CS 0368-4246: Combinatorial Methods in Algorithms (Spring 2025) 24/3, 2025

Lecture 2: Randomized Pattern Finding And Theta Graphs

Instructor: *Or Zamir* Scribes: *Sacha Hallermeier*

1 Introduction

Today, we explore efficient randomized algorithms for pattern finding in graphs. We begin by addressing the problem of detecting a cycle of length four (C_4) within a given graph.

2 Finding a C_4 Subgraph

The goal is to efficiently determine whether a given graph contains a C_4 subgraph.

Claim 1. *If $m \geq 100 \cdot n^{1.5}$, then we can find a C_4 in expected $O(m)$ time.*

Lemma 2. *Given an edge, we can determine in $O(m)$ time whether it is part of a C_4.*

Proof. We iterate over all other edges to check if they form a cycle of length four together with the given edge. □

Lemma 3. *In a graph with $m \geq 100 \cdot n^{1.5}$, at least 99% of the edges belong to some C_4.*

Proof. Let E' be the set of edges in E that do not belong to any C_4. The subgraph (V, E') contains no C_4. By a theorem from the previous lecture, $\text{ex}(C_4, n) \leq n^{1.5}$, implying $|E'| \leq n^{1.5}$. Given that $m \geq 100 \cdot n^{1.5}$, it follows that $|E'| \leq |E|/100$, meaning at most 1% of the edges are outside a C_4. □

Proof of Theorem. We design the following algorithm:

1. Uniformly at random, select an edge.
2. Check in $O(m)$ time whether it belongs to a C_4.
3. If not, repeat until a C_4 is found.

Since at least 99% of the edges belong to a C_4, the number of iterations follows a geometric distribution with success probability $p = 0.01$. The expected number of iterations is $100 = O(1)$, leading to an overall expected runtime of $O(m)$. □

Claim 4. *A C_4 subgraph can be found in time $O(m^{4/3})$ if one exists in the graph.*

Algorithm. Following the method from the previous lecture:

1. Iteratively remove all vertices with degree $\leq 100 \cdot m^{1/3}$ in linear time. Let n' and m' be the new number of nodes and edges.

2. If the remaining graph is nonempty with minimum degree at least $100 \cdot m^{1/3}$, then:

$$\begin{aligned} m' &\geq \frac{1}{2} n' \cdot m^{1/3} \\ &\geq \frac{1}{2} n' \cdot m'^{1/3} \end{aligned}$$

which implies $m' \geq \left(\frac{100n}{2}\right)^{4/3}$. Using Theorem 1, a C_4 can be found in linear time.

3. Otherwise, we remove all edges, leaving an empty graph. We analyze the removal process:

   - Each time a vertex v is removed, its edges are redirected to remaining nodes, forming a directed graph.
   - The out-degree satisfies $\deg_{out}(v) \leq 100 \cdot m^{1/3}$.
   - Iterate over all edges (u, v), considering edges directed outward from u and v.
   - Store these edge pairs in a hash map. A collision indicates two paths between the same pair of nodes, forming a C_4.

   Since the out-degree is bounded, the number of pairs examined is at most $m \cdot 100 \cdot m^{1/3} = O(m^{4/3})$, leading to the desired complexity.

□

3 Finding Paths and Cycles

We now seek to find P_k and C_k for general k in a graph. Note that when running a randomized algorithm with a chance of success p, we can run it t/p times to obtain a failure probability of $(1-p)^{t/p} \leq e^{-t}$.

Claim 5. *Given a graph and a node v, we can determine whether there is a P_k starting from v with high probability in time $O(k! \cdot m)$.*

Proof. We assign a random order to all vertices, except for v, which we place first. We then direct each edge from the smaller to the larger vertex according to this order. The resulting graph is a DAG, and the assigned order forms a topological sort. Using dynamic programming, we can determine in linear time the length of the longest path, and thereby check if a path of length k exists.

To analyze the probability of success, consider a specific P_k starting at v. We detect it if its order is preserved in our random order of vertices. By symmetry, the probability of this occurring is $1/k!$. Repeating the process $100 \cdot k!$ times results in a failure probability of at most e^{-100}, which is negligible. □

Note: Using this technique, we could search for a general P_k by adding a node s with an edge to all other nodes and searching for P_{k+1} starting from s.

