} q[R[v], l]$$

$$pf[v,l] = \sum_{p \in v\text{'s ancestors}} af[v,l]$$

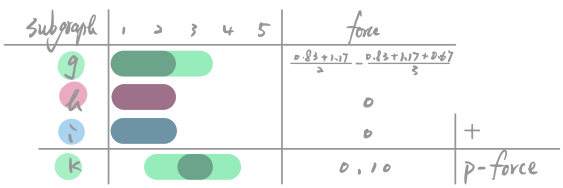
$$self[v,l] = \sum_{l \in \{A[v],A[v]+1,\ldots,R[v]\}} q[Ru[v],l] * p[v,l]$$
$$= q[Ru[v],l] - p[v,l] * \sum_{l \in \{A[v],A[v]+1,R[v]\}} q[R[v],l]$$
$$= q[Ru[v],l] - p[v,l] * BaseF[v]$$

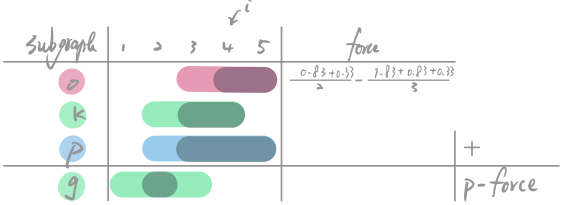
Array	baseF
g	$\frac{0.83 + 1.17 + 0.67}{3}$
h	$\frac{0.5 + 1.5}{2}$
i	$\frac{0.5 + 7.5}{2}$
j	$\frac{0.83 + 1.17}{2}$
k	$\frac{1.17\ 0.67\ 0.33}{3}$
l	$\frac{1.5\ 1.83}{2}$
m	$\frac{1.5\ 1.83}{2}$
n	$\frac{1.83\ 0.83}{2}$
o	$\frac{1.83\ 0.83\ 0.33}{3}$
p	$\frac{0.75\ 0.25\ 0.75\ 0.75}{4}$
q	$\frac{0.75\ 0.75}{2}$

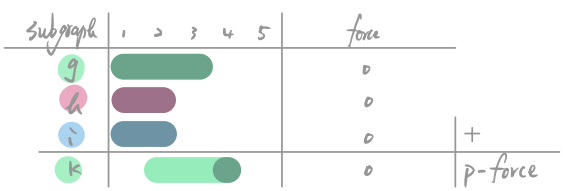
selff(v,l)	1	2	3	4	5
g	-0.06	0.38	-0.22		
h	-0.5	0.5			
i	-0.12	0.12			
j	-0.17	0.17			
k		0.44	-0.06	-0.39	
l		-0.17	0.17		
m		-0.17	0.17		
n			0.5	-0.5	
o			0.83	-0.17	-0.67
p		0.12	-0.38	0.12	0.12
q				0.00	0.00

