Lesson 21. Solving the Points-After-Touchdown Problem

- Let's solve the stochastic dynamic program we formulated for the points-after-touchdown problem in Lesson 20.
- Before doing anything else, let's import `StochasticDP` from the `stochasticdp` package:

```
In [2]: from stochasticdp import StochasticDP
```

Setting up the data

- In Lesson 20, we worked with the following data:

$$
\begin{aligned}
T &= \text{total number of possessions} \\
p_n &= \Pr\{\text{1-pt. conv. successful for Team } n \,|\, \text{1-pt. conv. attempted by Team } n\} && \text{for } n = \text{A, B} \\
q_n &= \Pr\{\text{2-pt. conv. successful for Team } n \,|\, \text{2-pt. conv. attempted by Team } n\} && \text{for } n = \text{A, B} \\
b_1 &= \Pr\{\text{1-pt. conv. attempted by Team B}\} \\
b_2 &= \Pr\{\text{2-pt. conv. attempted by Team B}\} \\
t_n &= \Pr\{\text{TD by Team } n \text{ in 1 possession}\} && \text{for } n = \text{A, B} \\
g_n &= \Pr\{\text{FG by Team } n \text{ in 1 possession}\} && \text{for } n = \text{A, B} \\
z_n &= \Pr\{\text{no score by Team } n \text{ in 1 possession}\} && \text{for } n = \text{A, B} \\
r &= \Pr\{\text{Team A wins in overtime}\}
\end{aligned}
$$

- Let's begin by defining numerical values for this data.
- We can find most of these values from Pro Football Reference.
- For now, let's assume that Team A and Team B are both average 2014 NFL teams.
  - Recall that in 2014, 1-pt. conversions started at the 2-yard line.
- Also, let's assume that Team A wins in overtime with probability 0.5.

```
In [3]: # Total number of possessions
        # Drive Averages: 2 * (#Dr) / (G * (# of teams))
        T = 23

        # 1-pt. conversion success probabilities
        # Kicking and Punting: XP%
        pA = 0.993
        pB = 0.993

        # 2-pt. conversion success probabilities
        # Scoring Offense: 2PM / 2PA
        qA = 0.483
        qB = 0.483

        # 1-pt. vs 2-pt attempts
        # 1-pt.: Scoring Offense: XPA / (XPA + 2PA)
        b1 = 0.954
        b2 = 1 - b1
```

```
# Possession outcome probabilities - Team A
# TD: (Scoring Offense: ATD) / (Drive Averages: #Dr)
# FG: (Scoring Offense: FGM) / (Drive Averages: #Dr)
tA = 0.218
gA = 0.172
zA = 1 - tA - gA

# Possession outcome probabilities - Team B
tB = 0.218
gB = 0.172
zB = 1 - tB - gB

# Probability that Team A wins in OT
r = 0.5
```

Initializing the stochastic dynamic program

- Stages:

$$
\begin{aligned}
t = 0, 1, \dots, T-1 \quad &\leftrightarrow \quad \text{end of possession } t \\
t = T \quad &\leftrightarrow \quad \text{end of game}
\end{aligned}
$$

```
In [4]: # Number of stages
        number_of_stages = T + 1
```

- States:

$$
\begin{aligned}
(n, k, d) \quad \leftrightarrow \quad & \text{Team } n\text{'s possession just ended} && \text{for } n \in \{\text{A}, \text{B}\} \\
& \text{Team } n \text{ just scored } k \text{ points} && \text{for } k \in \{0, 3, 6\} \\
& \text{Team A is ahead by } d \text{ points} && \text{for } d \in \{\dots, -1, 0, 1, \dots, \}
\end{aligned}
$$

- In Lesson 20, we did not assume a lower or upper bound on d, the values that represent Team A's lead.
- Since we need to have a finite number of states, let's assume $-20 \leq d \leq 20$.

- Some Python reminders:
  - We can construct a list by
    - first creating an empty list,
    - and then adding items to it using `.append()`.
  - For example:

```
my_list = []
for i in range(10):
    my_list.append(i)
```

  - `range(a)` iterates over the integers `0, 1, ..., a - 1`, while `range(a, b)` iterates over the integers `a, a + 1, ..., b - 1`.

```
In [5]: # List of states
        states = []
        for n in ['A', 'B']:
            for k in [0, 3, 6]:
                for d in range(-20, 21):
                    states.append((n, k, d))
```

