Project for NBA Hackathon Application

-- OBJECTIVE--

The objective of this project is to assign offensive and defensive ratings to basketball players given the play-by-play log for a game and the starting lineups for all four quarters of that game:

“Offensive Rating is defined as the team points scored per 100 possessions while the player is on the court. Defensive Rating is defined as the number of points per 100 possessions that the team allows while that individual player is on the court.”

-- PREPROCESSING --

Implementation is in preprocess.py:

1. Assign the correct team\_id for the rows by joining with Lineups.txt … write to Play\_by\_Play\_Processed.csv
2. Order the rows correctly using methodology indicated in the prompt (done in Excel) … write to Play\_by\_Play\_Processed\_Time.csv
3. Split Play\_by\_Play\_Processed\_Time.csv into multiple csv files- one for each game … write to “games” directory
   1. Store list of all 82 game ids … write to Game\_ids.txt
4. An end of possession can take multiple rows to identify, so go through each game and mark the end of every possession by writing our own special row- all 0’s except for PC time. Specifically, follow the possession end description given by the prompt, and then write the special row just before the PC-time changes … write to “games\_marked” directory

Use the csv files under “games\_marked” for processing.

-- PROCESSING --

Define a class called Game to hold important information when processing a game. Implementation is in models.py.

Notable fields:

1. **all\_players** (Pandas Dataframe) - full rosters and stats for both teams
   1. Rows- one per player … 30 total
   2. Columns- player\_Id, team\_Id, offensive score (off\_points), defensive score (def\_points), possessions (poss)
2. **on\_players** (Pandas Dataframe) - players on the court at the start of the possession
   1. Rows- one per player … 10 total
   2. Columns- player\_Id, team\_Id
3. **queue** (Pandas Dataframe) - substitutions since the start of the possession
   1. Rows- one per substitution … length varies
   2. Columns- Leaving player\_id (leaving), entering player\_id (entering), substitution pc\_time (sub\_time)

Notable methods:

1. Substitution (leaving\_player, entering\_player, sub\_time):
   1. append a row to **queue**
2. New\_possession (poss\_end\_time):
   1. Players on the court AT ANY POINT since the start of the possession get credit for it
   2. Take the set of a copy of on\_player’s player\_id’s, and loop through **queue**
      1. If substitution occurred before the score (sub\_time > score\_time), put the entering player in the set
   3. Use the resulting set to assign credit for the possession
   4. Loop through **queue** again
      1. Take leaving player out of **on\_players**, put entering player into **on\_players**
   5. **on\_players** now reflects the players on the court at the beginning of the next possession
3. Score (team\_id, score, score\_time):
   1. Only the players that were on the court LEADING UP to the score get credit for it
   2. Take the set of a copy of on\_player’s player\_id’s, and loop through **queue**:
      1. If substitution occurred before the score (sub\_time > score\_time), take out the leaving player and put in the entering player
   3. Use the resulting set to assign offensive or defensive score based on team
   4. Sanity check- The resulting set should contain 10 players, 5 for each team, every time
4. New\_period:
   1. Treat a new period as just a long list of substitutions between this period’s players and next period’s players
   2. Append this list of substitutions to the **queue**
   3. If next period has no players, it is the end of the game
      1. call Finish
5. Finish:
   1. Write the stats for this game to the “output” directory

With all the method logic now implemented in models.py, processing a game becomes trivial. Implementation is in process.py.

-- POST-PROCESSING –

Collect all the output csv files, consolidate and write to one file. Implementation is in post\_process.py.