In [1]:
```
import pandas as pd
import numpy as np
```

## Q.no. 1

### a.

In [2]:
```
df = pd.read_csv('NCI60_data.csv')
df = df.set_index('Unnamed: 0')
df_labels = pd.read_csv('NCI60_labs.csv')
df_labels = df_labels.set_index('Unnamed: 0')
```

### b.

In [3]:
```
df.describe()
```

Out[3]:

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | |
|---|---|---|---|---|---|---|---|---|---|
| **count** | 64.000000 | 64.000000 | 64.000000 | 64.000000 | 64.000000 | 64.000000 | 64.000000 | 64.000000 | 64. |
| **mean** | -0.019063 | -0.027813 | -0.019923 | -0.328673 | 0.026093 | 0.006718 | 0.019687 | -0.023126 | 0. |
| **std** | 0.441332 | 0.757433 | 0.433306 | 1.091905 | 0.485073 | 0.350432 | 0.370683 | 0.338629 | 0. |
| **min** | -1.060000 | -2.190000 | -1.710000 | -2.610000 | -0.825000 | -0.700000 | -0.920000 | -0.705000 | -0. |
| **25%** | -0.372500 | -0.404985 | -0.192485 | -1.322500 | -0.225000 | -0.156250 | -0.246250 | -0.204985 | -0. |
| **50%** | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0. |
| **75%** | 0.310005 | 0.352500 | 0.162490 | 0.692500 | 0.210000 | 0.184995 | 0.247505 | 0.160015 | 0. |
| **max** | 0.940000 | 2.240000 | 1.150000 | 1.500000 | 1.715000 | 1.160000 | 0.940000 | 0.724961 | 0. |

8 rows × 6830 columns

It's clearly evident from above that the numerical values in the dataset are not standardized as neither the mean is 0 for these columns nor is the standard deviation 1. In, other words, the columns have different means and standard deviations, which means that they are not standardised. So let's standardize them:

In [4]:
```
from sklearn import preprocessing
df.loc[:, df.columns!='Unnamed: 0'] = preprocessing.scale(df.loc[:, df.columns!='
```

In [5]:
```
df.describe()
```

Out[5]:

| | 1 | 2 | 3 | 4 | 5 | 6 | |
|---|---|---|---|---|---|---|---|
| **count** | 6.400000e+01 | 64.000000 | 6.400000e+01 | 6.400000e+01 | 6.400000e+01 | 6.400000e+01 | 6.40 |
| **mean** | -8.673617e-18 | 0.000000 | -3.014082e-17 | 3.989864e-17 | -2.428613e-17 | -5.204170e-18 | 2.42 |
| **std** | 1.007905e+00 | 1.007905 | 1.007905e+00 | 1.007905e+00 | 1.007905e+00 | 1.007905e+00 | 1.00 |
| **min** | -2.377270e+00 | -2.877193 | -3.931262e+00 | -2.105826e+00 | -1.768435e+00 | -2.032645e+00 | -2.55 |
| **25%** | -8.071713e-01 | -0.501898 | -4.013951e-01 | -9.173725e-01 | -5.217310e-01 | -4.687243e-01 | -7.23 |
| **50%** | 4.353664e-02 | 0.037011 | 4.634208e-02 | 3.033881e-01 | -5.421676e-02 | -1.932168e-02 | -5.35 |
| **75%** | 7.515195e-01 | 0.506077 | 4.243081e-01 | 9.426144e-01 | 3.821298e-01 | 5.127569e-01 | 6.19 |
| **max** | 2.190290e+00 | 3.017748 | 2.721339e+00 | 1.687994e+00 | 3.509280e+00 | 3.317043e+00 | 2.50 |

8 rows × 6830 columns

## c. i.

In [6]:
```python
cols = []
for i in range(64):
    cols.append("P" + str(i + 1))


from sklearn import decomposition
pca = decomposition.PCA()
df_plot = pd.DataFrame(pca.fit_transform(df), columns = cols, index=df.index)
df_plot
```

