Color Coding

Why randomized?

- A guaranteed error probability of 10^{-100} is as good as a deterministic algorithm. (Probability of hardware failure is larger!)
- Randomized algorithms can be more efficient and/or conceptually simpler.
- Can be the first step towards a deterministic algorithm.

Polynomial-time vs. FPT randomization

Polynomial-time randomized algorithms

- Randomized selection to pick a **typical**, **unproblematic**, **average** element/subset.
- Success probability is constant or at most polynomially small.

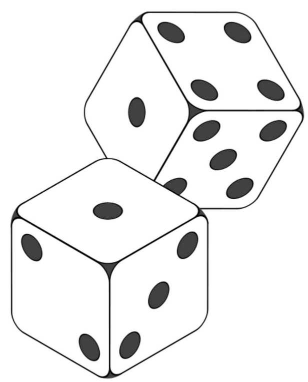
Randomized FPT algorithms

- Randomized selection to satisfy a **bounded number** of (unknown) constraints.
- Success probability might be exponentially small.

Randomization as reduction

Problem A
(what we want to solve)

Randomized magic



Problem B
(what we can solve)

Color Coding

k-PATH

Input: A graph G, integer k.
Find: A simple path on k vertices.

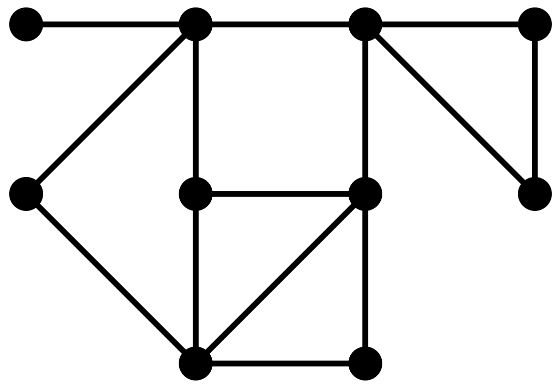
Note: The problem is clearly NP-hard, as it contains the HAMILTONIAN PATH problem.

Theorem

k-PATH can be solved in time $2^{O(k)} \cdot n^{O(1)}$.

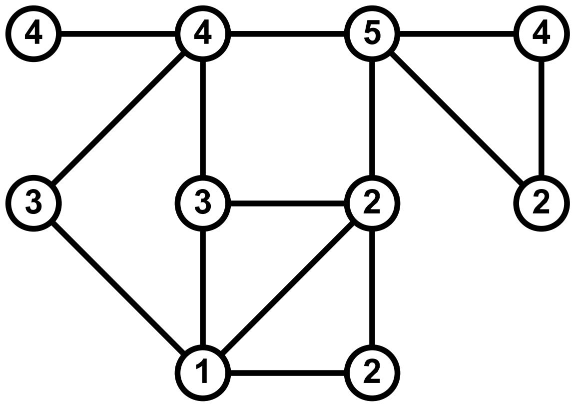
Color Coding

- Assign colors from $[k]$ to vertices $V(G)$ uniformly and independently at random.



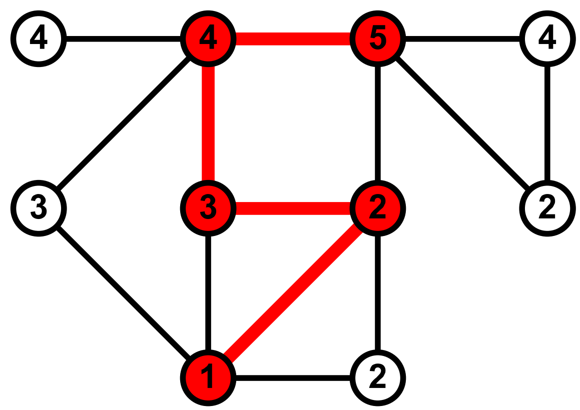
Color Coding

- Assign colors from $[k]$ to vertices $V(G)$ uniformly and independently at random.



Color Coding

- Assign colors from $[k]$ to vertices $V(G)$ uniformly and independently at random.



- Check if there is a path colored $1 - 2 - \cdots - k$; output "YES" or "NO".
  - If there is no k-path: no path colored $1 - 2 - \cdots - k$ exists $\Rightarrow$ "NO".
  - If there is a k-path: the probability that such a path is colored $1 - 2 - \cdots - k$ is k^{-k} thus the algorithm outputs "YES" with at least that probability.

