Lesson 19.

Solving the points-after-touchdown problem

Before we begin — upgrading and importing `stochasticdp`

- I've made some improvements to the internals of `stochasticdp`.
- To upgrade, open a WinPython Command Prompt and type:

```
pip install --upgrade stochasticdp
```

- Now we can import `StochasticDP` from `stochasticdp`:

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

Setting up the data

- In Lesson 18, we worked with the following data:

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

- 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
```

```python
# 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:

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

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

- States:

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

- In Lesson 18, 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]: # Maximum lead for Team A
        max_d = 20

        # List of states
        states = []
        for n in ['A', 'B']:
            for k in [0, 3, 6]:
                for d in range(-max_d, max_d + 1):
                    states.append((n, k, d))
```

- 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 [6]: # List of decisions
        decisions = [1, 2, 'none']
```

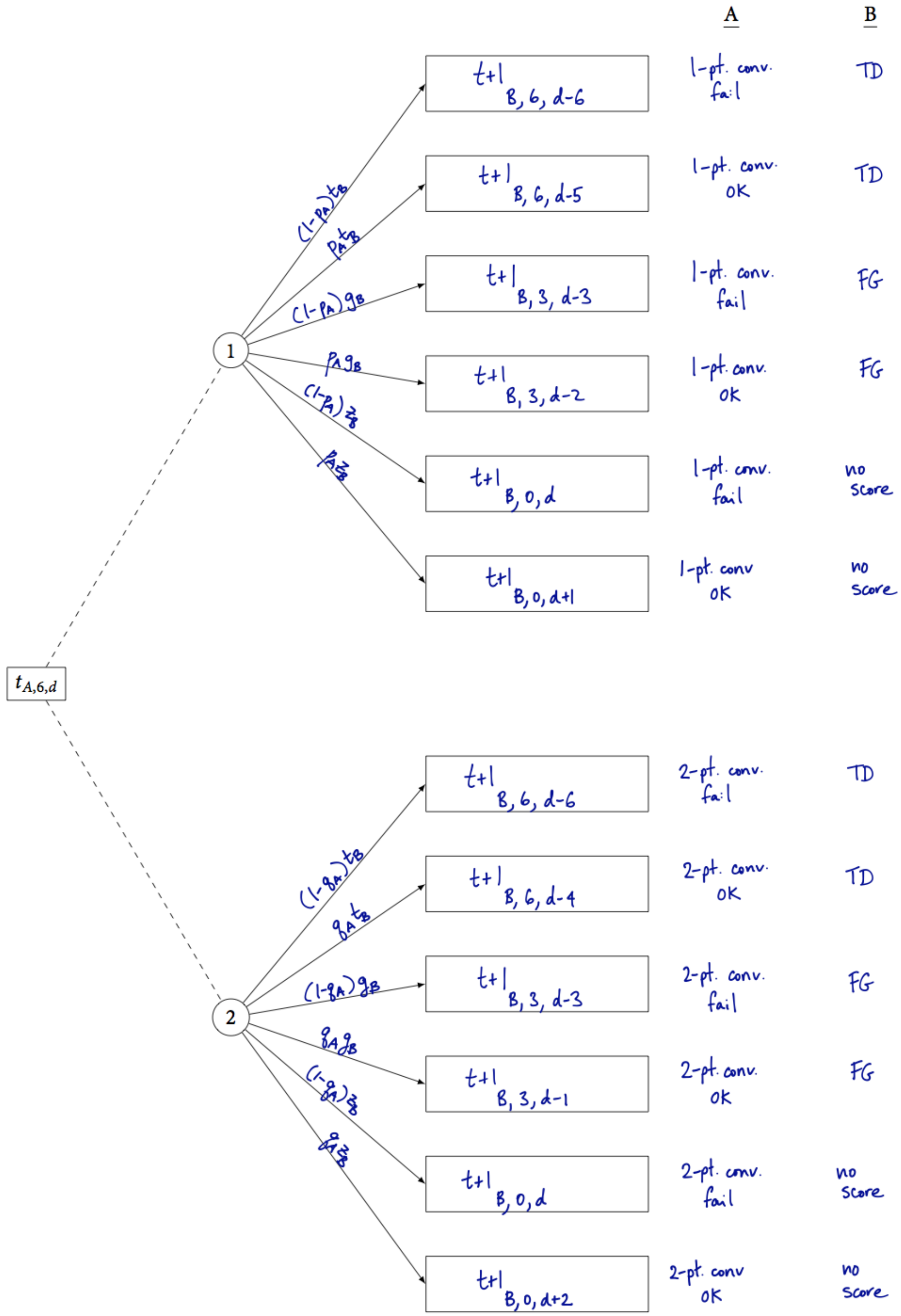
- Now we can initialize a stochastic DP object called `dp` as follows:

```
In [7]: # Initialize stochastic dynamic program - we want to maximize, so minimize = False
        dp = StochasticDP(number_of_stages, states, decisions, minimize=False)
```