Out[6]:

| | P1 | P2 | P3 | P4 | P5 | P6 | P7 | |
|---|---|---|---|---|---|---|---|---|
| **Unnamed: 0** | | | | | | | | |
| **V1** | -19.838042 | -3.555636 | -9.812399 | -0.824246 | 12.609984 | 7.471505 | 14.190730 | -3.1979 |
| **V2** | -23.089215 | -6.441460 | -13.478251 | 5.635308 | 8.035496 | 3.715178 | 10.143225 | -7.2927 |
| **V3** | -27.456114 | -2.465143 | -3.533054 | -1.341673 | 12.564846 | 17.344925 | 10.354857 | -2.6712 |
| **V4** | -42.816801 | 9.768358 | -0.890073 | 3.445043 | 42.269904 | 27.238815 | 17.520642 | -0.5543 |
| **V5** | -55.418530 | 5.198897 | -21.094558 | 15.849712 | 10.443273 | 12.991051 | 12.597895 | 32.5130 |
| **...** | ... | ... | ... | ... | ... | ... | ... | |
| **V60** | -17.996242 | 47.242414 | 5.398941 | -17.405145 | -0.293594 | -7.429801 | -15.572290 | 22.2701 |
| **V61** | -4.415510 | 42.309563 | 8.715183 | -2.805833 | 5.716318 | -3.375639 | -11.918083 | 6.7072 |
| **V62** | -22.966988 | 36.102038 | 18.116579 | -7.579957 | 5.874271 | 0.570517 | -7.703801 | -5.5856 |
| **V63** | -19.176007 | 50.398441 | 4.211153 | -3.041732 | -8.347417 | -3.660122 | -2.023234 | 11.8850 |
| **V64** | -13.232870 | 35.125249 | 3.433554 | -2.174010 | -1.001003 | -4.971377 | -10.267870 | 3.6266 |

64 rows × 64 columns

## c. ii.

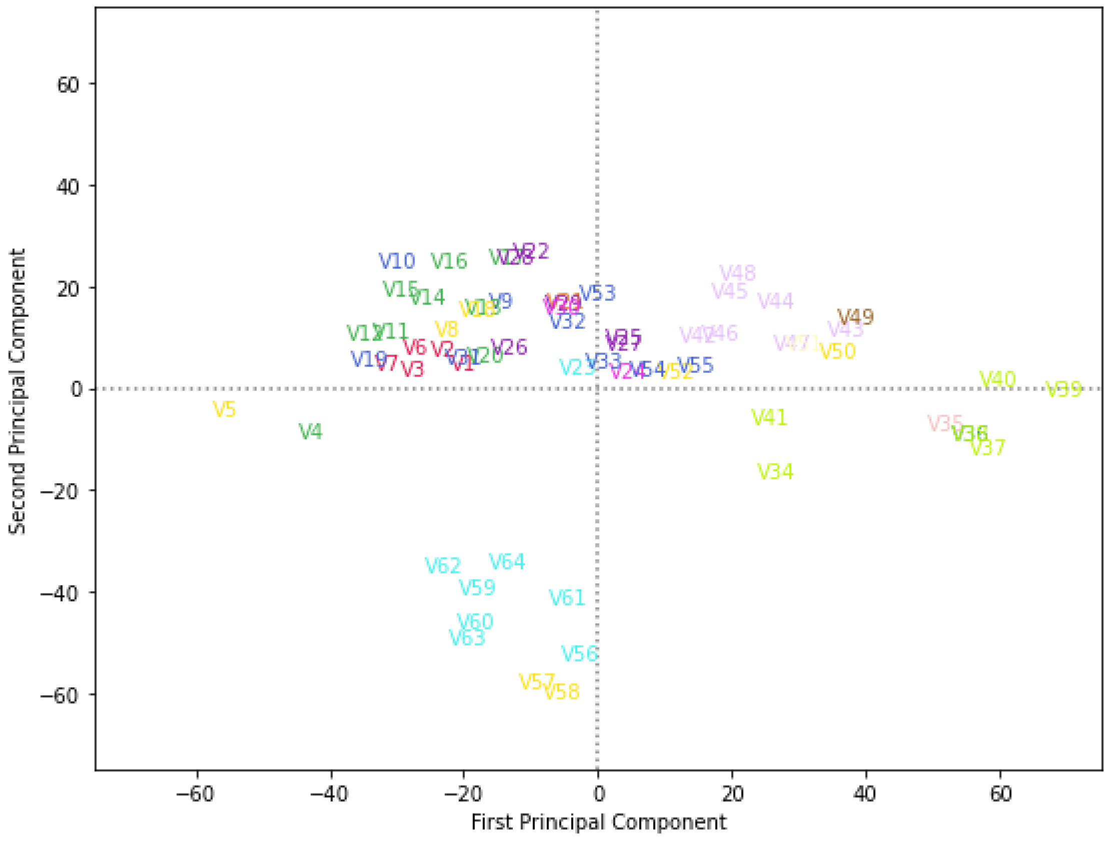
In [7]:
```
df_labels.index = df.index
df_labels['Cancer'] = df_labels['x']
color_map = dict()
color_list = ['#e6194b', '#3cb44b', '#ffe119', '#4363d8', '#f58231', '#911eb4', '
count = 0
for cancer_type in df_labels.x.unique():
    color_map[cancer_type] = color_list[count]
    count = count + 1

df_labels = df_labels.replace({"x": color_map})
df_labels.head(5)
```

