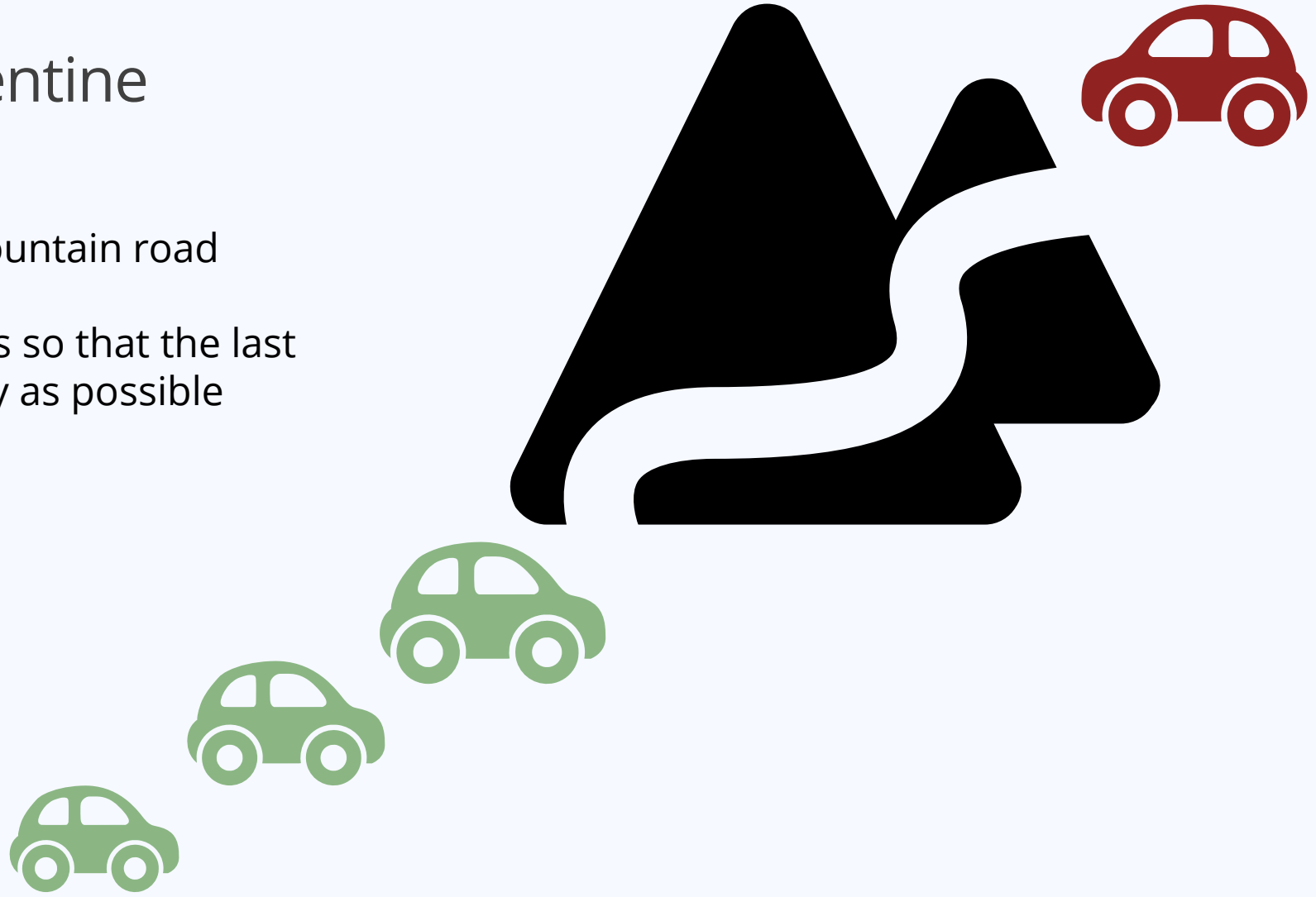
SINGLE-LANE SERPENTINE

PROBLEM H

Marius Spangenberger, Tobias Fröhlich

Single-lane Serpentine

- Narrow, single lane mountain road
- Two way traffic
- Schedule incoming cars so that the last car leaves road as early as possible



Quirks

Mountain road

- No overtaking
- No reordering cars at the ends of the road
- 10 seconds safety delay between two consecutive cars from the same direction
- Allowed to drive as soon as oncoming car leaves the road

Cars

- Individual arrival times at mountain road
- Individual travel times on mountain road

Input

Number of cars n
- $1 \leq n \leq 200$

Car direction
- "A" or "B"

Arrival time t
- $0 \leq t \leq 100000$

Travel time d
- $1 \leq d \leq 100000$

```
4
A     0    60
B    19    10
B    80    20
A    85   100
```

Sorted by arrival time, ascending
No two cars arrive at the same time

Output

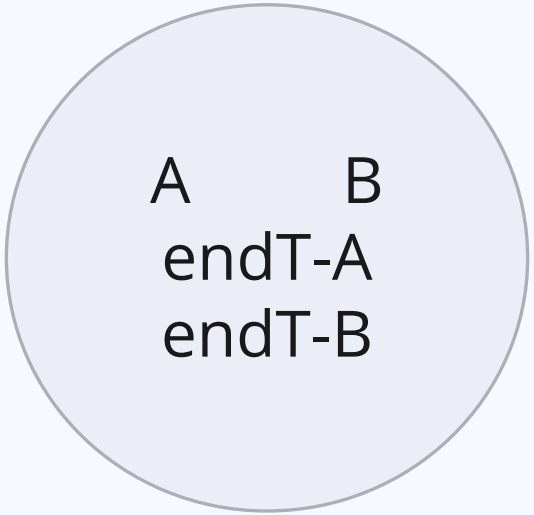
Optimal point in time, when the last car leaves the road

200

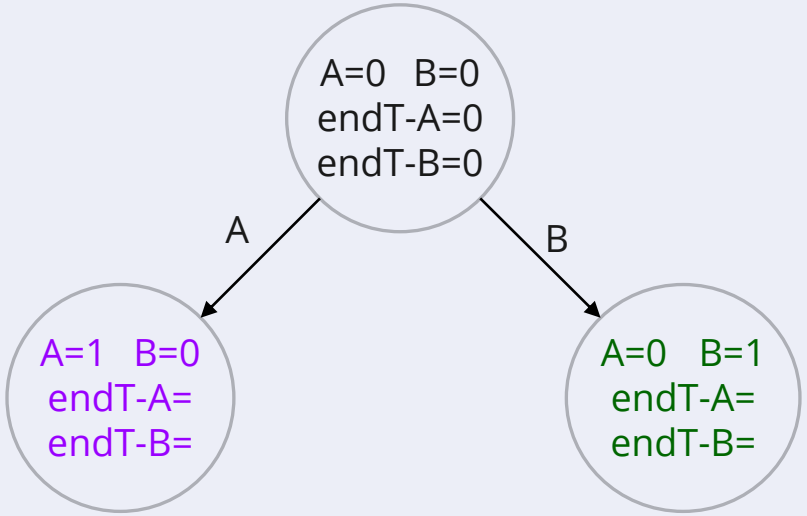
Solution Approach: Dynamic Program

3D-Array [a][b][c]:

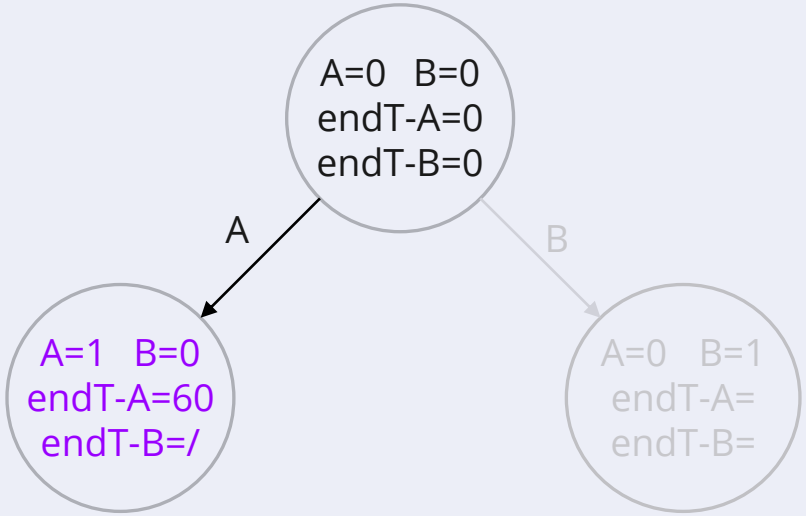
- [a]: Number of cars, that already crossed mountain road from direction A
- [b]: Number of cars, that already crossed mountain road from direction B
- [0][0][]: No cars crossed the road yet
- [a_{max}][b_{max}][]: all cars crossed the road
- [c]: Completion Time* for (a+b) cars, separated into
  - [0]: Last car that crossed the road had direction A
  - [1]: Last car that crossed the road had direction B



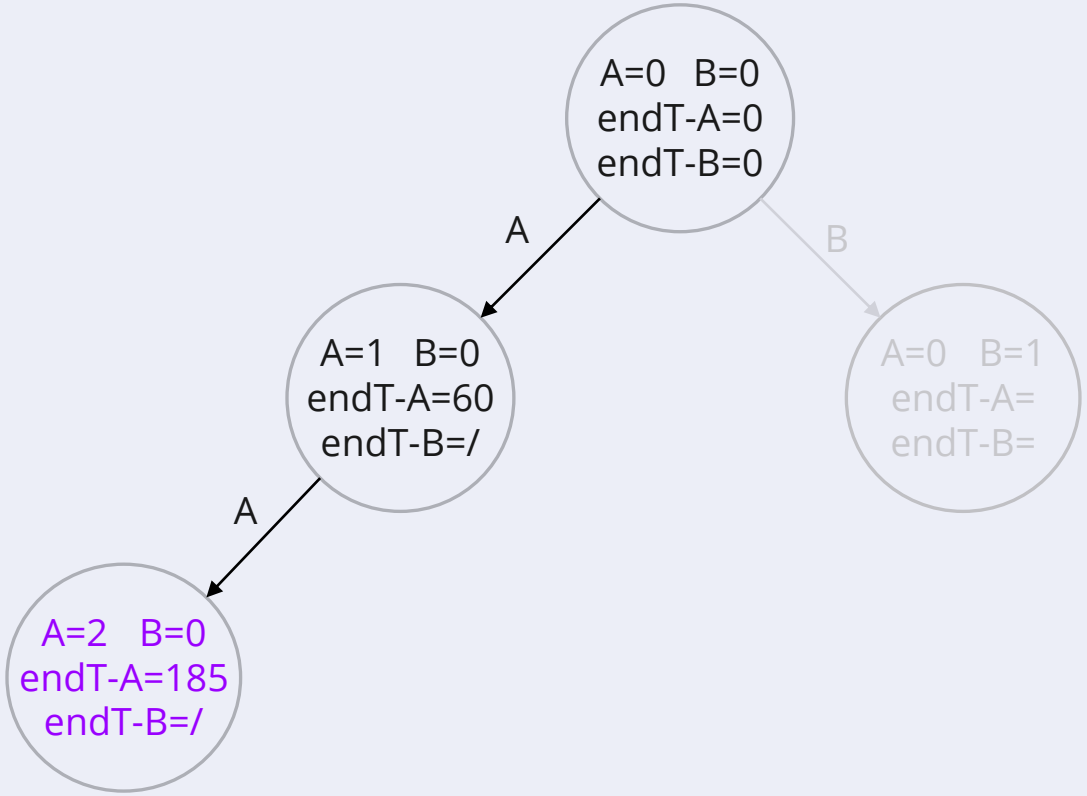
*Completion Time:
Time stamp, when car completed the mountain road/arrives at the end of the road



4		
A	0	60
B	19	10
B	80	20
A	85	100



4		
A	0	60
B	19	10
B	80	20
A	85	100



4		
A	0	60
B	19	10
B	80	20
A	85	100

$$\text{startT A} = \max(0 + 10, 85) = 85$$
$$\text{endT A} = \max(85 + 100, 60 + 10) = 185$$

