Part II Computational Projects 2020

C3318H

Attach to front of report

Project number:

17.1

17.1

Programs to generate random graphs from $\mathcal{G}(n,p)$ and $\mathcal{G}_k(n,p)$ can be found in page 6 named `random_graph.m` and `random_k_graph.m`.

Question 1

A program to apply the greedy algorithm to a graph with given ordering can be found in page 6 named `random_graph.m`. For a given graph G, the program `q1.m` in page 7 finds the number of colours used in when the vertices are ordered in the following ways: (i) by increasing degree, (ii) by decreasing degree, (iii) where v_j has minimum degree in the graph $G\{v_{j+1},\ldots,v_n\}$, (iv) at random. We run the script `q1_print.m` in page 7 to produce ten members of $\mathcal{G}(60,0.5)$ and $\mathcal{G}_3(60,0.75)$ and apply `q1.m` on them. The results are shown in table 1 and table 2 respectively.

	(i)inc	(ii)dec	(iii)min	(iv)rand
1	15	13	13	16
2	16	14	14	15
3	17	14	14	14
4	15	13	13	15
5	16	13	12	13
6	17	14	13	15
7	15	14	15	16
8	16	14	14	14
9	17	13	13	13
10	17	15	14	15

Table 1: $\mathcal{G}(60,0.5)$

	(i)inc	(ii)dec	(iii)min	(iv)rand
1	6	3	3	4
2	5	3	3	5
3	6	3	3	5
4	6	4	3	3
5	10	3	3	6
6	4	4	3	3
7	5	3	3	5
8	8	3	3	4
9	7	3	3	4
10	7	4	3	4

Table 2: $\mathcal{G}_3(60,0.75)$

Question 2

Since the set of vertices $V_k = \{i : 1 \leq n, i \equiv k \pmod 3\}$ are independent, applying greedy algorithm in following ordering will ensure $\chi(G) = 3$

$$1,4,7,\cdots 2,5,8,\cdots 3,6,9,\cdots$$

Now consider the graph G with $V(G) = \{1,2,\cdots 3n\}$ s.t ij is an edge if

1. $i \not\equiv j \pmod 3$ and $|i-j| > 3$ or
2. $i,j \in \{3n-2, 3n-1, 3n\}$ and $i \neq j$.

Obviously $G \in \mathcal{G}_3(n, 1/2)$, so $\chi(G) \leq 3$. Also $3n-2, 3n-1, 3n$ form a triangle, hence $\chi(G) = 3$.

Now if greedy algorithm is applied in numerical order, veritces $\{3i-2, 3i-1, 3i\}$ will get colour i for $1 \leq i \leq n-1$ and $3n-2, 3n-1, 3n$ will be coloured in $n, n+1, n+2$ respectively. So will need exactly $n+2$ colours.

Question 3

Let A_k be the event that the greedy algorithm finds a complete subgraph of order k. Now for a given set of vertices $W = \{w_1, \cdots w_k\}$ and an exterior point v, we have

$$\begin{aligned} P(W \subseteq \Gamma(v)) &= \prod_{i=1}^{k} P(w_i \in \Gamma(v)) \\ &= \prod_{i=1}^{k} \frac{1}{2} \\ &= 2^{-k} \end{aligned}$$

Hence

$$\begin{aligned} P(\exists v \text{ s.t } W \subseteq \Gamma(v)) &= 1 - P(W \nsubseteq \Gamma(v) \, \forall v \in G \ W) \\ &= 1 - \prod_{v \in G \ W} P(W \nsubseteq \Gamma(v)) \\ &= 1 - (1 - 2^{-k})^{n-k} \end{aligned}$$

So

$$P(A_{k+1} | A_k) = 1 - (1 - 2^{-k})^{n-k}$$

Since $A_{k+1} \subseteq A_k$, we have

$$\begin{aligned} P(A_{k+1} | A_k) &= \frac{P(A_{k+1} \cap A_k)}{P(A_k)} \\ &= \frac{P(A_{k+1})}{P(A_k)} \end{aligned}$$

Hence

$$\begin{aligned} P(A_{k+1}) &= P(A_{k+1} | A_k) P(A_k) \\ &= (1 - (1 - 2^{-k})^{n-k}) P(A_k) \end{aligned}$$

