Randomized Parameterized Algorithms
(and de-randomizations)
Design randomized algorithm first, then try to de-randomize it.
**k-Path**

**Input:** G, k

**Question:** Is there a path on k vertices in G?

**Parameter:** k

Will give an algorithm for k-path with running time $(2e)^{k+o(k)}n^{O(1)}$. 
Randomized Algorithm

Consider a random function $f : V(G) \rightarrow \{1...k\}$

For a set $S$ on $k$ vertices, what is the probability that all vertices get a different color?

\[
\frac{k!}{k^k} \geq \frac{1}{e^k}
\]

- Good colorings of $S$.  
- Possible colorings of $S$.  
- Stirling approximation
Randomized Algorithm

Repeat $e^k \cdot t$ times:

1. Pick random $f : V(G) \rightarrow \{1\ldots k\}$
2. Look for a colorful $k$-path.

If the algorithm finds a $k$-path, then $G$ definitely has one.

If there is a $k$-path, the algorithm will find it with probability at least

$$1 - \left(1 - \frac{1}{e^k}\right)^{e^k \cdot t} \geq 1 - \frac{1}{e^t}$$

For $x \geq 2$:

$$\frac{1}{4} \leq (1 - \frac{1}{x})^x \leq \frac{1}{e}$$
Finding a Colorful $k$-Path

«Exercise» Give a $k^{O(k)}$ time algorithm to determine whether a $k$-colored graph has a colorful $k$-path.
Finding a **Colorful k-Path**

Dynamic programming on the **colors** used by partial solutions.

$$T[S,v] = \begin{cases} 
    \text{true} & \text{If exists path on } |S| \text{ vertices ending in } v, \\
    \text{false} & \text{using all colors from } S.
\end{cases}$$

subset of \{1\ldots k\}
Dynamic Programming

\[ T[S, v] = \bigvee_{u \in N(v)} T[S \setminus f(v), u] \]

For each neighbor \( u \) of \( v \)

Is there a path ending in \( u \) that uses all colors in \( S \), except \( v \)'s color?

\( 2^k n \) table entries

\( O(n) \) time to fill each entry.

Total time: \( 2^k n^2 \)
Randomized Algorithm

Repeat \( e^k \cdot 100 \) times:

- Pick random \( f : V(G) \rightarrow \{1...k\} \)
- Look for a colorful \( k \)-path.

If the algorithm finds a \( k \)-path, then \( G \) definitely has one.

If there is a \( k \)-path, the algorithm will find it with probability at least \( 1 - \frac{1}{e^{100}} \)

Total time: \( O((2e)^k n^2) \).
De-randomization

How can we make the algorithm deterministic?

Let $F = f_1 \ldots f_t$ be a family of functions with $f_i : V(G) \rightarrow \{1 \ldots k\}$.

$F$ is a $k$-universal hash family $F$ if for every set $S \subseteq V(G)$ of size $k$, there is an $f_i \in F$ such that $f_i$ makes $S$ colorful.
Deterministic Algorithm

Construct a $k$-universal hash family $F$. For each $f \in F$:

Look for a colorful $k$-path.

If the algorithm finds a $k$-path, then $G$ definitely has one.

If there is a $k$-path, the algorithm will find it.

Total time: $O(t + |F| \cdot 2^{kn^2})$. 

Takes $t$ time

Takes $2^{kn^2}$ time
Constructing Hash Functions

\[ \text{NSS’95} \] Can construct a $k$-perfect hash family $F$ of size $e^{k+o(k)} \log n$ in time $e^{k+o(k)} n \log n$.

$k$-Path in time $(2e)^{k+o(k)} n^{O(1)} \leq 5.44^k n^{O(1)}$
Random Separation: Set Splitting

**Input:** Family $S_1...S_m$ of sets over a universe $U = v_1...v_n$, integer $k$.

**Question:** Is there a coloring $c : U \rightarrow \{0,1\}$ such that at least $k$ sets contain an element colored 0 and an element colored 1?

**Parameter:** $k$

Will give a $2^{kn^{O(1)}}$ time randomized, and a $4^{kn^{O(1)}}$ time deterministic algorithm.
Randomized Algorithm

Pick a random coloring \( c : V(G) \rightarrow \{1,2\} \).
If \( c \) splits at least \( k \) sets, return \( c \).

If the algorithm returns a coloring, then it is correct.

**Claim:** If there is a coloring \( \psi \) that splits at least \( k \) sets, then a random coloring will split at least \( k \) sets with probability at least \( \frac{1}{2^k} \).

The \( 2^k n^{O(1)} \) time randomized algorithm follows directly from the claim.
Proof of claim

Suppose $\psi$ splits the sets $S_1...S_k$.

Make a bipartite graph $G=(A \cup B, E)$ as follows:

- $A$ is a minimal hitting set for $S_1...S_k$ colored 0
- $B$ is a minimal hitting set for $S_1...S_k$ colored 1

For $i$ from 1 to $k$

Add one edge between a vertex in $S_i \cap A$ and a vertex in $S_i \cap B$. 
Set Splitting Graph
Proof of claim, continued.

If \( c \) properly colors \( G \) then all sets \( S_1 \ldots S_k \) are split.

\( G \) has \( \leq k \) edges and at most \( \leq 2k \) vertices

\( G \) has \( k+x \) vertices \( \Rightarrow \) \( G \) has \( \geq x \) components

Probability that \( c \) properly colors \( G \) is at least:

\[
\frac{2^x}{2^{k+x}} = \frac{1}{2^k}
\]
Set Splitting Randomized Algorithm

Repeat $100 \cdot 2^k$ times:

1. Pick a random coloring $c : V(G) \rightarrow \{1...k\}$.
2. If $c$ splits at least $k$ sets, return $c$.

Running time: $O(2^k nm)$

If the algorithm returns a coloring, then it is correct.