- *Quick check.* Let's inspect what we just created:

```
In [6]: # Print the list of states
        print(states)
```

```
[('A', 0, -20), ('A', 0, -19), ('A', 0, -18), ('A', 0, -17), ('A', 0, -16), ('A', 0, -15),
('A', 0, -14), ('A', 0, -13), ('A', 0, -12), ('A', 0, -11), ('A', 0, -10), ('A', 0, -9),
('A', 0, -8), ('A', 0, -7), ('A', 0, -6), ('A', 0, -5), ('A', 0, -4), ('A', 0, -3), ('A',
0, -2), ('A', 0, -1), ('A', 0, 0), ('A', 0, 1), ('A', 0, 2), ('A', 0, 3), ('A', 0, 4),
('A', 0, 5), ('A', 0, 6), ('A', 0, 7), ('A', 0, 8), ('A', 0, 9), ('A', 0, 10), ('A', 0,
11), ('A', 0, 12), ('A', 0, 13), ('A', 0, 14), ('A', 0, 15), ('A', 0, 16), ('A', 0, 17),
('A', 0, 18), ('A', 0, 19), ('A', 0, 20), ('A', 3, -20), ('A', 3, -19), ('A', 3, -18),
('A', 3, -17), ('A', 3, -16), ('A', 3, -15), ('A', 3, -14), ('A', 3, -13), ('A', 3, -12),
('A', 3, -11), ('A', 3, -10), ('A', 3, -9), ('A', 3, -8), ('A', 3, -7), ('A', 3, -6),
('A', 3, -5), ('A', 3, -4), ('A', 3, -3), ('A', 3, -2), ('A', 3, -1), ('A', 3, 0), ('A',
3, 1), ('A', 3, 2), ('A', 3, 3), ('A', 3, 4), ('A', 3, 5), ('A', 3, 6), ('A', 3, 7), ('A',
3, 8), ('A', 3, 9), ('A', 3, 10), ('A', 3, 11), ('A', 3, 12), ('A', 3, 13), ('A', 3, 14),
('A', 3, 15), ('A', 3, 16), ('A', 3, 17), ('A', 3, 18), ('A', 3, 19), ('A', 3, 20), ('A',
6, -20), ('A', 6, -19), ('A', 6, -18), ('A', 6, -17), ('A', 6, -16), ('A', 6, -15), ('A',
6, -14), ('A', 6, -13), ('A', 6, -12), ('A', 6, -11), ('A', 6, -10), ('A', 6, -9), ('A',
6, -8), ('A', 6, -7), ('A', 6, -6), ('A', 6, -5), ('A', 6, -4), ('A', 6, -3), ('A', 6,
-2), ('A', 6, -1), ('A', 6, 0), ('A', 6, 1), ('A', 6, 2), ('A', 6, 3), ('A', 6, 4), ('A',
6, 5), ('A', 6, 6), ('A', 6, 7), ('A', 6, 8), ('A', 6, 9), ('A', 6, 10), ('A', 6, 11),
('A', 6, 12), ('A', 6, 13), ('A', 6, 14), ('A', 6, 15), ('A', 6, 16), ('A', 6, 17), ('A',
6, 18), ('A', 6, 19), ('A', 6, 20), ('B', 0, -20), ('B', 0, -19), ('B', 0, -18), ('B', 0,
-17), ('B', 0, -16), ('B', 0, -15), ('B', 0, -14), ('B', 0, -13), ('B', 0, -12), ('B', 0,
-11), ('B', 0, -10), ('B', 0, -9), ('B', 0, -8), ('B', 0, -7), ('B', 0, -6), ('B', 0, -5),
('B', 0, -4), ('B', 0, -3), ('B', 0, -2), ('B', 0, -1), ('B', 0, 0), ('B', 0, 1), ('B', 0,
2), ('B', 0, 3), ('B', 0, 4), ('B', 0, 5), ('B', 0, 6), ('B', 0, 7), ('B', 0, 8), ('B', 0,
9), ('B', 0, 10), ('B', 0, 11), ('B', 0, 12), ('B', 0, 13), ('B', 0, 14), ('B', 0, 15),
('B', 0, 16), ('B', 0, 17), ('B', 0, 18), ('B', 0, 19), ('B', 0, 20), ('B', 3, -20), ('B',
3, -19), ('B', 3, -18), ('B', 3, -17), ('B', 3, -16), ('B', 3, -15), ('B', 3, -14), ('B',
3, -13), ('B', 3, -12), ('B', 3, -11), ('B', 3, -10), ('B', 3, -9), ('B', 3, -8), ('B', 3,
-7), ('B', 3, -6), ('B', 3, -5), ('B', 3, -4), ('B', 3, -3), ('B', 3, -2), ('B', 3, -1),
('B', 3, 0), ('B', 3, 1), ('B', 3, 2), ('B', 3, 3), ('B', 3, 4), ('B', 3, 5), ('B', 3, 6),
('B', 3, 7), ('B', 3, 8), ('B', 3, 9), ('B', 3, 10), ('B', 3, 11), ('B', 3, 12), ('B', 3,
13), ('B', 3, 14), ('B', 3, 15), ('B', 3, 16), ('B', 3, 17), ('B', 3, 18), ('B', 3, 19),
('B', 3, 20), ('B', 6, -20), ('B', 6, -19), ('B', 6, -18), ('B', 6, -17), ('B', 6, -16),
('B', 6, -15), ('B', 6, -14), ('B', 6, -13), ('B', 6, -12), ('B', 6, -11), ('B', 6, -10),
('B', 6, -9), ('B', 6, -8), ('B', 6, -7), ('B', 6, -6), ('B', 6, -5), ('B', 6, -4), ('B',
6, -3), ('B', 6, -2), ('B', 6, -1), ('B', 6, 0), ('B', 6, 1), ('B', 6, 2), ('B', 6, 3),
('B', 6, 4), ('B', 6, 5), ('B', 6, 6), ('B', 6, 7), ('B', 6, 8), ('B', 6, 9), ('B', 6,
10), ('B', 6, 11), ('B', 6, 12), ('B', 6, 13), ('B', 6, 14), ('B', 6, 15), ('B', 6, 16),
('B', 6, 17), ('B', 6, 18), ('B', 6, 19), ('B', 6, 20)]
```

- Allowable decisions x_t at stage t and state (n, k, d):

$$
\begin{aligned}
x_t &\in \{1, 2\} && \text{if } n = \text{A and } k = 6 \\
x_t &= \text{none} && \text{if } n = \text{A and } k \in \{0, 3\} \\
x_t &= \text{none} && \text{if } n = \text{B and } k \in \{0, 3, 6\}
\end{aligned}
$$

```
In [7]: # List of decisions
        decisions = [1, 2, 'none']
```

- Now we can initialize a `StochasticDP` object called `dp` as follows:

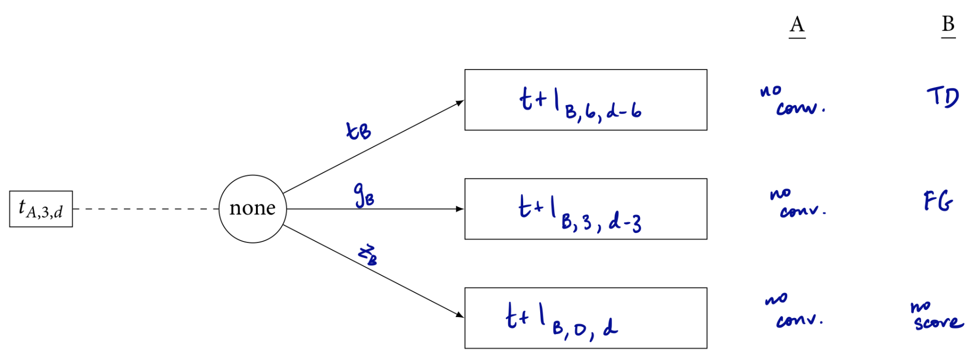
```
In [8]: # Initialize stochastic dynamic program - we want to maximize, so minimize = False
        dp = StochasticDP(number_of_stages, states, decisions, minimize=False)
```

Transition probabilities from stages $t = 0, 1, \dots, T - 2$

- Let's start with the easier transitions and work our way up to the more complicated ones.

