Lakers

March 23, 2020

[1]:
```
# We will make use of pandas and numpy to manipulate our dataset
import pandas as pd
import numpy as np
pd.set_option('display.max_columns', None)
```

1 Longest Name Chain

[2]:
```
names_data = pd.read_csv("NBA_Names.csv")
```

[3]:
```
#summary statistics of dataset
names_data.describe()
```

[3]:
```
        Draft Year  Years of Service
count  4768.000000       4768.000000
mean   1012.246435          4.692743
std    1003.080250          4.302366
min       0.000000          0.000000
25%       0.000000          1.000000
50%    1981.000000          3.000000
75%    2008.000000          7.000000
max    2019.000000         21.000000
```

[4]:
```
#dimensions of dataset
names_data.shape
```

[4]:
```
(4768, 4)
```

[5]:
```
#first 5 rows of dataset
names_data.head()
```

[5]:
```
  First Name Last Name  Draft Year  Years of Service
0      Larry     Jones           0                 2
1     Darren      Daye           0                 5
2       Theo   Ratliff        1995                16
3    Antonis    Fotsis        2001                 1
4       Bill    Buntin           0                 1
```

1.1 Testing a single name

[6]:
```
#initialize the chain to a single name and create a variable for the last name␣
 ↪since we'll be using this to search for
# chain "links"
chain = ""
x = names_data.iloc[3844]["Last Name"]
chain = names_data.iloc[3844]["First Name"] + " " + names_data.iloc[3844]["Last␣
 ↪Name"]
chain
```

[6]:
```
'Kobe Bryant'
```

[7]:
```
temp_df = names_data[names_data["First Name"] == x]
temp_df
```

[7]:
```
     First Name Last Name  Draft Year  Years of Service
3747     Bryant     Stith        1992                10
4125     Bryant    Reeves        1995                 6
```

[8]:
```
chain = chain + " " + temp_df.iloc[0]["Last Name"]
seen_list = [temp_df.iloc[0]["Last Name"]]
print(chain)
print(seen_list)
```

```
Kobe Bryant Stith
['Stith']
```

[9]:
```
x = chain[chain.rfind(" ") + 1:]
temp_df = names_data[names_data["First Name"] == x]
temp_df
```

[9]:
```
Empty DataFrame
Columns: [First Name, Last Name, Draft Year, Years of Service]
Index: []
```

[10]:
```
chain = chain[:chain.rfind(" ")]
x = chain[chain.rfind(" ") + 1:]
print(chain)
print(x)
```

```
Kobe Bryant
Bryant
```

[11]:
```
temp_df = names_data[(names_data["First Name"] == x) & (~names_data["Last␣
 ↪Name"].isin(seen_list))]
```

[12]:
```
chain = chain + " " + temp_df.iloc[0]["Last Name"]
seen_list.append(temp_df.iloc[0]["Last Name"])
print(chain)
print(seen_list)
```

```
Kobe Bryant Reeves
['Stith', 'Reeves']
```

[13]:
```
x = chain[chain.rfind(" ") + 1:]
temp_df = names_data[names_data["First Name"] == x]
temp_df
```

[13]:
```
Empty DataFrame
Columns: [First Name, Last Name, Draft Year, Years of Service]
Index: []
```

[14]:
```
chain = chain[:chain.rfind(" ")]
x = chain[chain.rfind(" ") + 1:]
print(chain)
print(x)
```

```
Kobe Bryant
Bryant
```

[15]:
```
chain[:chain.find(" ")]
```

[15]:
```
'Kobe'
```

[16]:
```
temp_df = names_data[(names_data["First Name"] == x) & (~names_data["Last␣
 ↪Name"].isin(seen_list))]
temp_df
```

[16]:
```
Empty DataFrame
Columns: [First Name, Last Name, Draft Year, Years of Service]
Index: []
```

1.2 Methodology

I want to perform a depth-first search iterating through all the possible name chains from a starting name. The program will only be ran on a subset of the dataset where it's possible for a name chain to start (the last name is contained in list of first names).

At first, I created a recursive function to call itself to iterate through the possible name chains. However, python does not handle recursion well and is computationally expensive so it was changed to an iterative process. We keep track of all the names we've seen at a specific branch using nested lists. The function will return the longest chain.

