APPLICATIONS OF ADJACENCY MATRICES

ARAV BHATTACHARYA

1. Introduction

There are many approaches to solving problems in graph theory. One of these approaches uses matrices to encapsulate graphs:

Definition 1.1. The *adjacency matrix* of a finite graph with n vertices is an n by n matrix whose entries indicate whether pairs of vertices are adjacent in the matrix.

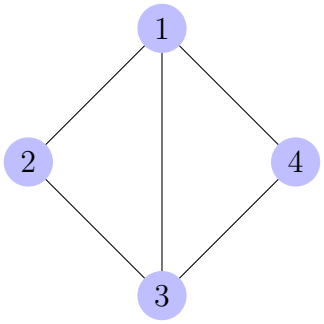

Figure 1. A graph G with vertices indexed

The graph above in Figure 1 has adjacency matrix

$$A = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}.$$

Notice that an undirected graph will have a symmetric adjacency matrix. This is important because we can apply the *spectral theorem*, which says that every symmetric matrix is orthogonally diagonalizable, on these matrices. In particular, this means that we can factor the adjacency matrix A into a product of an invertible $n \times n$ matrix M whose columns are (possibly orthonormal) eigenvectors of A, a diagonal matrix D with diagonal entries that are eigenvalues of A, and M^{-1}. For example, the matrix A above can be diagonalized as $A = MDM^{-1}$ where

$$M = \begin{bmatrix} -1 & 0 & 1 - \frac{1}{\sqrt{2}} & 1 + \frac{1}{\sqrt{2}} \\ 0 & -1 & -\sqrt{2} & \sqrt{2} \\ 1 & 0 & 1 - \frac{1}{\sqrt{2}} & 1 + \frac{1}{\sqrt{2}} \\ 0 & 1 & 1 & 1 \end{bmatrix}, \quad D = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 - \sqrt{2} & 0 \\ 0 & 0 & 0 & 1 + \sqrt{2} \end{bmatrix}.$$

Date: March 21, 2022.

Often, an explicit diagonalization of the adjacency matrix is less important than certain properties of adjacency matrices that arise as a result of them being symmetric. One such property is that the eigenvalues of A^2 are the squares of the eigenvalues of A, because $A^2 = MDM^{-1}MDM^{-1} = MD^2M^{-1}$ we can diagonalize A^2 as MD^2M^{-1}. In general, we can diagonalize A^n as MD^nM^{-1}, which the reader can verify.

Because of the prevalence of using the spectral theorem in graph theory problems where adjacency matrices are used, the part of graph theory in which this approach to encapsulating graphs is used is called spectral graph theory.

Another technique that can be useful in graph theoretic problems is creating algorithms that can traverse a graph and find some desired quality of that graph, such as whether it contains a given subgraph, whether it is planar, or the length of its minimum spanning tree. This technique is often used in computer science problems.

Spectral graph theoretic algorithms take a large amount of storage space, because adjacency matrices have n^2 entries. In comparison, an adjacency list can encapsulate a graph using an amount of storage proportional to the sum of the number of vertices and the number of edges in the graph (see [CLRS09]). As a result, algorithms using adjacency lists is much more space-efficient for graphs with relatively few edges compared to vertices. However, for *dense graphs* (graphs which have a close to maximal number of edges), adjacency lists and adjacency matrices take up similar amounts of space and as a result spectral algorithms are especially useful for these graphs.

In the coming sections, we will discuss some applications of both of these techniques, ultimately building up to the extremal graph theoretic question of forced subgraphs.

2. Friendship Theorem

We begin by looking at the *friendship theorem*, which states that if in a group of people every pair has exactly one common friend, then there is someone who is friends with everyone (excluding themselves) in the group.

To state this theorem formally, let's restate its premise in terms of graph theory. We can represent a group of people with a graph where each person in the group is represented as a vertex and each friendship is represented with an edge. A graph satisfies the premise of the friendship theorem if there is exactly one path of length 2 between any two of its vertices, and this graph satisfies the theorem if it has a vertex that shares an edge with all other vertices.