From states $(A, 3, d)$

- Let's consider the transitions from states $(A, 3, d)$ for $d = -20, \dots, 20$ in stages $t = 0, 1, \dots, T - 2$:



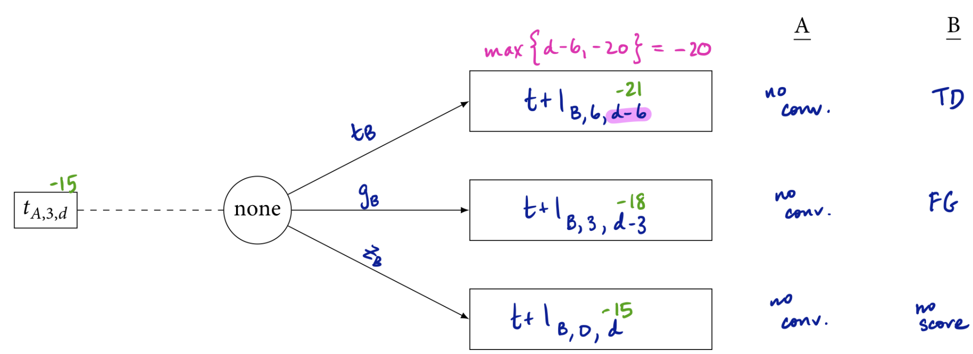
- Following the Lesson 19, we can put in these transitions like this:

```
In [9]: # Transition probabilities from (A, 3, d) up to stage T - 2
        for t in range(T - 1):
            for d in range(-20, 21):
                dp.add_transition(t, ('A', 3, d), 'none', ('B', 6, d - 6), tB, 0)
                dp.add_transition(t, ('A', 3, d), 'none', ('B', 3, d - 3), gB, 0)
                dp.add_transition(t, ('A', 3, d), 'none', ('B', 0, d), zB, 0)
```

```
    ---------------------------------------------------------------------------

    KeyError                                  Traceback (most recent call last)

    <ipython-input-9-d39ee919bf1f> in <module>()
      2 for t in range(T - 1):
      3     for d in range(-20, 21):
----> 4         dp.add_transition(t, ('A', 3, d), 'none', ('B', 6, d - 6), tB, 0)
      5         dp.add_transition(t, ('A', 3, d), 'none', ('B', 3, d - 3), gB, 0)
      6         dp.add_transition(t, ('A', 3, d), 'none', ('B', 0, d), zB, 0)


    ~/Dropbox/Development/teaching/stochasticdp/stochasticdp/stochasticdp.py in add_transition(self, stage, from_state, decisi
    152     def add_transition(self, stage, from_state, decision, to_state,
    153                        probability, contribution):
--> 154         if (self.probability[to_state, from_state, stage, decision] or
    155                 self.contribution[to_state, from_state, stage, decision]):
```

```
    156             warn("The transition from state {0} to state {1} in stage {2} "


~/Dropbox/Development/teaching/stochasticdp/stochasticdp/stochasticdp.py in __getitem__(self, key)
     38                 (t not in range(self.number_of_stages)) or
     39                 (x not in self.decisions)):
---> 40             raise KeyError('Invalid state, stage, or decision')
     41 
     42         if key not in self:


KeyError: 'Invalid state, stage, or decision'
```

- Why are we getting an `'Invalid state, stage, or decision'` message?
  - Remember that in Lesson 20, we assumed that d could take on an infinite number values.
  - On the other hand, here, we have limited d to be between -20 and 20.
- How can we work around this?
  - Let's do this: if d is supposed to be less than -20, then we simply assume that it is the same as having $d = -20$.
  - For example, suppose $d = -16$ in the diagram above. Then we can model the transitions like this:



- So... let's start over:

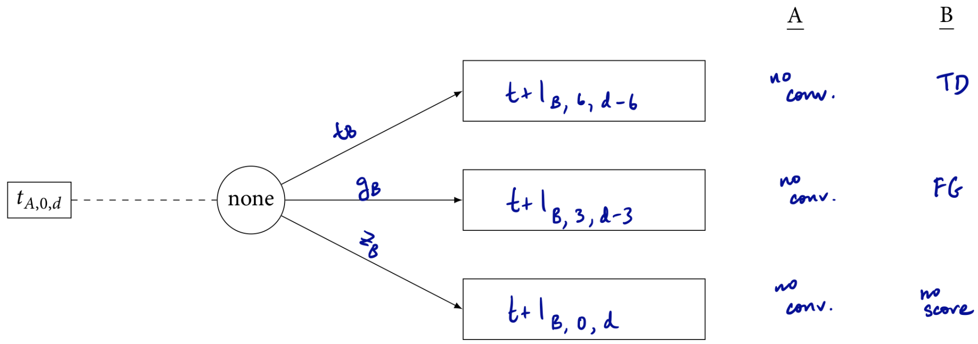
```
In [10]: # Initialize stochastic dynamic program - we want to maximize, so minimize = False
         dp = StochasticDP(number_of_stages, states, decisions, minimize=False)
```

- Using the idea above, we can write the following code to represent the transitions from states $(A, 3, d)$ for $d = -20, \dots, 20$ and $t = 0, 1, \dots, T-2$:

```
In [11]: # Transition probabilities from (A, 3, d) up to stage T - 2
         for t in range(T - 1):
             for d in range(-20, 21):
                 dp.add_transition(t, ('A', 3, d), 'none', ('B', 6, max(d - 6, -20)), tB, 0)
                 dp.add_transition(t, ('A', 3, d), 'none', ('B', 3, max(d - 3, -20)), gB, 0)
                 dp.add_transition(t, ('A', 3, d), 'none', ('B', 0, d), zB, 0)
```

From states $(A, 0, d)$

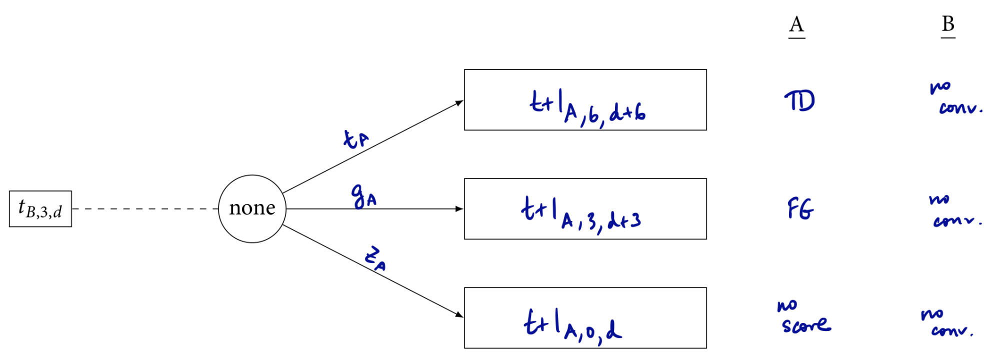
- Similarly, for the transitions from states $(A, 0, d)$ for $d = -20, \dots, 20$ in stages $t = 0, 1, \dots, T-2$:



