Topics in Theoretical Computer Science: Randomized Algorithms
EPFL, Fall semester, 2025

Homework 3 Solutions

Barras Simon <simon.barras@epfl.ch>
Zachary Doll <zachary.doll@epfl.ch>

---

1 Problem 1 (Frugal Vertex Coloring)

Let $G = (V, E)$ be a graph with maximum degree Δ. A vertex coloring $\chi : V \to C$ is called *β-frugal* if for every vertex v, no color appears more than β times in its neighborhood $N(v)$. That is,

$$|\{u \in N(v) : \chi(u) = c\}| \leq \beta \quad \text{for all } v \in V, c \in C.$$

A coloring is *proper* if $\chi(u) \neq \chi(v)$ for all edges $\{u, v\} \in E$.

Show that for any constant integer $\beta \geq 1$, there exists a β-frugal coloring of G using $Q = O(\Delta^{1+1/\beta})$ colors.

In fact, a stronger statement is true: there is a coloring that is both proper and β-frugal. But we only require to prove the above weaker statement.

1.1 Solution

Proof. Consider a color set C with cardinality $|C| := Q$. Let us color each vertex of $G = (V, E)$ uniformly at random:

$$\chi(v) \overset{\text{i.i.d.}}{\sim} \mathcal{U}\{Q\}, \quad \forall v \in V$$

For each vertex $v \in V$ all colours $c \in C$, and any subset $S \subseteq N(v)$ of size $|S| = \beta + 1$, define the bad event:

$$B(v, c, S) = \{\text{all vertices in S have color } c\}$$

If such a bad event exists, then by definition, the coloring χ is not β-frugal over G.

From the uniform color assignment, we have that the probability p of the bad event occuring for a fixed tuple (v, c, S) is given by:

$$\mathbb{P}[B(v, c, S)] = Q^{-(\beta+1)} \overset{\triangle}{=} p$$

Let us now find an upper-bound for the number of events $d^\star$ on which $B(u, c, S)$ depends, for a fixed vertex $u \in S$:

1. Because each color is assigned independently for each vertex, the event $B(u, c, S)$ only depends on the colors in S. In other words, for the events $B(u, c, S)$ and $B(w, \tilde{c}, \tilde{S})$ to be dependent, it must be the case that $u \in \tilde{S} \subseteq N(w)$, i.e. w and u must be neighbors.

By hypothesis, we know that $\delta(v) \leq \Delta$ for all $v \in V$, so there are at most Δ choices for an initial vertex.

2. We have at most $\binom{\Delta-1}{\beta}$ choices for the remaining β vertices in $\tilde{S} \subseteq N(w) \backslash \{u\}$.

3. We have Q choices for the color $\tilde{c} \in C$.

Thus,

$$d^{\star} \leq \Delta \binom{\Delta - 1}{\beta} Q - 1$$

Where we subtracted 1 for the event itself. Union-bounding over the fixed cardinality $|S| = \beta + 1$ yields an upper-bound to the total number of events d that any non-fixed event depends on:

$$\begin{aligned}
d &\leq (\beta+1)d^{\star} \\
&\leq Q(\beta+1)\Delta \binom{\Delta-1}{\beta} - 1 \\
&\leq Q(\beta+1)\Delta \left(\frac{e\Delta}{\beta}\right)^{\beta} - 1 \\
&= Q(\beta+1)\left(\frac{e}{\beta}\right)^{\beta} \Delta^{\beta+1} - 1 \stackrel{\triangle}{=} QC_{\beta}\Delta^{\beta+1} - 1
\end{aligned}$$

Where $C_{\beta} = (\beta+1)\left(\frac{e}{\beta}\right)^{\beta}$, and where we used stirling's bound in the third line:

$$k! \geq \left(\frac{k}{e}\right)^{k} \implies \frac{n^k}{k!} \leq \frac{n^k}{(k/e)^k} = \left(\frac{en}{k}\right)^{k}$$

The Local Lovász Lemma states that, if $ep(d+1) \leq 1$, then with positive probability, no bad event occurs, meaning a β-frugal coloring exists. Thus, plugging in the computed value for p and the above bound for d into the LLL inequality, it suffices to have

$$eQ^{-(\beta+1)} \cdot QC_{\beta}\Delta^{\beta+1} \leq 1 \implies eC_{\beta}Q^{-\beta}\Delta^{\beta+1} \leq 1$$

Finally, solving for Q gives:

$$Q^{\beta} \geq eC_{\beta}\Delta^{\beta+1} \implies Q \geq (eC_{\beta})^{1/\beta}\Delta^{1+1/\beta}$$

And thus

$$Q = O\left(\Delta^{1+1/\beta}\right)$$

as required.