3.1 Finding Paths of Length at Least k

Question: What if we want to determine, for each $u \neq v$, whether there is a path of length $\geq k$ from v to u?

We can use the same dynamic programming algorithm, but repeat it $100 \cdot \log n \cdot k!$ times. This results in a failure probability of $1/100n$ for each node. By the union bound, the probability of any error is at most $1/100$.

Question: Can we determine for each node whether there is a path of exactly length k leading to it?

Yes, we extend our dynamic programming algorithm by defining $\text{dp}[i][j]$ to be 1 if there exists a path of length j from v to node i. To compute $\text{dp}[i][j]$, we check $\text{dp}[u'][j-1]$ for each node u' with an incoming edge to i.

3.2 Finding Cycles of Length k

Claim 6. *We can find a C_k in a graph in time $O(k! \cdot k \cdot nm)$.*

Proof. We run the previous algorithm for each node v to find P_{k-1}. Then, for each neighbor of v, we check if there is a P_{k-1} leading to it. Since the path is simple, it does not use the edge between these nodes (otherwise, it would only be P_1), so joining them creates a C_k. □

Claim 7. *We can find either a C_k or P_k in time $O(k! \cdot \log k \cdot n^\omega)$ by using matrix multiplication instead of dynamic programming.*

Proof. We can use matrix multiplication instead of dynamic programming when counting paths in the DAG by raising the adjacency matrix to the power of k. □

4 Color Coding Technique

[Alon, Yuster, and Zwick (1995)]

In the previous algorithm, the complexity was in $O(k!)$. We seek to improve it from factorial to exponential in k, achieving a complexity of $2^{O(k)}$.

The technique is as follows: instead of directing edges, we will color the nodes with k random colors. We define a *colorful path* as a path where each color appears exactly once.

We should note two properties of colorful paths:

1. They are simple (repeating a node would mean repeating a color). 2. What is the probability that any given P_{k-1} becomes colorful? By counting, we see that the number of valid colorings of it (where each node receives a unique color) is $k!$, while the total number of ways to assign k colors to k nodes is k^k. Thus, the probability is:

$$\frac{k!}{k^k} \approx \frac{\sqrt{2\pi k}(k/e)^k}{k^k} = \frac{\sqrt{2\pi k}}{e^k},$$

where we used Stirling's approximation to justify the correctness of this estimate. This probability is approximately exponential in $1/k$.

Claim 8. *We can find, in $O(2^k km)$ time, for each node, all the lengths of colorful paths starting from v with length $\leq k$.*

Proof. We solve this with dynamic programming. Define $\text{dp}[i][S]$ to be 1 if there is a path from v to i with exactly the colors in the set S appearing in it.

We iterate over smaller subsets first, then for each subset, iterate over all nodes u. For each color c and neighbor u' of u, we update:

$$\text{dp}[u][S] = 1 \quad \text{if} \quad \text{dp}[u'][S \setminus \{c\}] = 1.$$

This ensures we only consider paths that respect the coloring constraints.

Following the same approach as before, we can find a C_k in $O(2^{O(k)}mn)$. □

4.1 Matrix Multiplication Approach

Can we improve the complexity to $O(2^{O(k)}n^{\omega})$?

Algorithm: We define the matrix A_i as the adjacency matrix restricted to edges that start at color i and do not end at color i. We also define R_i to be a matrix where rows corresponding to nodes of color i are filled with ones, and the rest are zero.

By definition, the product:

$$(R_1 A_2 \dots A_k)_{i,j}$$

counts the number of paths from i to j that pass through colors $1, 2, \dots, k$ exactly in that order.

Since we seek to detect any P_k, we want to compute:

$$\sum_{\sigma} R_{\sigma(1)} A_{\sigma(2)} \dots A_{\sigma(k)},$$

where the sum is taken over all permutations σ of $\{1, 2, \dots, k\}$. However, directly computing this sum is infeasible due to the $k!$ permutations.