Out[7]:

| | x | Cancer |
|---|---|---|
| **Unnamed: 0** | | |
| **V1** | #e6194b | CNS |
| **V2** | #e6194b | CNS |
| **V3** | #e6194b | CNS |
| **V4** | #3cb44b | RENAL |
| **V5** | #ffe119 | BREAST |

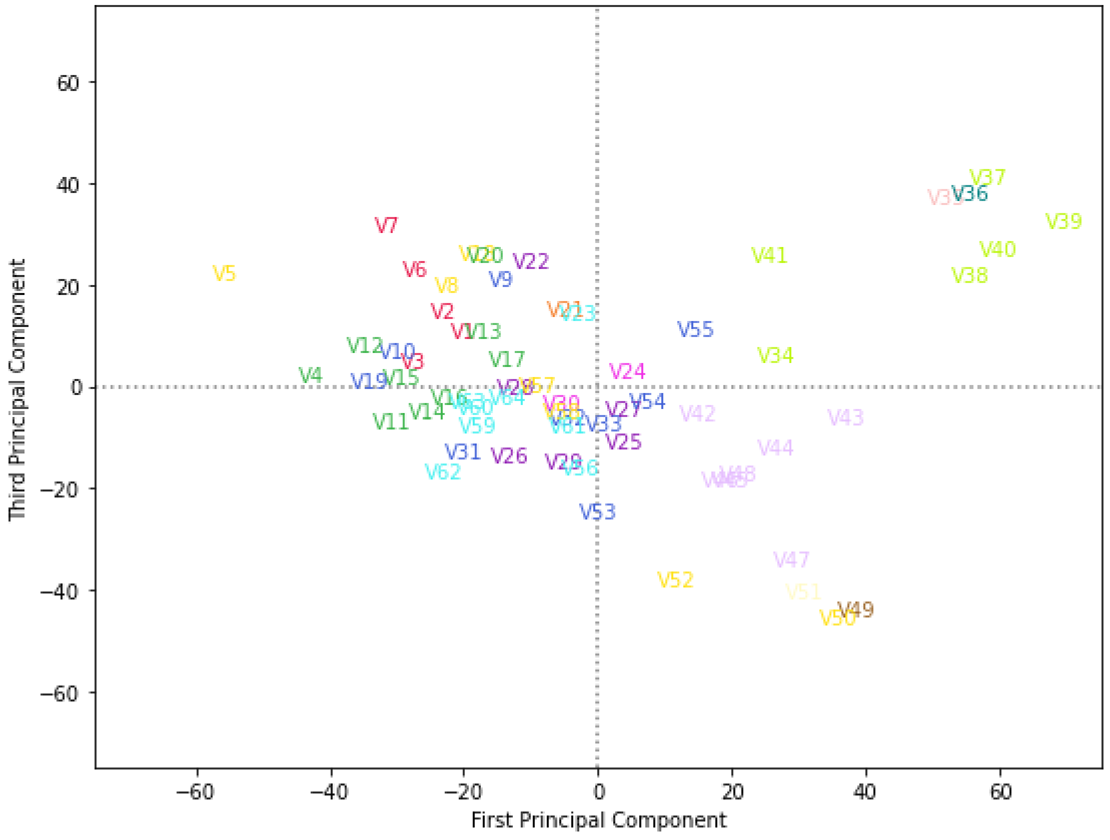
```
In [8]: import matplotlib.pyplot as plt
        fig , ax1 = plt.subplots(figsize=(9,7))

        ax1.set_xlim(-75,75)
        ax1.set_ylim(-75,75)

        # Plot Principal Components 1 and 2
        for i in df_plot.index:
            color = df_labels[df_labels.index == i]['x']
            ax1.annotate(i, (df_plot.P1.loc[i], -df_plot.P2.loc[i]), ha='center', color =

        # Plot reference lines
        ax1.hlines(0,-75,75, linestyles='dotted', colors='grey')
        ax1.vlines(0,-75,75, linestyles='dotted', colors='grey')

        ax1.set_xlabel('First Principal Component')
        ax1.set_ylabel('Second Principal Component')
```

```
Out[8]: Text(0, 0.5, 'Second Principal Component')
```