□

2 Problem 2 (Concentration for Euclidean MST)

Let $X_1, \ldots, X_n$ be n points chosen independently and uniformly at random from the unit square $[0,1]^2$. Let $L(X_1, \ldots, X_n)$ denote the total length of the Minimum Spanning Tree (MST) on these points, using Euclidean distances.

Let $\mu = \mathbb{E}[L(X_1, \ldots, X_n)]$ be the expected length of the MST. Prove that for any $\epsilon > 0$, the probability of deviating from the mean by ϵn is exponentially small in n. Specifically, show that:

$$\Pr(|L - \mu| \geq \epsilon n) \leq 2 \exp\left(-\frac{\epsilon^2 n}{25}\right)$$

Hint: You may use the following fact without proof.

Fact: Any Euclidean MST on points in the 2D plane (using the L_2 norm) has a maximum vertex degree of at most 5.

2.1 Solution

We aim to prove that L is a Lipschitz function, and then apply McDiarmid's inequality to obtain the desired concentration bound.

Proof. We claim that L is a Lipschitz function. That is, for any two configurations $X = (X_1, \ldots, X_n)$ and $X' = (X_1, \ldots, X_i', \ldots, X_n)$ that differ only in one coordinate i, the following holds:

$$|L(X) - L(X')| \leq c$$

for some constant c.

Let T be the minimum spanning tree (MST) of X, and let T' be the tree obtained from T by replacing the vertex X_i with X_i'. Although T' may not be the MST of X', it is still a valid spanning tree on the modified set of points. let $T'^{\star}$ denote the MST of X', then:

$$\operatorname{len}(T'^{\star}) \leq \operatorname{len}(T')$$

Since all points lie within the unit square $[0,1]^2$, the maximum Euclidean distance between any two point is

$$\sqrt{1^2 + 1^2} = \sqrt{2}$$

Given the fact that in planar MST, each vertex has maximum degree of 5. Therefore, when a single point X_i is moved to X_i', at most five edges are affected, each by changing by at most $\sqrt{2}$ in length. Hence

$$|\operatorname{len}(T) - \operatorname{len}(T')| \leq 5\sqrt{2}$$

Since $\operatorname{len}(T'^{\star}) \leq \operatorname{len}(T')$, it follows that:

$$|L(X) - L(X')| \leq |\operatorname{len}(T) - \operatorname{len}(T'^{\star})| \leq 5\sqrt{2}$$

Thus, L is c-Lipschitz with $c = 5\sqrt{2}$.

We can now apply McDiarmid's inequality, which states that if L is c_i-Lipschitz in each coordinate, then:

$$\Pr\left[|L - \mu| \geq t\right] \leq 2 \exp\left(\frac{-2t^2}{\sum_{i=1}^n c_i^2}\right)$$

Since each $c_i = 5\sqrt{2}$, we have

$$\sum_{i=1}^{n} c_i^2 = n \cdot (5\sqrt{2})^2 = 50n$$

Setting $t = \varepsilon n$, we obtain:

$$\Pr\left[|L - \mu| \geq \varepsilon n\right] \leq 2\exp\left(\frac{-2\varepsilon^2 n^2}{50n}\right) = 2\exp\left(\frac{-\varepsilon^2 n}{25}\right)$$

□

3 Problem 3 (A randomized algorithm for k-SAT)

Consider a satisfiable k-CNF Φ on n variables. One try of the algorithm: start at uniform $x_0 \in \{0,1\}^n$; for T steps $t = 0, 1, \ldots, T-1$, if x_t satisfies Φ return x_t, else pick an unsatisfied clause C, choose a uniform random literal $\ell \in C$ and flip its variable to obtain x_{t+1} from x_t. If no solution within T steps, restart. Fix a satisfying assignment $x^\star$ and let $D_t = \|x_t - x^\star\|_1$.

(a) Show that whenever $D_t > 0$, $\Pr[D_{t+1} = D_t - 1 \mid x_t] \geq 1/k$.