For our initial analysis, we will include suffixes ("Jr.", "Sr.", "III", etc.) essentially disqualifying any player with a suffix. We will revisit this for the second part of the question.

[17]:
```
# function to determine if the current name chain is larger than our current␣
 ↪longest
def is_max(chain, tokens, max_chain, len_max):
    if len(tokens) > len_max:
        max_chain = chain
        len_max = len(tokens)
    return max_chain, len_max
```

[18]:
```
#logic to create list of names we've already seen
def checked_names(seen_list, depth, chained_name):
    if depth == len(seen_list):
            seen_list.insert(len(seen_list), [chained_name])
    elif depth<len(seen_list) and len(seen_list) > 0:
        seen_list[depth].append(chained_name)
    return seen_list
```

[19]:
```
def chain_checker(df, player_first, player_last, seen_list, chain, max_chain,␣
 ↪len_max, depth):
    while True:
        # break special cases
        if player_first==player_last:
            return chain + " " + player_last, 3
            break
        # pop those names to allow them to be used by others
        if abs(depth-len(seen_list)) == 3:
            seen_list.pop()
        # creating subset of names to link onto chain
        if abs(depth-len(seen_list)) == 1:
            temp_df = df[(df["First Name"] == player_last)]
        else:
            temp_df = df[(df["First Name"] == player_last) & (~df["Last Name"].
 ↪isin(seen_list[depth+1]))]
        tokens = chain.split(" ")
        max_chain, len_max = is_max(chain, tokens, max_chain, len_max)
        # returning to base case then break out of loop and return the max␣
 ↪chain
        if temp_df.empty and depth == -1:
            return max_chain, len_max
            break
        # chop off the last name of the link and decrease depth by 1
        elif temp_df.empty:
            chain = chain[:chain.rfind(" ")]
            depth -= 1
            (player_first, player_last) = (chain[:chain.rfind(" ")],␣
 ↪chain[chain.rfind(" ") + 1:])
            continue
            break
        # add a name on the chain and increase the depth by 1
        else:
            chain = chain + " " + temp_df.iloc[0]["Last Name"]
            chained_name = chain[chain.rfind(" ") + 1:]
            depth += 1
            checked_names(seen_list, depth, chained_name)
            (player_first, player_last) = (player_last, chained_name)
            continue
            break
```

Spot checking certain special cases

[20]:
```
names_data.iloc[348]
```

[20]:
```
First Name          Booker
Last Name           Booker
Draft Year               0
Years of Service         2
Name: 348, dtype: object
```

[21]:
```
chain_checker(names_data, names_data.iloc[348]["First Name"], names_data.
 ↪iloc[348]["Last Name"], \
             [], names_data.iloc[348]["First Name"] + " " + names_data.
 ↪iloc[348]["Last Name"], \
             names_data.iloc[348]["First Name"] + " " + names_data.
 ↪iloc[348]["Last Name"], 2, -1)
```

[21]:
```
('Booker Booker Booker', 3)
```

[22]:
```
chain_checker(names_data, names_data.iloc[3844]["First Name"], names_data.
 ↪iloc[3844]["Last Name"], \
             [], names_data.iloc[3844]["First Name"] + " " + names_data.
 ↪iloc[3844]["Last Name"], \
             names_data.iloc[3844]["First Name"] + " " + names_data.
 ↪iloc[3844]["Last Name"], 2, -1)
```

[22]:
```
('Kobe Bryant Stith', 3)
```

Creating our list of potential starting names

[23]:
```
common_list = names_data[names_data["Last Name"].isin(names_data["First Name"])]
```

[24]:
```
common_list[common_list["Last Name"] == "Paul"]
```

[24]:
```
     First Name Last Name  Draft Year  Years of Service
872     Brandon      Paul        2013                 1
1377      Chris      Paul        2005                14
```

[25]:
```
common_list.shape
```

[25]:
```
(656, 4)
```

[26]:
```
common_list.index
```

[26]:
```
Int64Index([   6,   18,   21,   23,   25,   33,   52,   53,   56,   68,
            ...
            4706, 4716, 4721, 4726, 4729, 4734, 4737, 4748, 4761, 4764],
           dtype='int64', length=656)
```