Error probability

Useful fact

If the probability of success is at least p, then the probability that the algorithm **does not** say "YES" after $1/p$ repetitions is at most

$$(1-p)^{1/p} < \left(e^{-p}\right)^{1/p} = 1/e \approx 0.38$$

Error probability

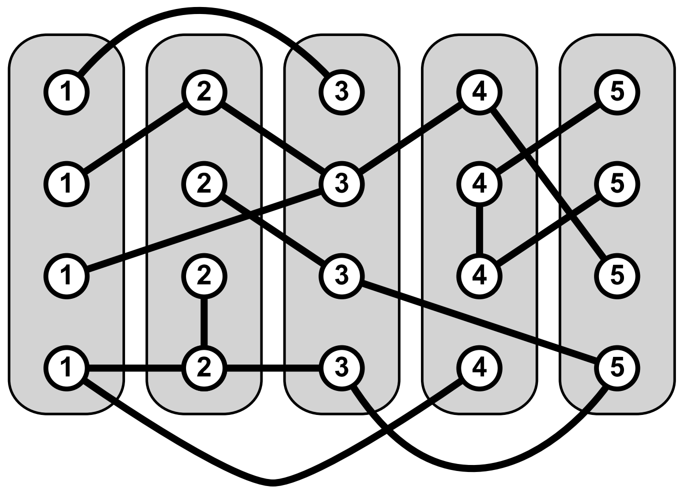
Useful fact

If the probability of success is at least p, then the probability that the algorithm **does not** say "YES" after $1/p$ repetitions is at most

$$(1-p)^{1/p} < \left(e^{-p}\right)^{1/p} = 1/e \approx 0.38$$

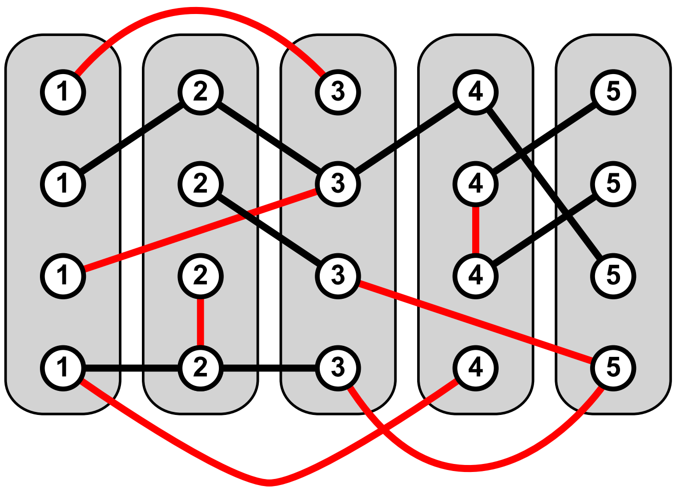
- Thus if $p > k^{-k}$, then error probability is at most $1/e$ after k^k repetitions.
- Repeating the whole algorithm a constant number of times can make the error probability an arbitrary small constant.
- For example, by trying $100 \cdot k^k$ random colorings, the probability of a wrong answer is at most $1/e^{100}$.

Finding a path colored $1 - 2 - \cdots - k$



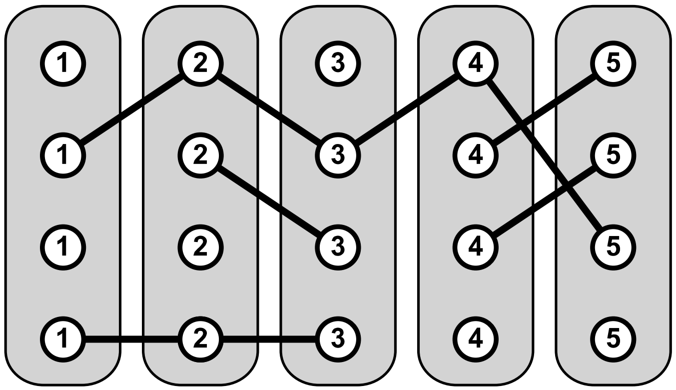
- Edges connecting nonadjacent color classes are removed.
- The remaining edges are directed towards the larger class.
- All we need to check if there is a directed path from class 1 to class k.