(b) If $D_0 = d$, prove $\Pr[\text{hit 0 within } d \text{ steps}] \geq (1/k)^d$ (via d consecutive decreases).

(c) For x_0 uniform, $D_0 \sim \mathrm{Bin}(n, 1/2)$. Show $\Pr[\text{success in one try}] \geq \left(\frac{k+1}{2k}\right)^n$.

(d) Argue that $T = n$ suffices to capture the event in (b), and conclude the expected time $\tilde{O}\left((\frac{2k}{k+1})^n\right)$; specialize to $k = 3$ as $\tilde{O}((\frac{3}{2})^n)$.

Remark (Schöning's bound). If in (b) you instead bound $\Pr[\text{ever hit } 0 \mid D_0 = d] \geq (1/(k-1))^d$ using a biased random-walk/gambler's-ruin argument with step -1 w.p. $1/k$ and $+1$ w.p. $1 - 1/k$, then averaging as in (c) yields per-try success $\left(\frac{k}{2(k-1)}\right)^n$ and expected time $\tilde{O}\left((2 - \frac{2}{k})^n\right)$ (e.g., $\tilde{O}((\frac{4}{3})^n)$ for 3-SAT).

3.1 Solution

3.1.1 Part A

Proof. Pick an arbitrary assignment x_t with distance $D_t > 0$ from a satisfying $x^\star$, and assume x_t does not satisfy Φ (in which case the algorithm would terminate).

Because $x^\star$ satisfies Φ, every clause C contains at least one literal $\ell^\star \in C$ set to true under the assignment $x^\star$. Conversely, as C is unsatisfied under x_t, every literal of C must be false in x_t. In particular, we have that $\ell^\star \in C$ is false in x_t. Because the algorithm picks a literal $\ell \in C$ u.a.r. among the k possible choices, and at least one of them ($\ell^\star$) would decrease the Hamming distance by 1, it follows that

$$\Pr[D_{t+1} = D_t - 1 \,|\, x_t] \geq \frac{1}{k}$$

□

3.1.2 Part B

Proof. We assume the satisfying assignment $x^\star$ is unique, as per clarifications on the exercise.

Consider the event that each of the first d steps decrease the distance by exactly 1. From part (a), we have that

$$\Pr\left[\bigcap_{t=0}^{d-1}\{D_{t+1} = D_t - 1\}\right] = \prod_{t=0}^{d-1}\Pr[D_{t+1} = D_t - 1 \mid x_t] \geq \left(\frac{1}{k}\right)^d$$

Thus:

$$\Pr[\text{ hit 0 within } d \text{ steps} \mid D_0 = d\,] \geq \left(\frac{1}{k}\right)^d$$

□

3.1.3 Part C

Proof. the result from part (b) gives us

$$\Pr[\text{ success in one try } \mid D_0 = d\,] \geq \left(\frac{1}{k}\right)^d$$

From which it immediately follows that

$$\begin{aligned}
\Pr[\text{ success in one try }] \geq \mathbb{E}\left[\left(\frac{1}{k}\right)^{D_0}\right] &= \prod_{i=1}^{n}\mathbb{E}\left[\left(\frac{1}{k}\right)^{Z}\right], \quad Z \sim \mathrm{Ber}(1/2) \\
&= \prod_{i=1}^{n}\left(\frac{1}{2} + \frac{1}{2}\cdot\frac{1}{k}\right) \\
&= \left(\frac{k+1}{2k}\right)^n
\end{aligned}$$

Where we simply used the MGF of $D_0 \sim \mathrm{Bin}(n, 1/2)$. □

3.1.4 Part D

The event from (b), namely:

$$D_0 = d \Longrightarrow \Pr[\text{ hit 0 within } d \text{ steps }] \geq (1/k)^d$$

is fully contained in the first d steps. Because we consider the K-SAT problem over n variables, it is clear that the maximal possible initial distance is $D_0 = n$, i.e. when all variables from $x_0 \sim \mathcal{U}\{0,1\}^n$ differs from the satisfying assignment $x^\star$. Thus, $D_0 = d \leq n$, and choosing $T = n$ suffices to capture the event fully.