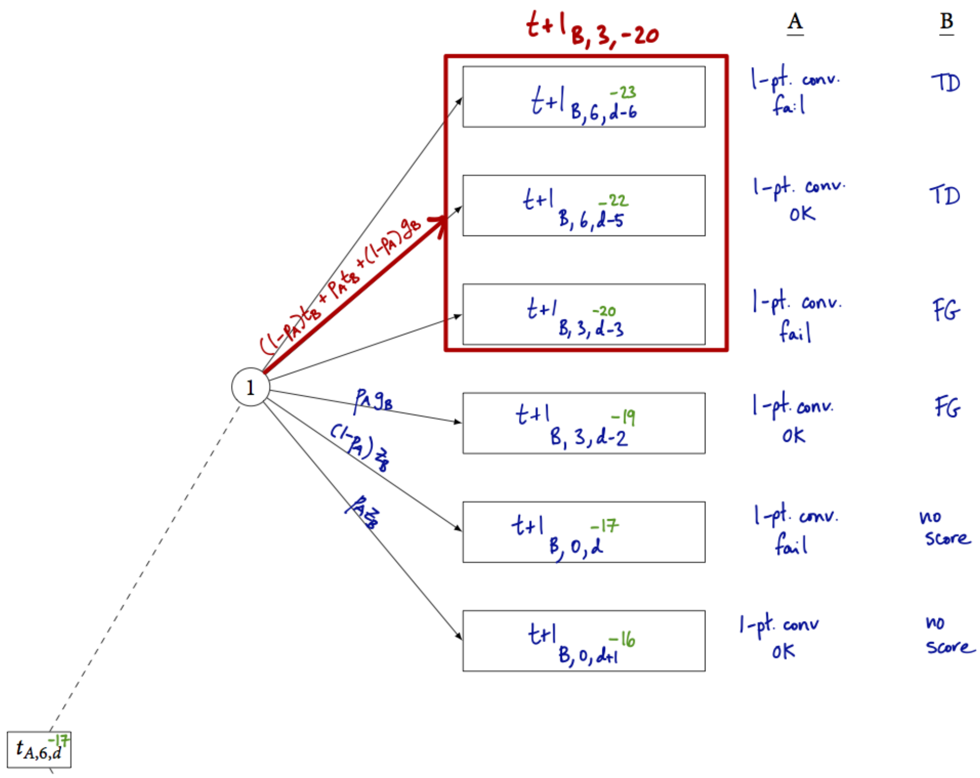
Transition probabilities from stages $t = 0, 1, \ldots, T - 2$

- First, let's tackle transitions from the state $(A, 6, d)$ for $d = -20, \ldots, 20$ in stages $t = 0, 1, \ldots, T - 2$:

- In Lesson 18, 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 does this change our transition probabilities?
- For example, suppose $d = -17$ in the diagram above. Then we can model the transition probabilities like this:


A
B
$t_{A,6,d}$
1
$(1-p_A)t_B$
$p_A t_B$
$(1-p_A)g_B$
$p_A g_B$
$(1-p_A)\bar{z}_B$
$p_A \bar{z}_B$
t+1 B, 6, d-6
1-pt. conv. fail
TD
t+1 B, 6, d-5
1-pt. conv. OK
TD
t+1 B, 3, d-3
1-pt. conv. fail
FG
t+1 B, 3, d-2
1-pt. conv. OK
FG
t+1 B, 0, d
1-pt. conv. fail
no score
t+1 B, 0, d+1
1-pt. conv OK
no score
2
$(1-q_A)t_B$
$q_A t_B$
$(1-q_A)g_B$
$q_A g_B$
$(1-q_A)\bar{z}_B$
$q_A \bar{z}_B$
t+1 B, 6, d-6
2-pt. conv. fail
TD
t+1 B, 6, d-4
2-pt. conv. OK
TD
t+1 B, 3, d-3
2-pt. conv. fail
FG
t+1 B, 3, d-1
2-pt. conv. OK
FG
t+1 B, 0, d
2-pt. conv. fail
no score
t+1 B, 0, d+2
2-pt. conv OK
no score

State $(A, 6, d)$



State $(A, 6, -17)$ under decision $x_t = 1$

- In other words, if d is supposed to be less than -20, then we simply assume that it is the same as having $d = -20$.
- We can do the same thing when d is supposed to be greater than 20.
- To implement this easily, we can define the transition probabilities like in the cell below.

- *Notes.*
  - In Python,

    ```
    a += 3
    ```

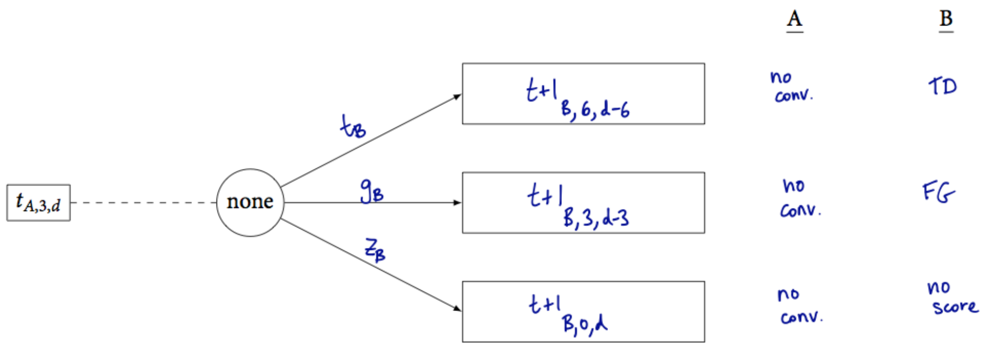
    is the same as

    ```
    a = a + 3
    ```
  - Remember that the transition probabilities and contributions are all initialized to 0.

```
In [8]: # Transition probabilities from (A, 6, d) up to stage T - 2
        for t in range(T - 1):
            for d in range(-max_d, max_d + 1):
                # 1-point conversion
                dp.transition[('B', 6, max(d - 6, -max_d)), ('A', 6, d), t, 1] += (1 - pA) * tB
                dp.transition[('B', 6, max(d - 5, -max_d)), ('A', 6, d), t, 1] += pA * tB
                dp.transition[('B', 3, max(d - 3, -max_d)), ('A', 6, d), t, 1] += (1 - pA) * gB
                dp.transition[('B', 3, max(d - 2, -max_d)), ('A', 6, d), t, 1] += pA * gB
                dp.transition[('B', 0, d),                  ('A', 6, d), t, 1] += (1 - pA) * zB
                dp.transition[('B', 0, min(d + 1, max_d)),  ('A', 6, d), t, 1] += pA * zB

                # 2-point conversion
```

```
            dp.transition[('B', 6, max(d - 6, -max_d)), ('A', 6, d), t, 2] += (1 - qA) * tB
            dp.transition[('B', 6, max(d - 4, -max_d)), ('A', 6, d), t, 2] += qA * tB
            dp.transition[('B', 3, max(d - 3, -max_d)), ('A', 6, d), t, 2] += (1 - qA) * gB
            dp.transition[('B', 3, max(d - 1, -max_d)), ('A', 6, d), t, 2] += qA * gB
            dp.transition[('B', 0, d),                  ('A', 6, d), t, 2] += (1 - qA) * zB
            dp.transition[('B', 0, min(d + 2, max_d)),  ('A', 6, d), t, 2] += qA * zB
```