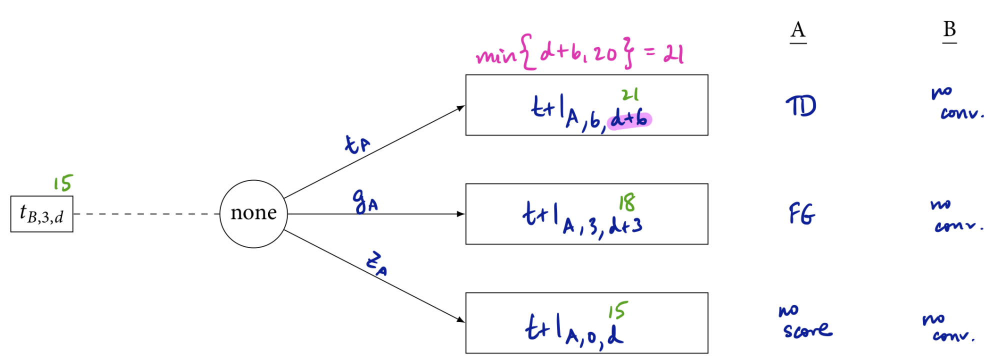

```
In [12]: # Transition probabilities from (A, 0, d) up to stage T - 2
         for t in range(T - 1):
             for d in range(-20, 21):
                 dp.add_transition(t, ('A', 0, d), 'none', ('B', 6, max(d - 6, -20)), tB, 0)
                 dp.add_transition(t, ('A', 0, d), 'none', ('B', 3, max(d - 3, -20)), gB, 0)
                 dp.add_transition(t, ('A', 0, d), 'none', ('B', 0, d), zB, 0)
```

From states $(B, 3, d)$

- Next, the transitions from states $(B, 3, d)$ for $d = -20, \dots, 20$ in stages $t = 0, 1, \dots, T-2$:



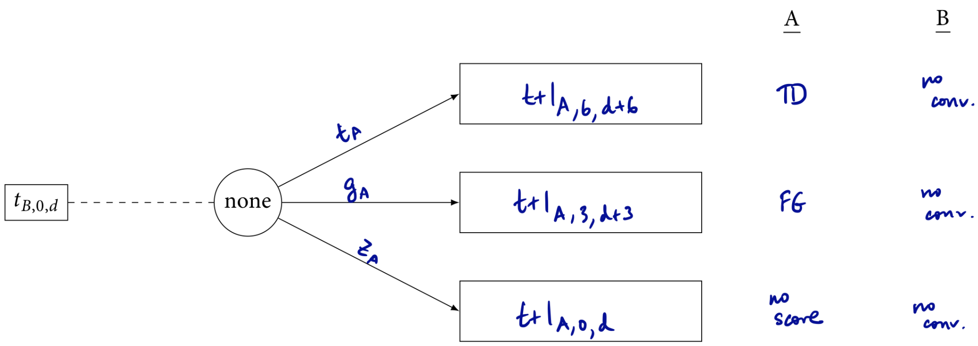
- We run into a similar problem: if we're not careful, we can end up with values of d greater than 20.
- Let's do this: if d is supposed to be greater than 20, then we simply assume that it is the same as having $d = 20$, like this:



```
In [13]: # Transition probabilities from (B, 3, d) up to stage T - 2
         for t in range(T - 1):
             for d in range(-20, 21):
                 dp.add_transition(t, ('B', 3, d), 'none', ('A', 6, min(d + 6, 20)), tA, 0)
                 dp.add_transition(t, ('B', 3, d), 'none', ('A', 3, min(d + 3, 20)), gA, 0)
                 dp.add_transition(t, ('B', 3, d), 'none', ('A', 0, d), zA, 0)
```

From states $(B, 0, d)$

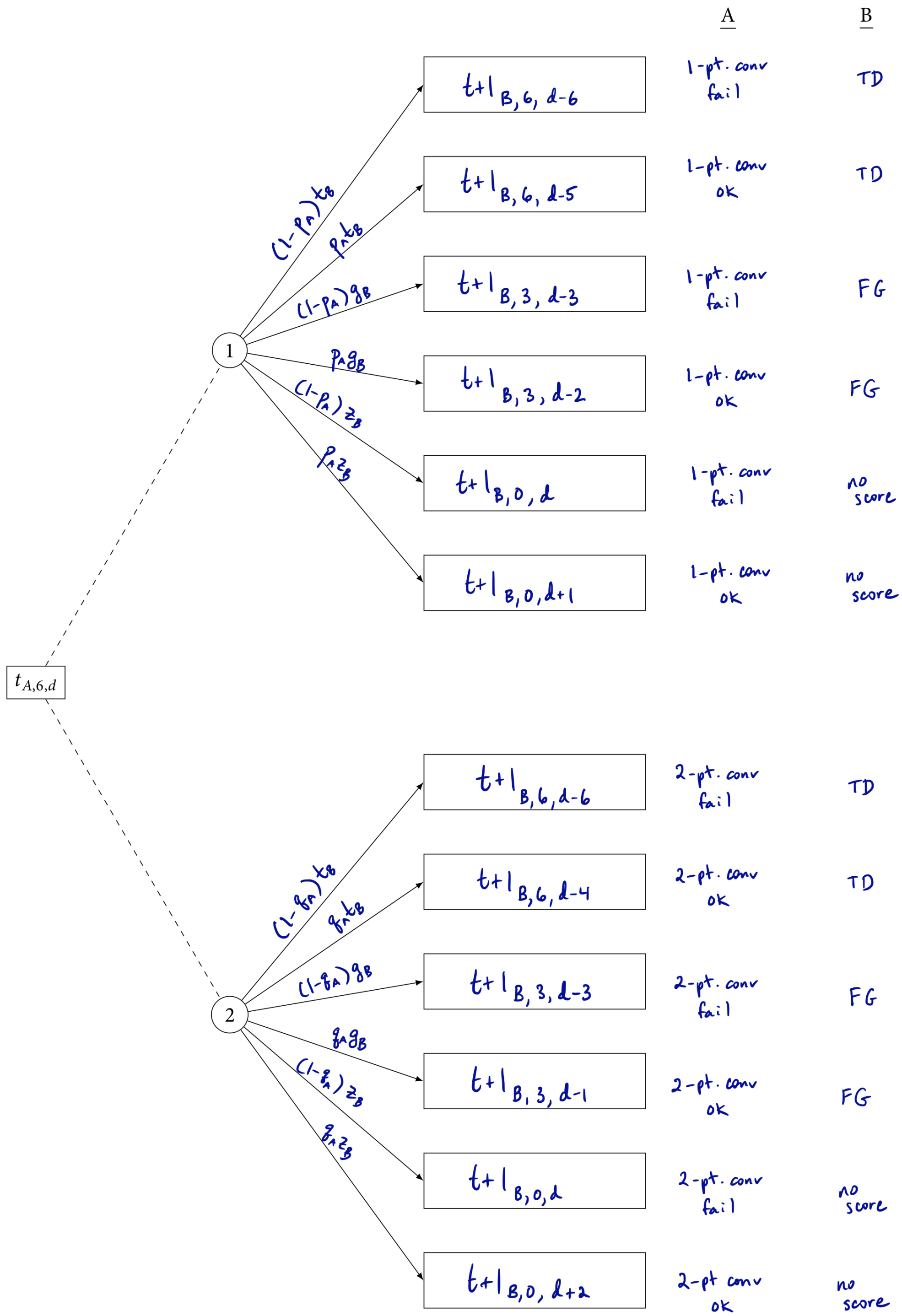
- Next, the transitions from states $(B, 0, d)$ for $d = -20, \ldots, 20$ in stages $t = 0, 1, \ldots, T-2$:



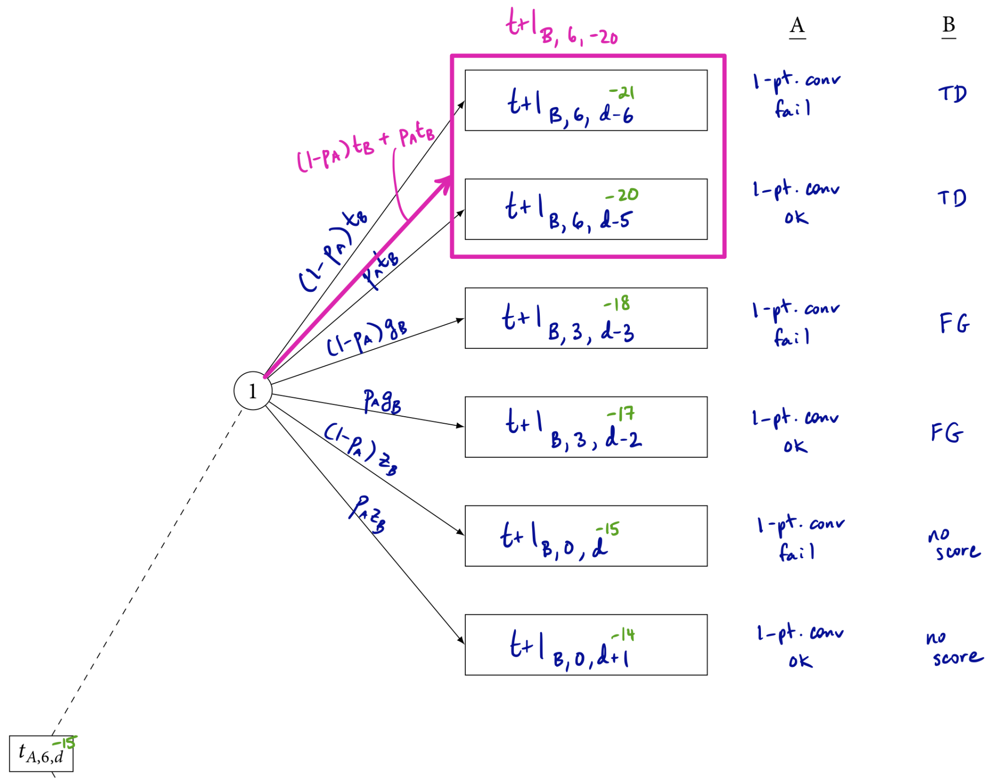

```
In [14]: # Transition probabilities from (B, 0, d) up to stage T - 2
         for t in range(T - 1):
             for d in range(-20, 21):
                 dp.add_transition(t, ('B', 0, d), 'none', ('A', 6, min(d + 6, 20)), tA, 0)
                 dp.add_transition(t, ('B', 0, d), 'none', ('A', 3, min(d + 3, 20)), gA, 0)
                 dp.add_transition(t, ('B', 0, d), 'none', ('A', 0, d), zA, 0)
```

From states $(A, 6, d)$

- Next up: transitions from the state $(A, 6, d)$ for $d = -20, \ldots, 20$ in stages $t = 0, 1, \ldots, T-2$:



- In this case, we can also have values of d lower than -20 or higher than 20 if we're not careful.
- Because of this, we can also have multiple transitions to the same state, like this:



- To take care of this, we simply need to merge these transitions.
- The code becomes a bit more complicated, though, since we need to consider various cases.
- First, let's consider the 1-point conversion decision and break down the cases depending on the value of d:

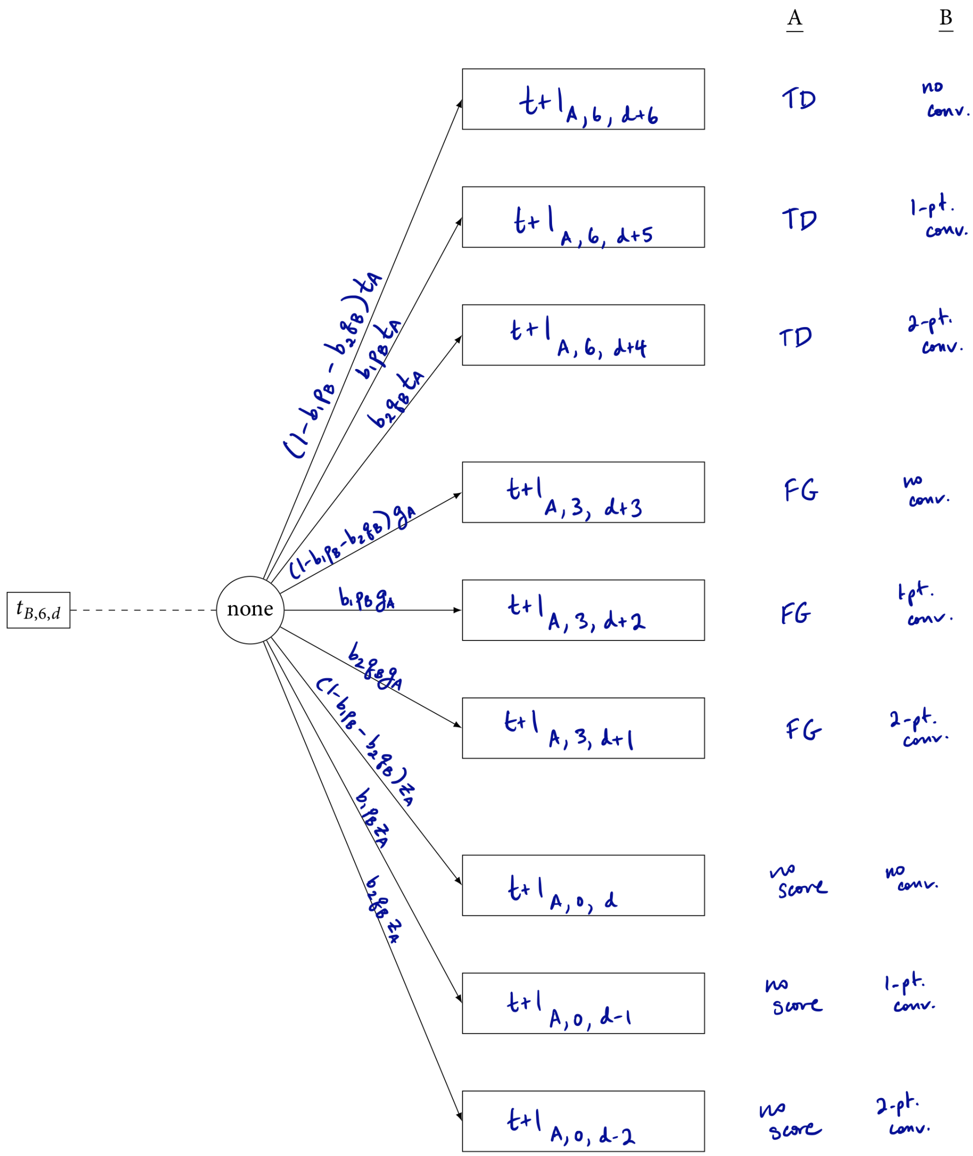
```
In [15]: # Remember:
         # dp.add_transition(stage, from_state, decision, to_state, probability, contribution)

         # Transition probabilities from (A, 6, d) up to stage T - 2
         # Decision: 1-point conversion
         for t in range(T - 1):
             for d in range(-20, 21):

                 if d - 5 <= -20:
                     dp.add_transition(t, ('A', 6, d), 1, ('B', 6, -20), (1 - pA) * tB + pA * tB, 0)
                 else:
                     dp.add_transition(t, ('A', 6, d), 1, ('B', 6, max(d - 6, -20)), (1 - pA) * tB, 0)
                     dp.add_transition(t, ('A', 6, d), 1, ('B', 6, max(d - 5, -20)), pA * tB, 0)

                 if d - 2 <= -20:
                     dp.add_transition(t, ('A', 6, d), 1, ('B', 3, -20), (1 - pA) * gB + pA * gB, 0)
                 else:
                     dp.add_transition(t, ('A', 6, d), 1, ('B', 3, max(d - 3, -20)), (1 - pA) * gB, 0)
                     dp.add_transition(t, ('A', 6, d), 1, ('B', 3, max(d - 2, -20)), pA * gB, 0)

                 if d >= 20:
                     dp.add_transition(t, ('A', 6, d), 1, ('B', 0, 20), (1 - pA) * zB + pA * zB, 0)
                 else:
                     dp.add_transition(t, ('A', 6, d), 1, ('B', 0, min(d, 20)), (1 - pA) * zB, 0)
                     dp.add_transition(t, ('A', 6, d), 1, ('B', 0, min(d + 1, 20)), pA * zB, 0)
```

- Let's do the same for the 2-point conversion decision:

```
In [16]: # Remember:
         # dp.add_transition(stage, from_state, decision, to_state, probability, contribution)

         # Transition probabilities from (A, 6, d) up to stage T - 2
         # Decision: 2-point conversion
         for t in range(T - 1):
             for d in range(-20, 21):

                 if d - 4 <= -20:
                     dp.add_transition(t, ('A', 6, d), 2, ('B', 6, -20), (1 - qA) * tB + qA * tB, 0)
                 else:
                     dp.add_transition(t, ('A', 6, d), 2, ('B', 6, max(d - 6, -20)), (1 - qA) * tB, 0)
                     dp.add_transition(t, ('A', 6, d), 2, ('B', 6, max(d - 4, -20)), qA * tB, 0)

                 if d - 1 <= -20:
                     dp.add_transition(t, ('A', 6, d), 2, ('B', 3, -20), (1 - qA) * gB + qA * gB, 0)
                 else:
                     dp.add_transition(t, ('A', 6, d), 2, ('B', 3, max(d - 3, -20)), (1 - qA) * gB, 0)
                     dp.add_transition(t, ('A', 6, d), 2, ('B', 3, max(d - 1, -20)), qA * gB, 0)

                 if d >= 20:
                     dp.add_transition(t, ('A', 6, d), 2, ('B', 0, 20),  (1 - qA) * zB + qA * zB, 0)
                 else:
                     dp.add_transition(t, ('A', 6, d), 2, ('B', 0, min(d, 20)), (1 - qA) * zB, 0)
                     dp.add_transition(t, ('A', 6, d), 2, ('B', 0, min(d + 2, 20)), qA * zB, 0)
```

From states $(B, 6, d)$

- We need to take similar care for the transitions from states $(B, 6, d)$ for $d = -20, \dots, 20$ in stages $t = 0, 1, \dots, T-2$:

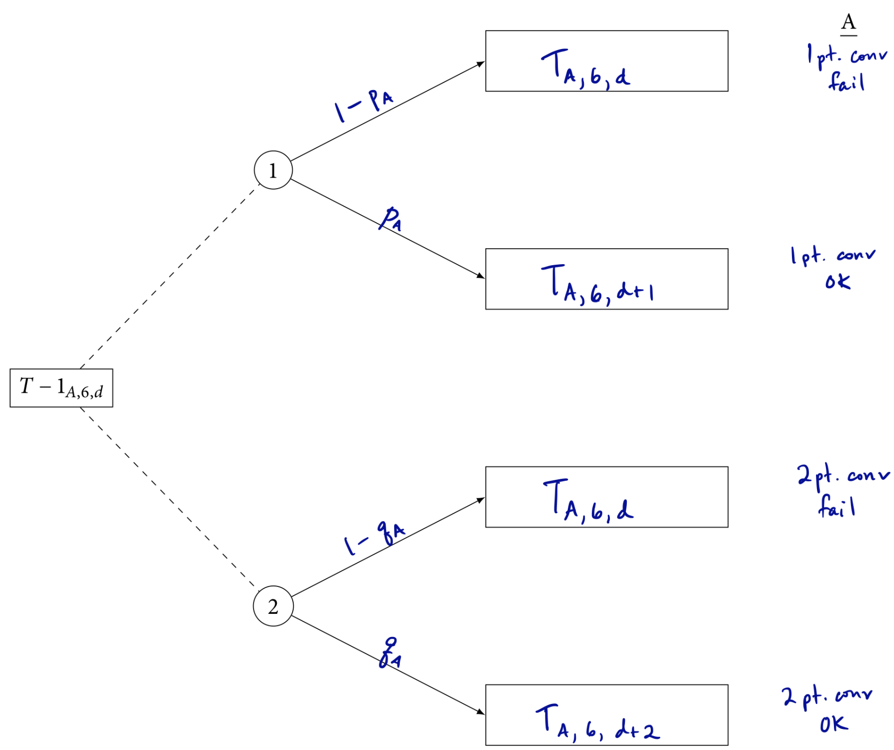



```
In [17]: # Transition probabilities from (B, 6, d) up to stage T - 2
         for t in range(T - 1):
             for d in range(-20, 21):
                 if d + 4 >= 20:
                     dp.add_transition(t, ('B', 6, d), 'none', ('A', 6, 20), (1 - b1*pB - b2*qB) *
                                       tA + b1*pB*tA + b2*qB*tA, 0)
                 elif d + 5 >= 20:
                     dp.add_transition(t, ('B', 6, d), 'none', ('A', 6, 20), (1 - b1*pB - b2*qB) *
                                       tA + b1*pB*tA, 0)
                     dp.add_transition(t, ('B', 6, d), 'none', ('A', 6, min(d + 4, 20)), b2*qB*tA, 0)
                 else:
                     dp.add_transition(t, ('B', 6, d), 'none', ('A', 6, min(d + 6, 20)), (1 - b1*pB
                                       - b2*qB) * tA, 0)
                     dp.add_transition(t, ('B', 6, d), 'none', ('A', 6, min(d + 5, 20)), b1*pB*tA, 0)
                     dp.add_transition(t, ('B', 6, d), 'none', ('A', 6, min(d + 4, 20)), b2*qB*tA, 0)

                 if d + 1 >= 20:
                     dp.add_transition(t, ('B', 6, d), 'none', ('A', 3, 20), (1 - b1*pB - b2*qB) *
                                       gA + b1*pB*gA + b2*qB*gA, 0)
                 elif d + 2 >= 20:
                     dp.add_transition(t, ('B', 6, d), 'none', ('A', 3, 20), (1 - b1*pB - b2*qB) *
                                       gA + b1*pB*gA, 0)
                     dp.add_transition(t, ('B', 6, d), 'none', ('A', 3, min(d + 1, 20)), b2*qB*gA, 0)
```

```
            else:
                dp.add_transition(t, ('B', 6, d), 'none', ('A', 3, min(d + 3, 20)), (1 - b1*pB
                                  - b2*qB) * gA, 0)
                dp.add_transition(t, ('B', 6, d), 'none', ('A', 3, min(d + 2, 20)), b1*pB*gA, 0)
                dp.add_transition(t, ('B', 6, d), 'none', ('A', 3, min(d + 1, 20)), b2*qB*gA, 0)

            if d <= -20:
                dp.add_transition(t, ('B', 6, d), 'none', ('A', 0, -20), (1 - b1*pB - b2*qB) *
                                  zA + b1*pB*zA + b2*qB*zA, 0)
            elif d - 1 <= -20:
                dp.add_transition(t, ('B', 6, d), 'none', ('A', 0, d), (1 - b1*pB - b2*qB) * zA, 0)
                dp.add_transition(t, ('B', 6, d), 'none', ('A', 0, -20), b1*pB*zA + b2*qB*zA, 0)
            else:
                dp.add_transition(t, ('B', 6, d), 'none', ('A', 0, d), (1 - b1*pB - b2*qB) * zA, 0)
                dp.add_transition(t, ('B', 6, d), 'none', ('A', 0, max(d - 1, -20)), b1*pB*zA, 0)
                dp.add_transition(t, ('B', 6, d), 'none', ('A', 0, max(d - 2, - 20)), b2*qB*zA, 0)
```

Transition probabilities from stage $T - 1$

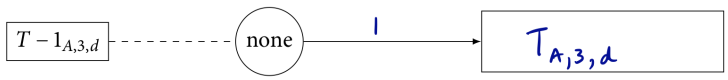
- Now, we can tackle the transitions from stage $T - 1$.
- From states $(A, 6, d)$ for $d = -20, \ldots, 20$ in stage $T - 1$:




```
In [18]: # Transition probabilities from (A, 6, d) in stage T - 1
         for d in range(-20, 21):
             # 1-point conversion
             if d >= 20:
                 dp.add_transition(T - 1, ('A', 6, d), 1, ('A', 6, 20), (1 - pA) + pA, 0)
             else:
                 dp.add_transition(T - 1, ('A', 6, d), 1, ('A', 6, d), 1 - pA, 0)
                 dp.add_transition(T - 1, ('A', 6, d), 1, ('A', 6, min(d + 1, 20)), pA, 0)

             # 2-point conversion
             if d >= 20:
                 dp.add_transition(T - 1, ('A', 6, d), 2, ('A', 6, 20), (1 - qA) + qA, 0)
             else:
                 dp.add_transition(T - 1, ('A', 6, d), 2, ('A', 6, d), 1 - qA, 0)
```

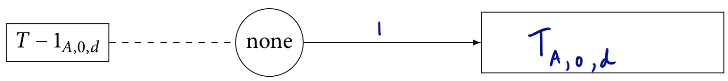
```
            dp.add_transition(T - 1, ('A', 6, d), 2, ('A', 6, min(d + 2, 20)), qA, 0)
```

- From states $(A, 3, d)$ for $d = -20, \ldots, 20$ in stage $T - 1$:



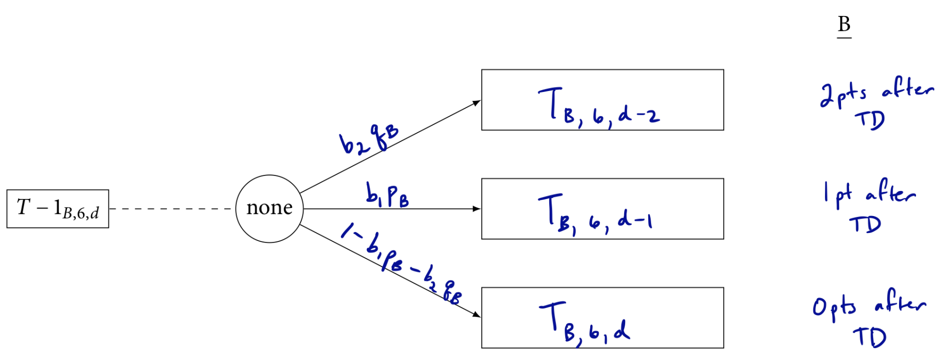

```
In [19]: # Transition probabilities from (A, 3, d) in stage T - 1
         for d in range(-20, 21):
             dp.add_transition(T - 1, ('A', 3, d), 'none', ('A', 3, d), 1, 0)
```

- From states $(A, 0, d)$ for $d = -20, \ldots, 20$ in stage $T - 1$:



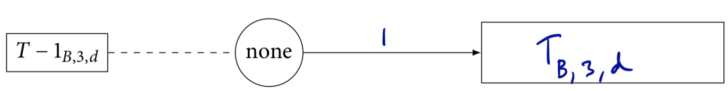

```
In [20]: # Transition probabilities from (A, 0, d) in stage T - 1
         for d in range(-20, 21):
             dp.add_transition(T - 1, ('A', 0, d), 'none', ('A', 0, d), 1, 0)
```

- From states $(B, 6, d)$ for $d = -20, \ldots, 20$ in stage $T - 1$:



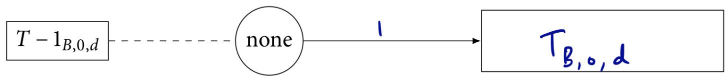

```
In [21]: # Transition probabilities from (B, 6, d) in stage T - 1
         for d in range(-20, 21):
             if d <= -20:
                 dp.add_transition(T - 1, ('B', 6, d), 'none', ('B', 6, -20), 1, 0)
             elif d <= -19:
                 dp.add_transition(T - 1, ('B', 6, d), 'none', ('B', 6, -20), b1*pB + b2*qB, 0)
                 dp.add_transition(T - 1, ('B', 6, d), 'none', ('B', 6, d), 1 - b1*pB - b2*qB, 0)
             else:
                 dp.add_transition(T - 1, ('B', 6, d), 'none', ('B', 6, max(d - 2, -20)), b2*qB, 0)
                 dp.add_transition(T - 1, ('B', 6, d), 'none', ('B', 6, max(d - 1, -20)), b1*pB, 0)
                 dp.add_transition(T - 1, ('B', 6, d), 'none', ('B', 6, d), 1 - b1*pB - b2*qB, 0)
```

- From states $(B, 3, d)$ for $d = -20, \ldots, 20$ in stage $T - 1$:



```
In [22]: # Transition probabilities from (B, 3, d) in stage T - 1
         for d in range(-20, 21):
             dp.add_transition(T - 1, ('B', 3, d), 'none', ('B', 3, d), 1, 0)
```

- From states $(B, 0, d)$ for $d = -20, \ldots, 20$ in stage $T - 1$:




```
In [23]: # Transition probabilities from (B, 0, d) in stage T - 1
         for d in range(-20, 21):
             dp.add_transition(T - 1, ('B', 0, d), 'none', ('B', 0, d), 1, 0)
```

Boundary conditions

- Finally, the boundary conditions:

$$f_T(n, k, d) = \begin{cases} 1 & \text{if } d > 0 \\ r & \text{if } d = 0 \\ 0 & \text{if } d < 0 \end{cases} \qquad \text{for } n \in \{\text{A}, \text{B}\}, k \in \{0, 3, 6\}, d = -20, \ldots, 20$$

```
In [24]: # Boundary conditions
         for n in ['A', 'B']:
             for k in [0, 3, 6]:
                 for d in range(-20, 21):
                     if d > 0:
                         dp.add_boundary(state=(n, k, d), value=1)
                     elif d == 0:
                         dp.add_boundary(state=(n, k, d), value=r)
                     else:
                         dp.add_boundary(state=(n, k, d), value=0)
```

Solving the stochastic dynamic program

```
In [25]: # Solve the stochastic dynamic program
         value, policy = dp.solve()
```

Interpreting output from the stochastic dynamic program

- What is the maximum probability that Team A wins after scoring a touchdown in the first possession?

```
In [26]: # Maximum probability that Team A wins after scoring a touchdown in the first possession
         print(value[0, ('A', 6, 6)])
```

```
0.7022345341399421
```

- What should Team A do after scoring a touchdown in the first possession?

```
In [27]: # Optimal strategy for Team A after scoring a touchdown in the first possession
         policy[0, ('A', 6, 6)]
```

```
Out[27]: {1}
```

- Suppose Team A just scored a touchdown, making it 4 points ahead. How does (1) the probability of Team A winning and (2) Team A's optimal strategy change depending on which possession this happened? Why do the trends you identified make sense?
  - What if Team A were 5 points ahead? 1 point behind?

Hint. Write a `for` loop that prints out the information you want.

```
In [28]: d = -1
         for t in range(number_of_stages - 1):
             print("Points ahead: {1}   Possession: {0}    Go for: {2}   Pr(win): {3}"
                   .format(t, d, policy[t, ('A', 6, d)], value[t, ('A', 6, d)]))
```

```
Points ahead: -1   Possession: 0    Go for: {2}   Pr(win): 0.5072953616034738
Points ahead: -1   Possession: 1    Go for: {2}   Pr(win): 0.44519240474106353
Points ahead: -1   Possession: 2    Go for: {2}   Pr(win): 0.50731028518891
Points ahead: -1   Possession: 3    Go for: {2}   Pr(win): 0.4424885299821573
Points ahead: -1   Possession: 4    Go for: {2}   Pr(win): 0.5074118904508018
Points ahead: -1   Possession: 5    Go for: {2}   Pr(win): 0.43931528946076176
Points ahead: -1   Possession: 6    Go for: {2}   Pr(win): 0.5075907168890346
Points ahead: -1   Possession: 7    Go for: {2}   Pr(win): 0.4355105592923179
Points ahead: -1   Possession: 8    Go for: {2}   Pr(win): 0.5078210778988483
Points ahead: -1   Possession: 9    Go for: {2}   Pr(win): 0.4308230454850312
Points ahead: -1   Possession: 10    Go for: {2}   Pr(win): 0.5080497243247094
Points ahead: -1   Possession: 11    Go for: {2}   Pr(win): 0.4248444944890374
Points ahead: -1   Possession: 12    Go for: {2}   Pr(win): 0.5081772915654194
Points ahead: -1   Possession: 13    Go for: {2}   Pr(win): 0.41687201090463577
Points ahead: -1   Possession: 14    Go for: {2}   Pr(win): 0.5080221801678291
Points ahead: -1   Possession: 15    Go for: {2}   Pr(win): 0.40558863804792583
Points ahead: -1   Possession: 16    Go for: {2}   Pr(win): 0.5072110127638884
Points ahead: -1   Possession: 17    Go for: {2}   Pr(win): 0.3881880953683321
Points ahead: -1   Possession: 18    Go for: {2}   Pr(win): 0.5047762309420285
Points ahead: -1   Possession: 19    Go for: {2}   Pr(win): 0.3576599174598068
Points ahead: -1   Possession: 20    Go for: {1}   Pr(win): 0.4987950978607944
Points ahead: -1   Possession: 21    Go for: {1}   Pr(win): 0.30286500000000005
Points ahead: -1   Possession: 22    Go for: {1}   Pr(win): 0.4965
```