Graphs and Combinatorics

Sunil Sitaram Shelke

Here, we discuss enumeration aspects of Graphs.

1 Introduction

1. Graph terminologies

   (a) To revise, an *undirected* graph $G = (V, E)$ is a pair of vertex set V and edge set E where $|V| \in N$ and $|E| \in N \cup \{0\}$. Edge set $E \subseteq V \times V$, discarding order among elements in pairs of $V \times V$. Vertex set of a graph G is generally written as $V(G)$ when there are multiple graphs in consideration.

   (b) A subgraph G' of a graph G is $G' = (V', E') \subseteq G = (V, E)$ such that $V' \subseteq V$ and $E' \subseteq E$.

   (c) Consider a vertex set V of G. A sequence of vertices and edges $W(v_{i_1}, v_{i_k}) = v_{i_1} - e_{i_1} - v_{i_2} - e_{i_2} - \cdots v_{i_k}$ which starts at some vertex v_{i_1} and end with some vertex v_{i_k} is called as a *walk* between terminal vertices.
   A walk $W(v_{i_1}, v_{i_k})$ where *none of the edges repeat* is called as a *path* $P(v_{i_1}, v_{i_k})$ between terminal vertices. So a path is one of the shortest walks by avoiding cycles by not tracing edges again. Vertices x and y are called connected when there exists a path between them.

   (d) A component in a graph G is a subgraph $C \subseteq G$ where $\forall_{x,y \in V(C)}\ \exists P(x, y)$. This means, every pair of vertices in C is connected. Component C itself is called a *connected component*. In general a graph may contain many components, each being connected. G is called connected if G itself is a component i.e every pair of vertices in $V(G)$ is connected.

   (e) A *cycle* C in a graph G is a subgraph of G which is a path with terminal vertices being same.

   (f) A subgraph $G' \subseteq G$ is called *edge maximal* with respect to a property PRO is G' is the largest possible graph containing largest possible edges without destroying property PRO. This means, adding even a single edge e to $E(G')$ would destroy property PRO. Similar considerations about *edgeminimal* (removing an edge here would destroy property), *vertex maximal* and *vertex minimal.*

   (g) A graph G is called *k-chromatic* if at least k colors are required for vertex coloring of $V(G)$ and it is written as $\chi(G) = k$.

   (h) A subset IS of $V(G)$ is called an *independent* set, if no 2 vertices of IS have direct edge between them. Maximal independent set is the maximum size independent set among all independent sets.

2. Combinatorics formulas

   (a) ${}^nC_r$:- Number of ways of *selecting* r *distinct* out of n *distinct without repeatation*

   (b) ${}^nP_r$:- Number of ways of *arranging/sequencing* r *distinct* out of n distinct *without repeatation*

(c) ${}^{n+r-1}C_{r-1}$:- Number of ways of *selecting* r out of n distinct elements *with repeatation allowed* = number of *non-negative* integer solutions of equation $x_1+x_2+\cdots+x_r = n$

(d) n^r :- Number of ways of *arranging/sequencing* r out of n distinct elements *with repeatation allowed*

2 Questions

1. Number of graphs on n labeled vertices for $n \in \mathbb{N}$

   - In a graph of n vertices, $0 \leq |E| \leq {}^nC_2$. So there are ${}^nC_2$ edges available.
   - Each of these ${}^nC_2$ edges has 2 choices:- whether to get picked up or not, *independently* of each other. Depending on how many edges are picked up and which are picked up, different graphs result. This problem maps to counting number of binary string of length k , where in our case $k = {}^nC_2$, 1 bit for each edge.
   - There are 2^k binary strings. As many string, those many graphs because each graph results into an unique binary string and each binary string corresponds to a unique graph. So there are $2^{{}^nC_2}$ graphs on n labeled vertices

2. The number of graphs on n vertices containing exactly $0 \leq k \leq {}^nC_2$ edges ?

   - There are ${}^nC_2$ edges available overall. Choosing exactly k from these many is exactly same as choosing k bits to be *ON* out of total ${}^nC_2$ available bits.
   - Answer= ${}^{{}^nC_2}C_k$. Straight selection problem.

3. The number of graphs on n vertices containing *at least* $0 \leq k \leq {}^nC_2$ edges?

   - Solve problem 2 above $\forall\ k \leq i \leq {}^nC_2$.
   - Answer= $\Sigma_{i=k}^{{}^nC_2}\ {}^{{}^nC_2}C_i$. Straight selection problem.