[27]:
```
longest = ""
len_long = 3
```

```
for i in common_list.index:
    #print(i)
    temp_chain, len_chain = chain_checker(names_data, names_data.iloc[i]["First␣
↪Name"], names_data.iloc[i]["Last Name"], \
              [], names_data.iloc[i]["First Name"] + " " + names_data.
↪iloc[i]["Last Name"], \
              names_data.iloc[i]["First Name"] + " " + names_data.iloc[i]["Last␣
↪Name"], 2, -1)
    if len_chain > len_long:
        longest = temp_chain
        len_long = len_chain
print(longest, len_long)
```

```
Ronnie Lester Conner Henry James Thomas Jordan Mickey Dillard Crocker 10
```

1.3 Removing Suffixes

I first searched for last names with a space in them to identify potential suffixes. Then, created logic to split up the name and remove the suffix without affecting last names with two names such as “Van Exel”. Finally, I performed the same search I used on the original dataset.

[28]:
```
pd.set_option('display.max_rows', None)
names_data[names_data["Last Name"].str.contains(" ")]
```

[28]:
```
     First Name        Last Name  Draft Year  Years of Service
121        Nick         Van Exel        1993                13
139       Jaren      Jackson Jr.        2018                 2
148     Charles        Brown Jr.        2019                 1
181      Marcus       Morris Sr.        2011                 8
275     Michael       Porter Jr.        2018                 1
821        Wade       Baldwin IV        2016                 3
895       Keith         Van Horn        1997                 9
1059     Lonnie        Walker IV        2018                 2
1084    Michael       Porter Jr.        2018                 1
1117      Nando          De Colo        2009                 2
1162       Zach      Norvell Jr.        2019                 1
1180     Lonnie        Walker IV        2018                 2
1369    Derrick       Walton Jr.        2017                 2
1382       Matt     Williams Jr.        2017                 1
1404        Luc     Mbah a Moute        2008                10
1456     Danuel        House Jr.        2016                 4
1459        Tom     Van Arsdale            0                12
1471      Vinny       Del Negro         1988                12
1483       Zach      Norvell Jr.        2019                 1
1559     Dennis        Smith Jr.        2017                 3
1657      Brian         Bowen II        2019                 1
1677       Dick     Van Arsdale            0                12
1706     Marvin       Bagley III        2018                 2
```

1745	James	Webb III	2016	1
1770	Gary	Payton II	2016	4
1799	Kevin	Porter Jr.	2019	1
1821	Robert	Williams III	2018	2
1842	James	Ennis III	2013	5
2151	Tim	Hardaway Jr.	2013	6
2179	Gary	Trent Jr.	2018	2
2182	Kevin	Knox II	2018	2
2209	Casper	Ware Jr.	2012	1
2339	Norm	Van Lier	0	10
2359	Log	Vander Velden	0	1
2423	Kevin	Knox II	2018	2
2563	Walt	Lemon Jr.	2014	2
2647	Melvin	Frazier Jr.	2018	2
2735	Gary	Trent Jr.	2018	2
2851	Wendell	Carter Jr.	2018	2
2852	Charles	Brown Jr.	2019	1
2887	Brian	Bowen II	2019	1
2950	Wade	Baldwin IV	2016	3
2951	Whitey	Von Nieda	0	1
3035	Harry	Giles III	2017	2
3038	Kevin	Porter Jr.	2019	1
3044	Roger	Mason Jr.	2002	10
3164	Larry	Nance Jr.	2015	4
3198	Kevin	Porter Jr.	2019	1
3360	Jan	Van Breda Kolff	0	7
3369	Marvin	Bagley III	2018	2
3392	Kelly	Oubre Jr.	2015	4
3477	Troy	Brown Jr.	2018	2
3636	Glenn	Robinson III	2014	5
3646	Troy	Brown Jr.	2018	2
3751	Metta	World Peace	1999	16
3852	Andrew	White III	2017	1
3894	Butch	Van Breda Kolff	0	4
3922	Derrick	Jones Jr.	2016	4
3980	Brian	Bowen II	2019	1
4115	Walt	Lemon Jr.	2014	2
4123	Robert	Williams III	2018	2
4282	Otto	Porter Jr.	2013	6
4314	Perry	Jones III	2012	3
4350	Larry	Drew II	2013	2
4466	John	Lucas III	2005	8
4486	Johnny	O'Bryant III	2014	3
4568	Zach	Norvell Jr.	2019	1
4608	Jaren	Jackson Jr.	2018	2
4685	George H.	Bon Salle	0	1

