To compile the code:
1) put makefile, pagerank123.cpp, pagerank4.cpp, pagerank5.cpp in your workspace
2) type “make” in terminal, make sure the Linux system have g++ compiler
3) you should find pagerank123.x, pagerank4.x, pagerank5.x, in your workspace

To run the code:
1) for question 1-3, type ./pagerank123.cpp in terminal
2) for question 4, type ./pagerank4.cpp in terminal
3) for question 5, type ./pagerank5.cpp in terminal

Question 1
when running the code, input 10 as the matrix size, following output appeared:

Please input the size of matrix:
10
Initializing the matrix A;
Matrix generation successful!
The matrix is:
0 0 0 0 0 0 0 0 0 0
1 0 0 1 0 0 0 1 0 1
0 1 0 0 1 1 0 1 0 0
0 0 0 0 0 0 0 0 1 1
0 1 1 0 0 0 0 1 0 0
1 0 0 0 0 0 1 0 1 0
1 0 0 1 1 1 0 1 1 1
0 1 0 1 1 1 0 0 1 1
0 0 0 0 0 0 0 0 0 0
1 1 1 1 0 0 0 1 1 0

The matrix was randomly generated, If There is a column of zero, the matrix will be discarded, until the matrix generation is successful.

Question 2
Output:

The matrix M is:
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.250 0.000 0.000 0.250 0.000 0.000 0.000 0.200 0.000 0.250
0.000 0.250 0.000 0.000 0.333 0.333 0.000 0.200 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.200 0.250
0.000 0.250 0.500 0.000 0.000 0.000 0.000 0.200 0.000 0.000
0.250 0.000 0.000 0.000 0.000 0.000 1.000 0.000 0.200 0.000
0.250 0.000 0.000 0.250 0.333 0.333 0.000 0.200 0.200 0.250
0.000 0.250 0.000 0.250 0.333 0.333 0.000 0.000 0.200 0.250
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.250 0.250 0.500 0.250 0.000 0.000 0.000 0.200 0.200 0.000

The page rank(calculated by method 1) result is:
0.015

0.0806522
0.134289
0.045138
0.114496
0.156795
0.160068
0.148734
0.015
0.129826

Question 3
Output:

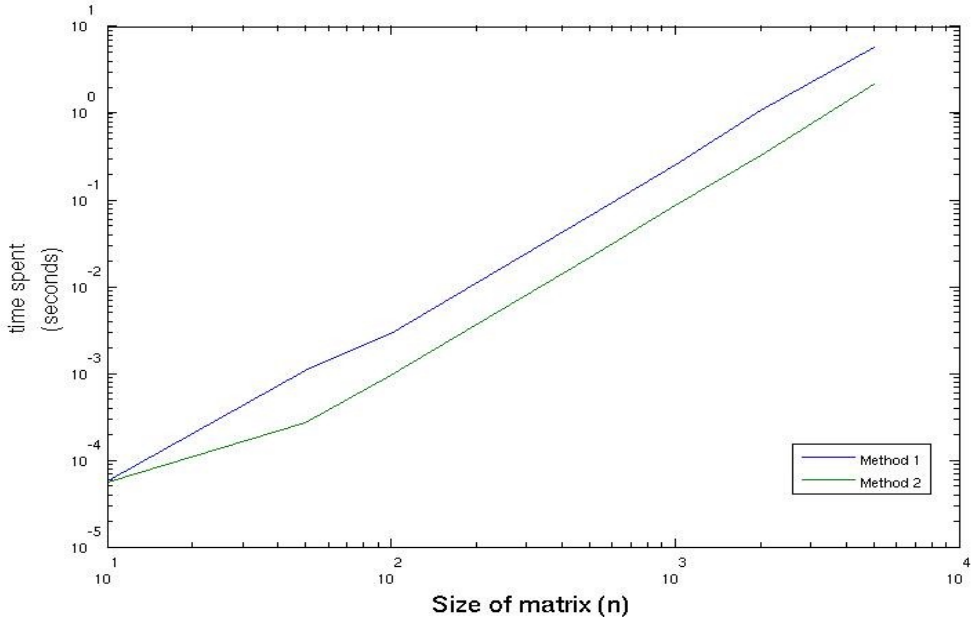
The page rank(calculated by method 2) result is:
0.015
0.0806522
0.134289
0.045138
0.114496
0.156795
0.160068
0.148734
0.015
0.129826

We can see that method 2 and 3 generated the same result, which indicate the successfully implementation of these two method.

