In [9]:
```
import math
import numpy as np
import pandas as pd

from matplotlib import pyplot as plt

%matplotlib inline
```

In [11]:
```
df1 = pd.read_csv("./Udacity¥project2_cvs¥p1.csv")
df2 = pd.read_csv("./Udacity¥project2_cvs¥p2.csv")
df3 = pd.read_csv("./Udacity¥project2_cvs¥p3.csv")
df4 = pd.read_csv("./Udacity¥project2_cvs¥p4.csv")
df_ALL = pd.read_csv("./Udacity¥project2_cvs¥pALL.csv")
```

df1 is the .csv file from problem1, df2 from problem2, def3 from problem3, def4 from problem4 df_ALL is a .csv where all contained in one file

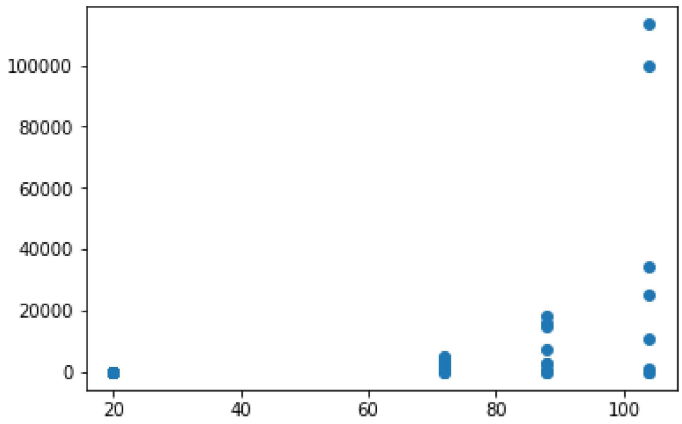
Use a table or chart to analyze the number of nodes expanded against number of actions in the domain

In [12]:
```
df_ALL[['problem','search functions', 'Actions', 'Expansions']].sort_values(by=['Actions'], ascend
ing=True).head()
```

Out[12]:

	problem	search functions	Actions	Expansions
0	problem1	breadth_first_search	20	43
1	problem1	depth_first_graph_search	20	21
2	problem1	uniform_cost_search	20	60
3	problem1	greedy_best_first_graph_search with h_unmet_goals	20	7
4	problem1	greedy_best_first_graph_search with h_pg_levelsum	20	6

In [13]:
```
plt.scatter(df_ALL['Actions'], df_ALL['Expansions'])
plt.show()
```



ANSWER> The more 'Actions', the more 'Expansions'. The range of 'Expansions' values seems getting wider for increasing 'Actions'.

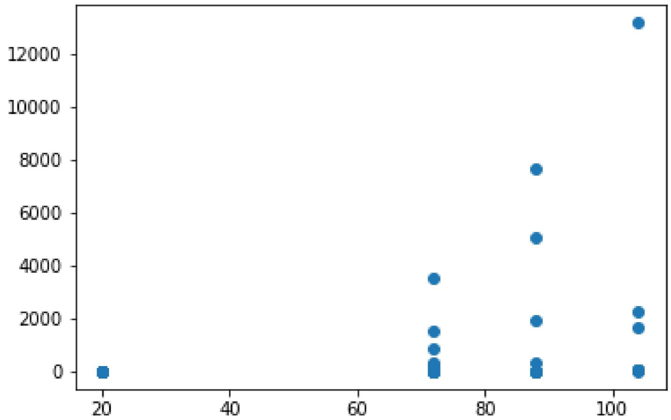
Use a table or chart to analyze the search time against the number of actions in the domain

In [14]:
```
df_ALL[['problem','search functions', 'Actions', 'Time elapsed in seconds']].sort_values(by=['Acti
ons'], ascending=True).head()
```

Out[14]:

	problem	search functions	Actions	Time elapsed in seconds
0	problem1	breadth_first_search	20	0.004595
1	problem1	depth_first_graph_search	20	0.002539
2	problem1	uniform_cost_search	20	0.005909
3	problem1	greedy_best_first_graph_search with h_unmet_goals	20	0.000980
4	problem1	greedy_best_first_graph_search with h_pg_levelsum	20	0.330014

In [15]:
```
plt.scatter(df_ALL['Actions'], df_ALL['Time elapsed in seconds'])
plt.show()
```



ANSWER> The more 'Actions', the more 'Time elapsed..'. The range of 'Time elapsed' values seems getting wider for increasing 'Actions'.