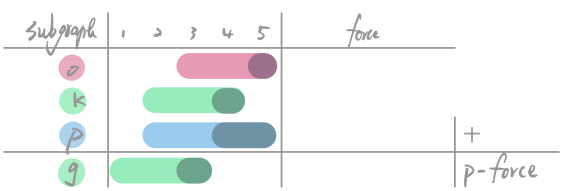

Subgraph
1 2 3 4 5
force
g
h
i
k
-0.685
p-force

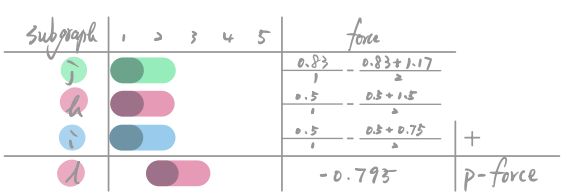

Subgraph
1 2 3 4 5
force
o
k
p
g
p-force

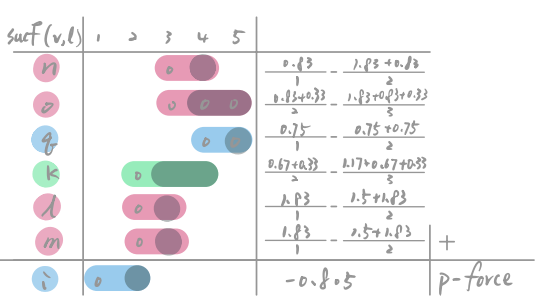

Subgraph
1 2 3 4 5
force
g
h
i
k
0.10
p-force


Subgraph
1 2 3 4 5
force
o
k
p
g
p-force

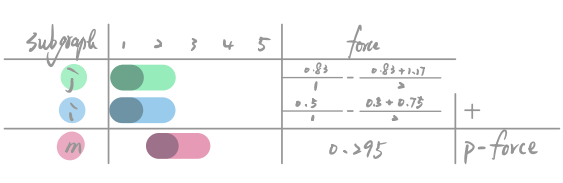

Subgraph
1 2 3 4 5
force
g
h
i
k
p-force

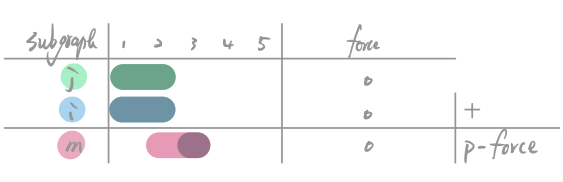

Subgraph
1 2 3 4 5
force
o
k
p
g
p-force


Subgraph
1 2 3 4 5
force
j
h
i
l
-0.795
p-force

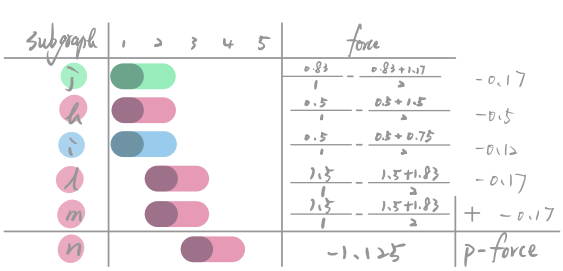

succ(v,l)
1 2 3 4 5
n
o
q
k
l
m
i
-0.805
p-force


Subgraph
1 2 3 4 5
force
j
h
i
l
p-force


Subgraph
1 2 3 4 5
force
j
i
m
0.295
p-force


Subgraph
1 2 3 4 5
force
j
i
m
p-force


$q_r(v,l)$
1 2 3 4 5
*
+
/

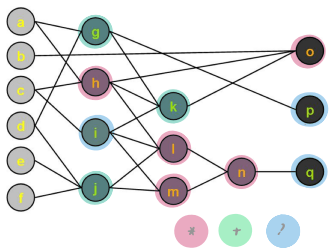

Subgraph
1 2 3 4 5
force
j
h
i
l
m
n
-1.125
p-force


Subgraph
1 2 3 4 5
force

j		0	
k		0	
i		0	
l		0	
m		0	+
n		0	p-force

Subgraph	1	2	3	4	5	force	
g						$\frac{0.83}{1} - \frac{0.83+1.17+0.67}{3}$	-0.66
k						$\frac{0.5}{1} - \frac{0.5+1.5}{2}$	+ -0.5
i						$\frac{0.5}{1} - \frac{0.5+0.75}{2}$	-0.12
k						$\frac{1.17}{1} - \frac{1.17+0.67+0.33}{3}$	0.44
o						-0.236	p-force

Subgraph	1	2	3	4	5	force	
g						$\frac{0.83+1.17}{2} - \frac{0.83+1.17+0.67}{3}$	
k						0	+
i						0	
k						$\frac{1.17+0.67}{2} - \frac{1.17+0.67+0.33}{3}$	
o						-0.1383	p-force



./espresso -Dexact a.pla
./espresso -Dexact b.pla
./espresso -Dxor a.pla b.pla > o.pla
./espresso -Dexact o.pla