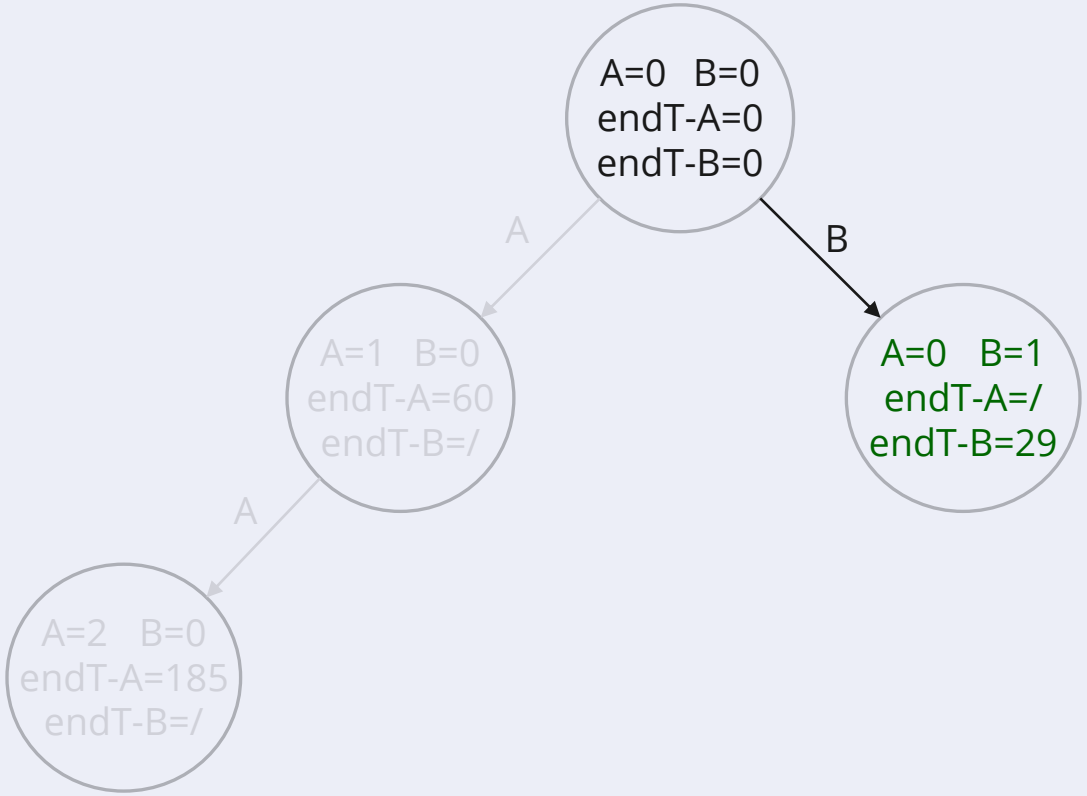
Calculating End Time

Swap

- $\text{startT} = \max(\text{endT previous}, \text{arrival time})$
- $\text{endT} = \text{startT} + \text{travel time}$

No swap

- $\text{startT} = \max(\text{startT previous} + 10, \text{arrival time})$
- $\text{endT} = \max(\text{startT} + \text{travel time}, \text{endT previous} + 10)$



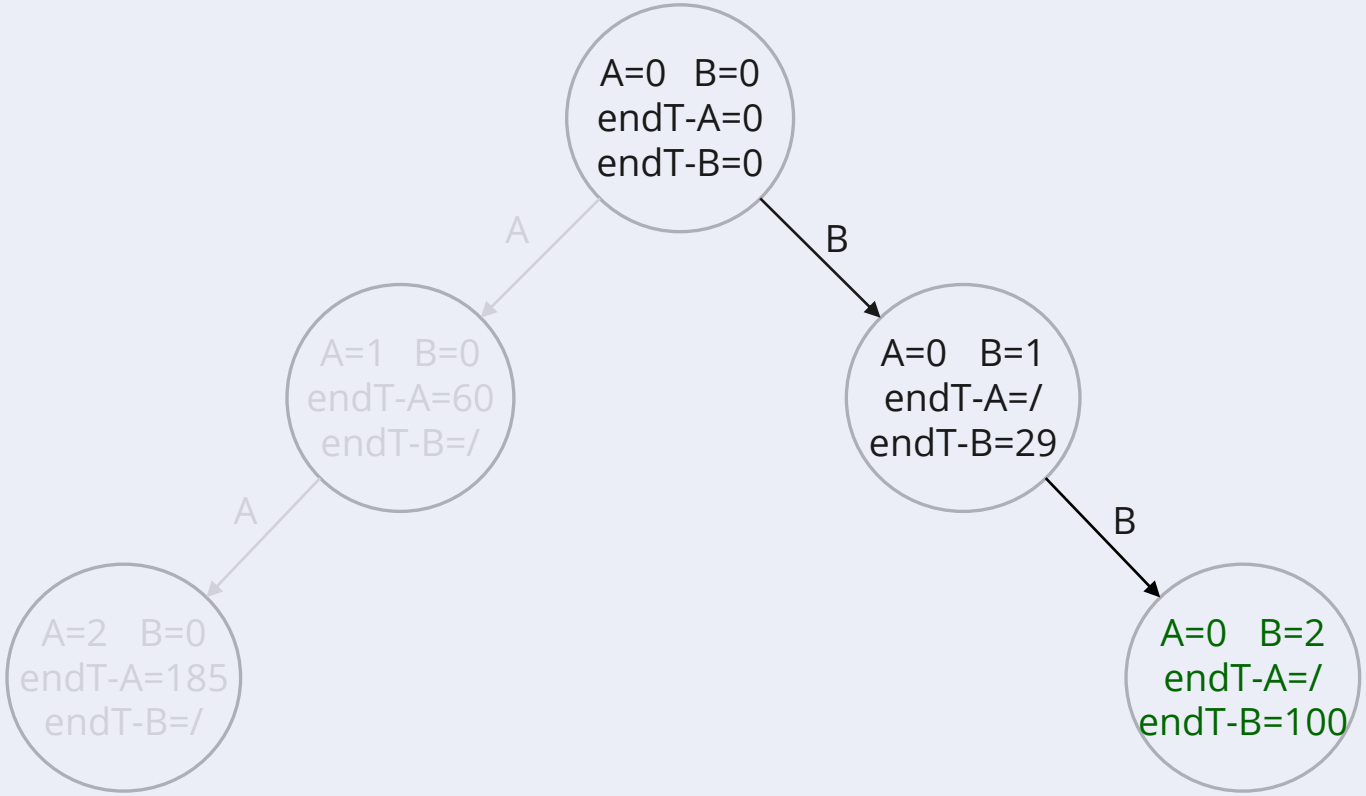
4		
A	0	60
B	19	10
B	80	20
A	85	100

swap.startT = max(endT previous, arrival time)

swap.endT = startT + travel time

noswap.startT = max(startT previous + 10, arrival time)

noswap.endT = max(startT + travel time, endT previous + 10)



4		
A	0	60
B	19	10
B	80	20
A	85	100

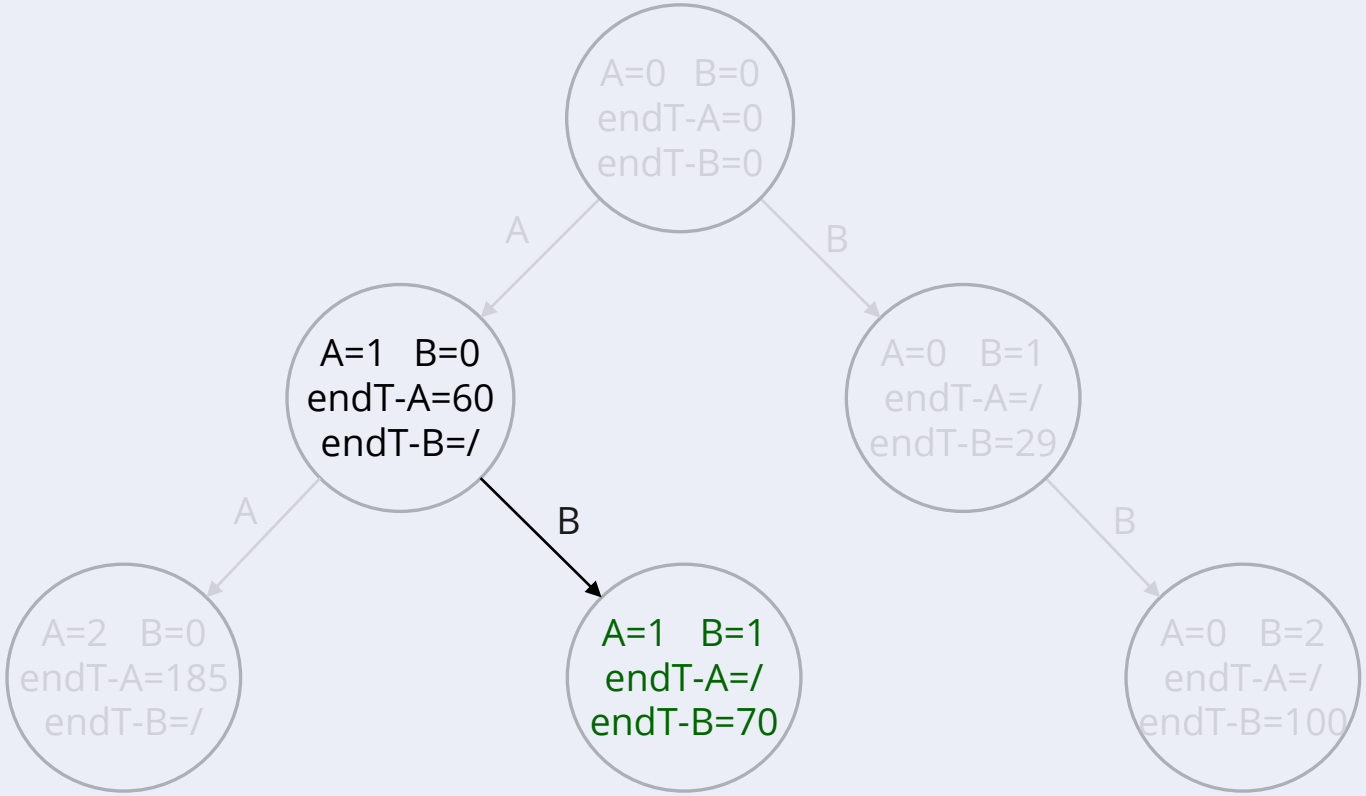
$$\text{startT B} = \max(19 + 10, 80) = 80$$
$$\text{endT B} = \max(80 + 20, 29 + 10) = 100$$

swap.startT = max(endT previous, arrival time)

swap.endT = startT + travel time

noswap.startT = max(startT previous + 10, arrival time)

noswap.endT = max(startT + travel time, endT previous + 10)



4		
A	0	60
B	19	10
B	80	20
A	85	100

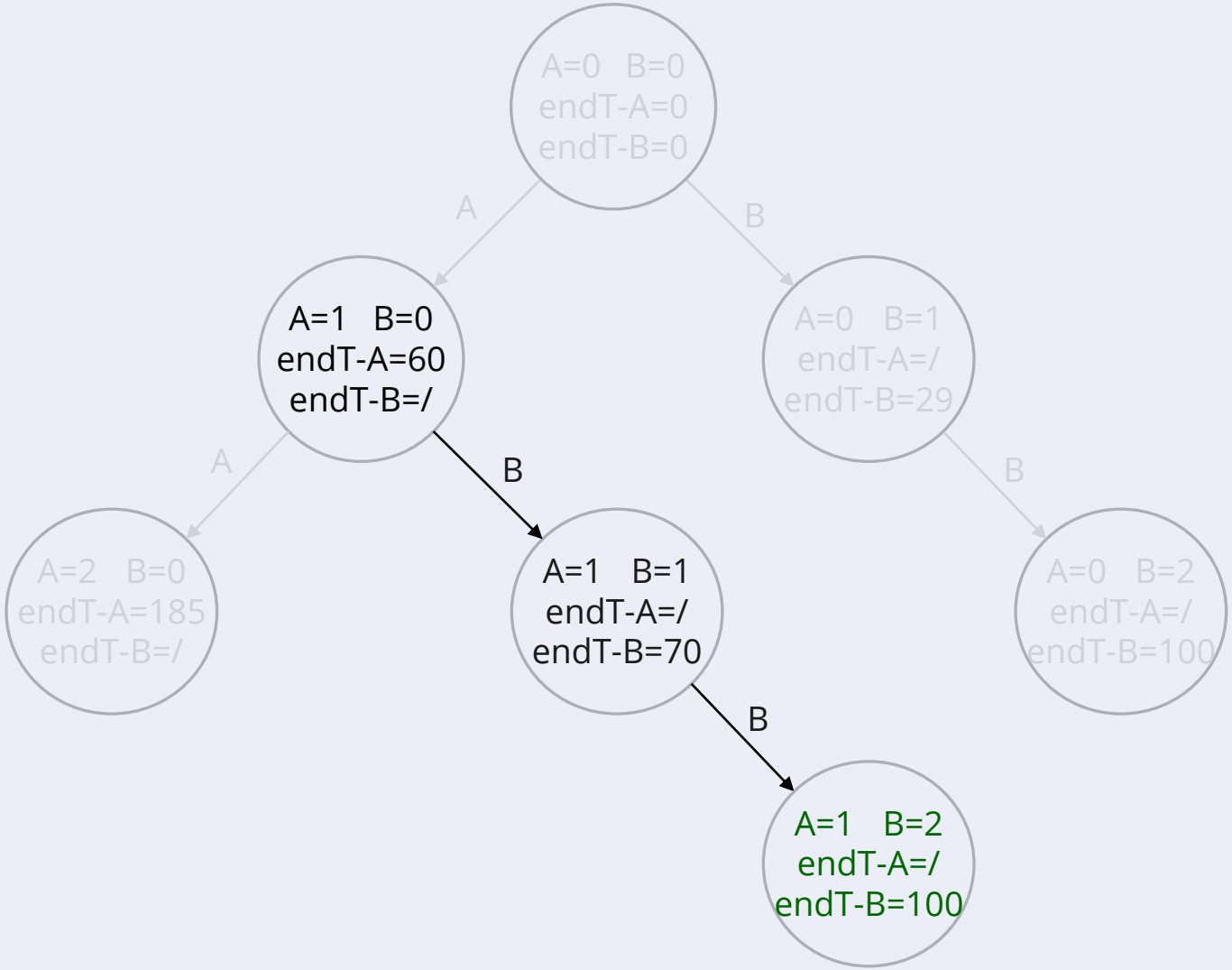
$$\text{startT B} = \max(60, 19) = 60$$
$$\text{endT B} = 60 + 10 = 70$$

swap.startT = max(endT previous, arrival time)

swap.endT = startT + travel time

noswap.startT = max(startT previous + 10, arrival time)

noswap.endT = max(startT + travel time, endT previous + 10)