[29]:
```
pd.set_option('display.max_rows', 15)
names_data[names_data["Last Name"].str.endswith(".") | names_data["Last Name"].
 ↪str.endswith("I") | names_data["Last Name"].str.endswith("V")]
```

[29]:
```
     First Name     Last Name  Draft Year  Years of Service
139       Jaren   Jackson Jr.        2018                 2
148     Charles     Brown Jr.        2019                 1
181      Marcus    Morris Sr.        2011                 8
275     Michael    Porter Jr.        2018                 1
821        Wade    Baldwin IV        2016                 3
...         ...           ...         ...               ...
4350      Larry       Drew II        2013                 2
4466       John     Lucas III        2005                 8
4486     Johnny  O'Bryant III        2014                 3
4568       Zach   Norvell Jr.        2019                 1
4608      Jaren   Jackson Jr.        2018                 2

[55 rows x 4 columns]
```

[30]:
```
suffixes = list(names_data[names_data["Last Name"].str.endswith(".") | \
                           names_data["Last Name"].str.endswith("I") | \
                           names_data["Last Name"].str.endswith("V")]["Last␣
 ↪Name"].str.split(" ").str[1].unique())
suffixes
```

[30]:
```
['Jr.', 'Sr.', 'IV', 'II', 'III']
```

[31]:
```
names_data[names_data["Last Name"].str.endswith(".") \
           | names_data["Last Name"].str.endswith("I") \
           | names_data["Last Name"].str.endswith("V")]["Last Name"].str.
 ↪split(" ").str[0]
```

[31]:
```
139      Jackson
148        Brown
181       Morris
275       Porter
821      Baldwin
          ...
4350        Drew
4466       Lucas
4486    O'Bryant
4568     Norvell
4608     Jackson
Name: Last Name, Length: 55, dtype: object
```

[32]:
```
f = lambda x: ' '.join([item for item in x.split() if item not in suffixes])
no_suffix_df = names_data
no_suffix_df["Last Name"] = names_data["Last Name"].apply(f)
no_suffix_df.iloc[[121, 139, 821, 895, 4350]]
```

[32]:
```
     First Name Last Name  Draft Year  Years of Service
121        Nick   Van Exel        1993                13
139       Jaren    Jackson        2018                 2
821        Wade    Baldwin        2016                 3
895       Keith   Van Horn        1997                 9
4350      Larry       Drew        2013                 2
```

[33]:
```
no_suffix_common_list = no_suffix_df[no_suffix_df["Last Name"].
 ↪isin(no_suffix_df["First Name"])]
```

[34]:
```
no_suffix_common_list.shape
```

[34]:
```
(673, 4)
```

[35]:
```
pd.set_option('display.max_rows', None)
no_suffix_common_list[no_suffix_common_list["Last Name"] == "Jackson"]
```

[35]:
```
     First Name Last Name  Draft Year  Years of Service
139       Jaren   Jackson        2018                 2
170       Myron   Jackson           0                 1
273       Tracy   Jackson           0                 3
533       Ralph   Jackson           0                 1
537     Michael   Jackson           0                 3
569     Stephen   Jackson        1997                14
615     Stanley   Jackson           0                 1
1287       Luke   Jackson        2004                 4
1480       Tony   Jackson           0                 1
1488      Jaren   Jackson           0                12
1520       Mark   Jackson        1987                17
1599         Al   Jackson           0                 1
1920    Darnell   Jackson        2008                 3
2304     Cedric   Jackson        2009                 1
2319     Reggie   Jackson        2011                 8
2367     Justin   Jackson        2017                 3
2487  Demetrius   Jackson        2016                 3
2589       Phil   Jackson           0                12
2665       Greg   Jackson           0                 1
2793       Josh   Jackson        2017                 3
2889    Lucious   Jackson           0                 8
3125      Frank   Jackson        2017                 2
3202    Wardell   Jackson           0                 1
3248     Pierre   Jackson        2013                 1
3832       Marc   Jackson        1997                 7
3895    Randall   Jackson           0                 0
3964      Aaron   Jackson           0                 1
4141      Bobby   Jackson        1997                12
4341        Jim   Jackson        1992                14
4589   Jermaine   Jackson           0                 5
4608      Jaren   Jackson        2018                 2
```

```
[36]: pd.set_option('display.max_rows', 15)
longest = ""
len_long = 3
for i in no_suffix_common_list.index:
    temp_chain, len_chain = chain_checker(no_suffix_df, no_suffix_df.
 ↪iloc[i]["First Name"], no_suffix_df.iloc[i]["Last Name"], \
              [], no_suffix_df.iloc[i]["First Name"] + " " + no_suffix_df.
 ↪iloc[i]["Last Name"], \
              no_suffix_df.iloc[i]["First Name"] + " " + no_suffix_df.
 ↪iloc[i]["Last Name"], 2, -1)
    if len_chain > len_long:
        longest = temp_chain
        len_long = len_chain
print(longest, len_long)
```

```
Ronnie Lester Conner Henry James Thomas Jordan Mickey Dillard Crocker 10
```