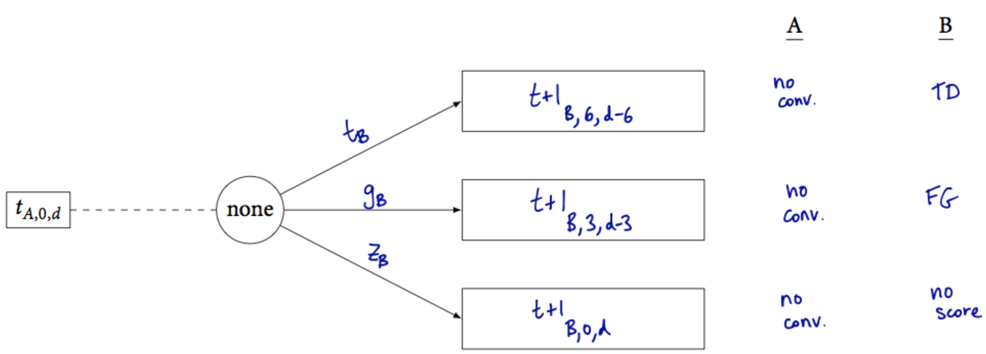
- In a similar fashion, we can define the remaining transition probabilities.
- From states $(A, 3, d)$ for $d = -20, \dots, 20$ in stages $t = 0, 1, \dots, T-2$:



State $(A, 3, d)$

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

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

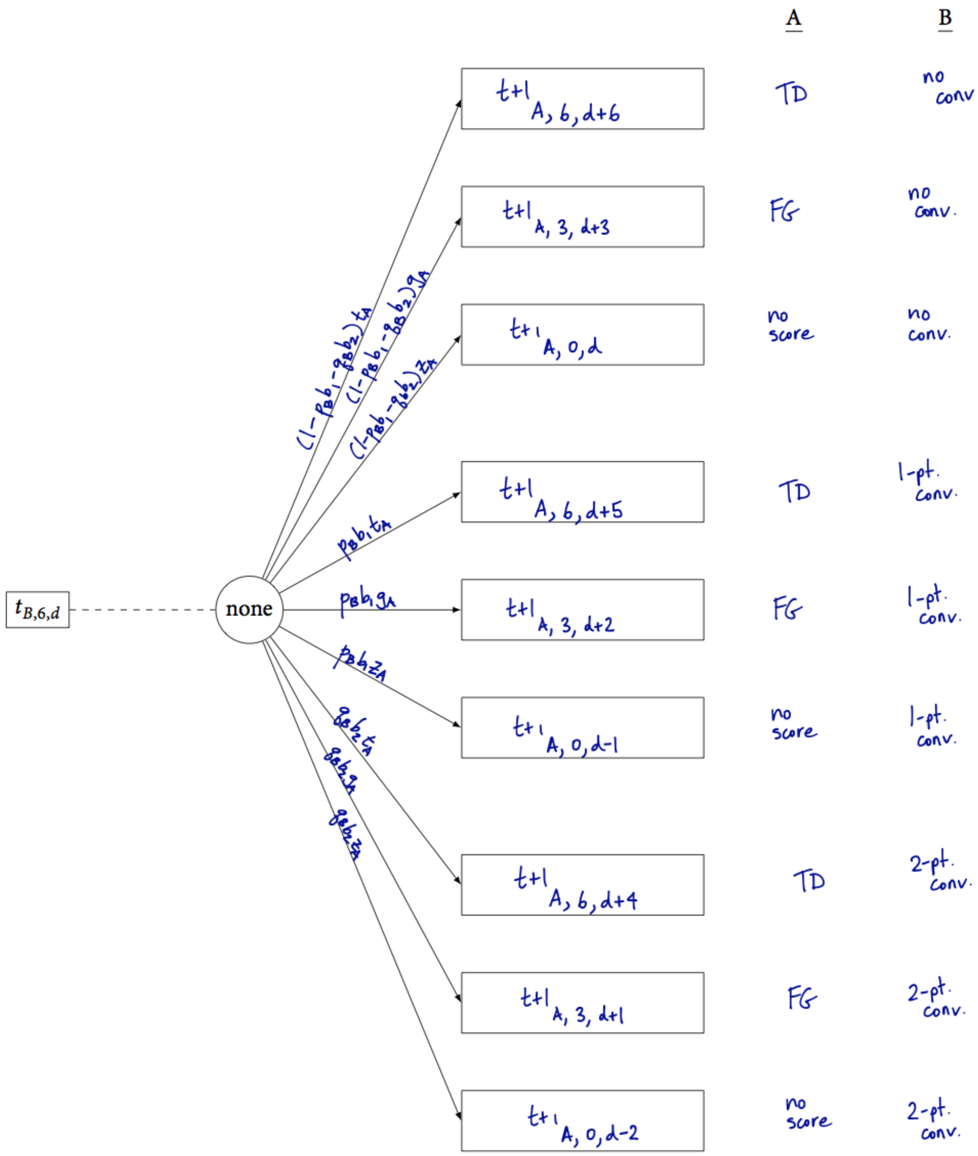


State $(A, 0, d)$

```
In [10]: # Transition probabilities from (A, 0, d) up to stage T - 2
         for t in range(T - 1):
             for d in range(-max_d, max_d + 1):
```

```
            dp.transition[('B', 6, max(d - 6, -max_d)), ('A', 0, d), t, 'none'] += tB
            dp.transition[('B', 3, max(d - 3, -max_d)), ('A', 0, d), t, 'none'] += gB
            dp.transition[('B', 0, d),                  ('A', 0, d), t, 'none'] += zB
```

- From states $(B, 6, d)$ for $d = -20, \ldots, 20$ in stages $t = 0, 1, \ldots, T-2$:

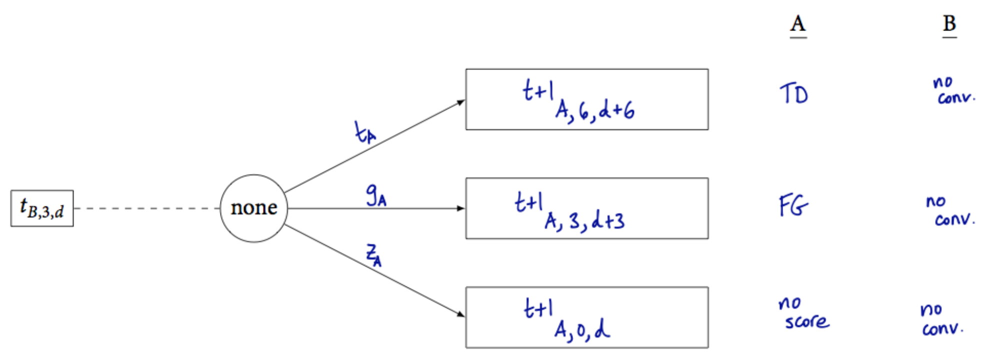


State $(B, 6, d)$

```
In [11]: # Transition probabilities from (B, 6, d) up to stage T - 2
         for t in range(T - 1):
             for d in range(-max_d, max_d + 1):
                 dp.transition[('A', 6, min(d + 6, max_d)), ('B', 6, d), t, 'none'] += (1 - b1*pB - b2*qB) * tA
                 dp.transition[('A', 3, min(d + 3, max_d)), ('B', 6, d), t, 'none'] += (1 - b1*pB - b2*qB) * gA
                 dp.transition[('A', 0, d),                 ('B', 6, d), t, 'none'] += (1 - b1*pB - b2*qB) * zA

                 dp.transition[('A', 6, min(d + 5, max_d)),  ('B', 6, d), t, 'none'] += b1 * pB * tA
                 dp.transition[('A', 3, min(d + 2, max_d)),  ('B', 6, d), t, 'none'] += b1 * pB * gA
```

```
            dp.transition[('A', 0, max(d - 1, -max_d)), ('B', 6, d), t, 'none'] += b1 * pB * zA

            dp.transition[('A', 6, min(d + 4, max_d)),  ('B', 6, d), t, 'none'] += b2 * qB * tA
            dp.transition[('A', 3, min(d + 1, max_d)),  ('B', 6, d), t, 'none'] += b2 * qB * gA
            dp.transition[('A', 0, max(d - 2, -max_d)), ('B', 6, d), t, 'none'] += b2 * qB * zA
```

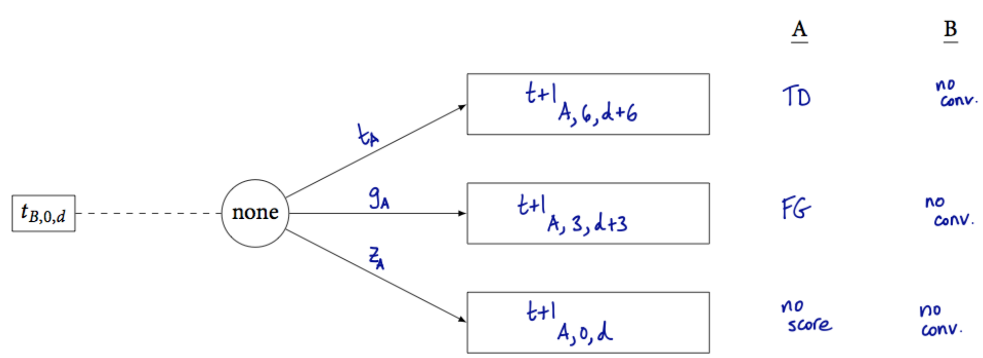
- From states $(B, 3, d)$ for $d = -20, \ldots, 20$ in stages $t = 0, 1, \ldots, T-2$:



State $(B, 3, d)$

```
In [12]: # Transition probabilities from (B, 3, d) up to stage T - 2
         for t in range(T - 1):
             for d in range(-max_d, max_d + 1):
                 dp.transition[('A', 6, min(d + 6, max_d)), ('B', 3, d), t, 'none'] += tA
                 dp.transition[('A', 3, min(d + 3, max_d)), ('B', 3, d), t, 'none'] += gA
                 dp.transition[('A', 0, d),                 ('B', 3, d), t, 'none'] += zA
```

