CSCI 3100 Software Engineering

Assignment 3

Jonathan Lam Pui Hong (1155093597)

Question 1

Please refer to Fig.1.1 and 1.2 in the appendix. A possible firing sequence for Q1.2 is

```
Patient1: P01 > P02 > P03 > P09 > P12 >           > P13 > P14 > P08
Patient2: P01 > P02 > P03 > P04 > P05 > P06 > P07 > P08
Patient3: P01                   > P02 > P03 > P04 > P05 > P06 > P07 > P08
Patient4: P01                   > P02 > P03 >           > P09 > P10 > P11 > P08

Director: 01  >           > 00        > 01  > 00        > 01  > 00  > 01
Doctor:   02  >           > 01        > 00        > 02        > 01  > 02
Nurse:    02  > 00        > 02  > 00  >     > 01  > 00  > 01  > 02  > 01  > 02
```

Question 2

Please refer to Fig.2.1, 2.2, and 2.3 in the appendix.

Question 3

Please refer to Fig.3 in the appendix. There are following assumptions:

- Clients' contact are assumed to be able to repeat, i.e. client can only have one contact, but contact can be referred by multiple clients.

- A branch's assets are those with relationship with branch, similar to the relationship between client and saving accounts. In other words, "assets" do not exist as an attribute of the branch.

Question 4.1

There are following assumptions:

- A statement is TRUE if and only if the number is a positive (non-zero and non-negative) integer.

In the tables provided, the leftmost 5 columns are the variables `c, x, y, i, j` in `valid_sequence()`; separated by a blank column, the remaining variables are those in `valid_layers()`. `c[i]` replaces c_i in the specification to denote the `i`th character in sequence `c`, where index `i` starts from 1. Elipses (…) are used to denote those neglected runs when 1) **`for all`** fails with one exception, denoted in red or 2) **`exists`** succeeds with one success, denoted in green. Light grey are used to denote those irrelevant information; in 4.1e onwards, some of these are omitted for clarity.

<u>Logic Specification</u>

```
valid_layer(i, j, m, n) ≡
(
    j = i and
    c[i] = “N”
) or
(
    n − m + 1 = 2 (j − i + 1) and
    j != i  and
    for all x (
        m <= x <= n and
        exists t
        implies (x − m = 3t − 1)
    ) implies c[x] = “S”
    and
    for all y (
        i <= y <= j and
        c[y] != “S”
    ) implies (
        (
            c[y] = “B” and
            for all w (
                2(y−i) <= w <= 3(y−i)−1
            ) implies c[m+w] = “B”
        ) or
        (
            c[y] = “N” and
            for all k (
                2(y−i) <= k <= 3(y−i)−1
            ) implies c[m+3k] = “N”
        )
    )
)
```

```
valid_sequence(m, n) ≡
(
    m = n and
    c[m] = “B”
) or
(
    c[m] = “N” and
    exists d implies (
        d >= 3 and
        for all c (3 <= c <= d) implies (
            exists x,y,i,j (
                y−x+1 = 2^(c−3) × 3 and
                j−i+1 = 2^(c−2) × 3 and
                j <= n and
                i = y + 1
            )implies valid_layer(x,y,i,j)
        )
    )
```

a. **FALSE.** There does not exists a d in which for all c there exists a pair {x, y, i, j} where all of which are non-negative integer.

b. **FALSE.** There does not exists a d in which for all c there exists a pair {x, y, i, j} that returns true in valid_layer().

															w	m+w	c[m+w]
c	x	y	i	j		n-m+1	j-i+1	x	t	c[x]	y	c[y]	2(y-i)	3(y-i)-1	k	m+3k	c[m+3k]
3	2	4	5	10		6	6	10	2	S							
	2	4	5	10				9	1.7								
	2	4	5	10				8	1.3								
	2	4	5	10				7	1	S							
	2	4	5	10				6	0.7								
	2	4	5	10				5	0.3								
	2	4	5	10							4	S					
	2	4	5	10							3	N	2	2	2	11	
	2	4	5	10							2	N	...				
	1	3	4	9		6	6	9	2	B							
	1	3	4	9				8	1.7								
	1	3	4	9				7	1.3								
	1	3	4	9				6	1	N							
	1	3	4	9				5	0.7								
	1	3	4	9				4	0.3								

c. **TRUE.** There exists a d (d = 3) in which for all c (c in [3, 3]) there exists a pair {x = 14, y = 16, i = 17, j = 22} that returns true in valid_layer().

															w	m+w	c[m+w]
c	x	y	i	j	n-m+1	j-i+1	x	t	c[x]	y	c[y]	2(y-i)	3(y-i)-1	k	m+3k	c[m+3k]
3	17	19	20	25	6	6	25	2	S							
	17	19	20	25			24	1.7								
	17	19	20	25			23	1.3								
	17	19	20	25			22	1	S							
	17	19	20	25			21	0.7								
	17	19	20	25			20	0.3								
	17	19	20	25						19	S					
	17	19	20	25						18	N	2	2	2	26	
	17	19	20	25						17	N	...				
	16	18	19	24	6	6	24	2	N							
	16	18	19	24			23	1.7								
	16	18	19	24			22	1.3								
	16	18	19	24			21	1	...							
	16	18	19	24			20	0.7								
	16	18	19	24			19	0.3								
	15	17	18	23	6	6	23	2	N							
	15	17	18	23			22	1.7								
	15	17	18	23			21	1.3								
	15	17	18	23			20	1	...							
	15	17	18	23			19	0.7								
	15	17	18	23			18	0.3								
	14	16	17	22	6	6	22	2	S							
	14	16	17	22			21	1.7								
	14	16	17	22			20	1.3								
	14	16	17	22			19	1	S							
	14	16	17	22			18	0.7								
	14	16	17	22			17	0.3								
	14	16	17	22						16	S					
	14	16	17	22						15	N	2	2	2	23	N
	14	16	17	22						14	N	0	-1	TRUE		
	...	...	...	...												