To circumvent this, we reorder the summation: we first sum over R_i, then sum over all permutations excluding i:

$$\sum_{i=1}^{k} R_i \cdot \sum_{\sigma \in [k] \setminus \{i\}} A_{\sigma(1)} A_{\sigma(2)} \dots A_{\sigma(k-1)}.$$

Claim 9. *We can compute:*

$$\sum_{\sigma \in [k-1]} A_{\sigma(1)} A_{\sigma(2)} \dots A_{\sigma(k-1)}$$

in $O(2^k n^{\omega})$ time.

Proof. We use dynamic programming on subsets. Define:

$$\text{dp}[S] = \sum_{\sigma \in S} A_{\sigma(1)} A_{\sigma(2)} \dots A_{\sigma(|S|)}.$$

Our goal is to compute $\text{dp}[[k]]$.

Using the same reordering trick as before, we obtain the recurrence:

$$\text{dp}[S] = \sum_{j \in S} A_j \cdot \text{dp}[S \setminus \{j\}].$$

We compute this from smaller to larger sets. There are 2^{k-1} subsets, and each requires k matrix multi-

plications, leading to a total time complexity of:

$$O(2^{k-1}kn^{\omega}) = O(2^k n^{\omega}).$$

□

4.2 Results in Pattern Finding

Using these techniques, we obtain the following results:

- Finding C_3: $\min(n^{\omega}, m^{2\omega/(1+\omega)})$.
- Finding C_4: $\min(n^2, m^{4/3})$.

  Note: In both cases, if $\omega = 2$, we get $O(n^2)$ complexity.

Theorem 10. *For each k, we can find C_{2k} in $O(2^{O(k)}n^2)$ time.*

Proof. Follows from previous results on color-coding and matrix multiplication. □

Theorem 11. *For each k, we can find C_{2k+1} in $O(2^{O(k)}n^{\omega})$ time.*

Proof. This follows from the claim on subset DP and matrix multiplication. □

5 Theta Graphs

5.1 Definition

A *theta-graph* is a graph obtained from two cycles sharing a single edge. A theta-graph has a girth greater than t if both the cycles that form it have a length greater than t.

5.2 Theorem and Proof

Theorem 12. *If a graph has $m \geq 2tn$, then there is a subgraph of girth greater than t, and we can find it in $O(m)$ time.*

Proof. As we have proved in Lecture 1, there is a subgraph with minimal degree of at least $2t$, and we know how to find it in linear time.

We shall now find a maximal (cannot be extended) simple path in the subgraph, which can be done in linear time by taking the current path and extending it until we cannot extend it further in both directions. Let this path be denoted as $P = (v_1, \ldots, v_k)$. We can notice that all the neighbors of v_1 are on the path; otherwise, it could be extended. Let us take the first $2t$ neighbors of v_1 on the path (since the minimal degree is at least $2t$) at indices $i_1 < i_2 < \cdots < i_{2t}$.

Now, we consider the path from v_1 to v_{i_t} and from v_{i_t} to $v_{i_{2t}}$ in P. We can close the first one into a cycle of length t with the edge $e_1 = (v_1, v_{i_t})$. Similarly, we can close the second with the same edge and the edge $e_2 = (v_1, v_{i_{2t}})$. Since v_{i_t} and $v_{i_{2t}}$ are at a distance of t, we obtain two cycles of length greater than t with one common edge (v_1, v_{i_t}). □

5.3 Periodic Coloring

Definition 13. *Given a graph and a coloring of its vertices, we say that the coloring is t-periodic if for each P_t, its two endpoints have the same color.*