./espresso -h
./espresso -Dminterms br.pla (秀) (不要用) (會爆)
./espresso -Dexact bs.pla
-Dverify bs.pla br.pla

ctag -R *.c

vi -t main
grep key-xor *.h
wc -l *.c *.h

ctrl 中括號

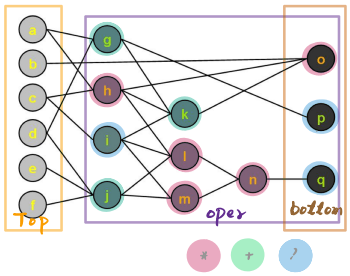
ctrl + t

pset family

F onset
R off set

猜的
pseudo

①



G	1	2	3	4	5
g	1/3	1/3	1/3		
h	1/2	1/2			
i	1/2	1/2			
j	1/2	1/2			
k		1/3	1/3	1/3	
l		1/2	1/2		
m		1/2	1/2		
n			1/2	1/2	
o			1/3	1/3	1/3
p			1/4	1/4	1/4	1/4
q				1/2	1/2

⑤

$q_r(r,l)$	1	2	3	4	5
*	0.5	1.5	1.83	0.83	0.33
+	0.83	1.17	0.67	0.33	0.00
?	0.5	0.75	0.25	0.75	0.75

②

```
0  initialization(G)
1    breath-first traverse all node √
2      classified v into "top" and "oper", "bottom"
3      label resource v used ;
4    for all node v in top
5      A(v) = -1 ;
6    for all node v in oper
7      A(v) = -∞ ;
8    push all node v in top into Q ;
9    while ( Q is not empty)
10     pop a node x from Q ;
       for all successor s of x
         if A(s) < A(x)+1
           A(s) = A(x)+1
           push all successor of s into Q
```

node_Array< unsigned > (G)

Indeg
outDeg

③

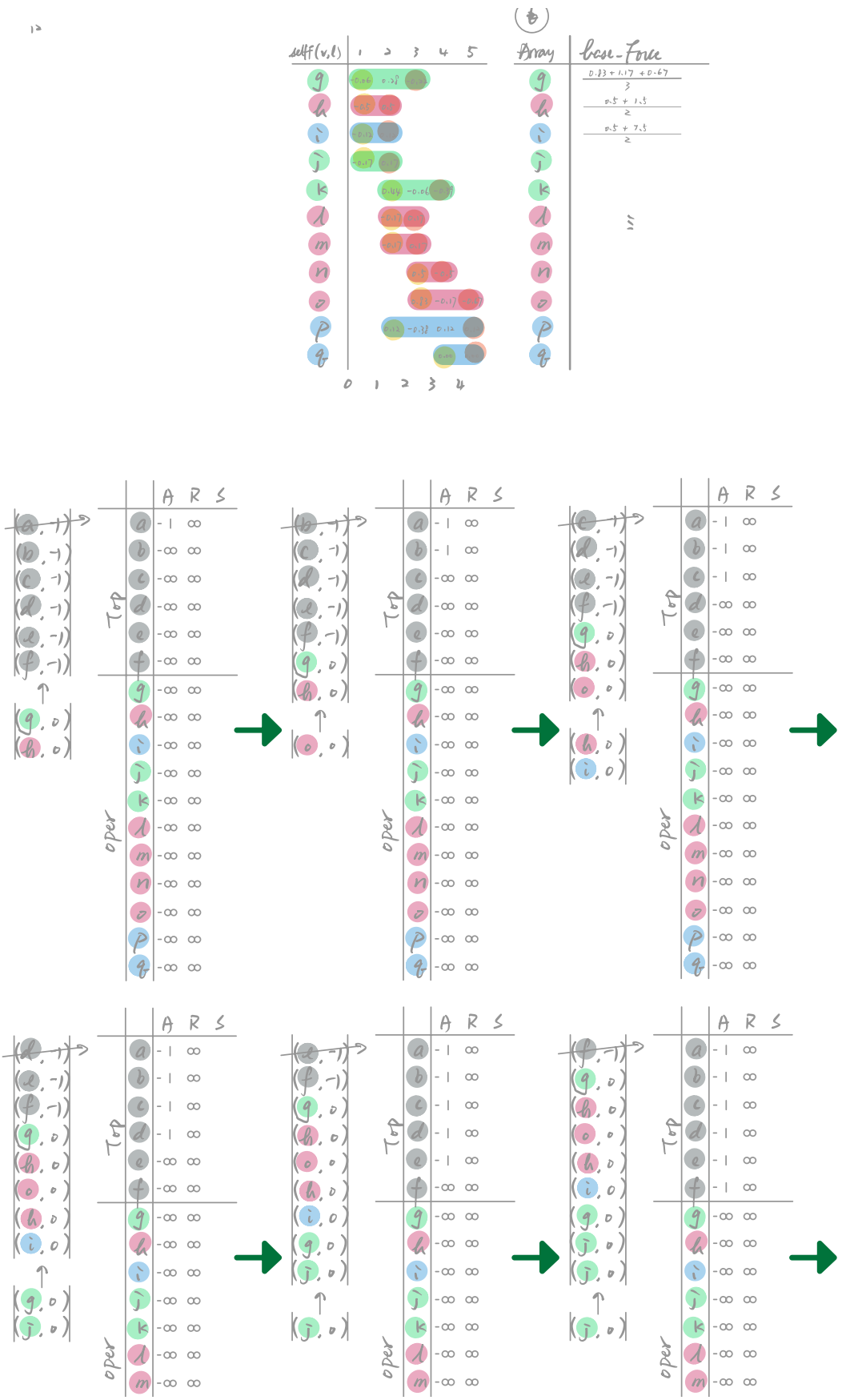
A(v) = node v 的 arrive time
R(v) =
S(v) =