Question 4
Output:

| n= | Method 1 time | Method 2 time |
|---|---|---|
| 10 | 0.0000567720 | 0.0000561660 |
| 50 | 0.0010849920 | 0.0002701950 |
| 100 | 0.0028970460 | 0.0009527350 |
| 500 | 0.0650460760 | 0.0217305450 |
| 1000 | 0.2517772040 | 0.0876978560 |
| 2000 | 1.0721262840 | 0.3288987520 |
| 5000 | 5.8098901860 | 2.1834978760 |

The efficient of both method was plotted as the following figure:




Discussion:
From the graph above, we can see that the running time of both method will increase as the size of matrix become larger. The efficiency of Method 1 and 2 do not have very significant difference when n is very small, however, when the size of matrix increase, the difference of efficient between the two method become nontrivial. The efficiency of Method 2 is 2-3 times higher than Method 1. However, these two method do not seems to have difference in order, since the slope of the two lines do not have significant difference when stable.

When stable, the slope of the two lines do not have significant changes, since the two lines keep straight when n is lager than 100. This phenomenon indicates the O(n) complexity of both method(since both axis are logged).

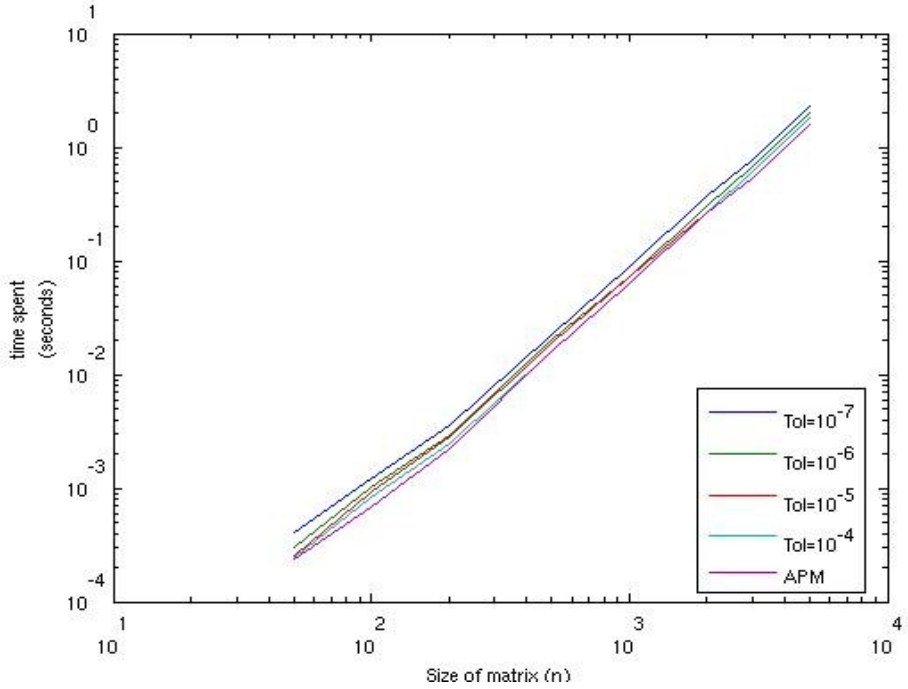
I think the Method 2 is much better since it have higher efficiency.

Question 5:
The most aggressive termination criterion I got is like this:
If the rank 10 websites haven't change (order and page IDs) for more than two iteration, termination will occur.

Output:

| n= | Tol=10^(-7) | Tol=10^(-6) | Tol=10^(-5) | Tol=10^(-4) | Aggressive Power Method |
|---|---|---|---|---|---|
| 50 | 0.0004056860 | 0.0002999100 | 0.0002580500 | 0.0002414330 | 0.0002383030 |
| 100 | 0.0012223650 | 0.0010155940 | 0.0009128790 | 0.0008284861 | 0.0006704390 |
| 200 | 0.0035090140 | 0.0028900020 | 0.0028280540 | 0.0024788520 | 0.0022069350 |
| 500 | 0.0222344610 | 0.0197417620 | 0.0188388590 | 0.0159563620 | 0.0158335940 |
| 1000 | 0.0898555320 | 0.0720767390 | 0.0720848150 | 0.0641959530 | 0.0642593730 |
| 2000 | 0.3688002800 | 0.2973763210 | 0.2661493390 | 0.2661079100 | 0.2661847910 |
| 3000 | 0.7825886279 | 0.6852184370 | 0.6144258410 | 0.6143517220 | 0.5429289320 |

5000 2.2695438100 1.9923332779 1.7942580300 1.7905002510 1.5981287500




From the output and the figure above, we can see that the Aggressive Power Method(APM) have highest efficiency compared with ordinary Power Method (when tolerance was set as 0.0001, 0.00001, 0.000001, 0.0000001). The efficiency of Aggressive Power Method is even higher when using efficient sorting algorithm to find top ten(The sorting I wrought is insertion sort ).