5.4 Examples

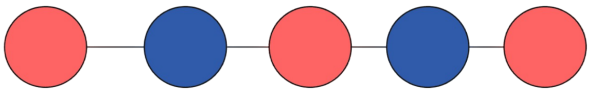

Figure 1: A 2-periodic coloring of a P5 graph

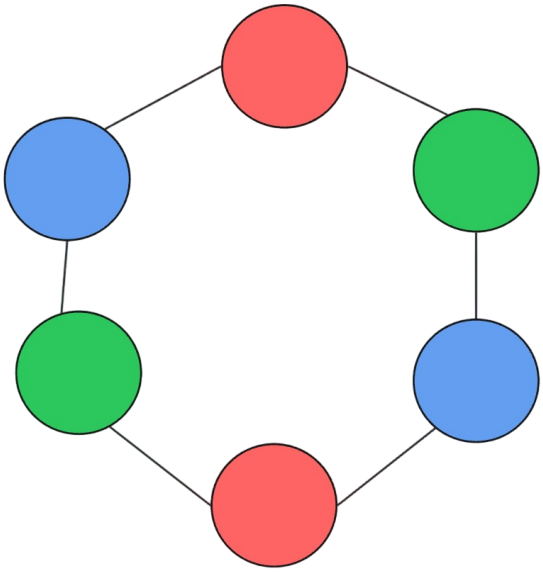

Figure 2: A 3-periodic coloring of a C6 graph

5.5 Claims and Proofs

Claim 14. *For a cycle C_l and a coloring of the vertices, the smallest t^* such that the coloring is t^*-periodic divides l.*

Proof. Let $t^* < l$ be the smallest period of the coloring. We note $l = \lfloor l/t^* \rfloor \cdot t^* + (l \mod t^*)$. We want to show that $l \mod t^* = 0$.

Assume by contradiction that $l \mod t^* \geq 1$. Consider two vertices at a distance of $l \mod t^*$. The short path between them has a length of $l \mod t^*$, but going around the other side of the cycle gives a

path of length $l-(l \mod t^*) = \lfloor l/t^* \rfloor \cdot t^*$. Since the coloring is t^*-periodic, it is also $\lfloor l/t^* \rfloor \cdot t^*$-periodic, and so the two vertices must have the same color. This means the graph is $(l \mod t^*)$-periodic, contradicting the assumption that t^* is the smallest period. □

Claim 15. *For a theta-graph with girth $> t$ and a coloring of its vertices that is t-periodic, if one of its three cycles is t^*-periodic for some $t^* \leq t$, then all its cycles are t^*-periodic.*

Proof. For any two vertices on a cycle at distance t, we can find paths of length $a \cdot t$ that start at these vertices and end in the t^*-periodic cycle. Since these paths end at vertices of distance t, they must have the same color. Furthermore, since the original two vertices are connected by paths of length $a \cdot t$ and the graph is t-periodic, they must also have the same color.

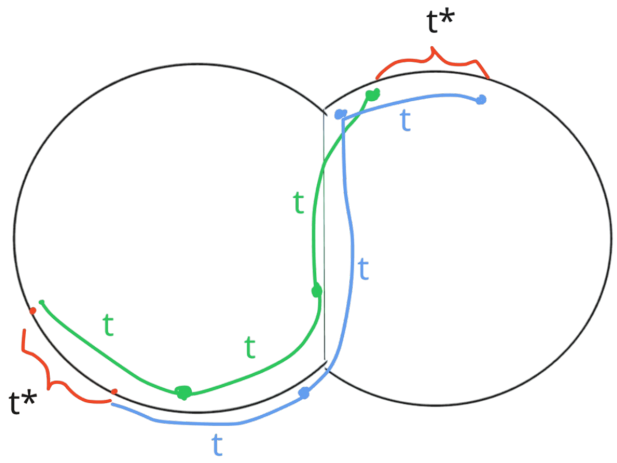

Figure 3: Illustration of jumps of size t keeping a distance of t* in a theta-graph.

□

5.6 Conclusion

Claim 16. *For a theta-graph with girth $> t$ and a coloring of its vertices that is t-periodic, the smallest period is at most 2.*

Proof. Every period t^* is a period of all three cycles. Let the lengths of the two cycles that form the graph be l_1 and l_2. The total cycle length is $l_1 + l_2 - 2$ (not counting the middle edge twice). Since $t^* \mid l_1$, $t^* \mid l_2$, and $t^* \mid (l_1 + l_2 - 2)$, it follows that:

$$t^* \mid (l_1 + l_2 - (l_1 + l_2 - 2)) = 2$$

Thus, since $t^* \mid 2$, we conclude that $t^* \leq 2$. □

References

[1] Noga Alon, Raphael Yuster, and Uri Zwick. "Color-coding". In: *J. ACM* 42.4 (July 1995), pp. 844–856. ISSN: 0004-5411. DOI: 10.1145/210332.210337. URL: https://doi.org/10.1145/210332.210337.