Finding a path colored $1 - 2 - \cdots - k$



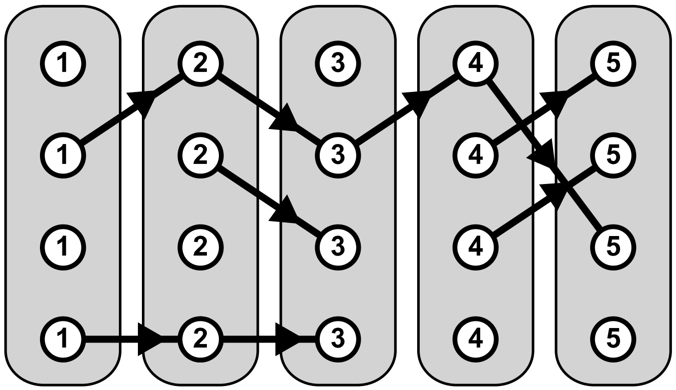
- Edges connecting nonadjacent color classes are removed.
- The remaining edges are directed towards the larger class.
- All we need to check if there is a directed path from class 1 to class k.

Finding a path colored $1 - 2 - \cdots - k$



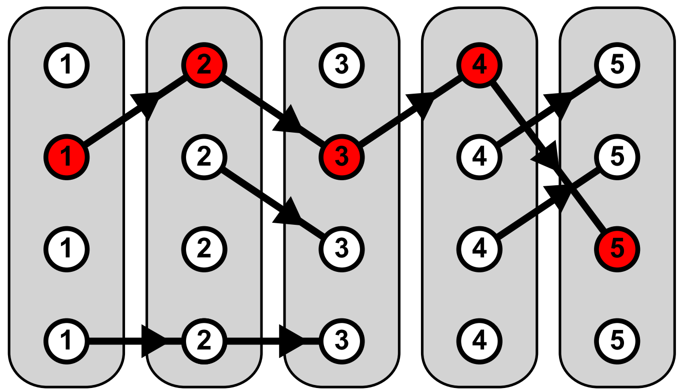
- Edges connecting nonadjacent color classes are removed.
- The remaining edges are directed towards the larger class.
- All we need to check if there is a directed path from class 1 to class k.

Finding a path colored $1 - 2 - \cdots - k$



- Edges connecting nonadjacent color classes are removed.
- The remaining edges are directed towards the larger class.
- All we need to check if there is a directed path from class 1 to class k.

Finding a path colored $1 - 2 - \cdots - k$

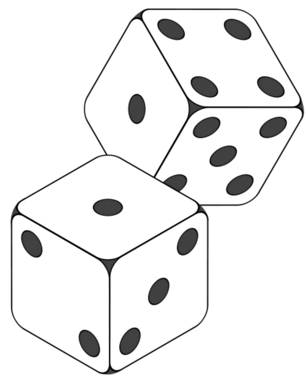


- Edges connecting nonadjacent color classes are removed.
- The remaining edges are directed towards the larger class.
- All we need to check if there is a directed path from class 1 to class k.

Color Coding

k-PATH

Color Coding
success probability: k^{-k}

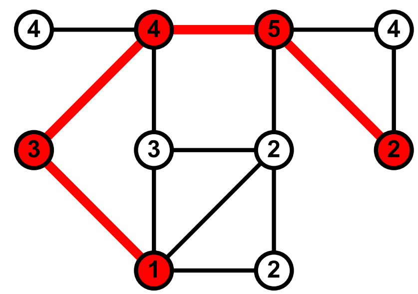


Finding a $1 - 2 - \cdots - k$ colored path

polynomial-time solvable

Improved Color Coding

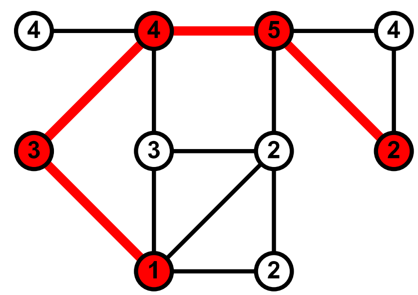
- Assign colors from $[k]$ to vertices $V(G)$ uniformly and independently at random.



- Check if there is a **colorful** path where each color appears exactly once on the vertices; output "YES" or "NO".

Improved Color Coding

- Assign colors from $[k]$ to vertices $V(G)$ uniformly and independently at random.