4. Same as problem 3. But calculated in a different way. *Illustration of simple Combinatorial Thinking*

   - Selecting at-least k edges out of ${}^nC_2$ is same as *rejecting at most* ${}^nC_2 - k$ edges. So we remove $0 \leq i \leq {}^nC_2 - k$ edges. Using ${}^nC_i = {}^nC_{n-i}$, selecting i edges $= {}^{{}^nC_2}C_i =$ rejecting ${}^nC_2 - i$ edges $= {}^{{}^nC_2}C_{({}^nC_2-i)}$, we have :-
   - Answer $= \Sigma_{i=0}^{{}^nC_2-k}\ {}^{{}^nC_2}C_i$

5. How many edges in an *edge maximal* graph G of n vertices with $\chi(G) = k$

   - $\chi(G) = k$ implies $V(G)$ is partitioned into k classes, a class for each color. Consider those sets to be $S_1, S_2, \cdots, S_k$ with $\forall_{1\leq i\leq k}\ |S_i| = n_i$ where $\forall_{1\leq i\leq k}\ 1 \leq n_i < n$ and $\Sigma_{i=1}^{k} n_i = n$.
   - We have distributed n_i vertices to i^{th} color class. What should we do increase the number of edges as much as possible? We should add an edge between every 2 vertices as long as they belong to different color class, because 2 vertices of same color class can't be joined by an edge.
   - Consider i^{th} color class S_i. Every every vertex of S_i should be joined with every vertex of class S_j $\forall i \neq j$. A vertex $v \in S_i$ adds n_j edges, one for each vertex in S_j. So, edges added due to $S_i = n_i n_j$. We do it for all pairs $1 \leq i < j \leq k$. So,
     Total-Edges= $\Sigma_{i=1}^{k-1}\ \Sigma_{j=i+1}^{k}\ n_i n_j$.
     So creating a complete k-partite graph $K_{n_1,n_2,\cdots,n_k}$ increases the number of edges.

But, there exist many such complete K-partite graphs depending on distribution of n into $(n_1, n_2, \cdots, n_k)$. There are ${}^{n+k-1-k}C_{k-1} = {}^{n-1}C_{k-1}$ complete K-partite graphs, to be precise. Which of them is *edge maximal*?

- Our aim is to solve following optimization problem:

$$Maximize\ \Sigma_{i=1}^{k-1}\Sigma_{j=i+1}^{k}\ n_i n_j \tag{1}$$

  over $\mathbb{N}^k$ with below constraints
  (1) $\forall_{1\leq i\leq k}\ 1 \leq n_i < n$
  (2) $\Sigma_{i=1}^{k}\ n_i\ = n$

- We can solve above integer optimization problem using integer optimization techniques. But, we will use graph argument to find an optimal solution.

- - Start with a nice distribution, say $D_1 = (\frac{n}{k}, \frac{n}{k}, \cdots, \frac{n}{k})$. Calculate number of edges $|E_1|$in D_1.
  - Remove a vertex, say v from any one of the color classes, say S_1 and put in another class, say, S_2, resulting in another distribution $D_2 = (\frac{n}{k}-1, \frac{n}{k}+1, \cdots, \frac{n}{k})$. Calculate $|E_2|$ of D_2.
  - Here we calculate *gain* in edges while moving from D_1 to D_2, comparing $|E_2|$ with E_1. Removing v from S_1 and moving to S_2 deletes $\frac{n}{k}$ edges between v and S_2. In turn, removal adds $\frac{n}{k} - 1$ edges from v to new S_1. Edges from v to other classes S_i $i \neq 1$ and $i \neq 2$ are left undisturbed. So, $|E_2| = |E_1| - edgeloss +\ edgegain$ $= |E_2| = |E_1| - \frac{n}{k} + \frac{n}{k} - 1 < |E_1|$.
  - So, disturbing $D_1 = (\frac{n}{k}, \frac{n}{k}, \cdots, \frac{n}{k},)$ even slightly reduces number of edges. Further removal will decrease even more. So, $D_1 = (\frac{n}{k}, \frac{n}{k}, \cdots, \frac{n}{k},)$ is the *edge maximal* distribution.
  - Putting $n_1 = n_2 = \cdots = n_k = \frac{n}{k}$ in equation (1), solves as $\frac{n^2(k-1)}{2k}$ number of edges.
  - $k = \chi(G)$, so finally, for number of edges $|E|$ in any $\chi(G)$ chromatic graph with n vertices, we have

$$|E| \leq \frac{n^2(k-1)}{2k} = \frac{n^2(\chi(G)-1)}{2\chi(G)} \tag{2}$$

- Eq 2 is an algebraic simplification of ${}^kC_2\frac{n^2}{k^2}$. If we already know that the edge maximal distribution is $(\frac{n}{k}, \frac{n}{k}, \cdots, \frac{n}{k})$ then above unsimplified can help us count *max* number of edges directly as follows:
  - In a single pair of color classes, there are $\frac{n^2}{k^2}$ edges. There are ${}^kC_2$ such pairs of color classes and we connect all of them to maximize edges. So, total max ${}^kC_2\frac{n^2}{k^2}$ edges.