1d. **FALSE.** There does not exists a d in which for all c there exists a pair {x, y, i, j} that returns true in valid_layer().

															w	m+w	c[m+w]
c	x	y	i	j	n-m+1	j-i+1	x	t	c[x]	y	c[y]	2(y-i)	3(y-i)-1	k	m+3k	c[m+3k]
3	2	4	5	10	6	6	10	2	S							
	2	4	5	10			9	1.7								
	2	4	5	10			8	1.3								
	2	4	5	10			7	1	S							
	2	4	5	10			6	0.7								
	2	4	5	10			5	0.3								
	2	4	5	10						4	S					
	2	4	5	10						3	B	2	2	2	7	S
	2	4	5	10						2	N	...				
	1	3	4	9	6	6	9	2	B							
	1	3	4	9			8	1.7								
	1	3	4	9			7	1.3								
	1	3	4	9			6	1	...							
	1	3	4	9			5	0.7								
	1	3	4	9			4	0.3								

1e. **FALSE.** There does not exists a d in which for all c there exists a pair {x, y, i, j} that returns true in valid_layer().

c	x	y	i	j	n-m+1	j-i+1	x	t	c[x]	y	c[y]	2(y-i)	3(y-i)-1	k	m+3k	c[m+3k]
3	14	16	17	22	6	6	22	2	S							
	14	16	17	22			21	1.7								
	14	16	17	22			20	1.3								
	14	16	17	22			19	1	S							
	14	16	17	22			18	0.7								
	14	16	17	22			17	0.3								
	14	16	17	22						16	S					
	14	16	17	22						15	B	2	2	2	19	S
	14	16	17	22						14	B	...				
	13	15	16	21	6	6	21	2	B							
	13	15	16	21			18	1	...							
	12	14	15	20	6	6	20	2	B							
	12	14	15	20			17	1	...							
	11	13	14	19	6	6	19	2	S							
	11	13	14	19			16	1	S							
	11	13	14	19						13	S					
	11	13	14	19						12	N	2	2	2	20	B
	11	13	14	19						11	B	...				
	10	12	13	18	6	6	18	2	B							
	10	12	13	18			15	1	...							
	9	11	12	17	6	6	17	2	B							
	9	11	12	17			14	1	...							
	8	10	11	16	6	6	16	2	S							
	8	10	11	16			13	1	S							
	8	10	11	16						10	S					
	8	10	11	16						9	B	2	2	2	13	S
	8	10	11	16						8	B	...				
	7	9	10	15	6	6	15	2	B							
	7	9	10	15			12	1	...							
	6	8	9	14	6	6	14	2	B							
	6	8	9	14			11	1	...							
	5	7	8	13	6	6	13	2	S							
	5	7	8	13			10	1	S							
	5	7	8	13						7	S					
	5	7	8	13						6	B	2	2	2	10	S
	5	7	8	13						5	N	...				
	4	6	7	12	6	6	12	2	N							
	4	6	7	12			9	1	...							
	3	5	6	11	6	6	11	2	B							
	3	5	6	11			8	1	...							
	2	4	5	10	6	6	10	2	S							
	2	4	5	10			7	1	S							
	2	4	5	10						4	S					
	2	4	5	10						3	B	2	2	2	7	S
	2	4	5	10						2	N	...				
	1	3	4	9	6	6	9	2	B							
	1	3	4	9			6	1	...							
...																

1f. **FALSE.** There does not exists a d in which for all c there exists a pair {x, y, i, j} that returns true in valid_layer(). (Ref: Appendix.)

Question 4.2

In valid_sequence(), there are two main cases:

1. when the sequence starts with B as the only letter, which will then pass the predicate, or;
2. when the sequence starts with N, and the sequence has at least 9 letters, which will then pass x, y, i, j to valid_layer().

Each pass to valid_layer() will only concern a sub-sequence between x to j with a length of $9 \times 2^{c-3}$, where c starts from 3. The sequence is further divided into two sub-sub-sequences between x to y and i to j, where x to y occupies one-third of the sub-sequence and i to j the remaining two-thirds.

In the following example, `ABCDEF123456789012` is the sub-sequence (with c = 4),

```
NXXXXXABCDEF123456789012XXXXXXX
      x    yi          j
```

In valid_layer(), given the call condition from valid_sequence(), the following sections have fixed outputs:

```
1. j = i and c[i] = "N"          FALSE
2. n - m + 1 = 2 ( j - i + 1)    TRUE
3. j != i                        TRUE
```

`for all x ( m <= x <= n and exists t implies (x - m = 3t - 1)) implies c[x] = "S"` checks whether the latter sub-sub-sequence is all composed of multiples of XXS.

`for all y ( i <= y <= j and c[y] != "S" ) implies ( ... )` checks whether the letters, excluding S, of the former sub-sub-sequence satisfy a condition. The condition depends on whether the letter is E or N; however, the condition is almost guaranteed to fail.

For example, say if c = {3, 4}, the first sub-sequence is all B, "v" marks y position, and "-" marks the range of m+w that the condition checks,

```
[ 3, 3]   v      -                  v    -    .
[ 5, 6]    v       --                 v    --  .
[ 7, 9]     v        ---                      .
[ 9,12]      v        ----                    .
[11,15]       v          -----                .
         BBBBBB12S45S78S01SXXX      BBB12S45S
         i     jm          n        i jm     n
```

If the `for all x () implies ()` predicate passes, the condition `for all w () implies c[m+w] = "B"` **always** fails.