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

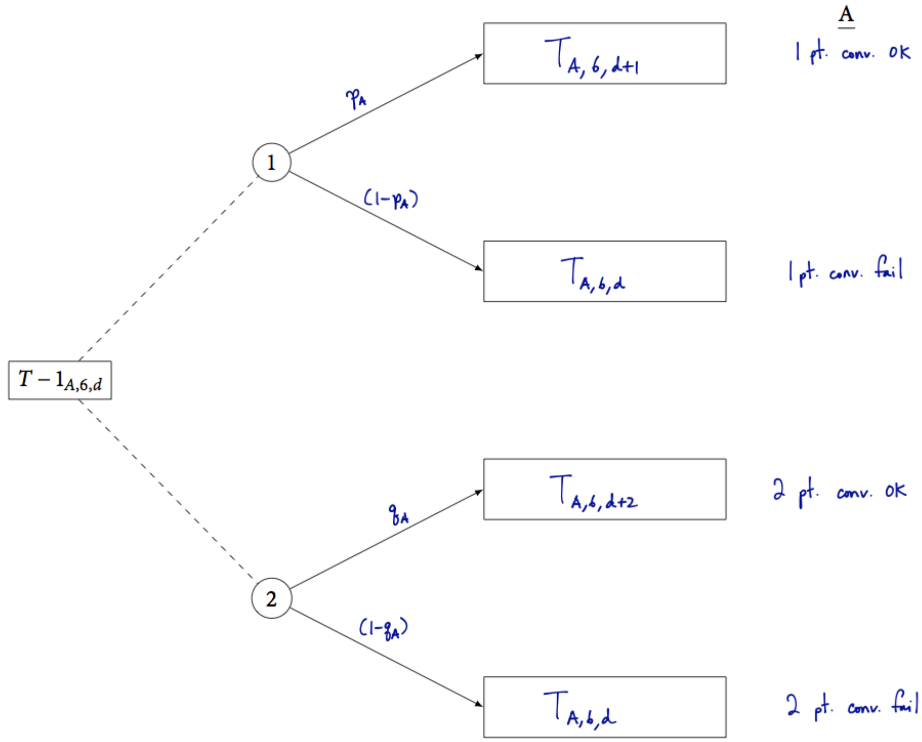


State $(B, 0, d)$

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

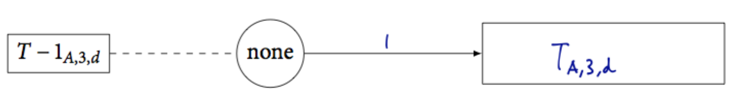
Transition probabilities from stage $T - 1$

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



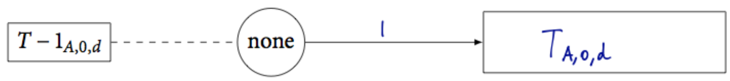

```
In [14]: # Transition probabilities from (A, 6, d) in stage T - 1
         for d in range(-max_d, max_d + 1):
             # 1-point conversion
             dp.transition[('A', 6, min(d + 1, max_d)),  ('A', 6, d), T - 1, 1] += pA
             dp.transition[('A', 6, d),                  ('A', 6, d), T - 1, 1] += 1 - pA

             # 2-point conversion
             dp.transition[('B', 0, min(d + 2, max_d)),  ('A', 6, d), T - 1, 2] += qA
             dp.transition[('B', 0, d),                  ('A', 6, d), T - 1, 2] += (1 - qA)
```

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



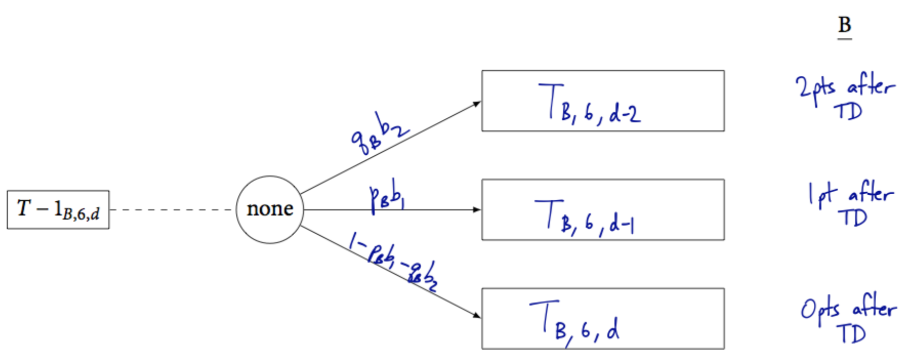
```
In [15]: # Transition probabilities from (A, 3, d) in stage T - 1
         for d in range(-max_d, max_d + 1):
             dp.transition[('A', 3, d), ('A', 3, d), T - 1, 'none'] += 1
```

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



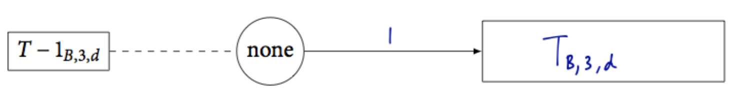

```
In [16]: # Transition probabilities from (A, 0, d) in stage T - 1
         for d in range(-max_d, max_d + 1):
             dp.transition[('A', 0, d), ('A', 0, d), T - 1, 'none'] += 1
```