4		
A	0	60
B	19	10
B	80	20
A	85	100

$$\text{startT B} = \max(60, 19) = 60$$
$$\text{endT B} = 60 + 10 = 70$$

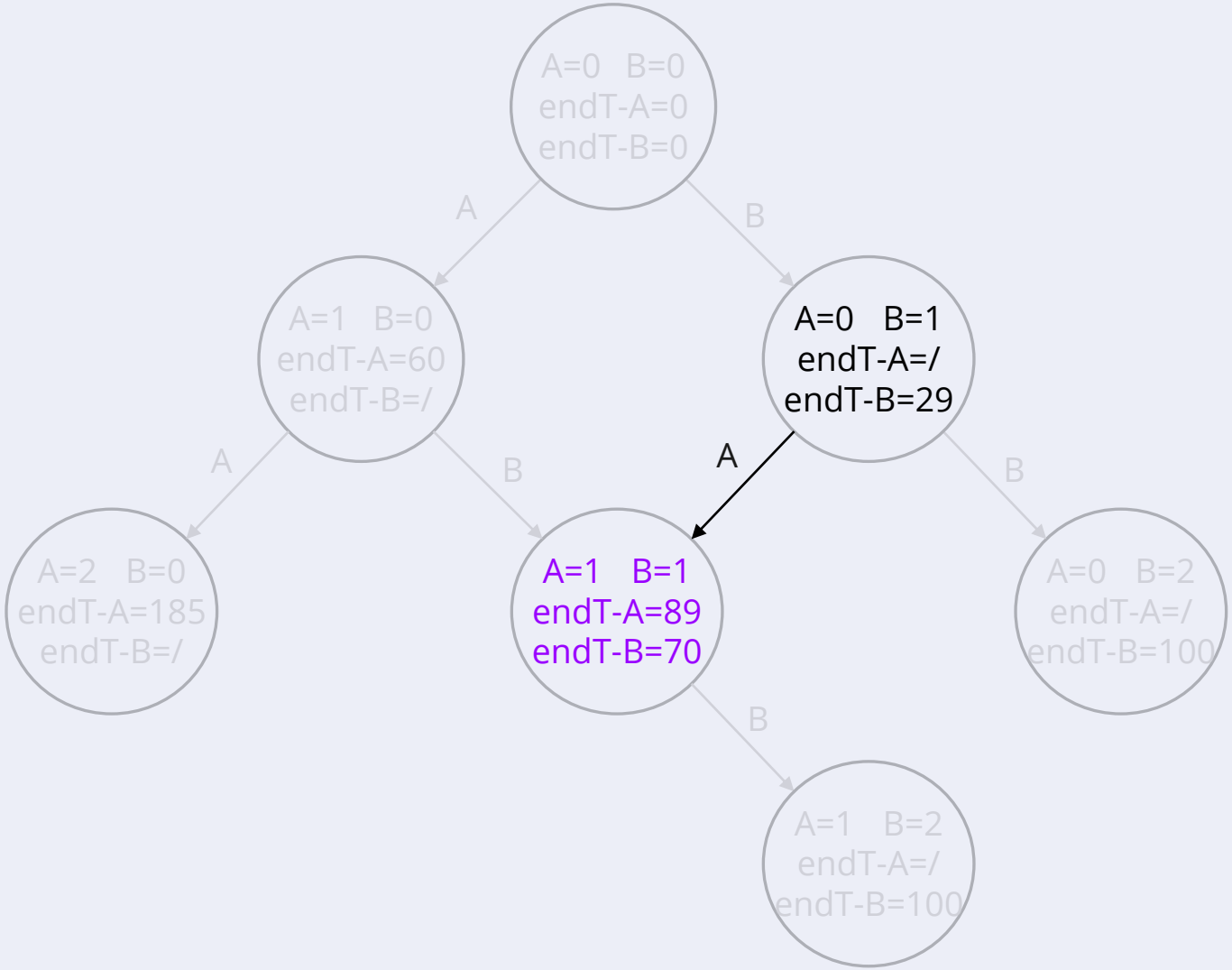
$$\text{startT B} = \max(70 + 10, 80) = 80$$
$$\text{endT B} = \max(80 + 20, 70 + 10) = 100$$

swap.startT = max(endT previous, arrival time)

swap.endT = startT + travel time

noswap.startT = max(startT previous + 10, arrival time)

noswap.endT = max(startT + travel time, endT previous + 10)



4		
A	0	60
B	19	10
B	80	20
A	85	100

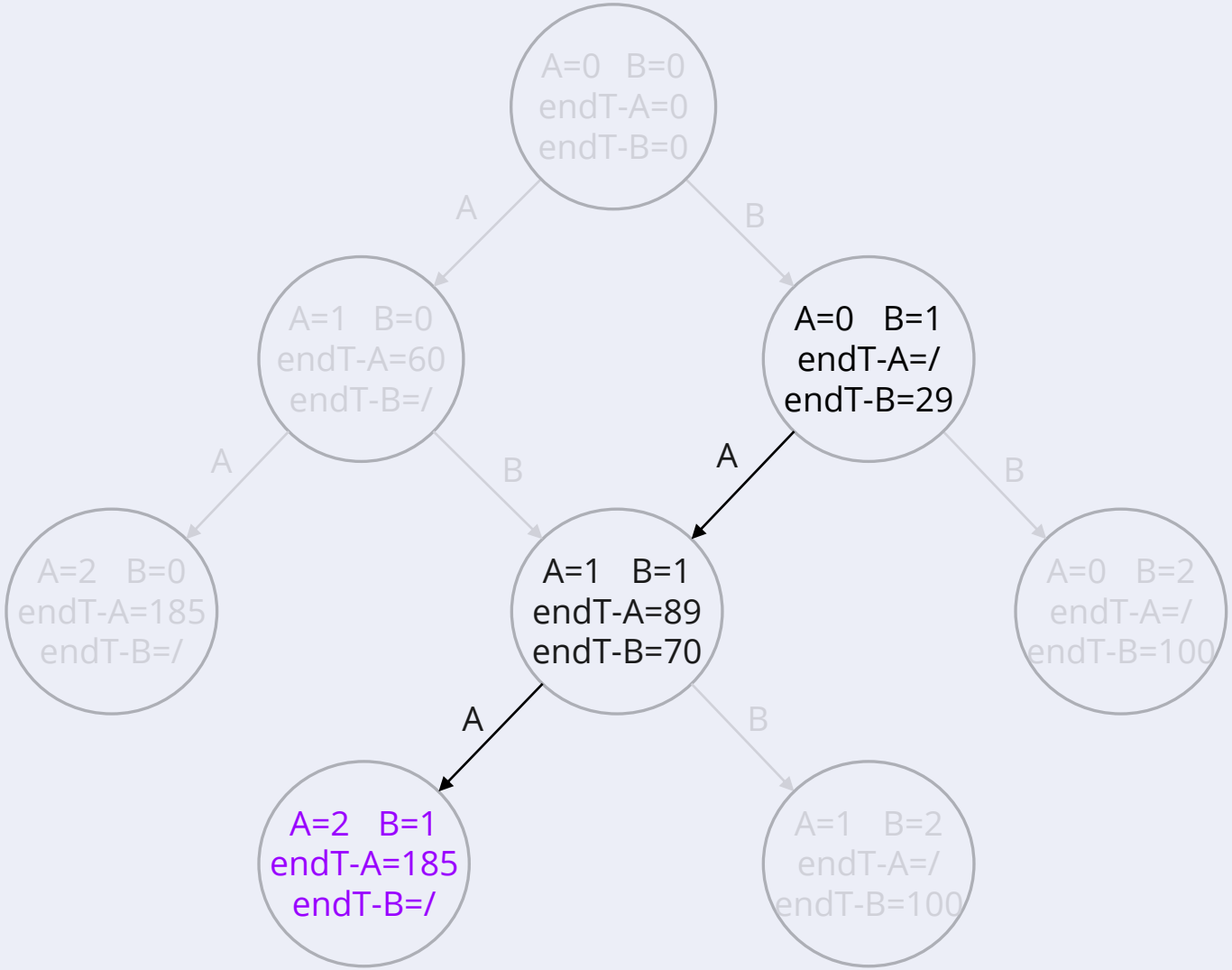
$$\text{startT A} = \max(29, 0) = 29$$
$$\text{endT A} = 29 + 60 = 89$$

swap.startT = max(endT previous, arrival time)

swap.endT = startT + travel time

noswap.startT = max(startT previous + 10, arrival time)

noswap.endT = max(startT + travel time, endT previous + 10)



4		
A	0	60
B	19	10
B	80	20
A	85	100

$$\text{startT A} = \max(29, 0) = 29$$
$$\text{endT A} = 29 + 60 = 89$$

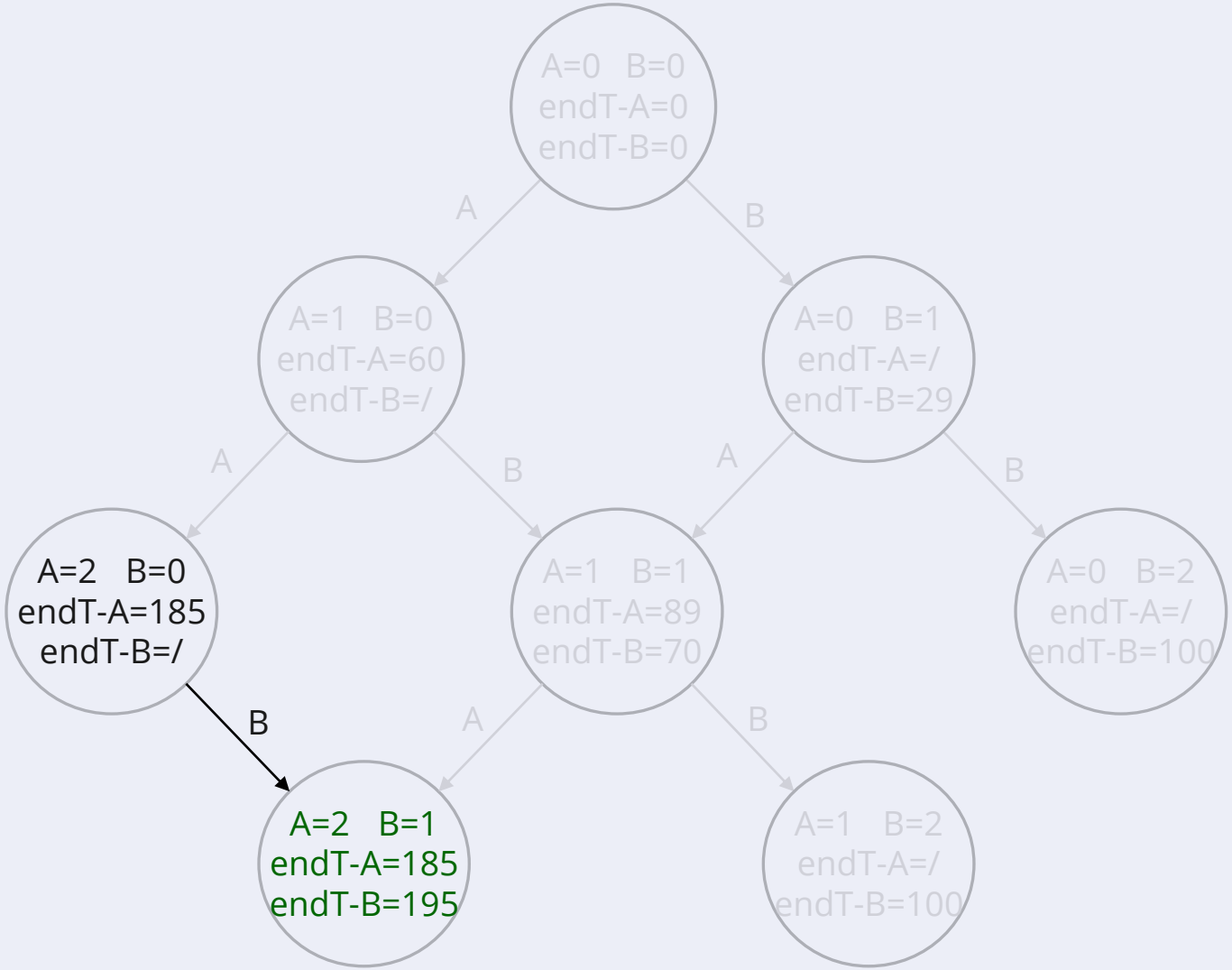
$$\text{startT A} = \max(29 + 10, 85) = 85$$
$$\text{endT A} = \max(85 + 100, 89 + 10) = 185$$

swap.startT = max(endT previous, arrival time)