Similarly, for example, say if c = {3, 4}, the first sub-sequence is all N, “v” marks y position, and “-“ marks the range of m+3k that the condition checks,

```
[ 7, 7]   v           -                v       -
[13,16]    v               ---...       v       ...
[19,26]     v                 ...               ...
[25,34]      v                ...               ...
[31,43]       v               ...               ...
         NNNNNN12S45S78S01SXXX...    BBB12S45SX...
         i     jm          n         i jm     n
```

If the **for all** x () **implies** () predicate passes, the condition **for all** k () **implies** c[m+3k] = **“N”** always fails **with the only exception** of k = 5.

In short, the valid_layer() call within valid_sequence() is problematic in a sense that the only time it returns TRUE is when

1. an “N” situates at the second (i+1) position of the first sub-sub-sequence AND an “N” situates at the seventh (m+4) position of the second sub-sub-sequence, or;
2. there are all “B”s or “N”s within the range of m+w or m+3k, respectively, that is out of bound of the sub-sequence.

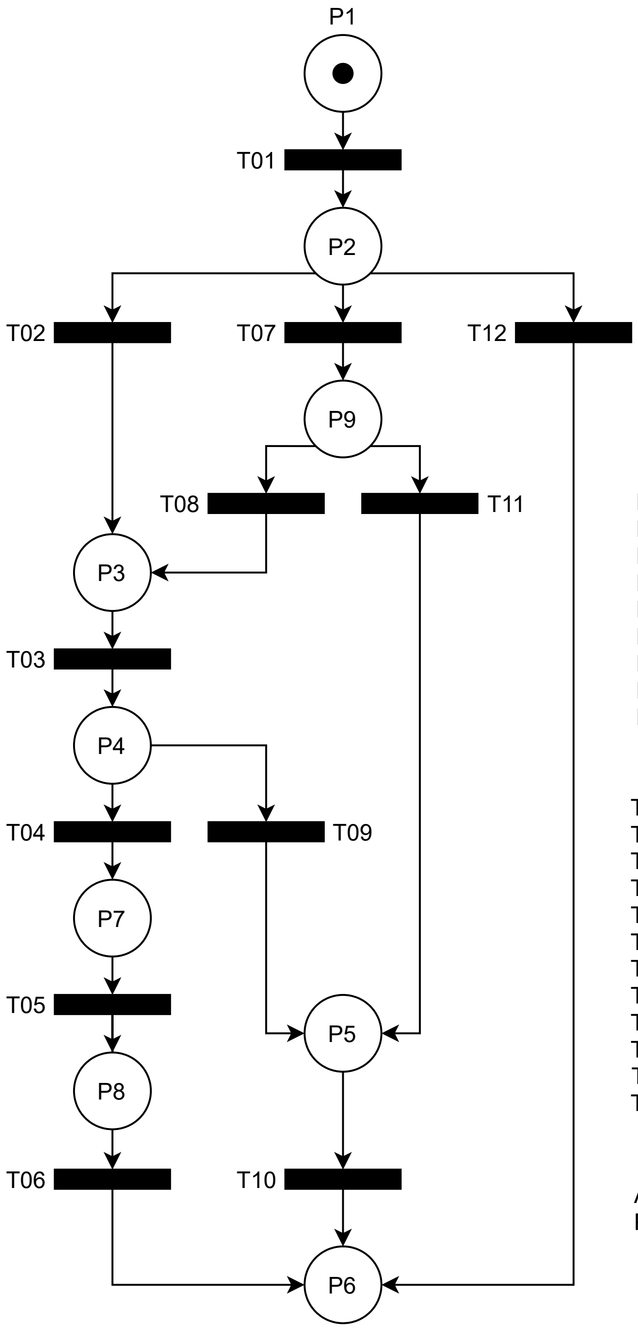
Therefore, given the above conclusion, I cannot think of what valid_sequence(m, n) will check in data structure. The whole question is rendered as obsolete since the combined use of these two logic specifications has no apparent meaningful outcome.

Question 4a

(So much time than planned have been spent on Q4 to decipher the logic behind these specification that I have to forfeit this problem. I am sure I have the ability to finish this bonus problem but at the same time regretful to come to this conclusion since HW3 has been too overwhelming. Especially in Q4, it appears to be designed only for the sake of homework, but not applicable in the foreseeable future. I apologize in advance if there were any error in my reasoning in Q4, but too much is too much.)