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



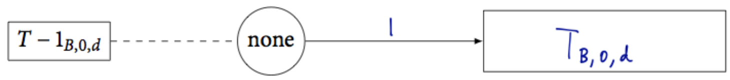

```
In [17]: # Transition probabilities from (B, 6, d) in stage T - 1
         for d in range(-max_d, max_d + 1):
             dp.transition[('B', 6, max(d - 2, -max_d)),  ('B', 6, d), T - 1, 'none'] += qB * b2
             dp.transition[('B', 6, max(d - 1, -max_d)),  ('B', 6, d), T - 1, 'none'] += pB * b1
             dp.transition[('B', 6, d),                  ('B', 6, d), T - 1, 'none'] += 1 - pB * b1 - qB * b2
```

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




```
In [18]: # Transition probabilities from (B, 3, d) in stage T - 1
         for d in range(-max_d, max_d + 1):
             dp.transition[('B', 3, d), ('B', 3, d), T - 1, 'none'] += 1
```

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




```
In [19]: # Transition probabilities from (B, 0, d) in stage T - 1
         for d in range(-max_d, max_d + 1):
             dp.transition[('B', 0, d), ('B', 0, d), T - 1, 'none'] += 1
```

Boundary conditions

- Finally, the boundary conditions:

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

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

Solving the stochastic dynamic program

```
In [21]: # Solve the stochastic dynamic program
         value, policy = dp.solve()
```

Interpreting output from the stochastic dynamic program

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

```
In [22]: value[0, ('A', 6, 6)]
```

```
Out[22]: 0.7022345341399421
```

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

```
In [23]: policy[0, ('A', 6, 6)]
```

```
Out[23]: {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?

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

```
In [24]: d = 4
         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: 4   Possession: 0    Go for: {1}   Pr(win): 0.6515937504061142
Points ahead: 4   Possession: 1    Go for: {1}   Pr(win): 0.5953309554275077
Points ahead: 4   Possession: 2    Go for: {1}   Pr(win): 0.6589513039084535
Points ahead: 4   Possession: 3    Go for: {1}   Pr(win): 0.6007578375899824
Points ahead: 4   Possession: 4    Go for: {1}   Pr(win): 0.6672665713194332
Points ahead: 4   Possession: 5    Go for: {1}   Pr(win): 0.6068027252465114
Points ahead: 4   Possession: 6    Go for: {1}   Pr(win): 0.6768164108937468
Points ahead: 4   Possession: 7    Go for: {1}   Pr(win): 0.613684035194822
Points ahead: 4   Possession: 8    Go for: {1}   Pr(win): 0.6879913865862931
Points ahead: 4   Possession: 9    Go for: {1}   Pr(win): 0.6217164148463916
Points ahead: 4   Possession: 10    Go for: {1}   Pr(win): 0.7013482767528225
Points ahead: 4   Possession: 11    Go for: {1}   Pr(win): 0.6313550712721305
Points ahead: 4   Possession: 12    Go for: {1}   Pr(win): 0.7177041574715131
Points ahead: 4   Possession: 13    Go for: {1}   Pr(win): 0.6432826232060777
Points ahead: 4   Possession: 14    Go for: {1}   Pr(win): 0.7383338660363578
Points ahead: 4   Possession: 15    Go for: {1}   Pr(win): 0.6586346048446324
Points ahead: 4   Possession: 16    Go for: {1}   Pr(win): 0.7654749169961569
Points ahead: 4   Possession: 17    Go for: {1}   Pr(win): 0.6797448781490858
Points ahead: 4   Possession: 18    Go for: {1}   Pr(win): 0.8038773271517671
Points ahead: 4   Possession: 19    Go for: {1}   Pr(win): 0.7130320885162305
Points ahead: 4   Possession: 20    Go for: {1}   Pr(win): 0.86647621838456
Points ahead: 4   Possession: 21    Go for: {2}   Pr(win): 0.7836036276200001
Points ahead: 4   Possession: 22    Go for: {1, 2}   Pr(win): 1.0
```