6. How many edges in an *edge maximal* graph G of n vertices with k components?

- Argument for this problem is almost same as in problem 5. Here too, we partition $V(G)$ into k disjoint classes, but the nature of a class is different than that in problem 5. In 5, no 2 vertices of same class had any edge because of coloring. In this problem, there will be edges inside a class, since a class here is a *connected component* and not a *color class.*

- So, again let there be k vertex classes $S_1, S_2, \cdots, S_k$, one for each connected component, with $\forall_{1\leq i\leq k}\ |S_i| = n_i$. What should we do to increase overall number of edges? Consider any component S_i. We should add an edge between every pair of vertices in S_i, since S_i is already connected and we force it to become *complete* component. Creating each of k components as complete subgraphs is the only way to increase number of edges.

- Consider a component S_i with $|S_i| = n_i$. Making it complete creates ${}^{n_i}C_2$ number of edges in component named S_i. So, total edges total-edges= $\Sigma_{i=1}^{k}\ {}^{n_i}C_2$.
  So, expressing G as a k-disjoint union of k *complete subgraphs* is the only way to achieve *edge maximal* configuration. But there are ${}^{n-1}C_{k-1}$ ways of expressing G as k-disjoint union of k *complete subgraphs* depending on the distribution $(n_1, \cdots, n_k)$ Which one of these distributions creates edge maximal configuration?
- Once again, our aim is now to solve following optimization problem:

$$Maximize\ \Sigma_{i=1}^{k}\ {}^{n_i}C_2 \tag{3}$$

  over $\mathbb{N}^k$ with below constraints
  (1) $\forall_{1\leq i\leq k}\ 1 \leq n_i < n$
  (2) $\Sigma_{i=1}^{k}\ n_i\ = n$
- – Once again we start with equi-distribution $D_1 = (\frac{n}{k}, \frac{n}{k}, \cdots, \frac{n}{k})$. Calculate number of edges $|E_1|$in D_1.
  – Remove a vertex, say v from any one of the components, say S_1 and put in another class, say, S_2, resulting in another distribution $D_2 = (\frac{n}{k}-1, \frac{n}{k}+1, \cdots, \frac{n}{k})$. Calculate $|E_2|$ of D_2.
  – Here we calculate *gain* in edges while moving from D_1 to D_2, comparing $|E_2|$ with E_1. Removing v from S_1 and moving to S_2 deletes $\frac{n}{k} - 1$ edges between v and remaining S_1. In turn, putting v int S_2 adds $\frac{n}{k}$ edges as there were already those many vertices in S_2. This makes new S_2 as a bigger complete graph with $\frac{n}{k}+1$ vertices. So, $|E_2| = |E_1| - edgeloss +\ edgegain = |E_2| = |E_1| - (\frac{n}{k} - 1) + \frac{n}{k} > |E_1|$. So there is an *edge gain* by disturbing distribution D_1. Is D_2 maximizing?
  – No. Removal of just one vertex increased edges, so we keep *weakening* 1^{st} component S_1 further and keep strengthening component S_2, until $|S_1|$ becomes 1. We stop here, else there will be one less component. By this time, $new|S_1| = 1$ and $new|S_2| = 2\frac{n}{k} - 1$.
  – We observe further that weakening S_1 to size 1 gained edges, so we weaken S_i, $\forall\ i \neq 2$. We repeat above steps with rest of the $k-2$ components, strengthening S_2 much more.
  – If we redistribute 1 vertex at a time, we have taken $(\frac{n}{k} - 1)$ steps to reduce a component to size 1. We weaken $(k-1)$ components to strengthen component S_2. So after $p = (k-1)(\frac{n}{k} - 1)$ steps, distribution D_p becomes $D_p = (1,\ n-k+1,\ 1, \cdots, 1)$, *gaining an edge at each step.*
  – We can't redistribute further since number of components are to kept as k. Finally we reach *edge maximal* distribution D_p expressing graph G_p as a k-disjoint union of k complete subgraphs namely $K_1,\ K_1,\ \cdots$ (k-1) such and K_{n-k+1} with $|E_p| = 0(k-1) + {}^{n-k+1}C_2 = {}^{n-k+1}C_2$.
  – So, *maximum* number of edges $|E_{maxK}|$ in a k-component graph satisfies

$$|E_{maxK}| \leq\ {}^{n-k+1}C_2 = \frac{(n-k+1)(n-k)}{2} \tag{4}$$

7. Given a graph G of n vertices, how many edges are *minimally sufficient* for G to be a k component graph?

- Upper bound in equation 4 is sufficient but not *minimally* as the bound calculated *highest* number of edges, but not *minimally* high to maintain k components.