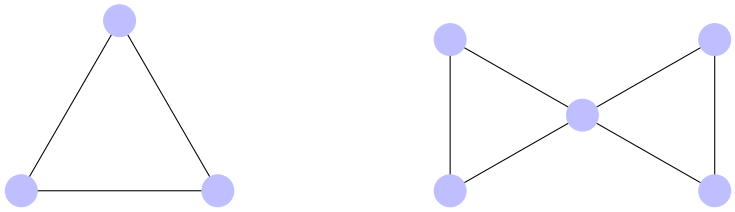

Figure 2. The 3-cycle C_3 (left) and butterfly graph (right)

The 3-cycle graph C_3 (Figure 2 left) satisfies this premise, and each vertex in this graph shares an edge with all other vertices within it so it also satisfies the friendship theorem. We

can also join two copies of C_3 at a given point to create a *butterfly graph* (Fig. 2 right) and once again verify the theorem using this graph. If we keep repeat this process of adjoining 3-cycles to each other at a single point, notice that we will continue to get graphs that satisfy the premise for the friendship theorem. We are now ready to define a general friendship graph using this process.

Definition 2.1. The *friendship graph* F_n is a planar graph created by joining n copies of the cycle graph C_3 with a common vertex.

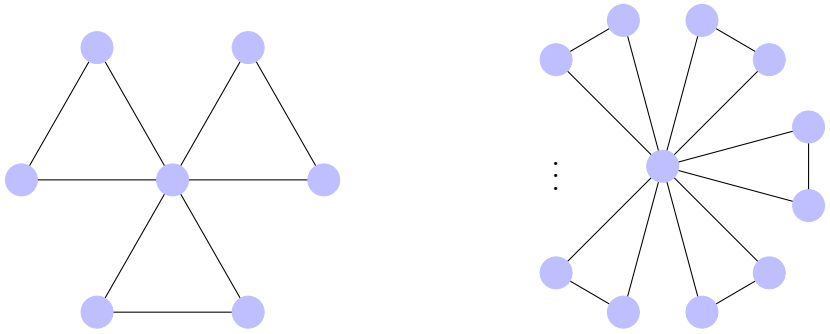

Figure 3. The friendship graph F_3 (left) and general friendship graph F_n (right)

Notice that any graph satisfying the theorem's condition cannot contain a 4-cycle C_4, because opposite vertices in that cycle would have two common neighbors. Informally, the groups of people that these graphs correspond to would have two people that share two common friends. Let's call this fact the C_4-*condition*. By the C_4-condition, the friendship theorem doesn't hold for infinite graphs. We can show this by taking a 5-cycle C_5 and repeatedly adding common neighbors to all points that don't yet have a common neighbor. This process creates an infinite graph H that satisfies the condition of the friendship theorem. For the sake of contradiction, suppose that H satisfies the friendship theorem. Then there is a vertex v of H adjacent to all other vertices of H. However, this violates the C_4-condition (see Fig. 4).

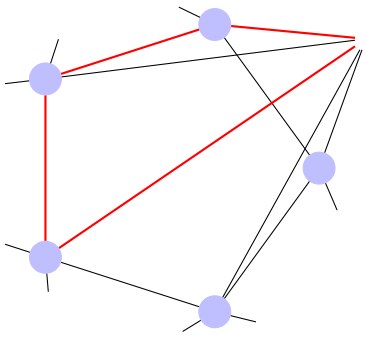

Figure 4. A depiction of H, with edges violating the C_4-condition in red

Since the friendship graphs all satisfy the friendship theorem, we can conjecture that these are the only graphs that satisfy the premise of the informal version of the theorem. In fact,

this conjecture is true and as a result we can write the friendship theorem in graph theory terms using this conjecture.