1.4 Hyphenated Names

I began by identifying hyphenated names. Then, I used the method listed out in this article to "explode", or split, the name into multiple rows and attached it to the end of the original dataset.

```
[37]: test_hyphen = names_data[names_data["Last Name"].str.contains("-")]
test_hyphen.head()
```

```
[37]:     First Name        Last Name  Draft Year  Years of Service
     11      Shareef      Abdur-Rahim        1996                12
     12         Wang          Zhi-zhi        1999                 5
     109   DeVaughn   Akoon-Purcell        2016                 1
     247     Michael  Kidd-Gilchrist        2012                 7
     466       Talen   Horton-Tucker        2019                 1
```

```
[38]: new_df = pd.DataFrame(test_hyphen["Last Name"].str.split('-').tolist(),␣
 ↪index=test_hyphen["First Name"]).stack()
```

```
[39]: new_df = new_df.reset_index([0, "First Name"])
```

```
[40]: new_df.columns = ['First Name', 'Last Name']
```

```
[41]: new_df.head()
```

```
[41]:   First Name Last Name
     0    Shareef     Abdur
     1    Shareef     Rahim
     2       Wang       Zhi
     3       Wang       zhi
     4   DeVaughn     Akoon
```

```
[42]: new_df = new_df.merge(test_hyphen[["First Name", "Draft Year", "Years of␣
 ↪Service"]], on = "First Name")
```

[43]:
```
hyphen_names = names_data.append(new_df, ignore_index = True)
hyphen_names.tail()
```

[43]:
```
     First Name Last Name  Draft Year  Years of Service
4871      Tariq     Wahad        1997                 6
4872     Xavier    Rathan        2017                 1
4873     Xavier     Mayes        2017                 1
4874     Willie    Cauley        2015                 4
4875     Willie     Stein        2015                 4
```

[44]:
```
hyphen_common_list = hyphen_names[hyphen_names["Last Name"].
 ↪isin(hyphen_names["First Name"])]
```

[45]:
```
hyphen_common_list.shape
```

[45]:
```
(687, 4)
```

[46]:
```
longest = ""
len_long = 3
for i in hyphen_common_list.index:
    temp_chain, len_chain = chain_checker(hyphen_names, hyphen_names.
 ↪iloc[i]["First Name"], hyphen_names.iloc[i]["Last Name"], \
                  [], hyphen_names.iloc[i]["First Name"] + " " + hyphen_names.
 ↪iloc[i]["Last Name"], \
                  hyphen_names.iloc[i]["First Name"] + " " + hyphen_names.
 ↪iloc[i]["Last Name"], 2, -1)
    if len_chain > len_long:
        longest = temp_chain
        len_long = len_chain
print(longest, len_long)
```

```
Ronnie Lester Conner Henry James Thomas Jordan Mickey Dillard Crocker 10
```

1.5 Reverse Names

I simply took the first and last name columns and added them to the end of the other. In this case, all names become possible matches to themselves.

