3. Code

```
a = 0.85; % alpha
% a = 0.75;
M = [
    1/11, (1-a)/11, (1-a)/11, (9*a+2)/22, (1-a)/11, (1-a)/11, (1-a)/11, (1-a)/11, (1-a)/11, (1-a)/11, (1-a)/11;
    1/11, (1-a)/11, (10*a+1)/11, (9*a+2)/22, (8*a+3)/33, (9*a+2)/22, (9*a+2)/22, (9*a+2)/22, (9*a+2)/22, (1-a)/11, (1-a)/11;
    1/11, (10*a+1)/11, (1-a)/11, (1-a)/11, (1-a)/11, (1-a)/11, (1-a)/11, (1-a)/11, (1-a)/11, (1-a)/11, (1-a)/11;
    1/11, (1-a)/11, (1-a)/11, (1-a)/11, (8*a+3)/33, (1-a)/11, (1-a)/11, (1-a)/11, (1-a)/11, (1-a)/11, (1-a)/11;
    1/11, (1-a)/11, (1-a)/11, (1-a)/11, (1-a)/11, (9*a+2)/22, (9*a+2)/22, (9*a+2)/22, (9*a+2)/22, (10*a+1)/11, (10*a+1)/11;
    1/11, (1-a)/11, (1-a)/11, (1-a)/11, (8*a+3)/33, (1-a)/11, (1-a)/11, (1-a)/11, (1-a)/11, (1-a)/11, (1-a)/11;
    1/11, (1-a)/11, (1-a)/11, (1-a)/11, (1-a)/11, (1-a)/11, (1-a)/11, (1-a)/11, (1-a)/11, (1-a)/11, (1-a)/11;
    1/11, (1-a)/11, (1-a)/11, (1-a)/11, (1-a)/11, (1-a)/11, (1-a)/11, (1-a)/11, (1-a)/11, (1-a)/11, (1-a)/11;
    1/11, (1-a)/11, (1-a)/11, (1-a)/11, (1-a)/11, (1-a)/11, (1-a)/11, (1-a)/11, (1-a)/11, (1-a)/11, (1-a)/11;
    1/11, (1-a)/11, (1-a)/11, (1-a)/11, (1-a)/11, (1-a)/11, (1-a)/11, (1-a)/11, (1-a)/11, (1-a)/11, (1-a)/11;
    1/11, (1-a)/11, (1-a)/11, (1-a)/11, (1-a)/11, (1-a)/11, (1-a)/11, (1-a)/11, (1-a)/11, (1-a)/11, (1-a)/11;
   ];
x = [1/11; 1/11; 1/11; 1/11; 1/11; 1/11; 1/11; 1/11; 1/11; 1/11; 1/11];
disp(M);
for i = 1:15 % iteration times
   x = M*x;
end
x
```

3 (a). M when $\alpha = 0.85$

0.0909	0.0136	0.0136	0.4386	0.0136	0.0136	0.0136	0.0136	0.0136	0.0136	0.0136
0.0909	0.0136	0.8636	0.4386	0.2970	0.4386	0.4386	0.4386	0.4386	0.0136	0.0136
0.0909	0.8636	0.0136	0.0136	0.0136	0.0136	0.0136	0.0136	0.0136	0.0136	0.0136
0.0909	0.0136	0.0136	0.0136	0.2970	0.0136	0.0136	0.0136	0.0136	0.0136	0.0136
0.0909	0.0136	0.0136	0.0136	0.0136	0.4386	0.4386	0.4386	0.4386	0.8636	0.8636
0.0909	0.0136	0.0136	0.0136	0.2970	0.0136	0.0136	0.0136	0.0136	0.0136	0.0136
0.0909	0.0136	0.0136	0.0136	0.0136	0.0136	0.0136	0.0136	0.0136	0.0136	0.0136
0.0909	0.0136	0.0136	0.0136	0.0136	0.0136	0.0136	0.0136	0.0136	0.0136	0.0136
0.0909	0.0136	0.0136	0.0136	0.0136	0.0136	0.0136	0.0136	0.0136	0.0136	0.0136
0.0909	0.0136	0.0136	0.0136	0.0136	0.0136	0.0136	0.0136	0.0136	0.0136	0.0136
0.0909	0.0136	0.0136	0.0136	0.0136	0.0136	0.0136	0.0136	0.0136	0.0136	0.0136

6 (a).