Appendix

Fig. 1.1

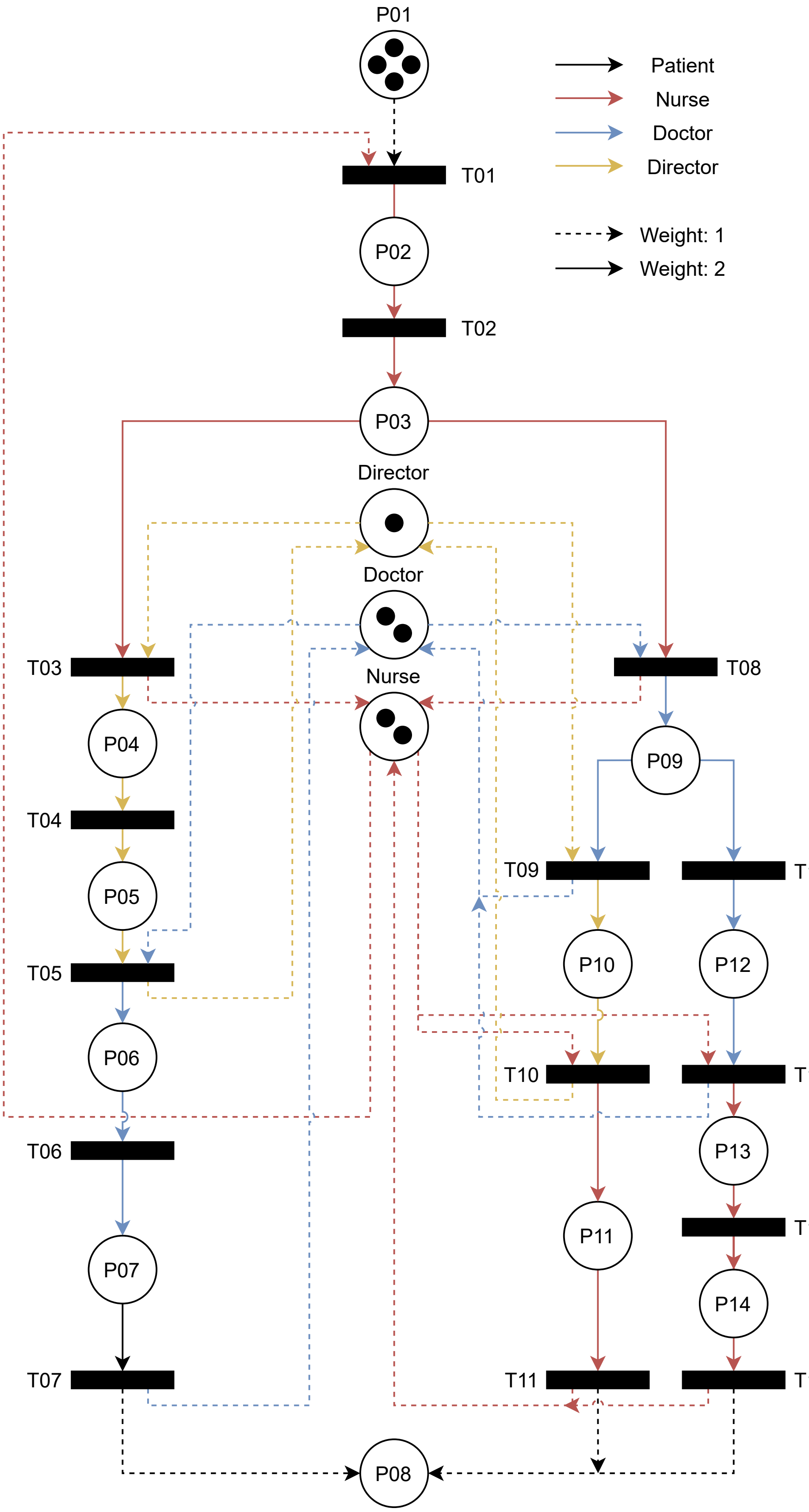


P1 student_arrival
P2 health_code_check
P3 send_to_hospital
P4 covid_test
P5 go_through_qurantine
P6 enter_campus
P7 medical_treatment
P8 student_cured
P9 measure_body_temperature

T01 health_code_check
T02 send_to_hospital
T03 conduct_covid_test
T04 medical_treatment
T05 cure_process
T06 enter_campus
T07 measure_body_temperature
T08 send_to_hospital
T09 quarantine
T10 enter_campus
T11 quarantine
T12 enter_campus

A possible firing sequence:
P1 > P2 > P9 > P5 > P6

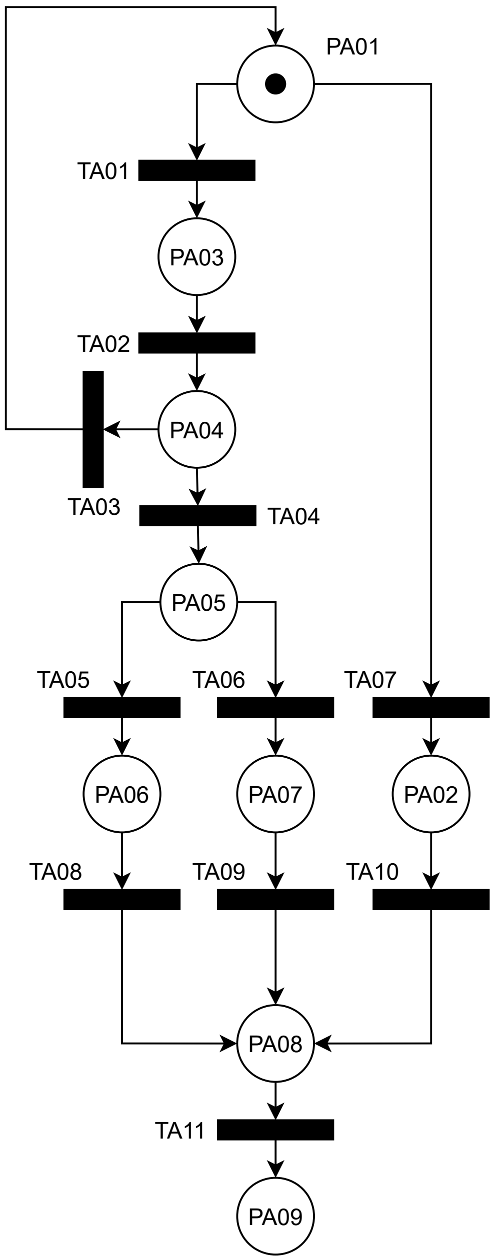
Fig. 1.2



P01	patient_arrive
P02	patient_accepted
P03	covid_tested
P04	case_double_checked
P05	fever_case_endorsed
P06	patient_treated
P07	patient_cured
P08	patient_left
P09	symptoms_checked
P10	transfer_endorsed
P11	patient_transferred_to_ARISE
P12	covid_patient_treated
P13	patient_transferred_to_ward
P14	covid_patient_cured

T01	accept_patient
T02	conduct_covid_test
T03	ask_for_double_check
T04	endorse_fever_case
T05	treat_fever
T06	await_for_fever_cured
T07	patient_leaves
T08	check_symptoms
T09	suggest_transfer
T10	transfer_patient_to_ARISE
T11	patient_leaves
T12	treat_covid
T13	transfer_patient_to_ward
T14	await_for_covid_cured
T15	patient_leaves