From part (c), we know that the probability of success in one try is:

$$P \geq \left(\frac{k+1}{2k}\right)^n$$

Therefore the expected number of necessary tries until success is

$$\mathbb{E}[\text{necessary tries until success}] = O(1/P) = O\left(\left(\frac{2k}{k+1}\right)^n\right)$$

Each try costs $T = n$ time, and thus:

$$\mathbb{E}[\text{time}] = O\left(n\left(\frac{2k}{k+1}\right)^n\right) = \tilde{O}\left(\left(\frac{2k}{k+1}\right)^n\right)$$

4 Problem 4 (The Long(est) Path Home)

Given a graph $G = (V, E)$, you want to find long simple paths in the graph in polynomial time.

(a) (Algorithm 1: Dead easy.) Show that you can find a path of length k (if such a path exists) in time $n\Delta^k$, where Δ is the maximum degree of G.

(b) (Easy.) If the graph were directed and acyclic (i.e., a DAG), then show that you can deterministically find the longest path in G in time $O(m+n)$. Here, and in general, $m = |E|$ and $n = |V|$.

(c) (Algorithm 2:) Consider running the following procedure n times, and outputing the longest path found in these n tries.

> Take a random permutation of the vertices, and direct each edge from the lower endpoint to the higher endpoint to create a DAG $\vec{G}$. Find a longest path in $\vec{G}$.

Show that for $k = c\frac{\log n}{\log\log n}$ for some constant $c > 0$, Algorithm 2 will find a path of length k (if it exists) with probability at least $1/2$.

(d) Now, consider a slight extension of this idea. Suppose you have a graph G, and you color the vertices using k colors (neighbors need not have different color). A path is called *polychromatic* if has $\ell \leq k$ vertices, and all the ℓ vertices have different colors.

i. Show that you can find a polychromatic path of length k in time that is $\text{poly}(n,k)2^k$. (So, this is polynomial time for $k = O(\log n)$).

ii. (Algorithm 3:) Consider running the following procedure n times, and outputing the longest path found in these n tries.

> Take a random coloring of the vertices using k colors, and find the polychromatic path of length at most k in G.

Show that for $k = c\log n$ for some constant $c > 0$, Algorithm 3 will find a path of length k (if it exists) with probability at least $1/2$. (Hint: Use Stirling's approximation.)

4.1 Solution

4.1.1 Part A

We aim to show that the brute-force algorithm for finding the longest path of length at most k in a graph $G = (V, E)$ runs in $O(n\Delta^k)$, where $n = |V|$ is the number of vertices and Δ is the maximum degree of any vertex in G.

Proof. The high-level algorithm (Algorithm 1) enumerates all vertices as potential starting points for the longest path. For each vertex $v \in V$, it calls a recursive procedure that explores all possible simple paths starting from v with length at most k. The outer loop thus contributes as factor of $O(n)$.

Algorithm 1 Find longest path in the graph

```
procedure LONGEST-PATH(G = (V, E), k)
    P ← ∅
    for v ∈ V do
        Q ← Longest-Path-Vertex(G, v, ∅, k)
        P ← max(P, Q)
    end for
    return P
end procedure
```

Algorithm 2 describes the recursive procedure that computes the longest path starting from a fixed vertex v. At each recursive step, the algorithm considers all neighbors u of v (excluding the predecessor vertex p) and recursively explores all paths of remaining length $k - 1$ starting from u.

Since each vertex has at most Δ neighbors, and the recursion proceeds to depth k, the total number of recursive calls can be upper-bounded by $O(\Delta^k)$. This corresponds to exploring all possible paths of length up to k starting from a given vertex.

Algorithm 2 Find longest path of a vertex

```
procedure LONGEST-PATH-VERTEX(G = (V, E), v, p, k)
    P ← ∅
    if k = 0 then
        return P
    end if
    for u ∈ V where {u, v} ∈ E and u ≠ p do
        Q ← Longest-Path-Vertex(G, u, v, k − 1)
        P ← max(P, Q)
    end for
    return P
end procedure
```

Combining both procedures, the overall time complexity is obtained by multiplying the outer $O(n)$ loop with the recursive exploration cost $O(\Delta^k)$. Therefore, the total running time of the algorithm is

$$O(n\Delta^k)$$

□

4.1.2 Part B

We aim to show that if the graph $G = (V, E)$ is directed and acyclic, there exist an algorithm to find the longest path in time $O(V + E)$.