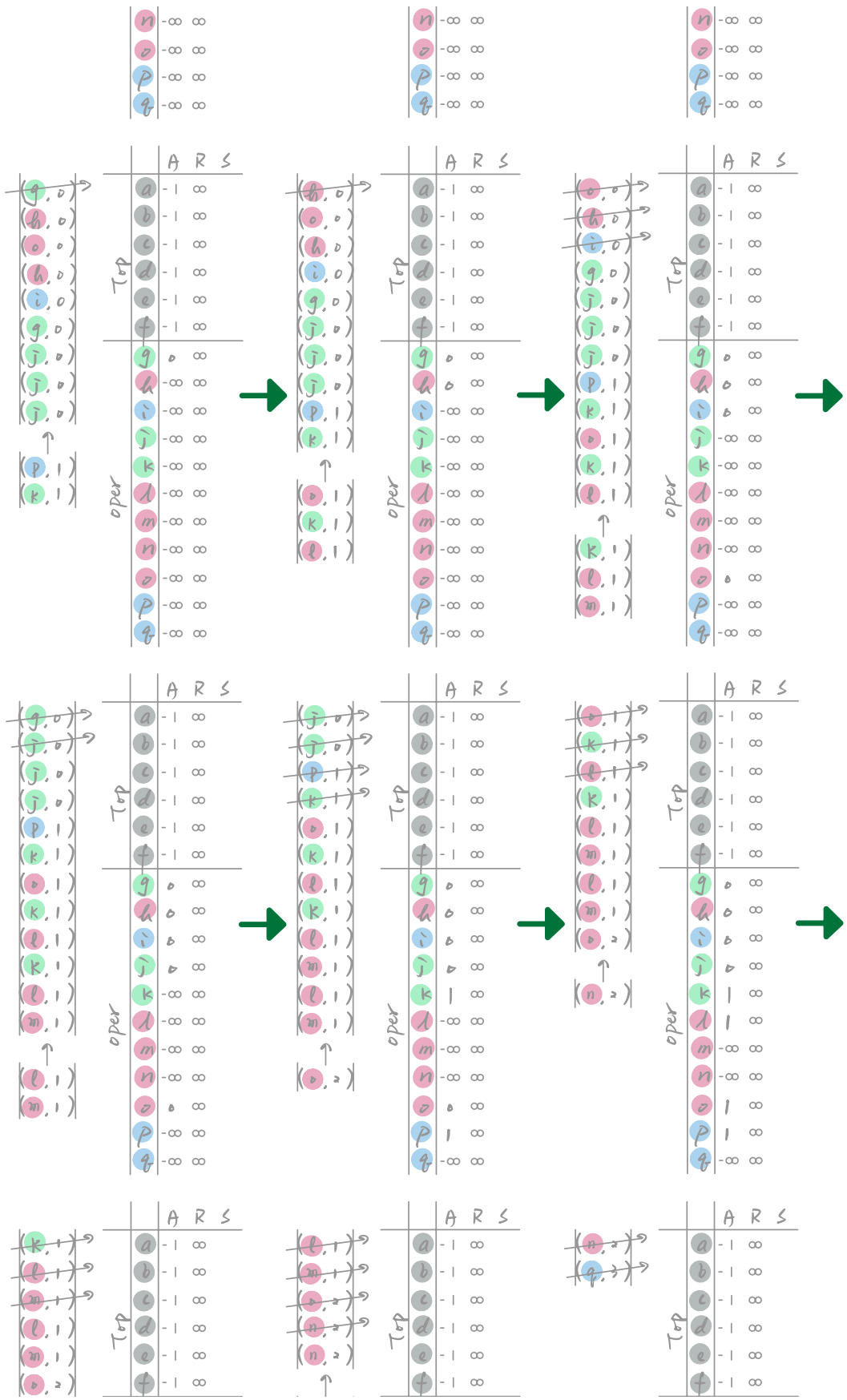
⑤ $q_r(r,l)$ = distribution of resource r in cycle l