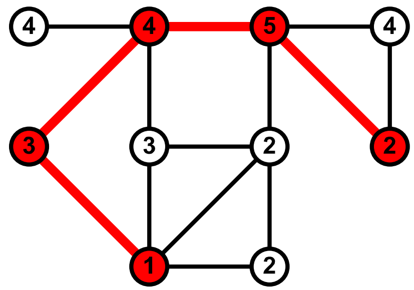
- Check if there is a **colorful** path where each color appears exactly once on the vertices; output "YES" or "NO".
  - If there is no k-path: no **colorful** path exists $\Rightarrow$ "NO".
  - If there is a k-path: the probability that it is **colorful** is

$$\frac{k!}{k^k} > \frac{\left(\frac{k}{e}\right)^k}{k^k} = e^{-k},$$

thus the algorithm outputs "YES" with at least that probability.

Improved Color Coding

- Assign colors from $[k]$ to vertices $V(G)$ uniformly and independently at random.



- Repeating the algorithm $100e^k$ times decreases the error probability to e^{-100}.

How to find a colorful path?

- Try all permutations ($k! \cdot n^{O(1)}$ time)
- Dynamic programming ($2^k \cdot n^{O(1)}$ time)

Finding a colorful path

Subproblems:

We introduce $2^k \cdot |V(G)|$ Boolean variables:

$x(v, C) = \text{TRUE}$ for some $v \in V(G)$ and $C \subseteq [k]$

$\Updownarrow$

There is a path P ending at v such that each color in C appears on P exactly once and no other color appears.

Answer:

There is a colorful path $\iff x(v, [k]) = \text{TRUE}$ for some vertex v.

Finding a colorful path

Subproblems:

We introduce $2^k \cdot |V(G)|$ Boolean variables:

$$x(v, C) = \text{TRUE for some } v \in V(G) \text{ and } C \subseteq [k]$$
$$\Updownarrow$$

There is a path P ending at v such that each color in C appears on P exactly once and no other color appears.

Initialization:

For every v with color r, $x(v, \{r\}) = \text{TRUE}$.

Recurrence:

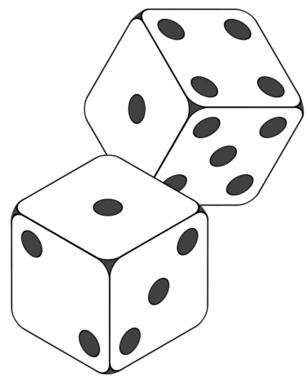
For every v with color r and set $C \subseteq [k]$

$$x(v, C) = \bigvee_{u \in N(v)} x(u, C \setminus \{r\}).$$

Improved Color Coding

k-PATH

Color Coding
success probability: e^{-k}



Finding a colorful path

Solvable in time $2^k \cdot n^{O(1)}$

Derandomization

Definition

A family $\mathcal{H}$ of functions $[n] \to [k]$ is a k-**perfect** family of hash functions if for every $S \subseteq [n]$ with $|S| = k$, there is an $h \in \mathcal{H}$ such that $h(x) \neq h(y)$ for any $x, y \in S$, $x \neq y$.

Theorem

There is a k-perfect family of functions $[n] \to [k]$ having size $2^{O(k)} \log n$ (and can be constructed in time polynomial in the size of the family).

Derandomization

Definition

A family $\mathcal{H}$ of functions $[n] \to [k]$ is a k-**perfect** family of hash functions if for every $S \subseteq [n]$ with $|S| = k$, there is an $h \in \mathcal{H}$ such that $h(x) \neq h(y)$ for any $x, y \in S$, $x \neq y$.

Theorem

There is a k-perfect family of functions $[n] \to [k]$ having size $2^{O(k)} \log n$ (and can be constructed in time polynomial in the size of the family).

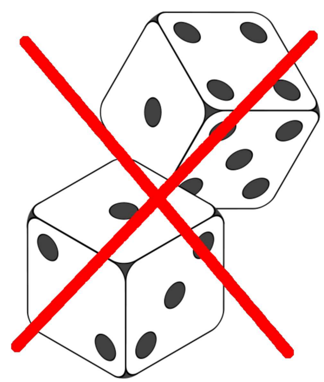
Instead of trying $O(e^k)$ **random colorings,** we go through a k-**perfect family** $\mathcal{H}$ of functions $V(G) \to [k]$.

If there is a solution S
$\Rightarrow$ The vertices of S are colorful for at least one $h \in \mathcal{H}$
$\Rightarrow$ Algorithm outputs "YES".
$\Rightarrow$ k-PATH can be solved in **deterministic** time $2^{O(k)} \cdot n^{O(1)}$.

Derandomized Color Coding

k-PATH

k-perfect family
$2^{O(k)} \log n$ functions



Finding a colorful path

Solvable in time $2^k \cdot n^{O(1)}$