In [11]:
```
import matplotlib.pyplot as plt
fig , ax1 = plt.subplots(figsize=(9,7))

ax1.set_xlim(-75,75)
ax1.set_ylim(-75,75)

# Plot Principal Components 1 and 3
for i in df_plot.index:
    color = df_labels[df_labels.index == i]['x']
    ax1.annotate(i, (df_plot.P1.loc[i], -df_plot.P3.loc[i]), ha='center', color=

# Plot reference lines
ax1.hlines(0,-75,75, linestyles='dotted', colors='grey')
ax1.vlines(0,-75,75, linestyles='dotted', colors='grey')

ax1.set_xlabel('First Principal Component')
ax1.set_ylabel('Third Principal Component')
```

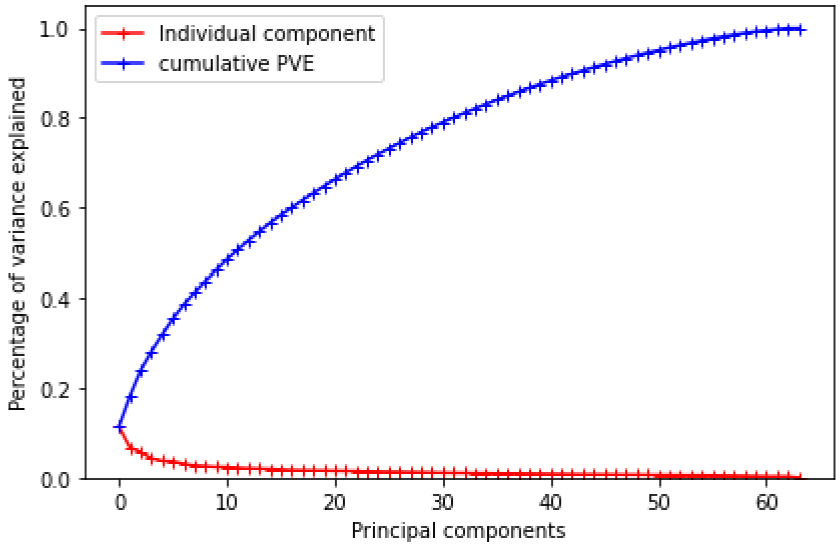
Out[11]:
```
Text(0, 0.5, 'Third Principal Component')
```