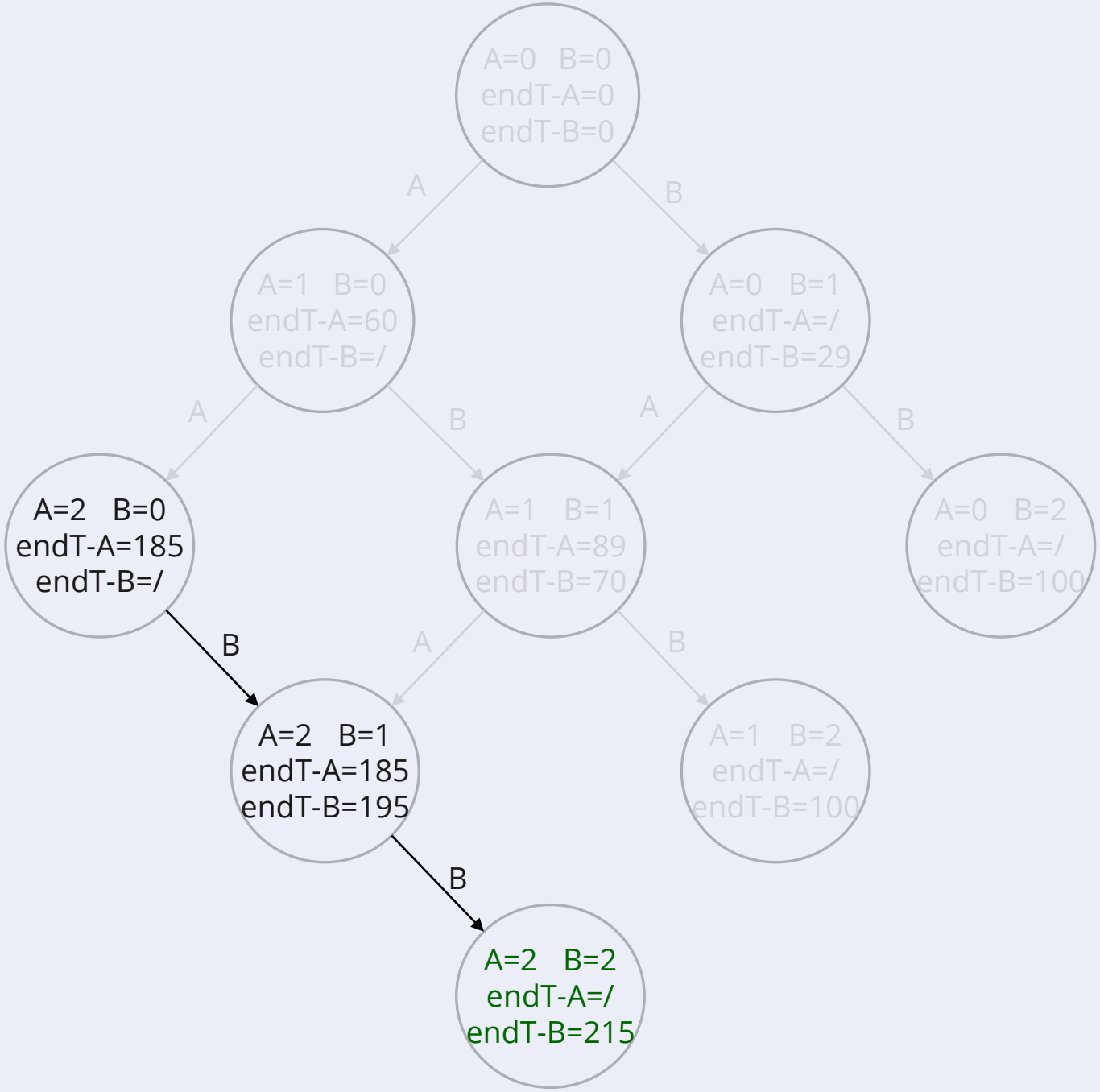
swap.endT = startT + travel time

noswap.startT = max(startT previous + 10, arrival time)

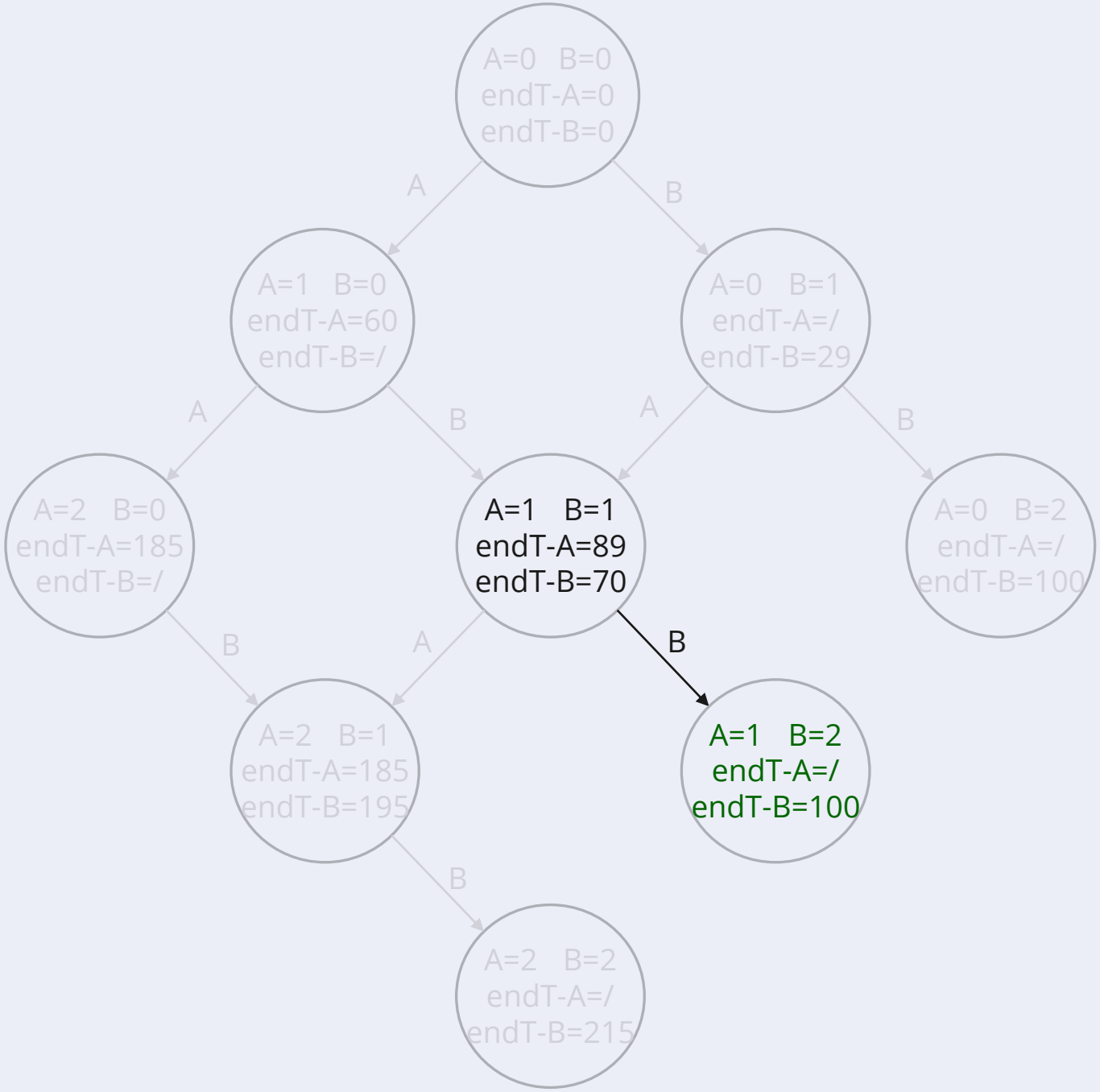
noswap.endT = max(startT + travel time, endT previous + 10)