This method have been tested multiple times with random generated matrix:
In order to check if the Aggressive Power Method can find top ten website correctly, the top ten website generated by Aggressive Power Method was compared with ordinary Power Method:

Top ten ID:
when n= 50:
ID of top ten webpage was printed out:
Tol=10^(-7): 35 18 3 23 7 24 5 10 37 22
Tol=10^(-6): 35 18 3 23 7 24 5 10 37 22
Tol=10^(-5): 35 18 3 23 7 24 5 10 37 22
Tol=10^(-4): 35 18 3 23 7 24 5 10 37 22
Aggressive method: 35 18 3 23 7 24 5 10 37 22
when n= 100:
ID of top ten webpage was printed out:
Tol=10^(-7): 63 95 53 29 11 99 92 85 1 98
Tol=10^(-6): 63 95 53 29 11 99 92 85 1 98
Tol=10^(-5): 63 95 53 29 11 99 92 85 1 98
Tol=10^(-4): 63 95 53 29 11 99 92 85 1 98
Aggressive method: 63 95 53 29 11 99 92 85 1 98
when n= 200:
ID of top ten webpage was printed out:
Tol=10^(-7): 56 72 185 135 146 121 66 75 195 47
Tol=10^(-6): 56 72 185 135 146 121 66 75 195 47

Tol=10^(-5): 56 72 185 135 146 121 66 75 195 47
Tol=10^(-4): 56 72 185 135 146 121 66 75 195 47
Aggressive method: 56 72 185 135 146 121 66 75 195 47
when n= 500:
ID of top ten webpage was printed out:
Tol=10^(-7): 330 324 121 22 185 372 231 340 226 40
Tol=10^(-6): 330 324 121 22 185 372 231 340 226 40
Tol=10^(-5): 330 324 121 22 185 372 231 340 226 40
Tol=10^(-4): 330 324 121 22 185 372 231 340 226 40
Aggressive method: 330 324 121 22 185 372 231 340 226 40
when n= 1000:
ID of top ten webpage was printed out:
Tol=10^(-7): 631 349 186 594 737 512 437 352 940 577
Tol=10^(-6): 631 349 186 594 737 512 437 352 940 577
Tol=10^(-5): 631 349 186 594 737 512 437 352 940 577
Tol=10^(-4): 631 349 186 594 737 512 437 352 940 577
Aggressive method: 631 349 186 594 737 512 437 352 940 577
when n= 2000:
ID of top ten webpage was printed out:
Tol=10^(-7): 869 176 1453 1676 1987 996 838 515 582 315
Tol=10^(-6): 869 176 1453 1676 1987 996 838 515 582 315
Tol=10^(-5): 869 176 1453 1676 1987 996 838 515 582 315
Tol=10^(-4): 869 176 1453 1676 1987 996 838 515 582 315
Aggressive method: 869 176 1453 1676 1987 996 838 515 582 315
when n= 3000:
ID of top ten webpage was printed out:
Tol=10^(-7): 1195 1262 2479 213 114 1650 2878 111 227 2244
Tol=10^(-6): 1195 1262 2479 213 114 1650 2878 111 227 2244
Tol=10^(-5): 1195 1262 2479 213 114 1650 2878 111 227 2244
Tol=10^(-4): 1195 1262 2479 213 114 1650 2878 111 227 2244
Aggressive method: 1195 1262 2479 213 114 1650 2878 111 227 2244
when n= 5000:
ID of top ten webpage was printed out:
Tol=10^(-7): 1222 1039 229 3701 509 3012 1492 1891 3630 602
Tol=10^(-6): 1222 1039 229 3701 509 3012 1492 1891 3630 602
Tol=10^(-5): 1222 1039 229 3701 509 3012 1492 1891 3630 602
Tol=10^(-4): 1222 1039 229 3701 509 3012 1492 1891 3630 602
Aggressive method: 1222 1039 229 3701 509 3012 1492 1891 3630 602

We can see all of them get the same page rank for top ten web page, which indicate the validation of Aggressive Power Method.

Question 6
The web page I refereed to is “95 SEO Tips and Tricks for Powerful Search Engine Optimization”
URL: http://webdesign.about.com/od/seo/tp/seo_tips_and_tricks.htm

In order to increase your page rank, one of the important methods is to ask other people for links to your web page. By doing so, you may have more other pages pointing to your web page, so your web

page will have higher score in adjacency matrix . Power method will generate a higher score.

Compared with links from non-famous web pages, links form reputable web page can be more helpful, because the reputable web page have more weight, so the linking will add more score onto your web page.

Another interesting trick is to try to get links form .edu and .gov sites. They are considered to be more reliable by search engine, so their link have higher weight.

It is also helpful to keep your page up-to-date and adding new content frequently,  which will also increase the ranking. Also the old page will have higher rank than new page, so keep the web page survive a long time will be helpful to the ranking. Which make sense because the frequently updated web page with  long history are considered more reliable by search engine.