Proof. Finding the the longest path in a DAG (Directed Acyclic Graph) can achieved efficiently using a DFS (Depth-First Search) combined with dynamic programming. This approach ensures that each vertex is processed only once, avoiding redundant computations and achieving the desired linear-time complexity.

The high-level procedure (Algorithm 3) iterates over all vertices, treating each as potential starting points for the longest path. For every vertex $v \in V$, it invokes a recursive routines that computes and memorizes the longest path starting from v. The outer loop thus contributes as factor of $O(V)$.

Algorithm 3 Find longest path in a DAG

```
procedure LONGEST-PATH-DAG(G = (V, E), k)
    P ← {∅} for |V|
    for v ∈ V do
        Longest-Path-Vertex-DAG(G, P, v)
    end for
    return max(P ∈ P)
end procedure
```

Algorithm 4 defines the recursive routine improved with dynamic programming that computes the longest path starting from a fixed vertex v. If the longest path for v has already been computed (i.e., $\mathcal{P}[v] \neq \emptyset$), the result is returned immediately. Otherwise, for each outgoing edge $(v, u) \in E$, the algorithm recursively computes the longest path starting from u and updates $\mathcal{P}[v]$ accordingly.

Algorithm 4 Find longest path of a vertex in DAG

```
procedure LONGEST-PATH-VERTEX-DAG(G = (V, E), P, v)
    if P[v] ≠ ∅ then return
    end if
    for e = {v, u} ∈ E do
        Longest-Path-Vertex-DAG(G, P, u)
        P[v] ← max(P[v], P[u] + e)
    end for
end procedure
```

Each vertex v is visited only once, and during its processing, the algorithm inspects all its outgoing edges. The total amount of work fo all vertices is the total number of edges E.

Thus combining bot procedures yields an algorithm that computes the longest path in a DAG in linear time.

$$O(V + E)$$

□

4.1.3 Part C

We claim that if we assign a random ordering to all vertices of an undirected graph $G = (V, E)$ and orient each edge from the vertex with smaller index to the one with larger index, we obtain a directed acyclic graph (DAG) $\vec{G} = (V, E)$.

Now, consider construction n independent random orientations $\vec{G}_1, \dots, \vec{G}_n$. We want to show that, for

$$k = c\frac{\log n}{\log\log n}$$

with a sufficiently large constant $c > 0$, the probability that at least one of these DAGs contains a path of length k is at least $\frac{1}{2}$:

$$\Pr\left[\exists i, P \subseteq \vec{G}_i, \mathrm{len}(P) = k\right] \geq \frac{1}{2}$$

Proof. Fix a specific path P of length k in G. It consists of $k+1$ vertices. When we randomly permute the vertices, there are $(k+1)!$ possible relative orderings of these vertices, and only one of them produces an increasing order consistent with the directed edges of $\vec{G}$. Hence

$$\Pr\left[P \subseteq \vec{G}\right] = \frac{1}{(k+1)!} =: p$$

The longer the path, the smaller this probability becomes. Let $P^\star$ be a longest path in G. For our family of random orientations, the probability that at least one of the $\vec{G}_i$ contains $P^\star$ is

$$\Pr\left[P \in \bigcup_{i=1}^{n} \vec{G}_i\right] \geq \Pr\left[P^\star \in \bigcup_{i=1}^{n} \vec{G}_i\right] = 1 - \Pr\left[P^\star \notin \bigcup_{i=1}^{n} \vec{G}_i\right] = 1 - (1-p)^n$$

Using the standard exponential bound $(1-p)^n \leq e^{-pn}$.

$$\Pr\left[P^\star \in \bigcup_{i=1}^{n} \vec{G}_i\right] \geq 1 - e^{-pn} = 1 - \exp\left(-n\frac{1}{(k+1)!}\right) = 1 - \exp\left(-n\frac{1}{\left(c\frac{\log n}{\log\log n} + 1\right)!}\right)$$

Using Stirling's bound

$$(k+1)! \geq \sqrt{2\pi(k+1)}\left(\frac{k+1}{e}\right)^{k+1}$$

we obtain

$$\frac{1}{(k+1)!} \leq \frac{1}{\sqrt{2\pi(k+1)}}\left(\frac{e}{k+1}\right)^{k+1}$$

Substituting $k = c\frac{\log n}{\log\log n}$

$$p \leq \frac{1}{\sqrt{2\pi k}}\left(\frac{e\log\log n}{c\log n}\right)^{c\frac{\log n}{\log\log n}}$$