[47]:
```
cols = ["Last Name", "First Name", "Draft Year", "Years of Service"]
reverse_names = hyphen_names[cols]
reverse_names.columns = ["First Name", "Last Name", "Draft Year", "Years of␣
 ↪Service"]
```

[48]:
```
reverse_names_df = hyphen_names.append(reverse_names, ignore_index = True)
```

[49]:
```
reverse_names_df.tail()
```

[49]:
```
     First Name Last Name  Draft Year  Years of Service
9747      Wahad     Tariq        1997                 6
9748     Rathan    Xavier        2017                 1
9749      Mayes    Xavier        2017                 1
9750     Cauley    Willie        2015                 4
```

```
9751      Stein     Willie        2015                    4
```

[50]:
```
reverse_common_list = reverse_names_df[reverse_names_df["Last Name"].
 ↪isin(reverse_names_df["First Name"])]
```

[51]:
```
reverse_common_list.shape
```

[51]:
```
(9752, 4)
```

[52]:
```
reverse_common_list.index
```

[52]:
```
Int64Index([   0,    1,    2,    3,    4,    5,    6,    7,    8,    9,
            ...
            9742, 9743, 9744, 9745, 9746, 9747, 9748, 9749, 9750, 9751],
           dtype='int64', length=9752)
```

[53]:
```
# longest = ""
# len_long = 3
# for i in reverse_common_list.index:
#     print(i)
#     temp_chain, len_chain = chain_checker(reverse_names_df, reverse_names_df.
 ↪iloc[i]["First Name"], reverse_names_df.iloc[i]["Last Name"], \
#               [], reverse_names_df.iloc[i]["First Name"] + " " +␣
 ↪reverse_names_df.iloc[i]["Last Name"], \
#               reverse_names_df.iloc[i]["First Name"] + " " + reverse_names_df.
 ↪iloc[i]["Last Name"], 3, -1)
#     if len_chain > len_long:
#         longest = temp_chain
#         len_long = len_chain
# print(longest, len_long)
```

In tweaking too many things, I broke the functionality for this case however I previously found that the longest chain was Brewer Ronnie Lester Conner Henry James Thomas Jordan Mickey Dillard Crocker

2 Nonzero Draft Year

My thought process was to start with Vince Carter and go backwards seeing who had the longest career in the year he started. After finding that player, repeat until I got back to 1955.

[54]:
```
names_data.head()
```

[54]:
```
  First Name Last Name  Draft Year  Years of Service
0      Larry     Jones           0                 2
1     Darren      Daye           0                 5
2       Theo   Ratliff        1995                16
3    Antonis    Fotsis        2001                 1
4       Bill    Buntin           0                 1
```

[55]:
```
nonzero_draft = names_data[names_data["Draft Year"] != 0].reset_index(drop =␣
 ↪True)
```

[56]:
```
nonzero_draft.head()
```

[56]:
```
  First Name  Last Name  Draft Year  Years of Service
0       Theo    Ratliff        1995                16
1    Antonis     Fotsis        2001                 1
2       Alex  Stepheson        2011                 0
3     Hilton  Armstrong        2006                 6
4        Rob       Kurz        2008                 1
```

[57]:
```
nonzero_draft['Career End'] = nonzero_draft.loc[:,['Draft Year','Years of␣
↪Service']].sum(axis=1)
nonzero_draft.head()
```

[57]:
```
  First Name  Last Name  Draft Year  Years of Service  Career End
0       Theo    Ratliff        1995                16        2011
1    Antonis     Fotsis        2001                 1        2002
2       Alex  Stepheson        2011                 0        2011
3     Hilton  Armstrong        2006                 6        2012
4        Rob       Kurz        2008                 1        2009
```

[58]:
```
nonzero_draft[nonzero_draft["Draft Year"] == 1955]
```

[58]:
```
    First Name Last Name  Draft Year  Years of Service  Career End
924    Maurice    Stokes        1955                 3        1958
```

[59]:
```
nonzero_draft[(nonzero_draft["Draft Year"] >= 1955) & (nonzero_draft["Draft␣
↪Year"] <= 1958)]
```

[59]:
```
    First Name Last Name  Draft Year  Years of Service  Career End
924    Maurice    Stokes        1955                 3        1958
```

[60]:
```
nonzero_draft.sort_values(by = ["Draft Year", "Years of Service"], ascending =␣
↪[False, False]).head(10)
```