Fig. 2.1



P*01 *_turn_ready
P*02 *_attack
P*03 *_critical_attack
P*04 check_^_HP
P*05 *_critical_attack_judgement
P*06 *_critical_attack_fails
P*07 *_critical_attack_succeeds
P*08 *_action_end
P*09 *_turn_wait

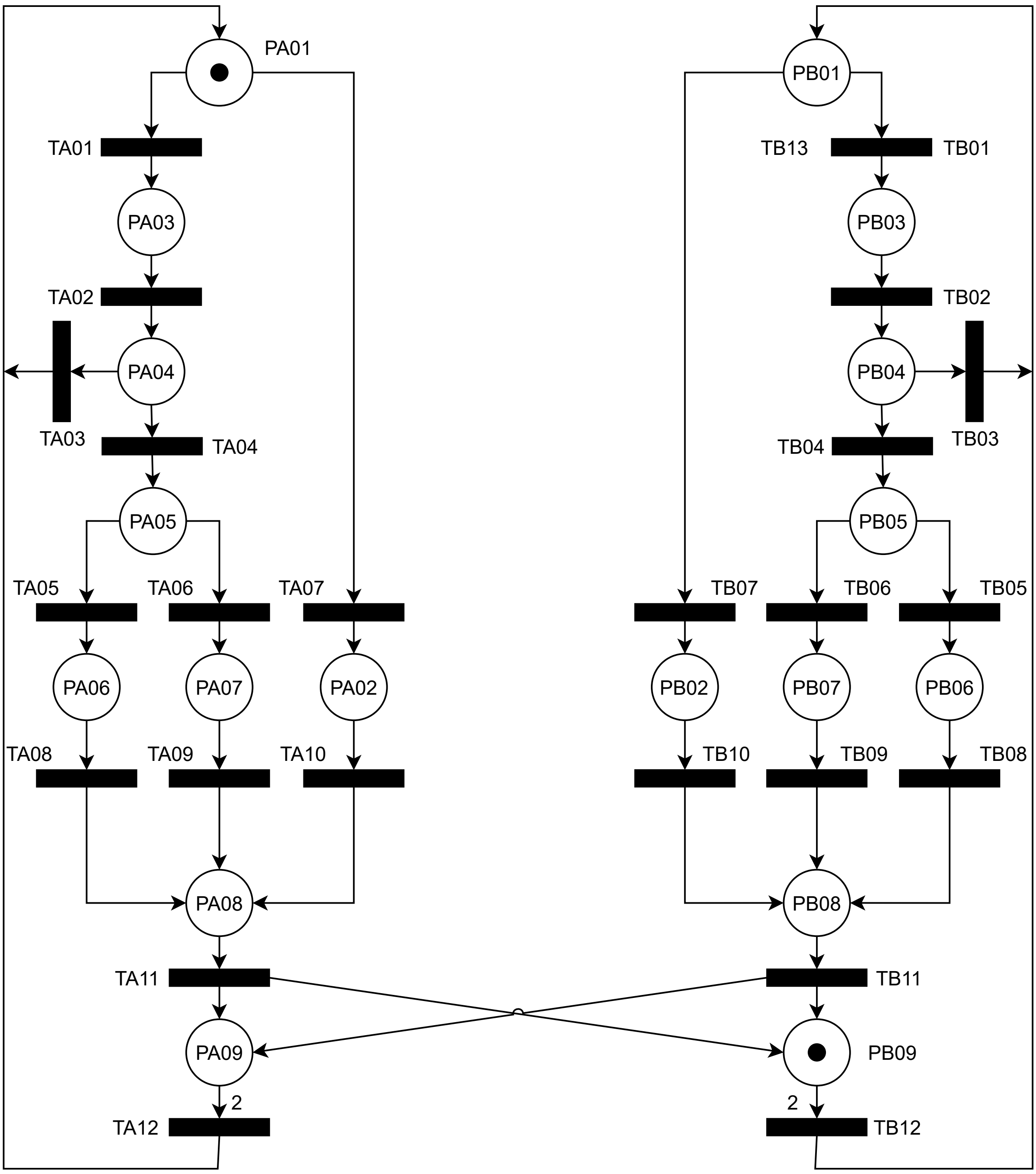
* A / B
^ B / A

T*01 *_chooses_crit
T*02 check_^_HP
T*03 return_*_ready
T*04 judge_*_crit_valid
T*05 fails_*_crit
T*06 succeeds_*_crit
T*07 *_chooses_attack
T*08 distribute_crit_fails_damage
T*09 distribute_crit_succeeds_damage
T*10 distribute_attack_damage
T*11 ready_*