Theorem 2.2 (friendship theorem, graph theory version)**.** *The finite graphs with the property that every two vertices have exactly one neighbor in common are exactly the friendship graphs.*

We will now build up a proof of Theorem 2.2, but to complete this proof we need the following lemma:

Lemma 2.3. *If the square root of a natural number is rational, then it is an integer.*

Dedekind's proof. Let m be a natural number such that $\sqrt{m}$ is rational, and let n_0 be the smallest natural number with $n_0\sqrt{m} \in \mathbb{N}$. If $\sqrt{m} \notin \mathbb{N}$, then there exists $\ell \in \mathbb{N}$ with $0 < \sqrt{m} - \ell < 1$. Setting $n_1 := n_0(\sqrt{m} - \ell) < n_0$, we find that $n_1 \in \mathbb{N}$ and

$$n_1\sqrt{m} = n_0(\sqrt{m} - \ell)\sqrt{m} = n_0 m - \ell(n_0\sqrt{m}) \in \mathbb{N},$$

since this last expression is a sum of two natural numbers. However, this means that n_1 contradicts the choice of n_0, thus proving our lemma. ■

Now we are ready to start our proof by contradiction. We begin this proof by assuming the existence of a counterexample $G = (V, E)$ to the friendship theorem. First, we will show that G is a *regular graph*, meaning that all its vertices have the same degree.

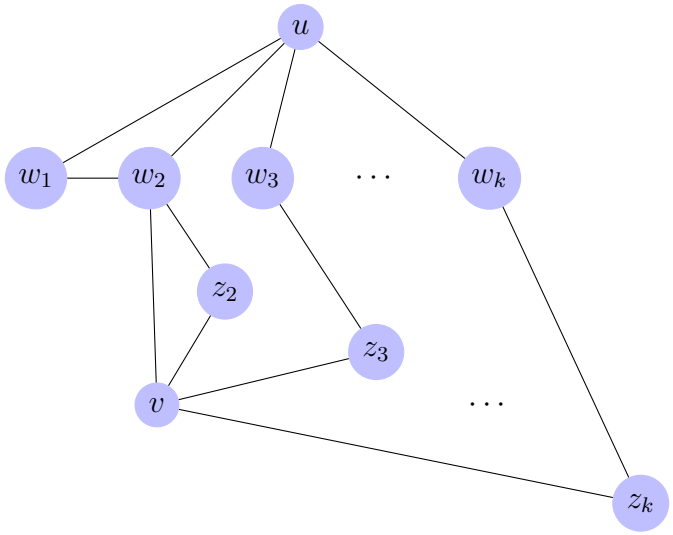

Figure 5. The construction of G

Before showing that G is regular, we will first show that any two *nonadjacent* vertices u and v of G have the same degree, meaning that $d(u) = d(v)$. Let $d(u) := k$ and label the neighbors of u as $w_1, \ldots, w_k$. The condition of the friendship theorem means that exactly one of these w_i, say w_2, is adjacent to v. Furthermore, exactly one of these w_i, say w_1, is adjacent to w_2. Then v has the common neighbor w_2 with w_1. By the condition of the theorem, v also has a common neighbor with each of w_i ($i \geq 2$). Label each of these neighbors as z_i (see Fig. 5 for a rough sketch of this construction). If each z_i is not unique, then there exist some n, m between 2 and k, inclusive, such that z_n is adjacent to w_n, w_m, and v, so we would have a 4-cycle between u, w_n, z_n, and w_m. Thus by the C_4-condition, each z_i is unique and thus

$d(v) \geq k = d(u)$. By symmetry, we could repeat this process starting at v to get $d(u) > d(v)$ and thus we have that $d(u) = d(v) = k$.

To show that G is regular, note that outside of w_2, each vertex of G is not a neighbor of at least one element of $\{u, v\}$, so by the result from the last paragraph we know that all vertices outside of w_2 have degree k. Lastly, notice that w_2 must have a non-neighbor so $d(w_2) = k$ as well. Thus G is k-regular, meaning that all vertices of G have degree k.