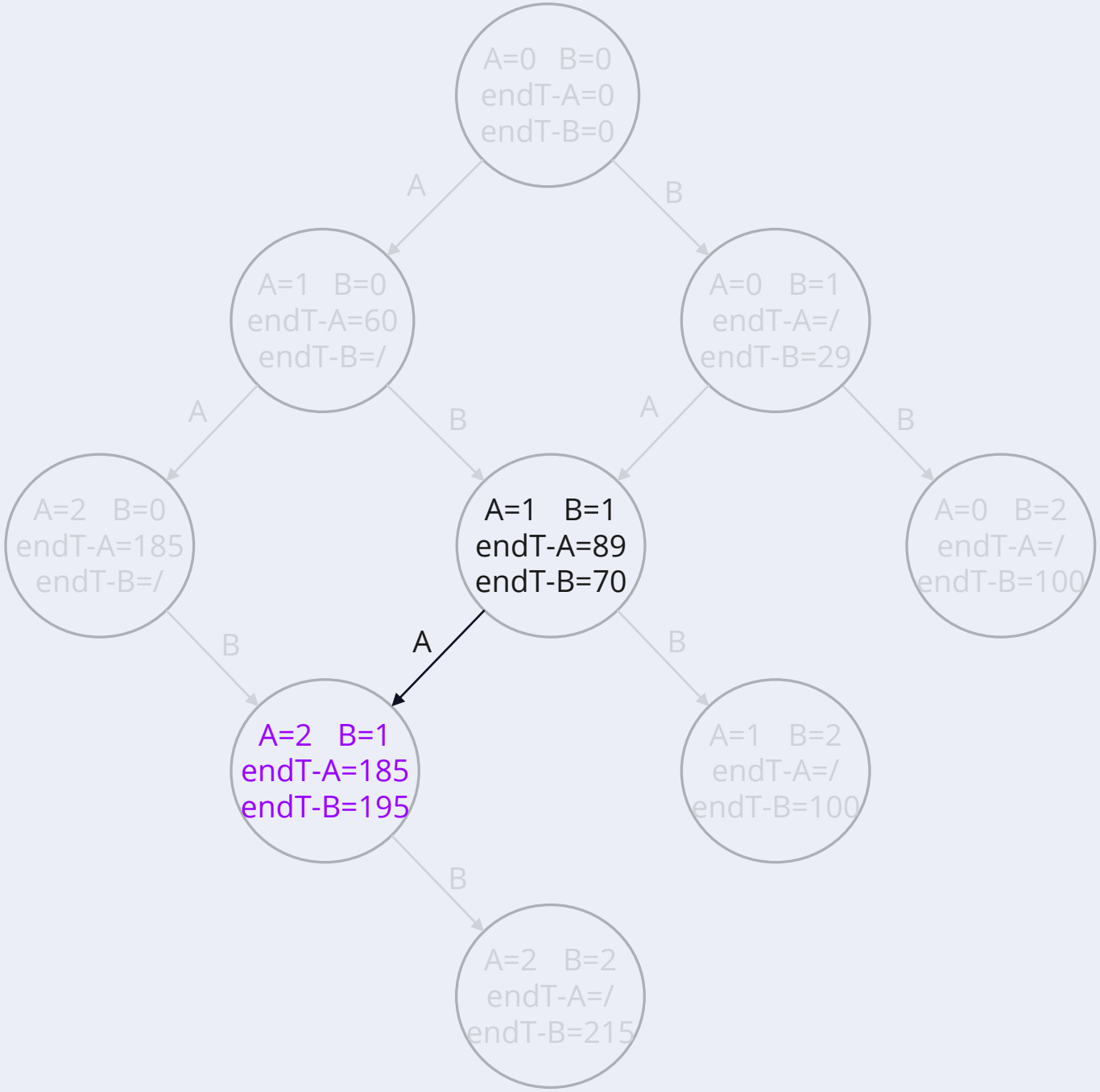
4		
A	0	60
B	19	10
B	80	20
A	85	100



4		
A	0	60
B	19	10
B	80	20
A	85	100

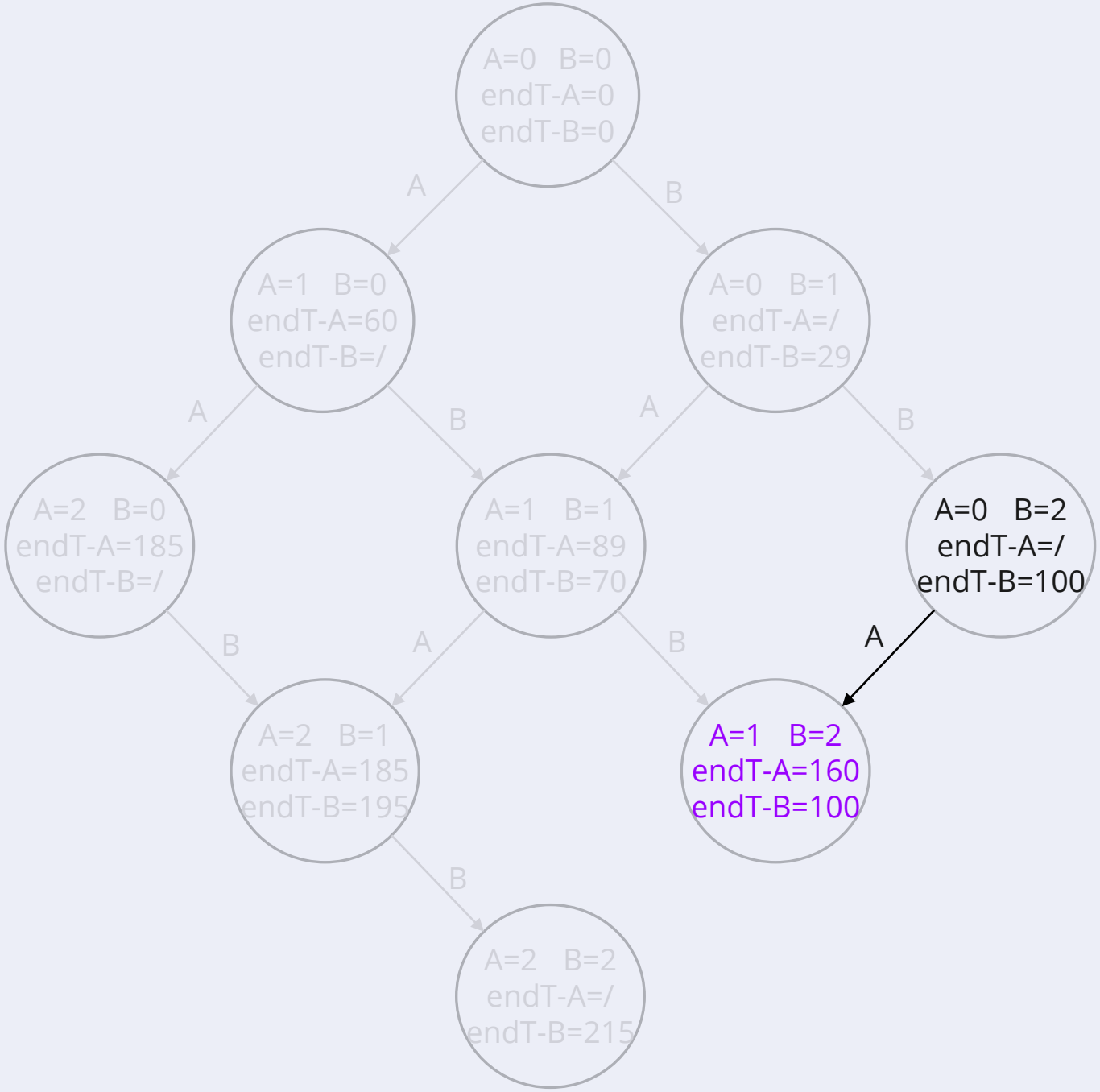


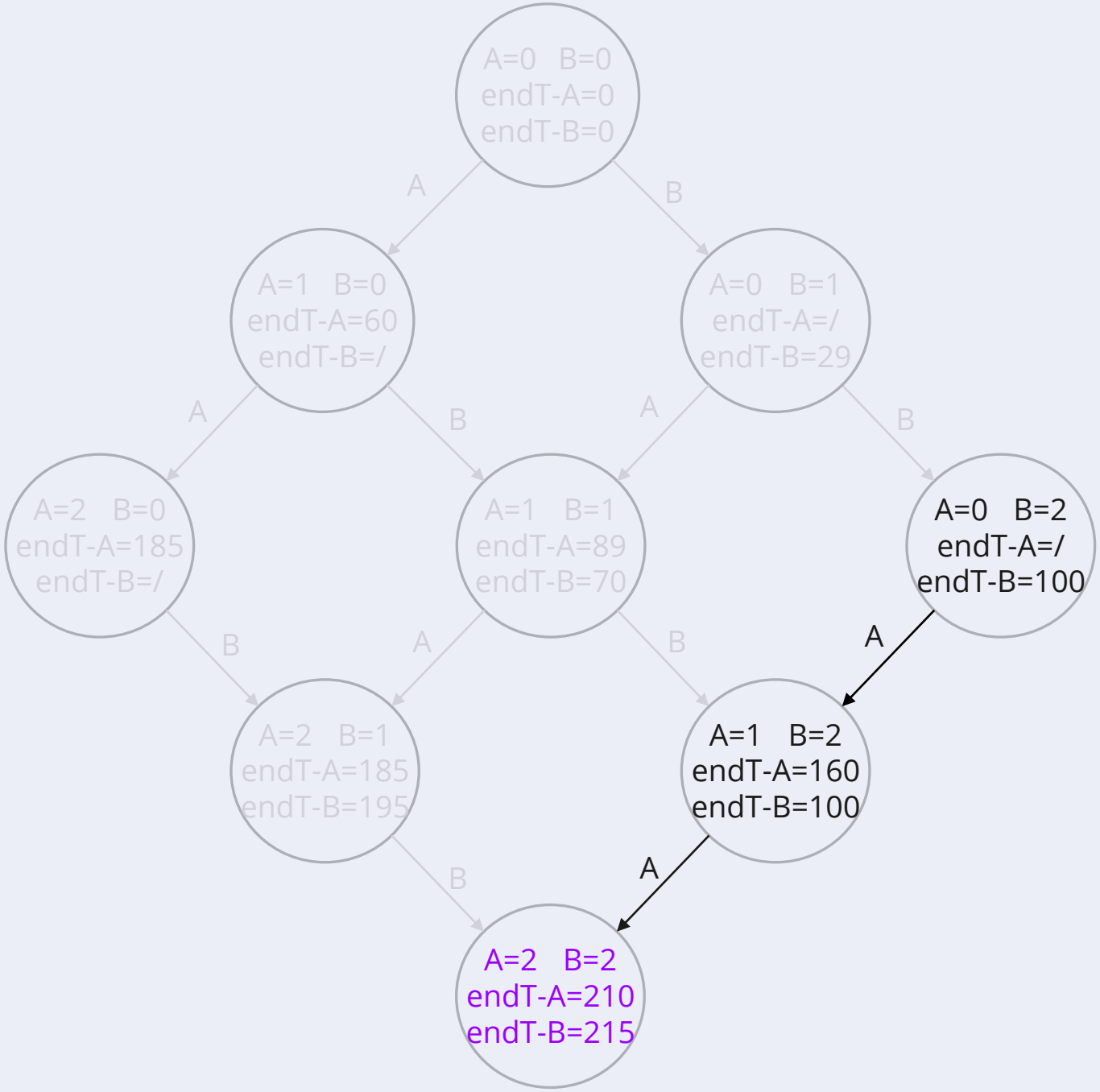
4		
A	0	60
B	19	10
B	80	20
A	85	100



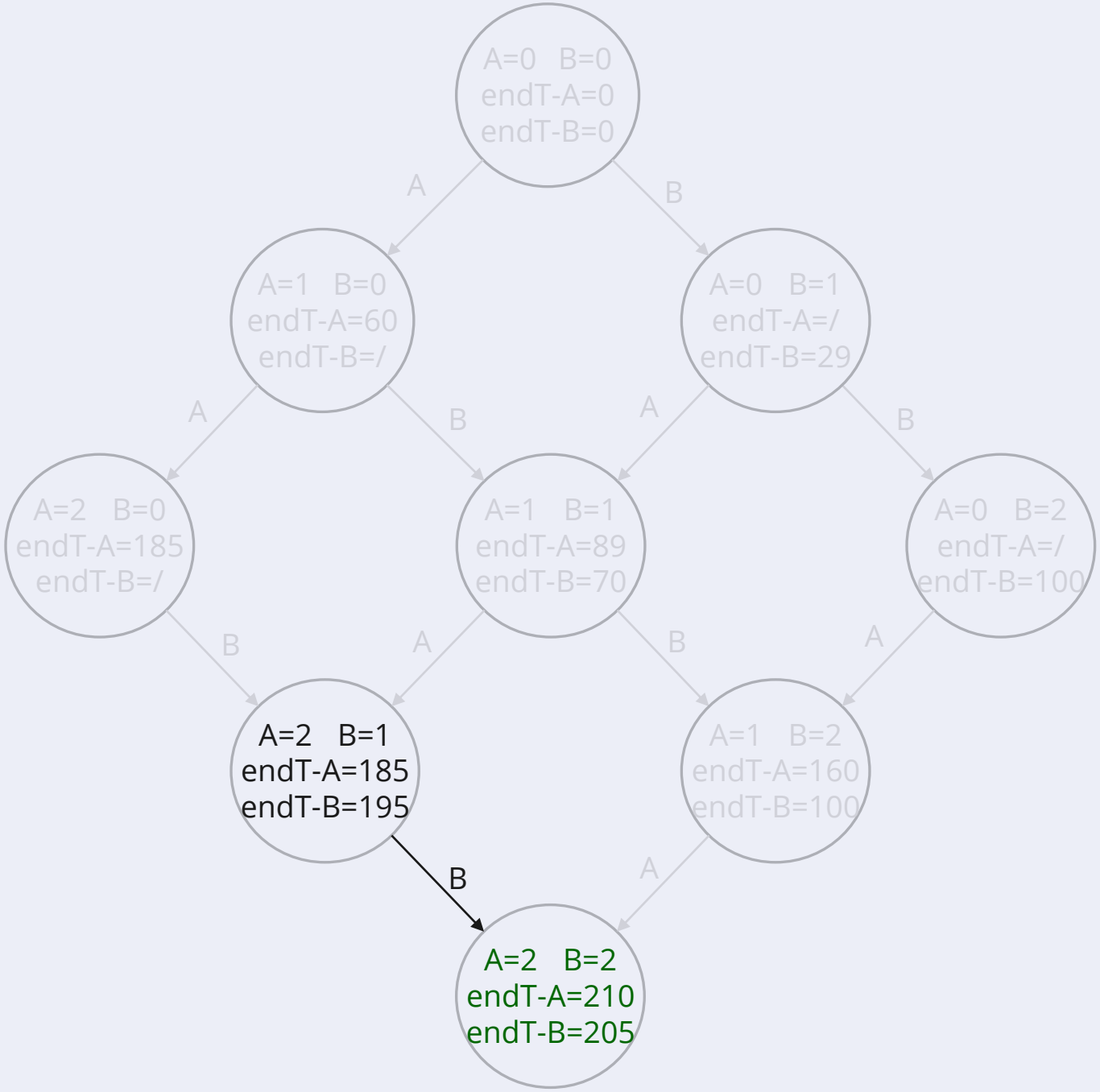
4		
A	0	60
B	19	10
B	80	20
A	85	100

4		
A	0	60
B	19	10
B	80	20
A	85	100

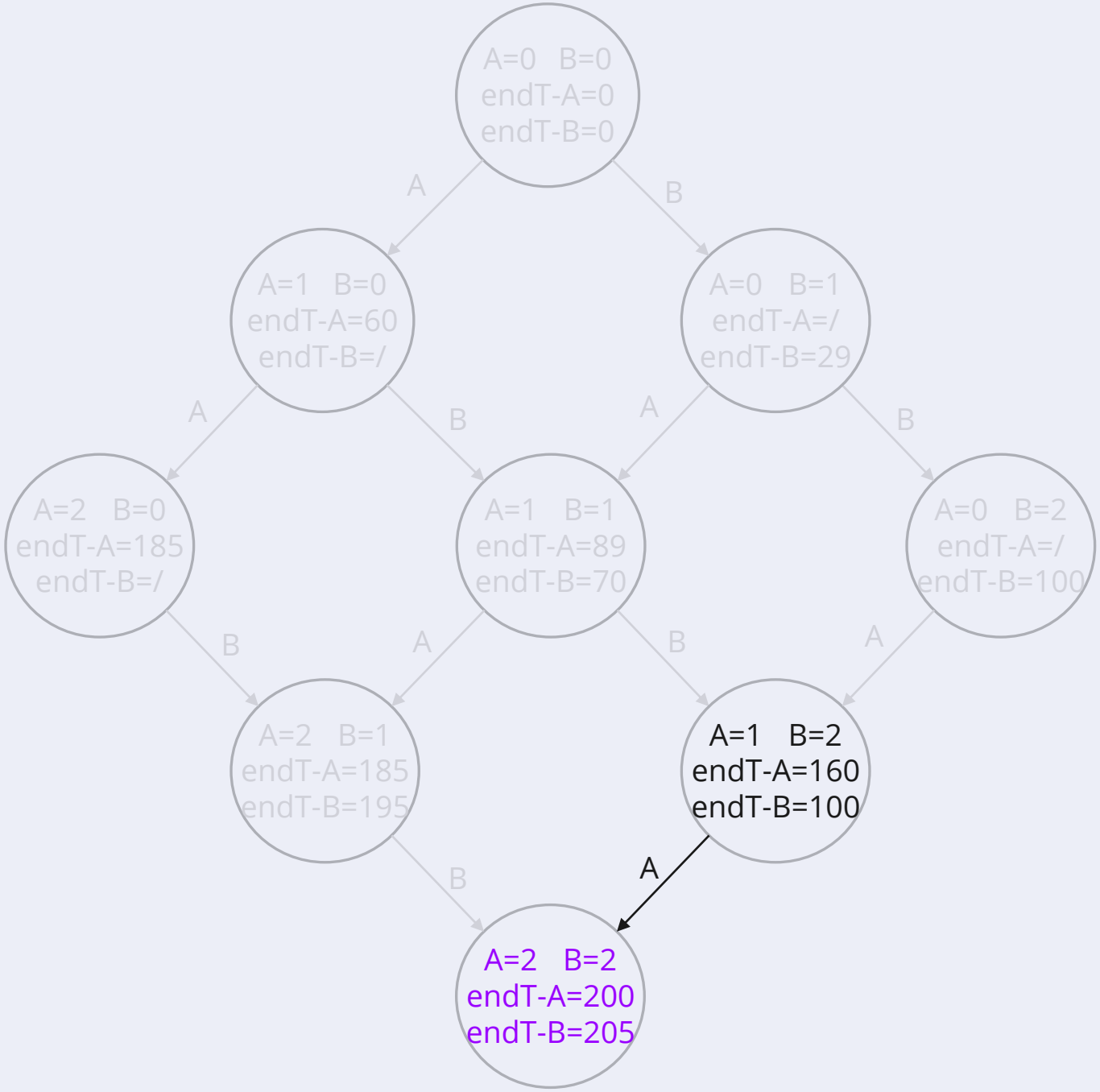