[60]:
```
    First Name    Last Name  Draft Year  Years of Service  Career End
29      Daniel     Gafford        2019                 1        2020
43      DaQuan    Jeffries        2019                 1        2020
67      Rayjon      Tucker        2019                 1        2020
68     Charles       Brown        2019                 1        2020
104       Eric    Paschall        2019                 1        2020
125      Tyler        Cook        2019                 1        2020
139       Amir      Coffey        2019                 1        2020
159    Admiral   Schofield        2019                 1        2020
160  Quinndary  Weatherspoon        2019                 1        2020
168      Chris     Clemons        2019                 1        2020
```

[61]:
```
nonzero_draft.sort_values(by = ["Years of Service"], ascending = False)
```

[61]:
```
     First Name Last Name  Draft Year  Years of Service  Career End
1503     Robert    Parish        1976                21        1997
1012      Vince    Carter        1998                21        2019
1045      Kevin    Willis        1984                21        2005
```

```
1171        Dirk  Nowitzki        1998                  20        2018
1446       Kevin   Garnett        1995                  20        2015
...          ...       ...         ...                 ...         ...
1651     Stanton      Kidd        2015                   0        2015
1610       Sasha      Kaun        2008                   0        2008
2377       Jacob    Pullen        2011                   0        2011
452      Michael  McDonald        1995                   0        1995
484     Michelle      Snow        2002                   0        2002

[2406 rows x 5 columns]
```

[62]:
```
nonzero_draft[(nonzero_draft["Career End"] == 2019)].sort_values(by = "Years of␣
→Service", ascending = False)
```

[62]:
```
     First Name   Last Name  Draft Year  Years of Service  Career End
1012      Vince      Carter        1998                21        2019
896       Tyson    Chandler        2001                18        2019
663        Kyle      Korver        2003                16        2019
430      LeBron       James        2003                16        2019
192     Carmelo     Anthony        2003                16        2019
...         ...         ...         ...               ...         ...
2161    Haywood   Highsmith        2018                 1        2019
388         Ray    Spalding        2018                 1        2019
1307      Kevin      Hervey        2018                 1        2019
2185     Jordan  McLaughlin        2018                 1        2019
941    Jemerrio       Jones        2018                 1        2019

[290 rows x 5 columns]
```

[63]:
```
nonzero_draft[(nonzero_draft["Career End"] == 1998)].sort_values(by = "Years of␣
→Service", ascending = False)
```

[63]:
```
     First Name Last Name  Draft Year  Years of Service  Career End
137        Rick    Mahorn        1980                18        1998
1199       Buck  Williams        1981                17        1998
1511      Eddie   Johnson        1981                17        1998
1436      Ricky    Pierce        1982                16        1998
1904      Clyde   Drexler        1983                15        1998
...         ...       ...         ...               ...         ...
827         God  Shammgod        1997                 1        1998
2053      Bubba     Wells        1997                 1        1998
720    Korleone     Young        1998                 0        1998
290     DeMarco   Johnson        1998                 0        1998
1125      Tyson   Wheeler        1998                 0        1998

[44 rows x 5 columns]
```

The players in whose careers ended between 1980-1983 did not have the longest careers or were nonexistent, so I continued my search until 1984

[64]:
```
nonzero_draft[(nonzero_draft["Career End"] == 1984)].sort_values(by = "Years of␣
↪Service", ascending = False)
```

[64]:
```
     First Name Last Name  Draft Year  Years of Service  Career End
1931        Bob    Lanier        1970                14        1984
975      Artis   Gilmore        1972                12        1984
```

[65]:
```
nonzero_draft[(nonzero_draft["Career End"] <= 1980) & (nonzero_draft["Career␣
↪End"] >= 1967)].sort_values(by = "Years of Service", ascending = False)
```

[65]:
```
     First Name  Last Name  Draft Year  Years of Service  Career End
588        Nate   Thurmond        1963                14        1977
1054      Oscar  Robertson        1960                14        1974
1277       Earl     Monroe        1967                13        1980
1630       Rick      Barry        1965                10        1975
2233      Steve    Malovic        1978                 1        1979
```

2.1 Longest Chain with overlapping careers

Using the same logic as before with additional logic in place to check if their careers overlapped

[66]:
```
def findnth(string, substring, n):
    parts = string.split(substring, n + 1)
    if len(parts) <= n + 1:
        return -1
    return len(string) - len(parts[-1]) - len(substring)
```