* A / B
^ B / A

A possible firing sequence:
PA01 > PA02 > PA08 > PA09

Fig. 2.2



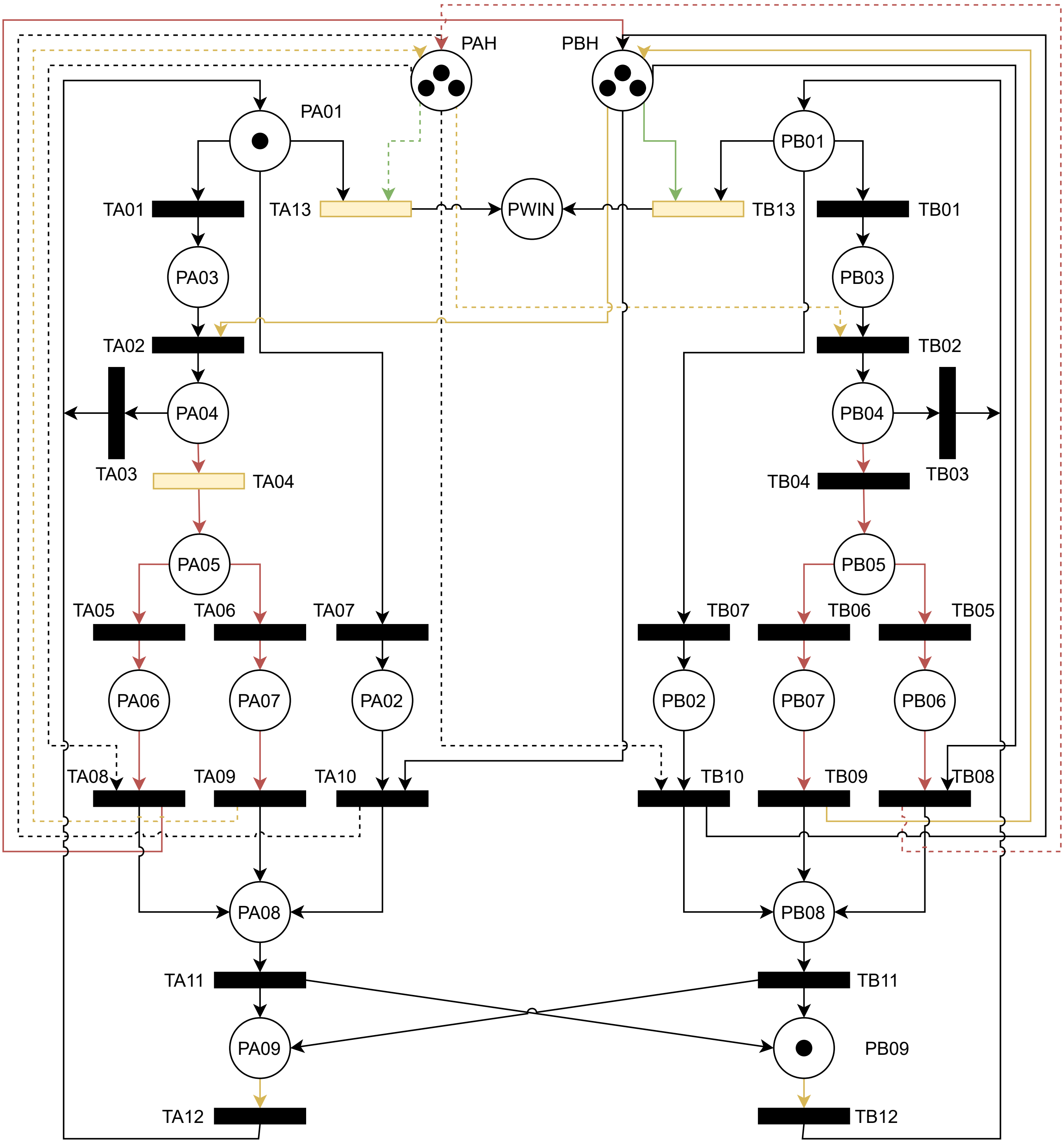
P*01	*_turn_ready
P*02	*_attack
P*03	*_critical_attack
P*04	check_^_HP
P*05	*_critical_attack_judgement
P*06	*_critical_attack_fails
P*07	*_critical_attack_succeeds
P*08	*_action_end
P*09	*_turn_wait
*	A / B
^	B / A

T*01	*_chooses_crit
T*02	check_^_HP
T*03	return_*_ready
T*04	judge_*_crit_valid
T*05	fails_*_crit
T*06	succeeds_*_crit
T*07	*_chooses_attack
T*08	distribute_crit_fails_damage
T*09	distribute_crit_succeeds_damage
T*10	distribute_attack_damage
T*11	ready_*
T*12	restart_*_round
*	A / B
^	B / A

A possible firing sequence:

PA01		
PA02		
PA08	>	PB09
PA09		PB01
		PB03
		PB04
		PB05
		PB06
		PB08
		PB09

Fig. 2.3



P*01	*_turn_ready
P*02	*_attack
P*03	*_critical_attack
P*04	check_^_HP
P*05	*_critical_attack_judgement
P*06	*_critical_attack_fails
P*07	*_critical_attack_succeeds
P*08	*_action_end
P*09	*_turn_wait
P*H	*_health
PWIN	game_ends
*	A / B
^	B / A