4		
A	0	60
B	19	10
B	80	20
A	85	100

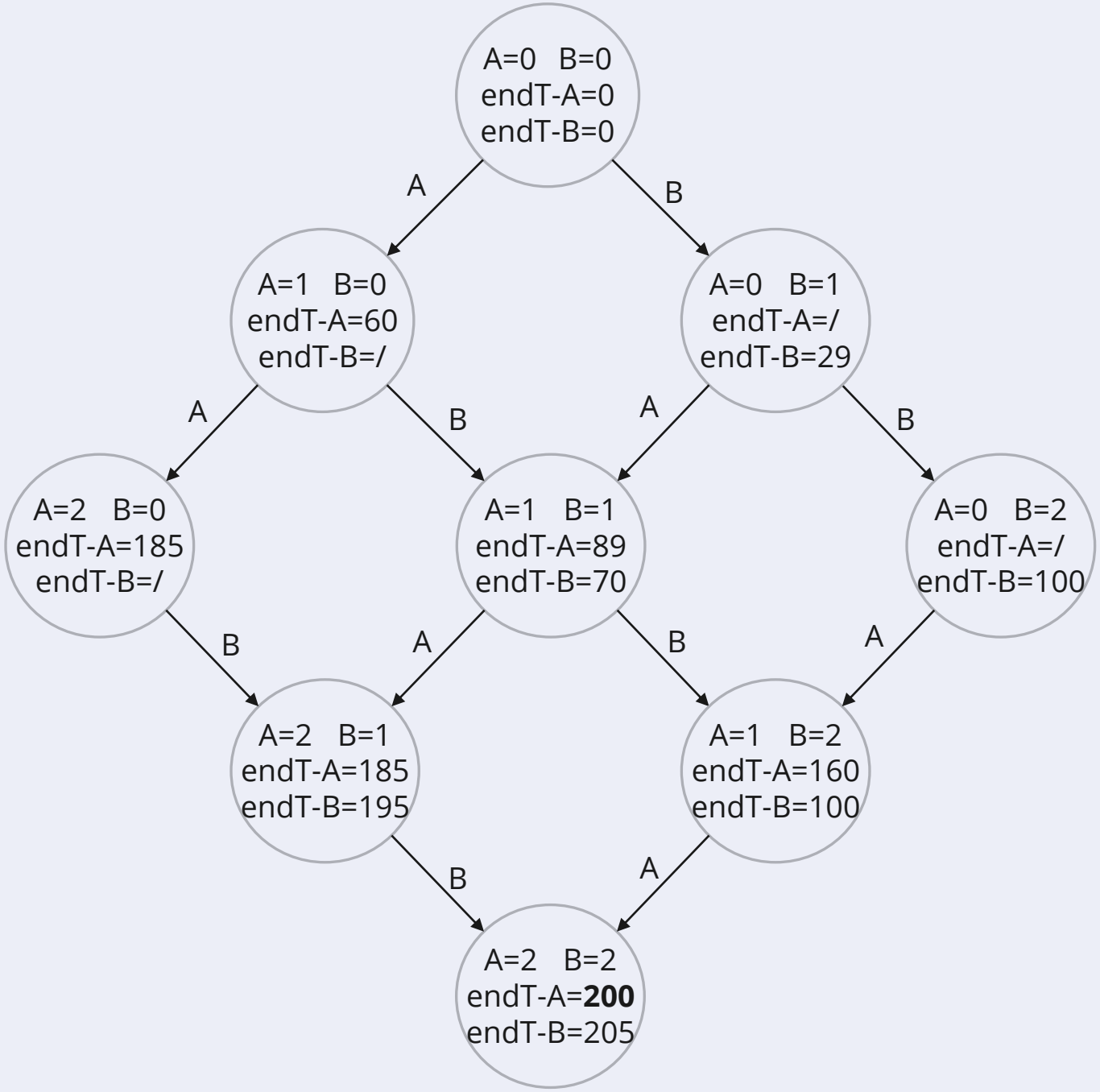


4		
A	0	60
B	19	10
B	80	20
A	85	100



4		
A	0	60
B	19	10
B	80	20
A	85	100

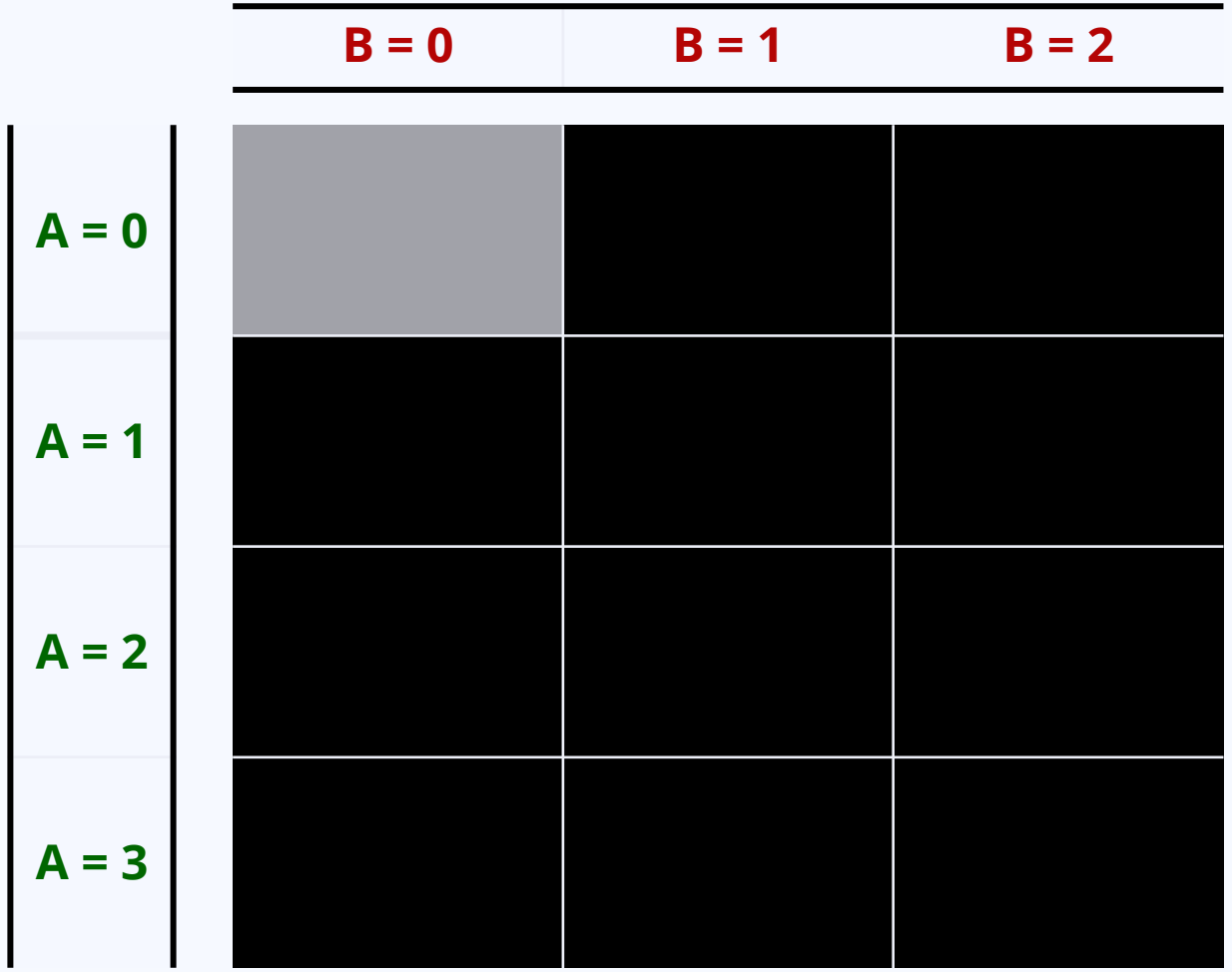
4		
A	0	60
B	19	10
B	80	20
A	85	100



Proof

Possible sequences:

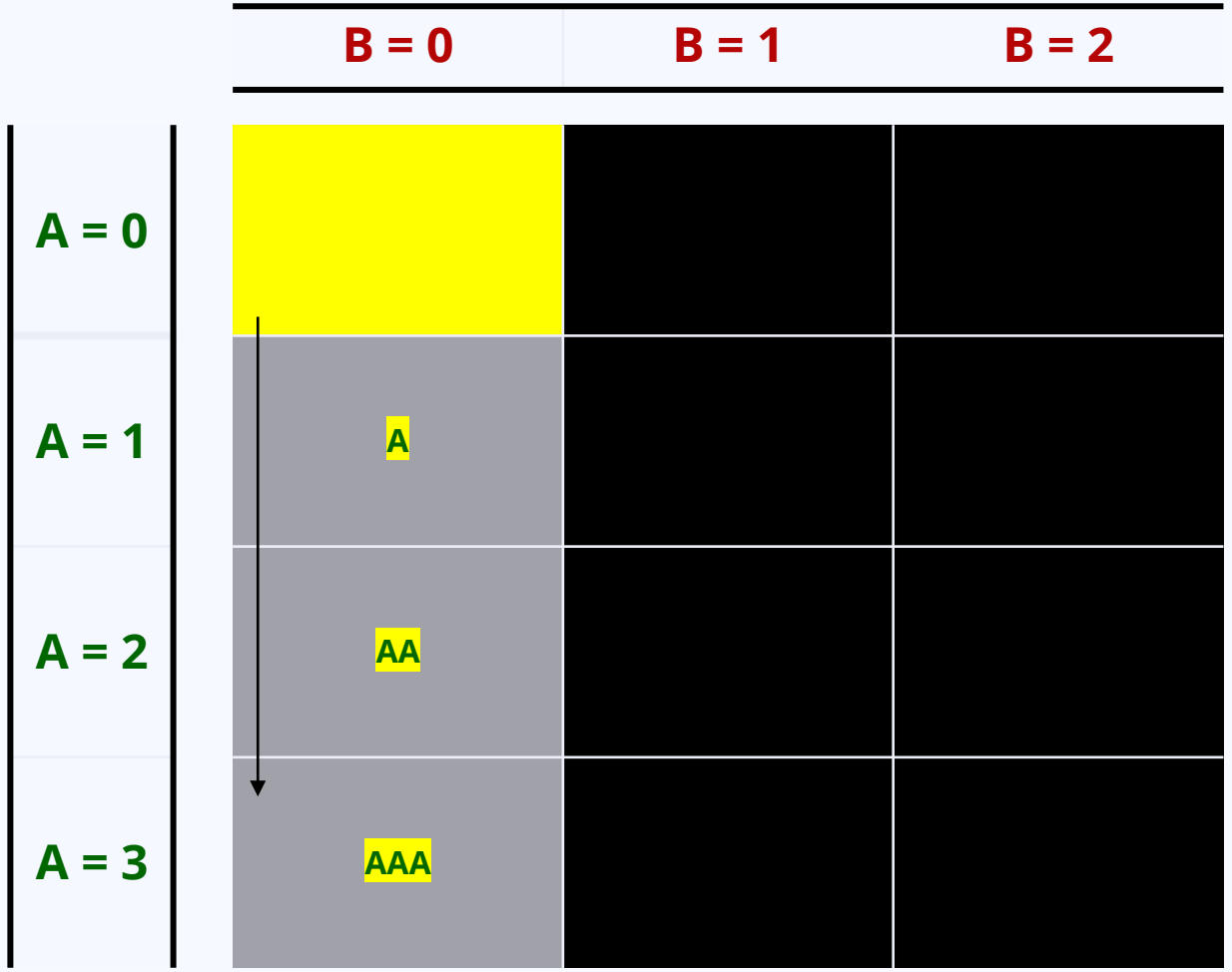
AAABB
AABAB
AABBA
ABAAB
ABABA
ABBAA
BAAAB
BAABA
BABAA
BBAAA



Proof

Possible sequences:

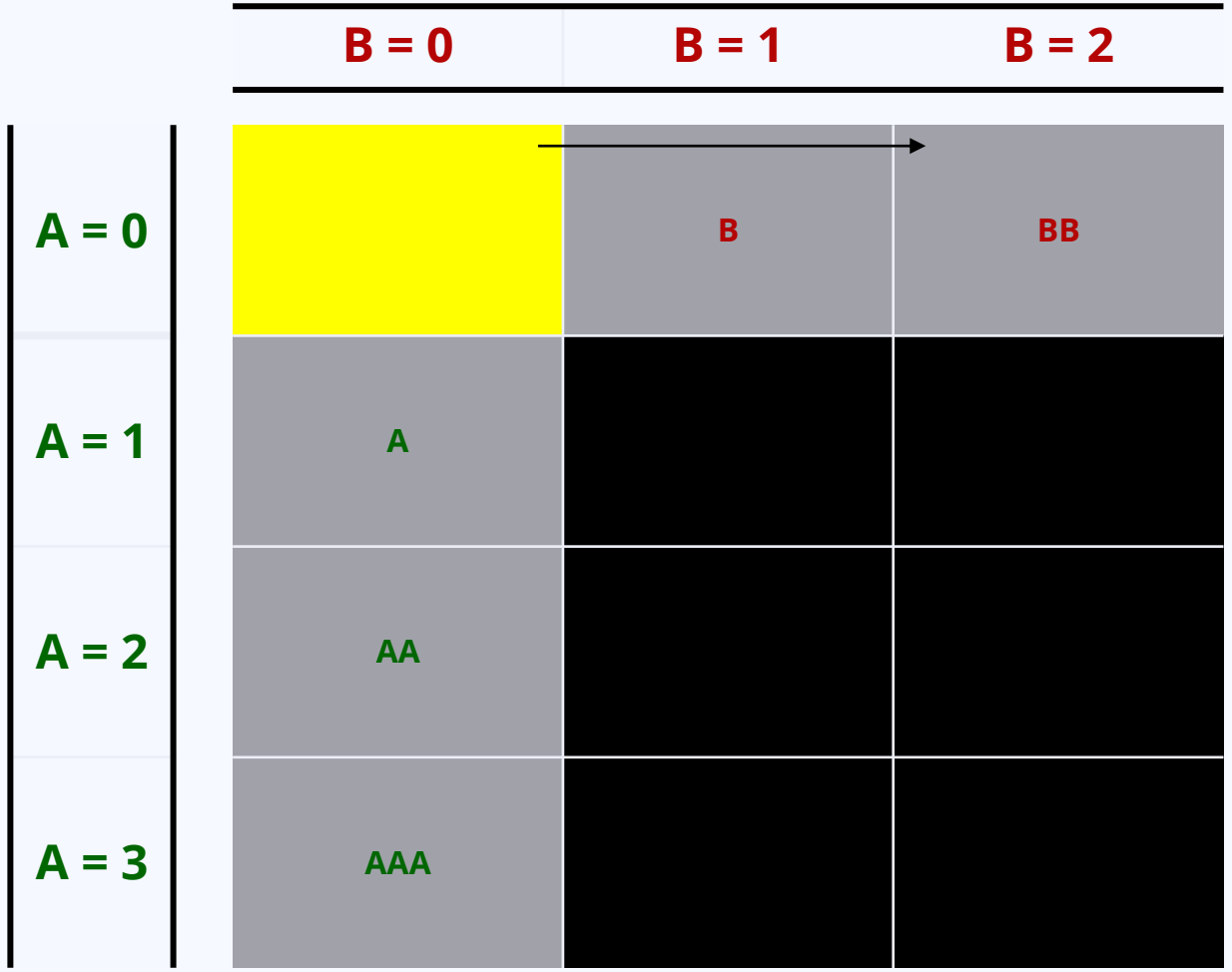
AAABB
AABAB
AABBA
ABAAB
ABABA
ABBAA
BAAAB
BAABA
BABAA
BBAAA



Proof

Possible sequences:

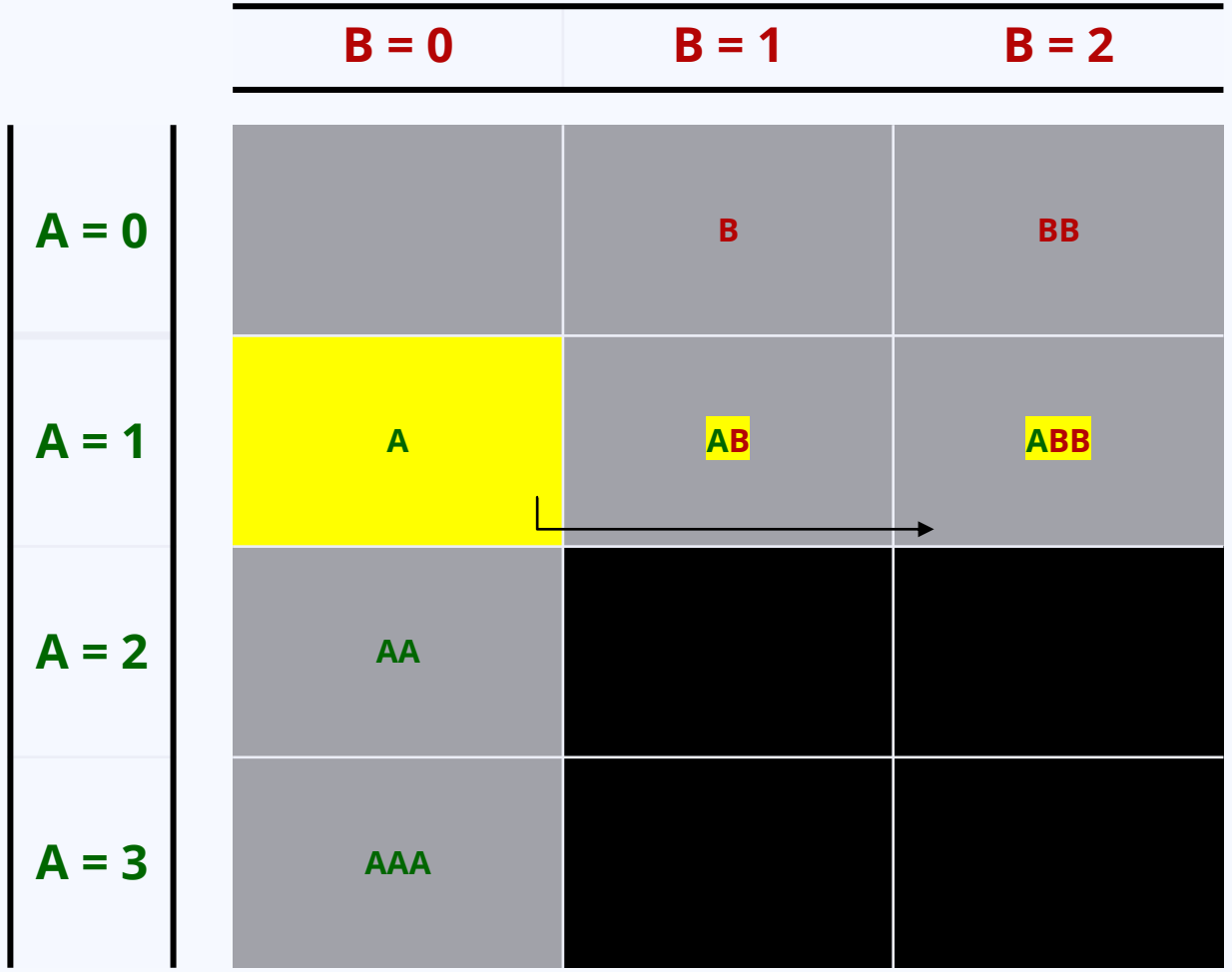
AAABB
AABAB
AABBA
ABAAB
ABABA
ABBAA
BAAAB
BAABA
BABAA
BBAAA



Proof

Possible sequences:

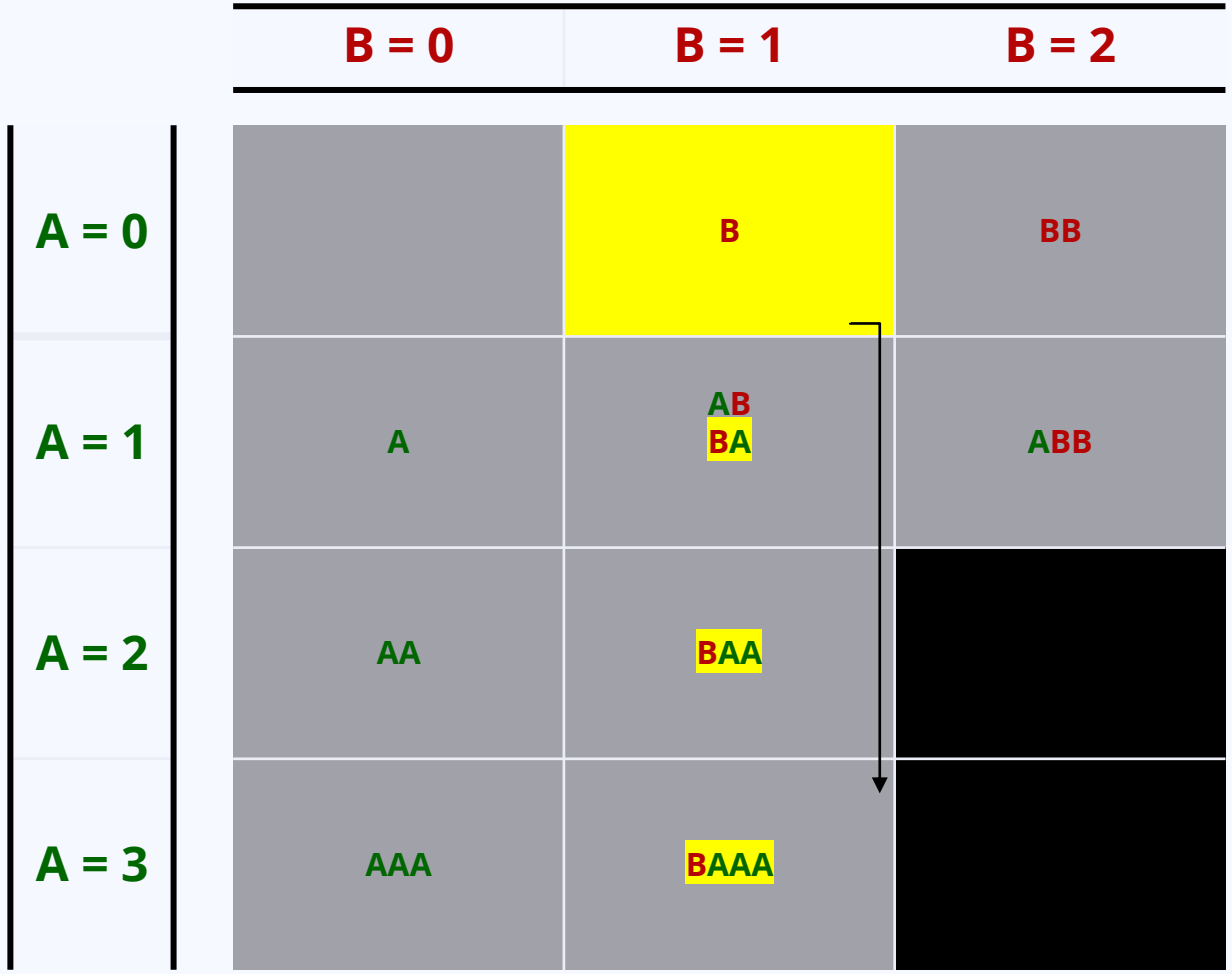
AAABB
AABAB
AABBA
ABAAB
ABABA
ABBAA
BAAAB
BAABA
BABAA
BBAAA



Proof

Possible sequences:

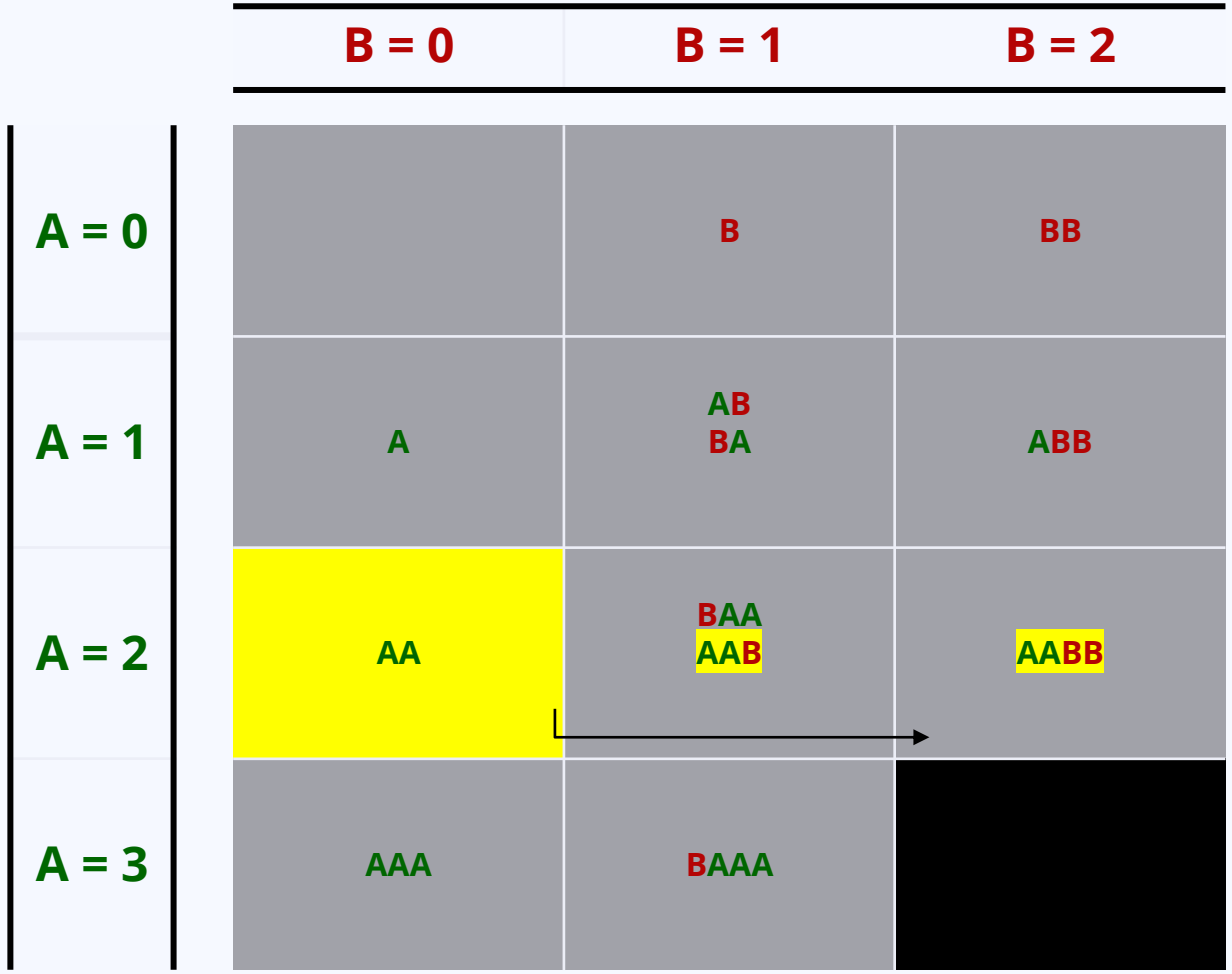
AAABB
AABAB
AABBA
ABAAB
ABABA
ABBAA
BAAAB
BAABA
BABAA
BBAAA



Proof

Possible sequences:

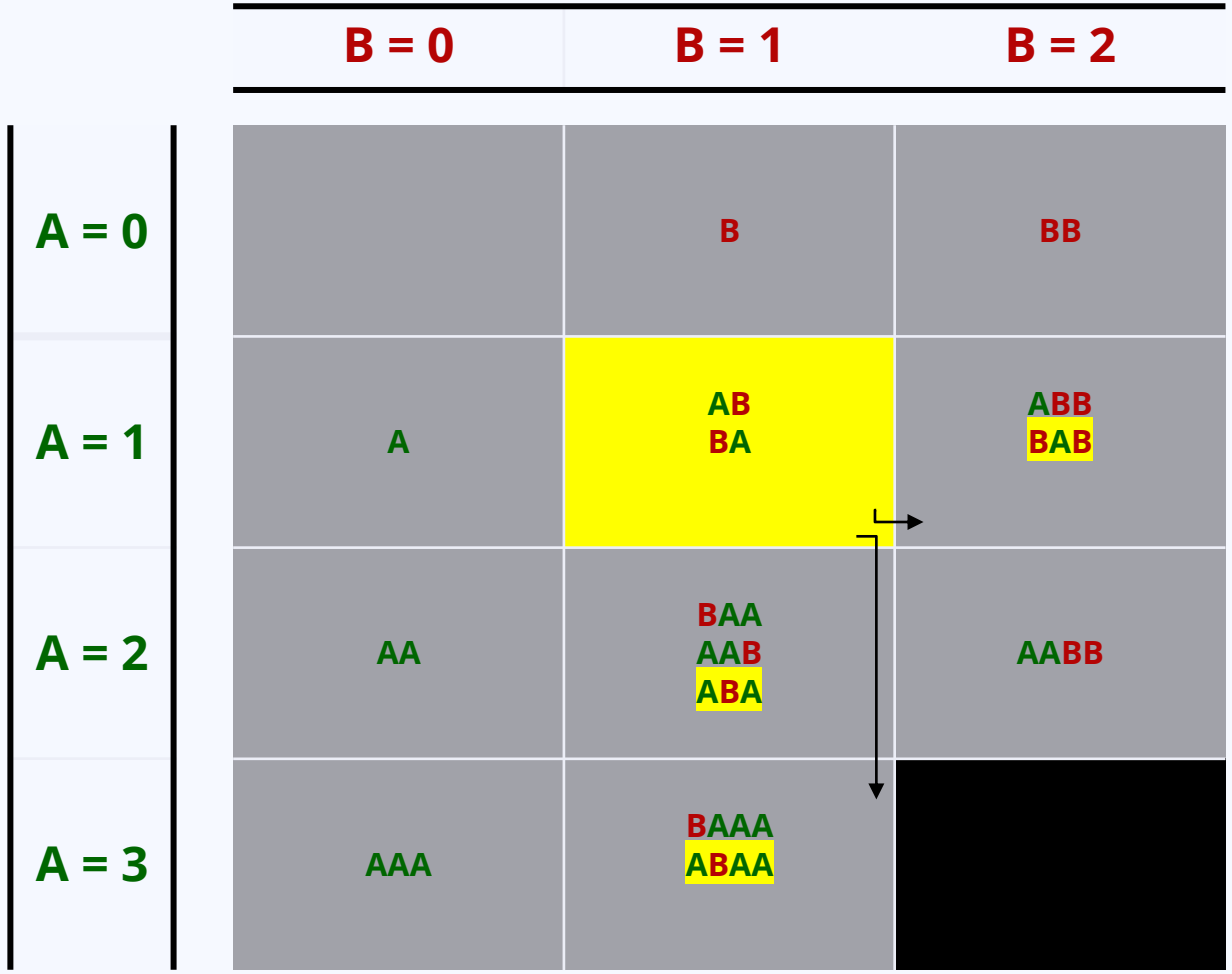
AAABB
AABAB
AABBA
ABAAB
ABABA
ABBAA
BAAAB
BAABA
BABAA
BBAAA



Proof

Possible sequences:

AAABB
AABAB
AABBA
ABAAB
ABABA
ABBAA
BAAAB
BAABA
BABAA
BBAAA



Proof

Possible sequences:

AAABB
AABAB
AABBA
ABAAB
ABABA
ABBAA
BAAAB
BAABA
BABAA
BBAAA

	B = 0	B = 1	B = 2
A = 0		B	BB
A = 1	A	AB BA	ABB BAB BBA
A = 2	AA	BAA AAB ABA	AABB BBAA
A = 3	AAA	BAAA ABAA	BBAAA

Proof

Possible sequences:

AAABB
AABAB
AABBA
ABAAB
ABABA
ABBAA
BAAAB
BAABA
BABAA
BBAAA

	B = 0	B = 1	B = 2
A = 0		B	BB
A = 1	A	AB BA	ABB BAB BBA
A = 2	AA	BAA AAB ABA	AABB BBAA
A = 3	AAA	BAAA ABAA AAAB	BBAAA AAABB

Proof

Possible sequences:

AAABB
AABAB
AABBA
ABAAB
ABABA
ABBAA
BAAAB
BAABA
BABAA
BBAAA

	B = 0	B = 1	B = 2
A = 0		B	BB
A = 1	A	AB BA	ABB BAB BBA
A = 2	AA	BAA AAB ABA	AABB BBAA BAAB ABAB
A = 3	AAA	BAAA ABAA AAAB AABA	BBAAA AAABB

Proof

Possible sequences:

AAABB
AABAB
AABBA
ABAAB
ABABA
ABBAA
BAAAB
BAABA
BABAA
BBAAA

	B = 0	B = 1	B = 2
A = 0		B	BB
A = 1	A	AB BA	ABB BAB BBA
A = 2	AA	BAA AAB ABA	AABB ABBA BBAA BABA BAAB ABAB
A = 3	AAA	BAAA ABAA AAAB AABA	BBAAA AAABB ABBAA BABAA

Proof

Possible sequences:

AAABB
AABAB
AABBA
ABAAB
ABABA
ABBAA
BAAAB
BAABA
BABAA
BBAAA

	B = 0	B = 1	B = 2
A = 0		B	BB
A = 1	A	AB BA	ABB BAB BBA
A = 2	AA	BAA AAB ABA	AABB ABBA BBAA BABA BAAB ABAB
A = 3	AAA	BAAA ABAA AAAB AABA	BBAAA ABAAB AAABB AABAB ABBAA BAAAB BABAA

Proof

Possible sequences:

AAABB
AABAB
AABBA
ABAAB
ABABA
ABBAA
BAAAB
BAABA
BABAA
BBAAA

	B = 0	B = 1	B = 2
A = 0		B	BB
A = 1	A	AB BA	ABB BAB BBA
A = 2	AA	BAA AAB ABA	AABB ABBA BBAA BABA BAAB ABAB
A = 3	AAA	BAAA ABAA AAAB AABA	BBAAA ABAAB AAABB AABAB ABBAA AABBA BABAA BAABA BAAAB ABABA

Proof

Possible sequences:

AAABB
AABAB
AABBA
ABAAB
ABABA
ABBAA
BAAAB
BAABA
BABAA
BBAAA

	B = 0	B = 1	B = 2
A = 0		B	BB
A = 1	A	AB BA	ABB BAB BBA
A = 2	AA	BAA AAB ABA	AABB ABBA BBAA BABA BAAB ABAB
A = 3	AAA	BAAA ABAA AAAB AABA	BBAAA ABAAB AAABB AABAB ABBAA AABBA BABAA BAABA BAAAB ABABA

Runtime

```
solveH()
{
  for  cars length do:
   {
     for B-cars length do:
      {
        driveAllRemainingAs()
        driveAllRemeiningBs()
      }
   }
}
```

n = cars length = A-cars length + B-cars length

if A-Cars has size 0 : n = B-cars length => max n iterations

can at most drive all A-cars => A-Cars length iterations

can at most drive all B-cars => B-Cars length iterations

$O(n)$

$O(n)$

$O(n)$

$O(n^3)$

VIEL SPAß! ☺