Hence

$$pn \leq n^{1-c+o(1)}$$

If we choose any $c > 1$, then $pn \to 0$ as $n \to \infty$, implying $1 - e^{-pn} \to 0$. Conversely, if $c < 1$, then $pn \to \infty$, and

$$1 - e^{-pn} \to 1$$

Thus, there exist a threshold constant $c^\star \approx 1$ such tact for $c < c^\star$ the probability exceeds $\frac{1}{2}$ for large n. $\square$

4.1.4 Part D

Consider a graph $G = (V, E)$ where each vertex is assigned one of k colors, such that no two adjacent vertices share the same color. A path P is said to be polychromatic if it contains at most one vertex of each color. Clearly, the length of any such path satisfies $\text{len}(P) \leq k$.

We claim that there exist a algorithms running in time $\text{poly}(n, k)2^k$ that finds the longest polychromatic path.

Proof. Consider a dynamic programming (DP) table defined as:

$$\text{DP}[C][v] = \begin{cases} \text{True} & \text{if there exists a path ending at vertex } v \text{ with } C \\ \text{False} & \text{otherwise} \end{cases}$$

Where C is a polychromatic subset.

The DP is initialized for single-vertex paths:

$$\text{DP}[\text{col}(v)][v] = \text{True}, \quad \forall v \in V$$

Then, it is updated recursively:

$$\text{DP}[C][v] = \bigvee_{u \in \text{N}(v)} \text{DP}[C \setminus \text{col}(v)][u]$$

Where $N(v)$ denotes the set of neighbors of v.

Each entry in the DP table represents whether a polychromatic path with color subset C ends at vertex v.

Since there are 2^k subsets of colors and n vertices, the table contains $O(n2^k)$ entries. For each vertex, we may check all its incident edges, leading to a total time complexity of:

$$O\left(2^k(n + m)\right) = \text{poly}(n, k)2^k$$

□

By repeating the random coloring of the graph $G = (V, E)$ independently n times to generate graphs $G_1^\bullet, \ldots, G_n^\bullet$, we claim that for

$$k = c \log n$$

with constant $c > 0$, the algorithm finds a polychromatic path of length k (if exists) with probability at least $\frac{1}{2}$:

$$\Pr\left[\exists i, P \subseteq G_i^\bullet, \text{len}(P) = k\right] \geq \frac{1}{2}$$

Proof. Fix a specific path P of length k in G. The path contains k distinct vertices, each independently assigned one of k color.

There are k^k possible colorings of the vertices along P, but only those in which all vertices have distinct colors yield a polychromatic path. The number of distinct-color colorings equals $k!$. Thus, for a single random coloring:

$$\Pr\left[\text{p}(P) \in G^\bullet\right] = \frac{k!}{k^k} =: p$$

Where $\mathrm{p}(P)$ is the polychromatic possible path of P.

For our family of random coloring, the probability that at least one of the $G_i^\bullet$ contains a valid $\mathrm{p}(P)$ of length k is

$$\Pr[\exists i : \mathrm{p}(P) \in G_i^\bullet] = 1 - (1-p)^n \geq 1 - e^{-pn} = 1 - \exp\left(-n\frac{k!}{k^k}\right)$$

To analyze this expression asymptotically, we apply Stirling's approximation:

$$k! \approx \sqrt{2\pi k}\left(\frac{k}{e}\right)^k$$

Thus,

$$\frac{k!}{k^k} \approx e^{-k}\sqrt{2\pi k}$$

Plugging this back, we get:

$$\Pr[\exists i : \mathrm{p}(P) \in G_i^\bullet] \geq 1 - \exp\left(e^{-k}\sqrt{2\pi k}\right)$$

Setting $k = c\log n$

$$ne^{-k}\sqrt{2\pi k} = ne^{-c\log n}\sqrt{2\pi c\log n} = n^{1-c}\sqrt{2\pi c\log n}$$

For this probability to be at least $\frac{1}{2}$, we require the exponent to be at least $\ln 2$. That is, $n^{1-c}\sqrt{\log n} = \Omega(1)$, which holds for any constant $c < 1$.

Therefore, for any constant $c < 1$,

$$k = c\log n \Rightarrow \Pr[\exists i : \mathrm{p}(P) \in G_i^\bullet] \geq \frac{1}{2}$$

$\square$