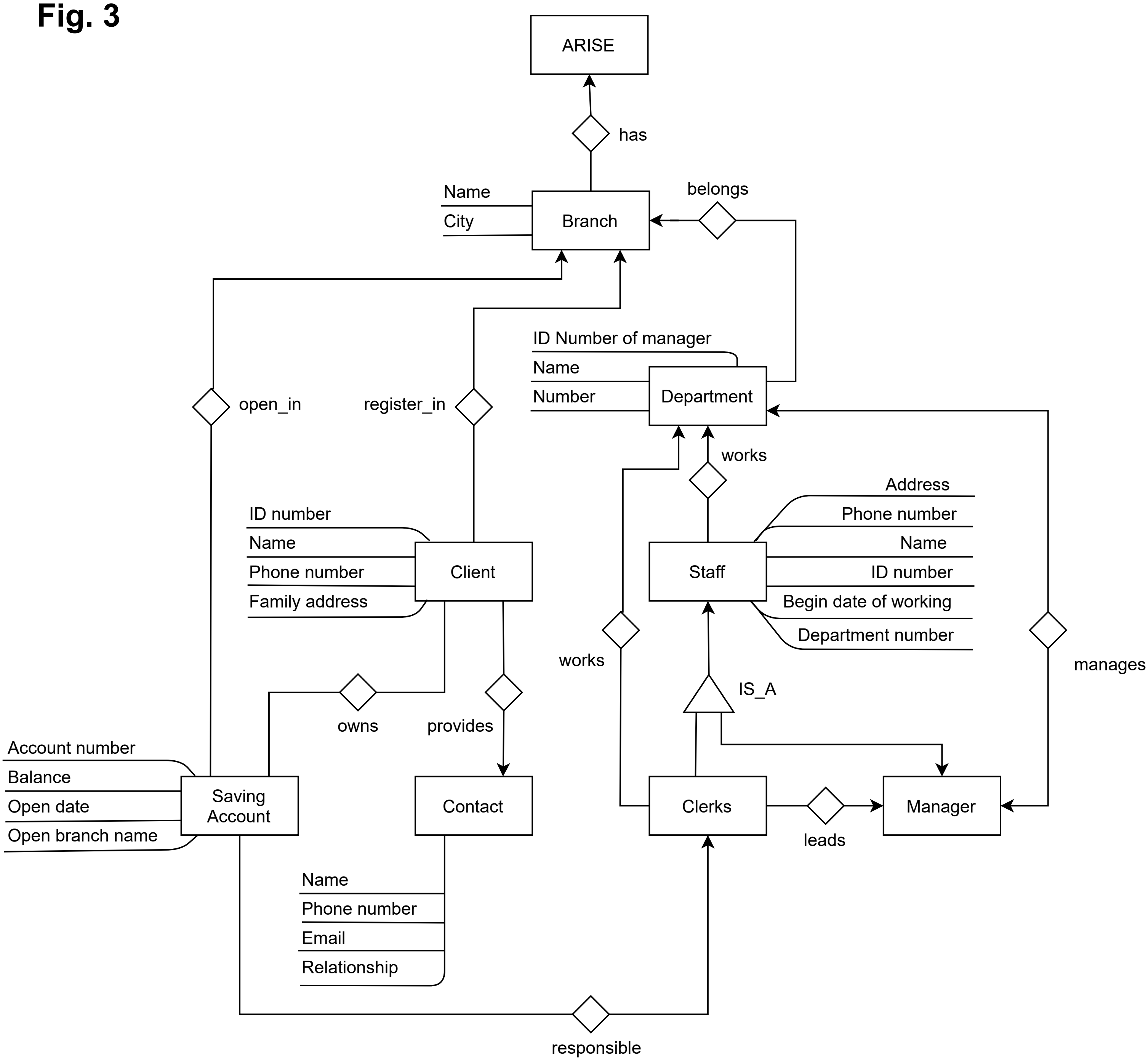
T*01	*_chooses_crit
T*02	check_^_HP
T*03	return_*_ready
T*04	judge_*_crit_valid
T*05	fails_*_crit
T*06	succeeds_*_crit
T*07	*_chooses_attack
T*08	distribute_crit_fails_damage
T*09	distribute_crit_succeeds_damage
T*10	distribute_attack_damage
T*11	ready_*
T*12	restart_*_round
T*13	*_wins
*	A / B
^	B / A

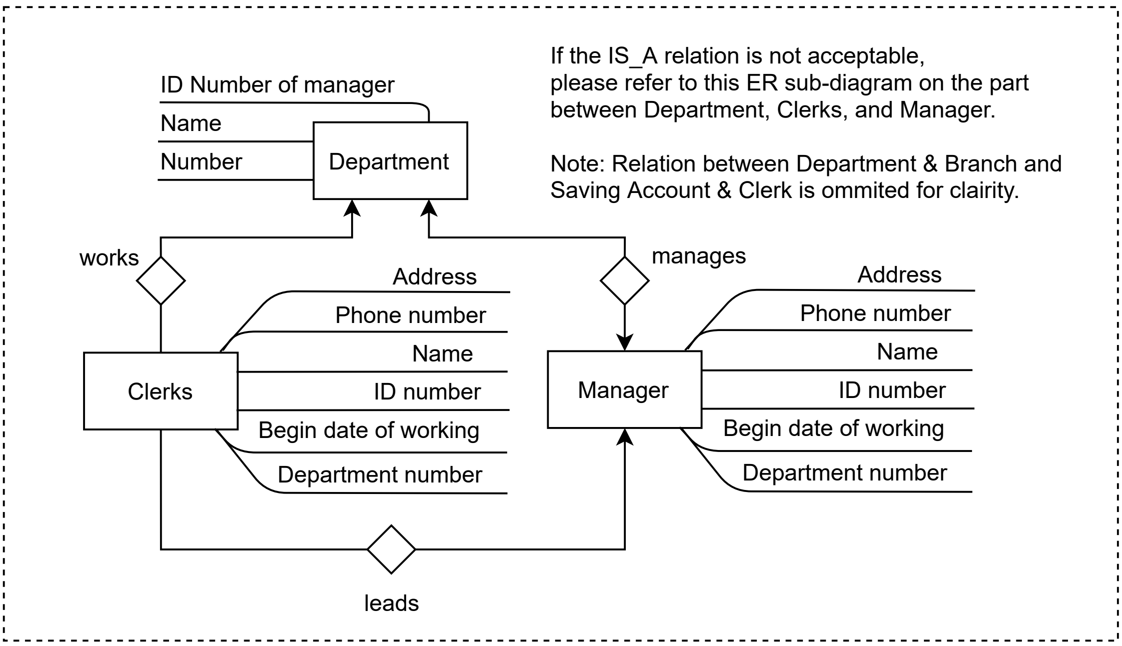
Weight: 1
Weight: 2
Weight: 3
Weight: 6

A HP
B HP

Priority: 2
Priority: 1

Fig. 3


ARISE
has
Branch
Name
City
belongs
open_in
register_in
Department
ID Number of manager
Name
Number
works
Client
ID number
Name
Phone number
Family address
Staff
Address
Phone number
Name
ID number
Begin date of working
Department number
works
manages
IS_A
owns
provides
Saving Account
Account number
Balance
Open date
Open branch name
Contact
Clerks
leads
Manager
Name
Phone number
Email
Relationship
responsible