To find the number of vertices of G, we can sum over the degrees of each of the k neighbors of u, since each vertex except u has exactly one common vertex with u. Thus we counted every vertex exactly once, with the exception of u which we counted k times. Thus the total number of vertices of the graph is $n = k^2 - (k-1) = k^2 - k + 1$.

We can use some standard linear algebra results along with Lemma 2.3 to finish the proof of the friendship theorem. We first note that $k > 2$, because otherwise the only possible regular graphs G could be are C_1 or C_3.

We can look at the adjacency matrix A of G, with elements a_{ij} defined as before:

$$a_{ij} = \begin{cases} 1, \text{if } i \neq j \text{ and } \{i, j\} \in E(G) \\ 0, \text{otherwise} \end{cases}$$

Obviously, A does not encode information about the common neighbors of two points. However, notice that the matrix A^2 with elements b_{ij} does:

$$b_{ij} = \sum_{k=1}^{n} a_{ik} a_{kj}.$$

This property of A^2 makes it extremely easy to find its entries:

$$A^2 = \begin{bmatrix} k & 1 & \cdots & 1 \\ 1 & k & & 1 \\ \vdots & & \ddots & \vdots \\ 1 & \cdots & 1 & k \end{bmatrix} = (k-1)I + J,$$

where I is the identity matrix and J is the matrix of all 1's. We can immediately verify that J has eigenvalues n (with multiplicity 1) and 0 (with multiplicity $n-1$). Thus A^2 has eigenvalues $k - 1 + n = k^2$ (of multiplicity 1) and $k - 1$ (of multiplicity $n - 1$).

Since A is an undirected graph with no loops (edges that connect a vertex to itself) we know that A is symmetric with 0's on its main diagonal and thus A has eigenvalues k (multiplicity 1) and $\pm\sqrt{k-1}$. Suppose that r of these eigenvalues are $\sqrt{k-1}$ and s are $-\sqrt{k-1}$, with $r + s = n - 1$. Then since the sum of the eigenvalues of A is equal to the sum of its diagonal entries, we get that $k + r\sqrt{k-1} - s\sqrt{k-1} = 0$. Since $r \neq s$, this means that

$$\sqrt{k-1} = \frac{k}{s-r}.$$

By Lemma 2.3 we can let $h = \sqrt{k-1} \in \mathbb{N}$ so $h(s-r) = k$, and since $k = (\sqrt{k-1})^2 + 1$ we find that $h(s-r) = h^2 + 1$ so $h = 1$ and thus $k = 2$. This is a contradiction since $k > 2$ so we proved the friendship theorem.

The friendship theorem gives us all the finite graphs with the property that between any two vertices there is precisely one path of length 2. A conjecture of Anton Kotzig asserts that this is not possible if the path length is greater than 2:

Conjecture 2.4 (Kotzig's Conjecture)**.** *Let $\ell > 2$. Then there are no finite graphs with the property that between any two vertices there is precisely one path of length ℓ.*

Kotzig's conjecture has been proved for small values of ℓ (see [Kos88]), but a general proof has not yet been found.

3. Finding Triangle Subgraphs

We now turn our attention to a completely different problem: finding triangle subgraphs. In other words, given a graph G, how can we find a 3-cycle (if there is any) within it? (this section drew heavy inspiration from [Mat10])

One obvious method for finding such a subgraph is taking every set of 3 points in our graph and checking if all 3 points are connected to each other via an edge (see Fig. 6).

```
procedure BruteForceTriangle(G=(V,E)):
    for {u,v,w} in V:
        set T to {{u,v},{v,w},{u,w}};
        if T is subset of E:
            return {u,v,w};
    return null;
```

Figure 6. A brute-force triangle-finding algorithm

However, `BruteForceTriangle` seems to be quite inefficient, since in the worst case we would have to check every single subgraph of 3 points to find a triangle. But how would we quantify this inefficiency? One way of doing so is by finding this algorithm's so-called *time complexity*.