In both the biplots, the cancers which are of the same type are clustered together which is evident from the labels(eg, V1, V2, etc) of the same color having similar scores in the plots. As such, we can conclude that the principal components in both the biplot appear to have significantly captured the variation in the data. However, the first biplot with the first two principal components is found to be more distinct in separation of groups of cancers of same types because the first two components combined always capture more variation than the first and the third component combined.

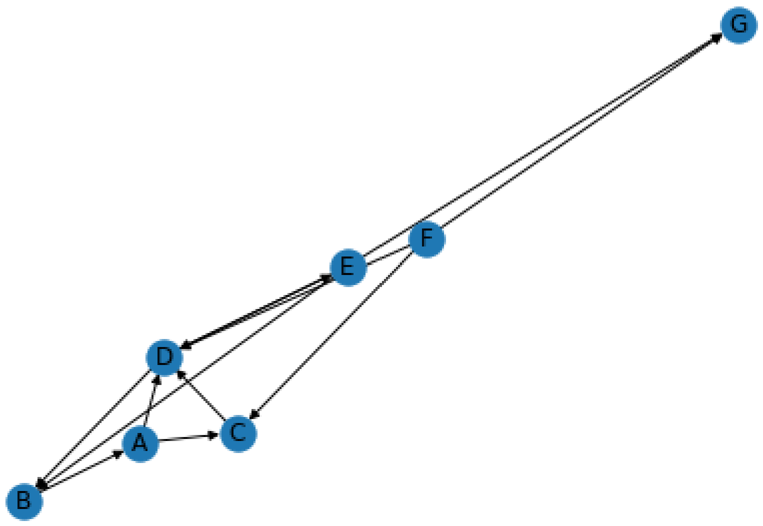
## c. iii.

In [10]:
```
# Instantiate PCA estimator
pca = decomposition.PCA()
# fit : Run PCA
df_PC = pca.fit(df)


# Percentage of Variance Explained (PVE) by each PC:
plt.figure(figsize=(6, 4))
plt.ylim(0, 1.05)

plt.plot(df_PC.explained_variance_ratio_, 'r+-', label='Individual component')
plt.plot(np.cumsum(df_PC.explained_variance_ratio_), 'b+-', label='cumulative PVE

plt.ylabel('Percentage of variance explained')
plt.xlabel('Principal components')
plt.legend(loc='best')
plt.tight_layout()
```