Since $P(A_1) = 1$, taking $k = 1, \cdots 13$ and multiplying, we get

$$P(A_{14}) = \prod_{k=1}^{13} (1 - (1 - 2^{-k})^{n-k})$$

We run a script to find out that it's equal to 0.043192, hence very unlikely.

```
p = 1;
n = 2000;
for k=1:13
    p = p*(1-( 1 - 2^(-k))^(n-k));
end
fprintf('%f \n',p)
```

Let C_k be the number of complete graphs of size k in a graph G. We have $\mathbb{E}(C_k) = \binom{2000}{k}2^{-\binom{k}{2}}$. Now by Markov's inequality

$$\mathbb{P}(C_{18} \geq 1) \leq \mathbb{E}(C_{18}) = 0.0033$$

So it's very unlikely that there will be any complete graphs of size 18.
Also $\mathbb{E}(C_{16}) = \binom{2000}{16}2^{-\binom{16}{2}} > 2218$. So there is high probability there will be a complete graph of size 16. So the size of the clique is likely to be somewhere between 16 and 18.

Question 4

A program to find a clique of a graph G can be found in page 8 named `max_clique.m`.

We run the script `q4.m` in page 8 to compare the clique size with the upper bound obtained from greedy algorithm for graphs randomly taken from $\mathcal{G}_7(60, 0.5)$ and $\mathcal{G}_{13}(60, 0.7)$. Results are shown in table 3 and 4 respectively.

	max clique	greedy alg
1	6	13
2	7	14
3	7	12
4	6	14
5	7	14
6	7	13
7	7	13
8	7	13
9	6	13
10	7	13

Table 3: $\mathcal{G}_7(60, 0.5)$

	max clique	greedy alg
1	11	21
2	11	19
3	11	19
4	11	19
5	11	19
6	10	20
7	10	19
8	10	19
9	10	18
10	10	18

Table 4: $\mathcal{G}_{13}(60, 0.7)$

We notice that the upper bounds obtained are roughly double of the lower bounds.

Question 5

A program to find an independent set of maximum order in a graph can be found in page 9 named `colouring_indep.m`.

We run the script `q5_print.m` in page 9 to run the program on ten members of $\mathcal{G}_7(60, 0.5)$, $\mathcal{G}_7(60, 0.4)$ and $\mathcal{G}_7(60, 0.6)$, results are shown in table 5, 6 and 7 respectively.

	greedy alg	colouring indep
1	13	9
2	13	9
3	13	9
4	13	7
5	12	10
6	13	8
7	14	8
8	13	8
9	12	9
10	12	8

Table 5: $\mathcal{G}_7(60, 0.5)$

	greedy alg	colouring indep
1	11	9
2	10	8
3	11	10
4	11	9
5	11	9
6	10	10
7	12	10
8	11	9
9	11	10
10	12	10

Table 6: $\mathcal{G}_7(60, 0.4)$

	greedy alg	colouring indep
1	15	7
2	14	8
3	15	7
4	14	7
5	14	7
6	14	7
7	15	9
8	16	7
9	14	8
10	14	7

Table 7: $\mathcal{G}_7(60, 0.6)$

We can see from table 6 and 7 that the bounds are much tighter when $p = 0.4$, in fact even the worst bound gives only three possible choice for $\chi(G)$. Whereas when $p = 0.6$, the upper bounds are nearly double of lower bounds.

Question 6

Suppose graph G has vertices $\{v_1, v_2, \cdots, v_n\}$.

Greedy algorithm: The algorithm has following steps

1. We start by colouring v_1 with colour 1.
2. Then for each $2 \leq i \leq n$, we find out the neighbours of v_i in $\{v_1, v_2, \cdots, v_{i-1}\}$, which requires $i - 1$ steps. Then starting from $j = 1$, we check whether colour j has already appeared among the neighbours of v_i, if not, we assign colour j to v_i. In worst case, we will need $(i - 1)$ operations for a j, hence at most $i(i - 1)$ operations.

So in total $1 + \sum_{i=2}^{n}(i - 1) + i(i - 1) + 1 = \mathcal{O}(n^2)$ operations.