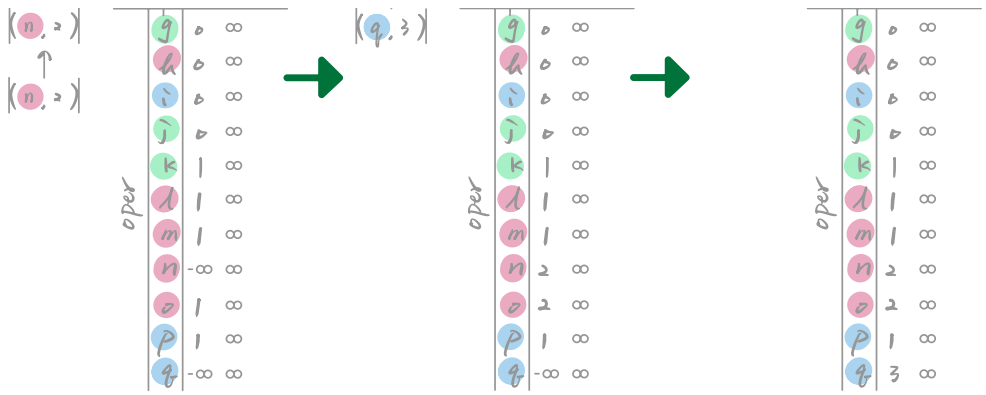
clock of feasible (S)

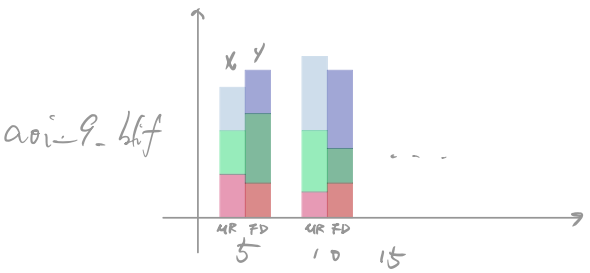
④ init-q_r(A, R, S)

```
0  FD-LCS (G, L)
1    initialization(G) ;
2    clock of feasible (S)
3    init-qr(A, R, S)
4
5
6
7
8
9
10
11
```


self(v,l)
Array
brute-force
top
oper
A R S


A R S
top
oper






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

```
int main ( int argc, char **argv){
  vector<string> from_benchmark;
  string extension;
  string dir_benchmark;
```

M_and M_not
M_or M_Time

MR fnb	5	10	15	20	25	...	500
0							
1							
2							
3							
4							
5							
6							
7							
8							
9							
10							