- Look at *edge maximal* graph G' with $k+1$ components. G' is already maximally populated with edges to maintain $k+1$ components. If we add even a single edge e to G', it will join some 2 vertices of *different* components, say S_i and S_j, which will *unite* these 2 components *because of* e, reducing number of components by from $k+1$ to k by 1. This is the start of k-component graphs.
- So, if $|E_{minK}|$ is the minimum number of edges to start getting k components, then

$$|E_{minK}| = |E_{max(K+1)} + 1| \;= {}^{n-k}C_2 + 1 \;= \frac{(n-k)(n-k-1)}{2} + 1 \tag{5}$$

8. From problems 6 and 7, the range on the *sufficient* number of edges $|E|$ for n vertex graph to have k components *for sure* is

$$\frac{(n-k)(n-k-1)}{2} + 1 \;\leq\; |E| \leq \; \frac{(n-k+1)(n-k)}{2} \tag{6}$$

As a direct application of above inequality, the minimum number of edges $|E_{ConMin}|$ required for n vertex graph to be *surely* connected *OR sufficient* number of edges to be *singly connected* is

$$|E_{ConMin}| = \frac{(n-1)(n-2)}{2} + 1 \tag{7}$$

9. Number of perfect matchings in a complete graph K_{2n}, $n \in \mathbb{N}$
   - First, number of perfect matchings = 0, if n is odd.
   - Perfect matching in K_{2n} is a sequence of $2n$ vertices $P = v_{\sigma_1}v_{\sigma_2}\cdots v_{\sigma_{2n-1}}v_{\sigma_{2n}}$ where σ is a permutation of the sequence $< 1, 2, 3, \cdots, 2n >$. This $2n$ length sequence can be looked as n length sequence of n matched pairs with $\{v_{\sigma_{2i-1}}, v_{\sigma_{2i}}\}$ as matched pair, $\forall\ 1 \leq i \leq n$. There are altogether $(2n)!$ permutations of $2n$ vertices. Not all are different from each other, as perfect matchings. A perfect matching P is counted many times. For eq. perfect matching $P = v_1v_2\cdots v_{2n-1}v_{en}$ with matched pairs $\{v_1, v_2\}$, $\{v_3, v_4\}$, $\cdots$, $\{v_{2n-1}, v_{2n}\}$ is counted multiple times as
     (a) $v_1v_2\cdots v_{2n-1}v_{en}$
     (b) $v_2v_1\cdots v_{2n-1}v_{en}$
     (c) $v_{2n-1}v_{2n}\cdots v_1v_2$
     (d) $v_{2n}v_{2n-1}\cdots v_1v_2$
     (e) $v_{2n}v_{2n-1}\cdots v_2v_1$, to count *at the least*
   - To be precise, in above example perfect match is counted $2^n(n!)$ times. How? There are n perfect match pairs. Each pair can be considered as a single object representing matching. Such n objects can permute among themselves in $n!$ ways. In addition, inside each pair $\{x, y\}$ they can permute in 2! ways independently of other pairs. All such instances are extra. So, there are $2^n(n!)$ repeatations of each distinct perfect matching.
   - Total number of permutations $= T = (2n)!$ and each distinct perfect match is counted $2^n(n!)$ times. So total number of distinct perfect matchings

$$PM = \frac{(2n)!}{2^n(n!)} \tag{8}$$

10. What is the *minimum* number of edges in a graph of n vertices to *guarantee* presence of a cycle?

- This can be solved using a 2-player adversarial game. First player X proposes as few number of edges e claiming sufficiency and second player Y defies sufficiency by drawing a graph of e edges without containing cycle.
- Sufficiency proposed by X is established when Y can no more win over X.
- Now, X already knows that a tree of n vertices has $n-1$ edges and tree is an acyclic graph. So, X will not propose $e = n - 1$. So, X proposes $e = n$.
- Y can not win over X on proposed value n. If graph is singly connected then it definitely is cyclic. If graph has say k components then $\forall_{1 \leq i \leq k}\ C_i$ contain n_i vertices with $\Sigma_{i=1}^{k}\ n_i = n$. If this graph is acyclic then there are at max $e' = \Sigma_{i=1}^{k}\ (n_i - 1) = n - k$ edges with each component being a tree. But we have n edges. Adding even one edge willmake at least one component cyclic.
- Above argument proves that n vertex graph with n edges can not be acyclic and with $n-1$ edges cycle is not guaranteed. So, minimum number of edges to guarantee a cycle is n.