Finding Clique: We find a clique recursively as follows

1. Suppose it requires $f(n)$ operations to find a clique for any n.

2. Take any vertex, say v_n, take the subgraph G' created by the neighbours of v_n, this will require n^2 operations.

3. We compute the clique of G' and $G - v_n$, suppose they have size u and w respectively. This will need $2f(n-1)$ operations at most. Now the biggest clique in G has size $\max(u+1, w)$.

So we have $f(n) = 2f(n-1) + O(n^2)$, suppose $f(n) \leq 2f(n-1) + an^2$ for some $a > 0$. Then,

$$f(n) + an^2 + 4an - 10a \leq 2(f(n-1) + a(n-1)^2 + 4a(n-1) - 10a)$$

Applying induction on n, we get

$$\begin{aligned} f(n) + an^2 + 4an - 10a &\leq 2^{n-1}(f(1) + a + 4a - 10a) \\ f(n) &\leq 2^{n-1}(f(1) + a + 4a - 10a) - (an^2 + 4an - 10a) \\ \therefore f(n) &= O(2^n) \end{aligned}$$

Colouring using independent set: We colour recursively as follows

1. Suppose it needs $g(n)$ operations.

2. We compute the complement graph G^c of G, this requires n^2 operations. Then we find a clique I_1 in G_c, which is an independent set in G. This needs $O(2^n)$ operations. We colour all the points in I_1 with colour 1.

3. Then we do the same procedure for $G - I_1$, this requires at most $g(n-1)$ operations.

Hence we get

$$g(n) = g(n-1) + O(2^n)$$

Applying induction

$$g(n) = O(2^n) + O(2^{n-1}) + \ldots O(1) = O(2^n)$$

For colouring a graph with exactly $\chi(G)$ colours, we can apply greedy algorithm on G for every possible order of the vertices, then choose the one with minimum colour. We prove that this ensures a colouring with $\chi(G)$ colours.

Suppose a colouring of G with $\chi(G)$ colours divide G into independent sets $I_1, \cdots I_{\chi(G)}$ where I_j are the vertices with colour j. Now we can assume that for $2 \leq i \leq n$, $\exists v \in I_i, w \in I_{i-1}$ s.t vw are connected, otherwise we can move v to I_{i-1}. Now we order vertices in I_1 randomly, then move to I_2 and so on and apply greedy algorithm on G in this ordering. Since I_1 is independent, all vertices in I_1 will get colour 1. Since every vertex in I_2 has a neighbour in I_1 and since I_2 is independent, all vertices in I_2 will get colour 2 and so on. Hence this ordering gives a colouring with $\chi(G)$ colours.

Now there are $n!$ ordering of the vertices and greedy algorithm requires $O(n^2)$ operations, so this method has complechity $O(n!.n^2) = O((n-1)!.n^3)$.

Programms

Listing 1: random_graph.m

```
function G = random_graph(n,p)

A = rand(n,n);
G = zeros(n,n);
for i = 1:n
    for j = 1:i-1
        if A(i,j) < p
            G(i,j) = 1;
            G(j,i) = 1;
        end
    end
end
end
```

Listing 2: random_k_graph.m

```
function G = random_k_graph ( k,n,p)

G = random_graph(n,p);

for i = 1:n
    for j = 1:i-1
        if mod(i-j,k) == 0
            G(i,j) = 0;
            G(j,i) = 0;
        end
    end
end
end
```

Listing 3: greedy_alg.m

```
function [chi,colours] = greedy_alg ( G )

[n,~] = size(G);
colours = zeros(1,n);
colours(1) = 1;

for i = 2:n
    colours_taken = G(i,1:(i-1)) .* colours(1:i-1) ;
    c = 1 ;
    while c < i
        if any(colours_taken == c)
            c = c + 1;
        else
            break
        end
    end

    colours(i) = c;
end
chi = max(colours);
end
```