[67]:
```
def chain_checker(df, player_first, player_last, seen_list, chain, max_chain,␣
↪len_max, depth):
    while True:
        if player_first==player_last:
            return chain + " " + player_last, 3
            break
        if abs(depth-len(seen_list)) == 3:
            seen_list.pop()
        if abs(depth-len(seen_list)) == 1:
            temp_df = df[(df["First Name"] == player_last) & (((df["Draft␣
↪Year"] >= \
                (df[(df["First Name"] == player_first) & (df["Last Name"] ==␣
↪player_last)].iloc[0]["Draft Year"])) | \
                (df["Draft Year"] + df["Years of Service"] >= \
                 df[(df["First Name"] == player_first) & (df["Last Name"] ==␣
↪player_last)].iloc[0]["Draft Year"])) & \
                 (df["Career End"] <= \
                df[(df["First Name"] == player_first) & (df["Last Name"] ==␣
↪player_last)].iloc[0]["Career End"]))]
        else:
            temp_df = df[(df["First Name"] == player_last) & \
                         (~df["Last Name"].isin(seen_list[depth+1])) &␣
↪(((df["Draft Year"] >= \
               (df[(df["First Name"] == player_first) & (df["Last Name"] ==␣
↪player_last)].iloc[0]["Draft Year"])) | \
               (df["Draft Year"] + df["Years of Service"] >= \
                df[(df["First Name"] == player_first) & (df["Last Name"] ==␣
↪player_last)].iloc[0]["Draft Year"])) & \
                (df["Career End"] <= \
               df[(df["First Name"] == player_first) & (df["Last Name"] ==␣
↪player_last)].iloc[0]["Career End"]))]
        tokens = chain.split(" ")
        max_chain, len_max = is_max(chain, tokens, max_chain, len_max)
        if temp_df.empty and depth == -1:
            return max_chain, len_max
            break
        elif temp_df.empty:
            chain = chain[:chain.rfind(" ")]
            depth -= 1
            (player_first, player_last) = (chain[findnth(chain, " ", depth) + 1:
↪chain.rfind(" ")], chain[chain.rfind(" ") + 1:])
            continue
            break
        else:
            chain = chain + " " + temp_df.iloc[0]["Last Name"]
            chained_name = chain[chain.rfind(" ") + 1:]
            depth += 1
            checked_names(seen_list, depth, chained_name)
            (player_first, player_last) = (player_last, chained_name)
            continue
            break
```

[68]:
```
chain_checker(nonzero_draft, nonzero_draft.iloc[430]["First Name"],␣
↪nonzero_draft.iloc[430]["Last Name"], \
             [], nonzero_draft.iloc[430]["First Name"] + " " + nonzero_draft.
↪iloc[430]["Last Name"], \
             nonzero_draft.iloc[430]["First Name"] + " " + nonzero_draft.
↪iloc[430]["Last Name"], 2, -1)
```

[68]:
```
('LeBron James Lang', 3)
```

[69]:
```
nonzero_list = nonzero_draft[nonzero_draft["Last Name"].
↪isin(nonzero_draft["First Name"])]
```

[70]:
```
nonzero_list.shape
```

[70]:
```
(266, 5)
```

[71]:
```
nonzero_list.index
```

[71]:
```
Int64Index([  10,   16,   23,   28,   42,   50,   56,   62,   66,   83,
            ...
            2328, 2329, 2338, 2362, 2382, 2386, 2391, 2392, 2398, 2404],
           dtype='int64', length=266)
```

[72]:
```
longest = ""
len_long = 2
for i in nonzero_list.index:
    #print(i)
    temp_chain, len_chain = chain_checker(nonzero_draft, nonzero_draft.
 ↪iloc[i]["First Name"], nonzero_draft.iloc[i]["Last Name"], \
             [], nonzero_draft.iloc[i]["First Name"] + " " + nonzero_draft.
 ↪iloc[i]["Last Name"], \
             nonzero_draft.iloc[i]["First Name"] + " " + nonzero_draft.
 ↪iloc[i]["Last Name"], 2, -1)
    if len_chain > len_long:
        longest = temp_chain
        len_long = len_chain
print(longest, len_long)
```

```
Carmelo Anthony Davis Bertans 4
```

[ ]:
```

```