Definition 3.1 (Big O notation)**.** An algorithm performed on an input with n data points is said to have time complexity $O(f(n))$ if there is a constant $k \in \mathbb{R}$ such that the algorithm takes at most $kf(n)$ steps for all n. We say an algorithm with time complexity $O(f(n))$ has greater time complexity than an $O(g(n))$ algorithm if $\lim \frac{f(n)}{g(n)} = \infty$.

Let's compute the time complexity of `BruteForceTriangle`. Let's say that G has n vertices, and let $T(G)$ represent the number of steps our algorithm takes. We then have $\binom{n}{3}$ choices for our set of 3 vertices. We perform a constant number of operations on each set of vertices (namely, we check for edges up to 3 times), and since

$$T(G) \leq 3\binom{n}{3} = 3\frac{n(n-1)(n-2)}{6} < 3\frac{n^3}{6} = \frac{1}{2}n^3,$$

we find that our algorithm has time complexity $O(n^3)$.

Could we create a faster algorithm? Yes, and we can do so via use of the adjacency matrix A (see Fig. 7).

The key insight in `AdjacencyTriangle` is that finding a triangle in a graph is the same as finding two points that both share an edge and have a common neighbor. Remember

```
procedure AdjacencyTriangle(G=(V,E)):
    set A to a |V| by |V| matrix with all entries 0;
    index vertices in V from 1 until |V|;
    for i:=0 step 1 until |V|:
        for j:=0 step 1 until |V|:
            if {V[i],V[j]} in E:
                set A[i][j] to 1;

    // Square is an arbitrary algorithm that squares a matrix
    set B to Square(A);

    set i and j to 1;
    for i:=0 step 1 until |V|:
        for j:=0 step 1 until |V|:
            if A[i][j] and B[i][j] are both not 0: goto 0;

    return null;

    label 0:
        for k:=0 step 1 until |V|:
            if A[i][k] and A[k][j] are both 1:
                return {V[i],V[j],V[k]};
```

Figure 7. A triangle-finding algorithm that uses the adjacency matrix

from Section 2 that the matrix A^2 counts the number of common neighbors two points have. Thus we can find a triangle in a graph by finding indices i, j such that the i, j-th element of both A and A^2 is nonzero.

The first part of `AdjacencyTriangle`, which indexed V and initializes elements of A, takes $O(n^2)$ operations since the indexing takes $O(n)$ operations and the initialization takes $O(n^2)$ operations.

The second part of `AdjacencyTriangle` computes A^2. In the naive algorithm for `Square`, we add n products of elements of A for each element of A^2, taking $O(n \cdot n^2) = O(n^3)$ operations. Incredibly, through more sophisticated algorithms (such as the Strassen algorithm, described more in [S+69], which is $O(n^{\log_2 7}) \approx O(n^{2.807})$) we can decrease the time complexity of `Square` below $O(n^3)$. Note, however, that currently even the algorithms `Square` with lowest time complexity still have greater time complexity than $O(n^2)$.

Although the third part of `AdjacencyTriangle` has three nested `for` loops, note that the innermost `for` loop (at `label 0`) will only execute at most once: if it executes then `AdjacencyTriangle` will terminate upon its completion. As a result, this final part of `AdjacencyTriangle` only takes $O(n^2)$ operations.

Since `Square` is the part of `AdjacencyTriangle` with highest time complexity, the time complexity of the whole `AdjacencyTriangle` algorithm will also decrease below 3 as a result. Note that these sophisticated algorithms may actually yield slower results for small matrices

than the naive algorithm. For example, the Strassen algorithm is slower than the naive approach until around $n = 100$.