Listing 4: q1.m

```
function [chi_inc,chi_dec,chi_min,chi_rand] = q1 ( G )

[n,~] = size(G);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

G_inc = G ;

for i = 1:n-1
    for j = i+1:n
        if sum(G_inc(i,:)) > sum(G_inc(j,:))
            G_inc(:,[i j]) = G_inc(:,[j i]) ;
            G_inc([i j],:) = G_inc([j i],:) ;
        end
    end
end

chi_inc = greedy_alg(G_inc);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

G_dec = zeros(n,n);

for i = 1:n
    for j = 1:n
        G_dec(i,j) = G_inc(n+1-j,n+1-i);
    end
end

chi_dec = greedy_alg(G_dec);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

G_min = G;

for i = 1:n
    for j = 1:n+1-i
        if sum(G_min(n+1-i , 1:n+1-i)) > sum(G_min(j , 1:n+1-i))
            G_min(:,[n+1-i j]) = G_min(:,[j n+1-i]) ;
            G_min([n+1-i j],:) = G_min([j n+1-i],:) ;
        end
    end
end

chi_min = greedy_alg(G_min);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

chi_rand = greedy_alg(G);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

end
```

Listing 5: q1_print.m

```
fileID1 = fopen('p1.csv','w');
fprintf(fileID1, 'inc, dec, min, rand \n');
for i=1:10
    G = random_graph(60,0.5);
    [chi_inc,chi_dec,chi_min,chi_rand] = q1 ( G );
```

```
    fprintf(fileID1,'%d, %d, %d, %d \n', chi_inc, chi_dec, chi_min, chi_rand );
end

fileID2 = fopen('p2.csv','w');
fprintf(fileID2, 'inc, dec, min, rand \n');
for i=1:10
    G = random_k_graph(3,60,0.75);
    [chi_inc,chi_dec,chi_min,chi_rand] = q1 ( G );
    fprintf(fileID2,'%d, %d, %d, %d \n', chi_inc, chi_dec, chi_min, chi_rand );
end
```

Listing 6: max_clique.m

```
ction clique = max_clique( G )

[n,~] = size(G);

if n == 1
    clique = 1 ;
    %chi = 1 ;

else
    neighbour = find(G(n,:));

    if isempty(neighbour) == 1
        v = n;
    else
        G_neighbour = G ( neighbour , neighbour );
        v = [ neighbour(max_clique(G_neighbour)) n];
    end

    u = max_clique( G(1:n-1 , 1:n-1));

    if length(v) > length(u)
        clique = v;
    else
        clique = u ;
    end
    %chi = length(clique);
end

end
```

Listing 7: q4.m

```
fileID1 = fopen('s1.csv','w');
fprintf(fileID1, ' clique , greedy \n');
for i=1:10
    G = random_k_graph(7,60,0.5);
    fprintf(fileID1,' %d, %d \n', length(max_clique(G)) , greedy_alg(G) );
end


fileID2 = fopen('s2.csv','w');
fprintf(fileID2, ' clique , greedy \n');
for i=1:10
    G = random_k_graph(13,60,0.7);
    fprintf(fileID2,' %d, %d \n', length(max_clique(G)) , greedy_alg(G) );
end
```

Listing 8: `colouring_indep.m`

```
function [chi, colours] = colouring_indep ( G )

[n,~] = size(G);

G_c = ones(n,n) - G - eye(n);

I = max_clique(G_c);

if length(I) == n
    colours = ones(1,n);
    chi = 1;

else
    colours = ones(1,n);
    V = setdiff( 1:n , I );
    [chi, U ] = colouring_indep ( G( V , V ) );
    chi = chi + 1;
    colours(V) = U + 1 ;

end
end
```

Listing 9: `q5_print.m`

```
fileID1 = fopen('r1.csv','w');
fprintf(fileID1, ' greedy , indep \n');
fileID2 = fopen('r2.csv','w');
fprintf(fileID2, ' greedy , indep \n');
fileID3 = fopen('r3.csv','w');
fprintf(fileID3, ' greedy , indep \n');

for i=1:10
    G1 = random_k_graph(7,60,0.5);
    fprintf(fileID1,' %d, %d \n', greedy_alg(G1) , colouring_indep(G1) );

    G2 = random_k_graph(7,60,0.4);
    fprintf(fileID2,' %d, %d \n', greedy_alg(G2) , colouring_indep(G2) );

    G3 = random_k_graph(7,60,0.6);
    fprintf(fileID3,' %d, %d \n', greedy_alg(G3) , colouring_indep(G3) );
end
```