Use a table or chart to analyze the length of the plans returned by each algorithm on all search problems

In [16]:
```
df_ALL.pivot(index='search functions', columns='problem', values ='Plan length')
```

Out[16]:

problem	problem1	problem2	problem3	problem4
search functions				
astar_search with h_pg_levelsum	6.0	9.0	12.0	15.0
astar_search with h_pg_maxlevel	6.0	9.0	12.0	NaN
astar_search with h_pg_setlevel	6.0	9.0	NaN	NaN
astar_search with h_unmet_goals	6.0	9.0	12.0	14.0
breadth_first_search	6.0	9.0	12.0	14.0
depth_first_graph_search	20.0	619.0	392.0	24132.0
greedy_best_first_graph_search with h_pg_levelsum	6.0	9.0	14.0	17.0
greedy_best_first_graph_search with h_pg_maxlevel	8.0	15.0	19.0	23.0
greedy_best_first_graph_search with h_pg_setlevel	6.0	10.0	19.0	NaN
greedy_best_first_graph_search with h_unmet_goals	6.0	9.0	15.0	18.0
uniform_cost_search	6.0	9.0	12.0	14.0

ANSWER> 'depth_first_graph search' is protruding. It is characteristic. About others, as the graphs become more complex, numbers of plans also increase.

Which algorithm or algorithms would be most appropriate for planning in a very restricted domain (i.e., one that has only a few actions) and needs to operate in real time?

In [17]:
```
df1[['search functions', 'Time elapsed in seconds']].sort_values(by=['Time elapsed in seconds'], ascending=True).head(10)
```

Out[17]:

	search functions	Time elapsed in seconds
3	greedy_best_first_graph_search with h_unmet_goals	0.000980
1	depth_first_graph_search	0.002539
0	breadth_first_search	0.004595
7	astar_search with h_unmet_goals	0.005580
2	uniform_cost_search	0.005909
4	greedy_best_first_graph_search with h_pg_levelsum	0.330014
8	astar_search with h_pg_levelsum	0.865519
9	astar_search with h_pg_maxlevel	0.903606
5	greedy_best_first_graph_search with h_pg_maxlevel	1.017564
6	greedy_best_first_graph_search with h_pg_setlevel	1.124558

ANSWER> 1: greedy_best_first_graph_search with h_unmet_goals 2: depth_first_graph_search 3: breadth_first_search

Which algorithm or algorithms would be most appropriate for planning in very large domains (e.g., planning delivery routes for all UPS drivers in the U.S. on a given day)

In [20]:
```
df4[['search functions', 'Time elapsed in seconds']].sort_values(by=['Time elapsed in seconds'], ascending=True).head(10)
```

Out[20]:

	search functions	Time elapsed in seconds
3	greedy_best_first_graph_search with h_unmet_goals	0.036535
4	greedy_best_first_graph_search with h_pg_levelsum	29.354749
6	astar_search with h_unmet_goals	40.914764
0	breadth_first_search	57.509488
2	uniform_cost_search	83.493080
7	astar_search with h_pg_levelsum	1668.088534
1	depth_first_graph_search	2246.683306
5	greedy_best_first_graph_search with h_pg_maxlevel	13149.995510

ANSWER> 1: greedy_best_first_graph_search with h_unmet_goals 2: greedy_best_first_graph_search with h_pg_levelsum 3: astar_search with h_unmet_goals

Which algorithm or algorithms would be most appropriate for planning problems where it is important to find only optimal plans?

ANSWER> greedy_best_first_graph_search with h_unmet_goals, uniform_cost_search or breadth_first_search, astar_search with h_unmet_goals.

==== About 'Actions', 'Expansions', 'Goal Tests', 'New Nodes', 'Plan length', 'Time elapsed in seconds'

In [37]:
```
from pandas.plotting import scatter_matrix
df_SCM = df_ALL.loc[:,['Actions', 'Expansions', 'Goal Tests', 'New Nodes', 'Plan length', 'Time elapsed in seconds']]
df_SCM.corr()
```

Out[37]:

	Actions	Expansions	Goal Tests	New Nodes	Plan length	Time elapsed in seconds
Actions	1.000000	0.387719	0.383388	0.387874	0.185040	0.300566
Expansions	0.387719	1.000000	0.997057	0.999798	0.105796	0.009645
Goal Tests	0.383388	0.997057	1.000000	0.997042	0.095992	0.001692
New Nodes	0.387874	0.999798	0.997042	1.000000	0.100115	0.017712
Plan length	0.185040	0.105796	0.095992	0.100115	1.000000	0.080513
Time elapsed in seconds	0.300566	0.009645	0.001692	0.017712	0.080513	1.000000

Goal Tests-Expansions, Goal Test-New Nodes, New Nodes-Expansions ; These three combinations are strongly correlated. Plan length and Time elapsed seems seem to be uncorrelated.

Since Actions are constant while each problem (ex. problem1.Actions are 20), the ideal of the better search will be 'with less time and plans, less Expansions, Goal Tests, New Nodes' for goal.