4. Extremal Graph Theory

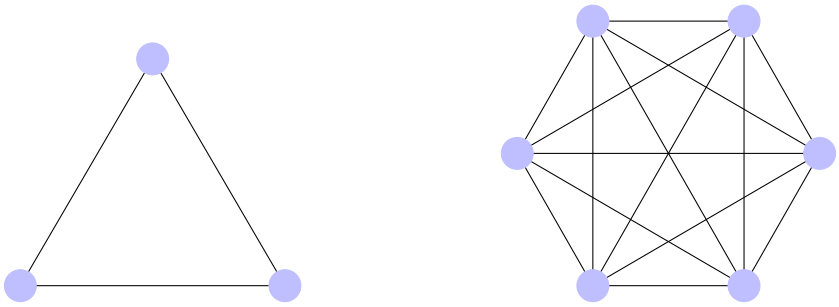

Figure 8. The 3-clique, or triangle, K_3 (left) and 6-clique K_6 (right)

In the last section, we looked at an algorithms that detect triangles (see Fig. 8, left) in a graph. What if instead we wanted to characterize the graphs that *must* contain a triangle subgraph? Thinking about it logically, we can conjecture that if a graph is sufficiently dense then it must contain a triangle. In other words, if there are enough edges in the graph relative to its number of vertices, then we can state definitively that it contains a triangle. The following theorem gives us the exact number of edges necessary:

Theorem 4.1. *A graph $G = (V, E)$ on n vertices contains a triangle if it has more than $\frac{n^2}{4}$ edges.*

Proof, adapted from [AZ99]. We will show that a graph $G = (V, E)$ with no triangle has at most $\frac{n^2}{4}$ edges. Let α be the size of the largest independent set A of G, and set $\beta = n - \alpha$. For each vertex i, its neighbors form an independent set (because G is triangle-free) and so $d_i \leq \alpha$. Notice that since A is independent, the set $V \setminus A$ of size β meets every edge of G, so we can count the edges of G by their endvertices outside of A to get $|E| \leq \sum_{i \notin A} d_i \leq \alpha\beta$. Applying the AM-GM inequality yields

$$|E| \leq \alpha\beta \leq \left(\frac{\alpha + \beta}{2}\right)^2 = \frac{n^2}{4}.$$

■

Turán's theorem allows us to generalize Theorem 4.1–this theorem characterizes the graphs with n vertices that must contain an r-clique K_r (see Figure 8, right):

Theorem 4.2 (Turán's theorem)**.** *For $r \geq 2$, a graph $G = (V, E)$ that has n vertices must contain a r-clique if it satisfies*

$$|E| > \left(1 - \frac{1}{r-1}\right)\frac{n^2}{2}.$$

Proof. In this proof, we will show by induction that any graph with n vertices that *does not* contain an r-clique has at most $\left(1 - \frac{1}{r-1}\right)\frac{n^2}{2}$ edges.

Note that if $n < r$, we have that

$$|E| < \binom{n}{2} = \frac{n(n-1)}{2} = \frac{n^2}{2} - \frac{n}{2} \leq \frac{n^2}{2} - \frac{n^2}{2(r-1)} = \left(1 - \frac{1}{r-1}\right)\frac{n^2}{2}.$$

Thus for the least (nontrivial) case $n = 3$ it suffices to check $r \in \{2, 3\}$. Notice that if $r = 2$, $\left(1 - \frac{1}{r-1}\right)\frac{(n-r+1)^2}{2} = 0$. A 2-clique K_2 is simply an edge, so a graph with no 2-cliques has no edges and thus this case is proved. The case of $r = 3$ follows from the proof of Thm. 4.1.