(0, 0, 0, x)

(unsigned, unsigned, unsigned, double)

MR_5_0_test

FD fnb	5	10	15	20	25	...	500
0							
1							
2							
3							
4							
5							
6							
7							
8							
9							
10							

MR_5_0_test

```
int main()
{
    // open for input and output and preposition file pointers to end-of-file
    // file mode argument see § 8.4 (p. 319)
    fstream inOut("copyOut",
                  fstream::ate | fstream::in |
                  fstream::out);
    if (!inOut) {
        cerr << "Unable to open file!" << endl;
        return EXIT_FAILURE; // EXIT_FAILURE see § 6.3.2 (p. 227)
    }
    // inOut is opened in ate mode, so it starts out positioned at the end
    auto end_mark = inOut.tellg();// remember original end-of-file
position
    inOut.seekg(0, fstream::beg); // reposition to the start of the file
    size_t cnt = 0;                  // accumulator for the byte count
    string line;                     // hold each line of input
    // while we haven't hit an error and are still reading the original data
    while (inOut && inOut.tellg() != end_mark
           && getline(inOut, line)) { // and can get another line of
input
        cnt += line.size() + 1;         // add 1 to account for the
newline
        auto mark = inOut.tellg();      // remember the read position
        inOut.seekp(0, fstream::end); // set the write marker to the
end
        inOut << cnt;                   // write the accumulated length
        // print a separator if this is not the last line
        if (mark != end_mark) inOut << " ";
        inOut.seekg(mark);              // restore the read position
    }
    inOut.seekp(0, fstream::end);      // seek to the end
    inOut << "\n";                      // write a newline at end-of-
file
    return 0;
}
```

1. abc\n
2. itdeln
3. f\n
4. itgit\n
5. EOF

1. abc\n
2. itdeln
3. f\n
4. itgit EOF

Primer, Fifth Edition

abcd \n
efg \n
hi \n
j \n
5 9 12 14 \n
\0

```
real    0m4.84s
user    0m1.030s
sys     0m3.43s
```

```
#include <stdlib.h>

int main (int argc, char *argv){

    string ext, dir_b ;                          M_and
                                                   or
    vector<string> fnb ;                           not
                                                   true
    for ( auto &b : fnb ){    i = 0 ;
      for ( unsigned l (3); l <= 300 ; l +=5 ){
          string nOutMR ("MR_"+ to_string (l) + "_"+ b ),
                 nOutFD ("FD_"+ to_string (l) + "_"+ b ),
                 oExt (".test"),
                 exe ("./les"),
                 condMR ( exe + " -s "+ dir_b + b+ ext + " " + to_string (l) + " > "+ nOutMR + oExt),
                 condFD ( exe + " "+ dir_b + b+ ext + "  +"to_string (l) + " > "+ nOutFD + oExt ) ;
    system ( condMR.c_str() ) ;
    system ( condFD.c_str() ) ;


    ifstream  net_MR ( outMR + oExt , ate | in | out ),
              net_FD ( outFD + oExt , ate | in | out );

    if (!net_MR ) {                    if (!net_FD ) {
        cerr << ...  ;                     cerr << ...  ;
        return EXIT_failure ;              return EXIT_failure ;
    }                                  }

    auto DM = seekg(-1, end) ;  ,  DF = seekg(-1, end);

    net_MR.seekg ( DM);                net_MR.seekg ( DF);
```

```
ret_MR >> s >> M_and[i][2];    ret_MR >> s >> M_and[i][2];
ret_MR >> s >> M_or[i][2];     ret_MR >> s >> M_or[i][2];
ret_MR >> s >> M_nt[i][2];     ret_MR >> s >> M_nt[i][2];

        // from                         // from

++i;

remove( outMR + oExt );
remove( outFD + oExt );

}
}
```

MR	品牌一	品牌二	品牌三
A店	50	100	55
B店	100	100	100
C店	77	88	99
FD	一	二	三
A	30	20	30
B	7	4	60
C	8	9	80