ConstructPdpe.m

```
function [P, d, p, e] = ConstructPdpe(G)
% Get size of matrix G
[row_cnt, col_cnt] = size(G);
% Init. P, d, p, e
P = sparse(row_cnt, col_cnt);
d = sparse(row_cnt, 1);
p = sparse(ones(row_cnt, 1) / row_cnt);
e = sparse(ones(row_cnt, 1));
% Set values
for j = 1:1:col_cnt
    % Get the count of all links from node j
    link_cnt = full(sum(G(:,j)));
    if link_cnt ~= 0
        for i = 1:1:row_cnt
            is_link = full(G(i, j));
            if is_link ~= 0
                P(i, j) = 1.0 / link_cnt;
            end
        end
    else
        % node j is a dead end
        d(j, 1) = 1;
    end
end
```

MyPageRank.m

```
function [p, iter] = MyPageRank(G, alpha)
tol = 0.0000001;
% Make Sparse Matrix P, d
[P, d, p, e] = ConstructPdpe(G);
% Get size of matrix G
[R, C] = size(G);
% Iterate
p_last = p;
iter = 0;
while 1
    p = alpha*(P*p+e*(transpose(d)*p)/R)+(1-alpha)*e/R;
    iter = iter + 1;
    % Check
    p_diff = abs(p - p_last);
    if max(p_diff(:,1)) < tol
        break;
    end
    p_last = p;
end
```

6 (b).

q6b.m

```
G = sparse(15, 15);
G(1, 5) = 1;
G(2, 1) = 1;
G(2, 3) = 1;
G(3, 2) = 1;
G(3, 4) = 1;
G(4, 8) = 1;
G(5, 2) = 1;
G(5, 9) = 1;
G(6, 3) = 1;
G(6, 9) = 1;
G(7, 2) = 1;
G(7, 12) = 1;
G(8, 3) = 1;
G(8, 12) = 1;
G(9, 1) = 1;
G(9, 13) = 1;
G(10, 5) = 1;
G(10, 6) = 1;
G(10, 7) = 1;
G(10, 9) = 1;
G(10, 14) = 1;
G(11, 6) = 1;
G(11, 7) = 1;
G(11, 8) = 1;
G(11, 12) = 1;
G(11, 14) = 1;
```

```
G(12, 4) = 1;
G(12, 15) = 1;
G(13, 10) = 1;
G(13, 14) = 1;
G(14, 13) = 1;
G(14, 15) = 1;
G(15, 11) = 1;
G(15, 14) = 1;
[p, iter] = MyPageRank(G, 0.75);
p
iter
```

Result:

```
p =

   (1,1)        0.0333
   (2,1)        0.0389
   (3,1)        0.0389
   (4,1)        0.0333
   (5,1)        0.0443
   (6,1)        0.0443
   (7,1)        0.0443
   (8,1)        0.0443
   (9,1)        0.0716
  (10,1)        0.1035
  (11,1)        0.1035
  (12,1)        0.0716
  (13,1)        0.1133
  (14,1)        0.1017
  (15,1)        0.1133


iter =

    23
```

6 (c).

q6c.m

```
load('math_uwaterloo.mat');

alpha = 0.85;
[p, iter] = MyPageRank(G, alpha);
[y I] = sort(p, 'descend');
for n = 1:min(length(I), 20)
    disp([num2str(n) ': '    U{I(n)}]);
end
```

```
1: https://www.facebook.com/help/568137493302217
2: https://uwaterloo.ca/support
3: https://uwaterloo.ca/admissions
4: https://uwaterloo.ca/about
5: https://www.facebook.com
6: http://uwaterloo.ca/support
7: http://uwaterloo.ca/admissions
8: http://uwaterloo.ca/about
9: https://uwaterloo.ca/math
10: https://uwaterloo.ca
11: http://uwaterloo.ca/math
12: https://www.facebook.com/university.waterloo
13: https://www.facebook.com/waterloo.math
14: https://www.facebook.com/privacy/explanation
15: https://uwaterloo.ca/applied-mathematics
16: https://uwaterloo.ca/combinatorics-and-optimization
17: https://uwaterloo.ca/pure-mathematics
18: http://uwaterloo.ca/applied-mathematics
19: http://uwaterloo.ca/combinatorics-and-optimization
20: http://uwaterloo.ca/pure-mathematics
```

6 (d).

q6d.m

```
load('math_uwaterloo.mat');

alpha = [0.15 0.6 0.75 0.95];
for m = 1:length(alpha)
    disp(alpha(m));
    [p, iter] = MyPageRank(G, alpha(m));
    disp(iter);
    [y I] = sort(p, 'descend');
    for n = 1:min(length(I), 20)
        disp([num2str(n) ': '    U{I(n)}]);
    end
end
```

Alpha (α)	**Iteration times**
0.15	6
0.6	21
0.75	37
0.95	201

When α increases, the number of iterations also increases. Also, the number of iterations increases become faster when α increases.

The reason for this relationship is that the convergence of algorithm becomes slower when α increases, and slower convergence of algorithm needs more iterations to make the final values of p have the given tolerance.