Let $G = (V, E)$ be a graph with n vertices and maximal edges that does not contain a r-clique. By the maximality of its edges, G must contain a $(r-1)$-clique and so we let $A \subseteq G$ be a $(r-1)$-clique. Then we can let B be $V \setminus A$. A contains $\binom{r-1}{2}$ edges, so we can now bound the number of edges in B and between A and B. We know by induction that B contains at most $\left(1 - \frac{1}{r-1}\right)\frac{(n-r+1)^2}{2}$ edges. Since G has no p-clique, each vertex in B can intersect at most $r-2$ vertices in A. Thus the number of edges with one vertex in A and another in B is at most $(r-2)(n-r+1)$. Then

$$|E| \leq \binom{r-1}{2} + \left(1 - \frac{1}{r-1}\right)\frac{(n-r+1)^2}{2} + (r-2)(n-r+1)$$

$$= \frac{(r-1)(r-2)}{2} + \left(1 - \frac{1}{r-1}\right)\frac{n^2 - 2n(r-1) + (r-1)^2}{2} + (r-2)(n-r+1)$$

$$= \left(1 - \frac{1}{r-1}\right)\frac{n^2}{2} + \frac{(r-1)(r-2) + (r-1)^2 - (2n+1)(r-1) + 2n + 2(r-2)(n-r+1)}{2}$$

$$= \left(1 - \frac{1}{r-1}\right)\frac{n^2}{2} + \frac{(r-1)(2r-3) - (r-1)(2r-4) + 2n(r-1) - (2n+1)(r-1)}{2}$$

$$= \left(1 - \frac{1}{r-1}\right)\frac{n^2}{2} + \frac{(r-1) - (r-1)}{2} = \left(1 - \frac{1}{r-1}\right)\frac{n^2}{2}.$$

■

Turán's theorem is a key result of *extremal graph theory*, a field within graph theory concerned with how global properties of a graph like its number of vertices or edges affect its local structure such as forced subgraphs. Through the use of spectral graph theory, Turán's theorem can be extended to give the number of edges past which an arbitrary graph must contain an r-clique (see [Chu05]).

Another extension of Turán's theorem gives us a bound on the number of edges a graph may have before it must contain a friendship graph as a subgraph:

Theorem 4.3. *A graph $G = (V, E)$ must contain the friendship graph F_r if it satisfies*

$$|E| \geq \left\lfloor \frac{n^2}{4} \right\rfloor + f(r), \qquad f(r) = \begin{cases} r^2 - r, & r \text{ odd} \\ r^2 - 3r/2, & r \text{ even} \end{cases}.$$

Furthermore, these bounds on $|E|$ are the best possible bounds.

Proof. See [EFGG95] for proof. ■

5. Recap

The graph theoretic methods of using adjacency matrices and creating algorithms as described in this paper are particularly useful in spectral graph theory and computer science problems, respectively. These methods, thanks to their usefulness in solving such vast types of problems, are essential tools in any graph theorist's toolkit.

References

[AZ99] Martin Aigner and Günter M Ziegler. Proofs from the book. *Berlin. Germany*, 1999.

[Chu05] Fan Chung. A spectral turán theorem. *Combinatorics, Probability and Computing*, 14(5-6):755–767, 2005.

[CLRS09] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2009.

[EFGG95] P. Erdos, Z. Furedi, R.J. Gould, and D.S. Gunderson. Extremal graphs for intersecting triangles. *Journal of Combinatorial Theory, Series B*, 64(1):89–100, 1995. URL: https://www.sciencedirect.com/science/article/pii/S009589568571026X, doi:https://doi.org/10.1006/jctb.1995.1026.

[Kos88] AV Kostochka. The nonexistence of certain generalized friendship graphs. In *Combinatorics (Eger, 1987)*, pages 341–356. North-Holland, Amsterdam, 1988.

[Mat10] Jiří Matoušek. *Thirty-three miniatures: Mathematical and Algorithmic applications of Linear Algebra*. American Mathematical Society Providence, 2010.

[S+69] Volker Strassen et al. Gaussian elimination is not optimal. *Numerische mathematik*, 13(4):354–356, 1969.

Email address: bhattacharya.arav05@gmail.com