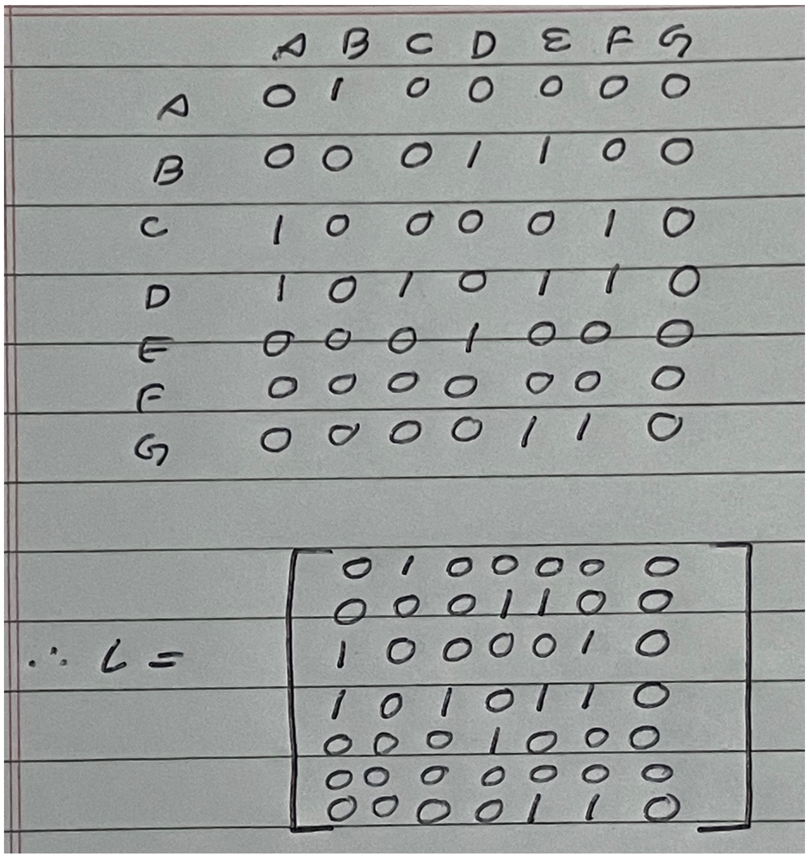
## Q.no. 2

In [11]:
```
import matplotlib.pyplot as plt
import networkx as nx
import numpy as np

G=nx.DiGraph(directed=True)

# a list of nodes:
pages = ["A","B","C","D","E","F","G"]
G.add_nodes_from(pages)


G.add_edges_from([('A','D'), ('A', 'C'), ('B','A'),('C','D'), ('D','B'),('D','E')

nx.draw(G, with_labels = True)
```



In [12]:
```
from IPython import display
display.Image("./link.jpg", width = 400, height = 300)
```

Out[12]:



$$\begin{array}{c|ccccccc} & A & B & C & D & E & F & G \\ \hline A & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ B & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ C & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ D & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ E & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ F & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ G & 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{array}$$

$$\therefore L = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$

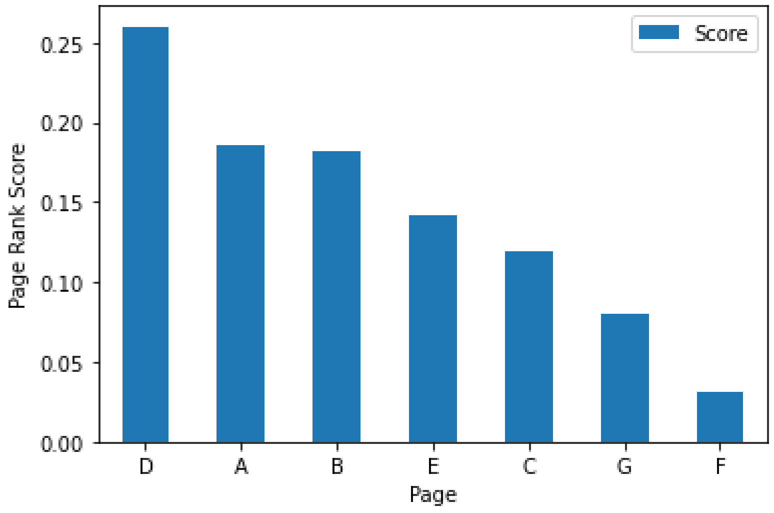
In [13]:
```
prank=nx.pagerank(G, alpha=0.85)
data_items = prank.items()
data_list = list(data_items)
df = pd. DataFrame(data_list, columns = ['Page', 'Score'])
df.head(7)
```

Out[13]:

| | Page | Score |
|---|---|---|
| **0** | A | 0.185800 |
| **1** | B | 0.181928 |
| **2** | C | 0.118956 |
| **3** | D | 0.260235 |
| **4** | E | 0.141762 |
| **5** | F | 0.031162 |
| **6** | G | 0.080157 |

In [14]:
```
df.sort_values(by='Score',ascending=False).plot(kind='bar', x='Page', rot=0)
plt.ylabel("Page Rank Score")
```

Out[14]:
```
Text(0, 0.5, 'Page Rank Score')
```



The pages that come to the top four list are D, A, B, and E