If the IS_A relation is not acceptable,
please refer to this ER sub-diagram on the part
between Department, Clerks, and Manager.
Note: Relation between Department & Branch and
Saving Account & Clerk is ommited for clairity.
Department
ID Number of manager
Name
Number
works
manages
Clerks
Address
Phone number
Name
ID number
Begin date of working
Department number
Manager
Address
Phone number
Name
ID number
Begin date of working
Department number
leads

c	x	y	i	j	n-m+1	j-i+1	x	t	c[x]	y	c[y]	2(y-i)	3(y-i)-1	w k	m+w m+3k	c[m+w] c[m+3k]
3	38	40	41	46	6	6	46	2	S							
	38	40	41	46			45	1.7								
	38	40	41	46			44	1.3								
	38	40	41	46			43	1	S							
	38	40	41	46			42	0.7								
	38	40	41	46			41	0.3								
	38	40	41	46						40	S					
	38	40	41	46						39	B	2	2	2	43	S
	38	40	41	46						38	B	...				
	37	39	40	45	6	6	45	2	B							
	37	39	40	45			44	1.7								
	37	39	40	45			43	1.3								
	37	39	40	45			42	1	...							
	37	39	40	45			41	0.7								
	37	39	40	45			40	0.3								
	36	38	39	44	6	6	44	2	B							
	36	38	39	44			43	1.7								
	36	38	39	44			42	1.3								
	36	38	39	44			41	1	...							
	36	38	39	44			40	0.7								
	36	38	39	44			39	0.3								
	35	37	38	43	6	6	43	2	S							
	35	37	38	43			42	1.7								
	35	37	38	43			41	1.3								
	35	37	38	43			40	1	S							
	35	37	38	43			39	0.7								
	35	37	38	43			38	0.3								
	35	37	38	43						37	S					
	35	37	38	43						36	B	2	2	2	40	S
	35	37	38	43						35	B	...				
	34	36	37	42	6	6	42	2	B							
	34	36	37	42			39	1	...							
	33	35	36	41	6	6	41	2	B							
	33	35	36	41			38	1	...							
	32	34	35	40	6	6	40	2	S							
	32	34	35	40			37	1	S							
	32	34	35	40						34	S					
	32	34	35	40						33	B	2	2	2	37	S
	32	34	35	40						32	B	...				
	31	33	34	39	6	6	39	2	B							
	31	33	34	39			36	1	...							
	30	32	33	38	6	6	38	2	B							
	30	32	33	38			35	1	...							
	29	31	32	37	6	6	37	2	S							
	29	31	32	37			34	1	S							
	29	31	32	37						31	S					

29	31	32	37						30	B	2	2	2	34	S
29	31	32	37						29	B	…				
28	30	31	36	6	6	36	2	B							
28	30	31	36			33	1	…							
27	29	30	35	6	6	35	2	B							
27	29	30	35			32	1	…							
26	28	29	34	6	6	34	2	S							
26	28	29	34			31	1	S							
26	28	29	34						28	S					
26	28	29	34						27	B	2	2	2	31	S
26	28	29	34						26	B	…				
25	27	28	33	6	6	33	2	B							
25	27	28	33			30	1	…							
24	26	27	32	6	6	32	2	B							
24	26	27	32			29	1	…							
23	25	26	31	6	6	31	2	S							
23	25	26	31			28	1	S							
23	25	26	31						25	S					
23	25	26	31						24	N	2	2	2	32	B
23	25	26	31						23	N	…				
22	24	25	30	6	6	30	2	B							
22	24	25	30			27	1	…							
21	23	24	29	6	6	29	2	B							
21	23	24	29			26	1	…							
20	22	23	28	6	6	28	2	S							
20	22	23	28			25	1	S							
20	22	23	28						22	S					
20	22	23	28						21	B	2	2	2	25	S
20	22	23	28						20	B	…				
19	21	22	27	6	6	27	2	B							
19	21	22	27			24	1	…							
18	20	21	26	6	6	26	2	B							
18	20	21	26			23	1	…							
17	19	20	25	6	6	25	2	S							
17	19	20	25			22	1	S							
17	19	20	25						19	S					
17	19	20	25						18	B	2	2	2	22	S
17	19	20	25						17	B	…				
16	18	19	24	6	6	24	2	N							
16	18	19	24			21	1	…							
15	17	18	23	6	6	23	2	N							
15	17	18	23			20	1	…							
14	16	17	22	6	6	22	2	S							
14	16	17	22			19	1	S							
14	16	17	22						16	S					
14	16	17	22						15	B	2	2	2	19	S
14	16	17	22						14	B	…				
13	15	16	21	6	6	21	2	B							

13	15	16	21			18	1	...							
12	14	15	20	6	6	20	2	B							
12	14	15	20			17	1	...							
11	13	14	19	6	6	19	2	S							
11	13	14	19			16	1	S							
11	13	14	19						13	S					
11	13	14	19						12	B	2	2	2	16	S
11	13	14	19						11	N	...				
10	12	13	18	6	6	18	2	B							
10	12	13	18			15	1	...							
9	11	12	17	6	6	17	2	B							
9	11	12	17			14	1	...							
8	10	11	16	6	6	16	2	S							
8	10	11	16			13	1	S							
8	10	11	16						10	S					
8	10	11	16						9	B	2	2	2	13	S
8	10	11	16						8	B	...				
7	9	10	15	6	6	15	2	B							
7	9	10	15			12	1	...							
6	8	9	14	6	6	14	2	B							
6	8	9	14			11	1	...							
5	7	8	13	6	6	13	2	S							
5	7	8	13			10	1	S							
5	7	8	13						7	S					
5	7	8	13						6	N	2	2	2	14	B
5	7	8	13						5	N	...				
4	6	7	12	6	6	12	2	B							
4	6	7	12			9	1	...							
3	5	6	11	6	6	11	2	N							
3	5	6	11			8	1	...							
2	4	5	10	6	6	10	2	S							
2	4	5	10			7	1	S							
2	4	5	10						4	S					
2	4	5	10						3	B	2	2	2	7	S
2	4	5	10						2	N	...				
1	3	4	9	6	6	9	2	B							
1	3	4	9			6	1	...							

...