If there is a coloring that splits $k$ sets, the algorithm will find one with probability $1 - 1/2^{100}$. 
Let $F = \{c_1 \ldots c_t\}$ be a family of colorings $V(G) \to \{0,1\}$.

$F$ is a $k$-universal coloring family if for every set $S$ on at most $k$ vertices and every way of coloring $S$ there is some $c_i \in F$ which colors $S$ exactly like that.
Set Splitting Algorithm

Construct $2k$-universal coloring family $F$

For each $c \in F$:

If $c$ splits at least $k$ sets, return $c$.

If the algorithm returns a coloring, then it is correct.

If there is a coloring that splits $k$ sets, the algorithm will find one, since the graph $G$ has $\leq 2k$ vertices.

Running time: $O(t + |F|nm)$
Construction of Universal Coloring Families

[NSW’95] Can construct a $k$-universal coloring family $F$ of size $2^{k+o(k)\log n}$ in time $2^{k+o(k)n\log n}$.

(We need a $2k$-universal coloring family)

Set Splitting in time $4^{k+o(k)n^{O(1)}}$. 
Induced Subgraph Isomorphism

**Input:** Graphs $G$ and $H$, $G$ has maximum degree $\Delta$, $|V(H)| = k$

**Question:** Does $G$ contain $H$ as an induced subgraph?

**Parameters:** $\Delta + |V(H)|$

Encodes the $k$-Clique problem

(so FPT just by $k$ is unlikely)

Naive algorithm: $n^{O(k)}$

$|V(G)|$

Will see a $\Delta^{O(k)}$ time algorithm
Random Separation

Will assume $H$ is connected

Color vertices of $G$ red with probability $p$, blue with probability $1-p$

Delete all blue vertices

Determine whether any (red) connected component is equal to $H$ using Graph Isomorphism in time $2^{\text{polylog}(k)}$
Success Probability

If $G$ does not contain $H$ then algorithm always says no

If $G$ contains $H$ then:

- All the vertices of $H$ are colored red with probability $p^k$
- All the neighbors of $H$ (in $G$) are colored blue
  with probability at least $(1 - p)^{\Delta k}$

Success probability: $p^k(1 - p)^{\Delta k} = \frac{1}{\Delta^k} (1 - \frac{1}{\Delta})^{\Delta k} \geq \frac{1}{(4\Delta)^k}$

Set $p = 1/\Delta$
Running time

Each run of the algorithm takes $2^{o(k)}$ time.

Repeat $(4\Delta)^k$ times for constant success probability.

Total runtime: $(4\Delta)^{k+o(k)}$
Derandomization

**Universal Coloring Families** $\rightarrow$ deterministic algorithm for **Induced Subgraph Isomorphism** with running time $2^{O(\Delta k)}$.

This can be improved to $\Delta^{O(k)}$. 
Other variants

**Simple extension 1:** Algorithm for the case where $H$ is not necessarily connected with essentially the same running time.

**Simple extension 2:** Algorithm for Subgraph Isomorphism problem (not induced) with similar ish running time.
Feedback Vertex Set

Feedback Vertex Set (FVS)

IN: $G, k$

Q: Is there a set $S$ of $\leq k$ vertices such that $G \setminus S$ is a forest?
FVS reduction rules

**R1:** Delete vertices of degree $\leq 1$

**R2:** Replace degree 2 vertices by edges (keep multiedges)
\( \alpha \)-cover Lemma

A set \( S \subseteq V(G) \) is an \( \alpha \)-cover if

\[
\sum_{v \in S} d(v) \geq \alpha \cdot \sum_{v \in V(G)} d(v) \quad (= \alpha \cdot 2m)
\]

Lemma: If R1 and R2 do not apply, then every feedback vertex set \( S \) of \( G \) is a \( \frac{1}{4} \)-cover.
**Lemma:** If $R_1$ and $R_2$ do not apply then every feedback vertex is a $\frac{1}{4}$-cover.

**Proof:**

$$\frac{\sum_{v \in S} d(v) - 2}{\sum_{v \in V(G)} d(v) + 2}$$
Algorithm for FVS

while $G$ is not empty
   Apply $R1$ and $R2$ on $G$ exhaustively
   Select a vertex $v$ with prob = $d(v) / 2m$.
   $S := S \cup \{v\}$
   $G := G \setminus v$
   If $|S| > k$ output NO
output $S$

Succeeds with probability $\frac{1}{4}$
Feedback Vertex Set

Runs in $O(k(n+m))$ time, succeeds with $\frac{1}{4^k}$ probability.

So $O(4^k k(n+m))$ time for constant success probability.

Expected output solution size is $\leq 4\text{OPT}$, this is a 4-approximation!
Feedback Vertex Set below $2^n$

Feedback Vertex Set (FVS)

**IN:** $G$, $k$

**Q:** Is there a set $S$ of $\leq k$ vertices such that $G\setminus S$ is a forest?

Saw a $4^k n^{O(1)}$ time algorithm

Can we beat $2^n$?

Branching is tricky

**Next:** $O((2-\frac{1}{4})^n) = O(1.75^n)$ time using $4^k n^{O(1)}$ as black box
Feedback Vertex Set algorithm

Given $n$, $k$, pick integer $t \leq k$.

Pick a set $S$ of size $t$ uniformly at random, put $S$ in solution.

(By deleting $S$ and decreasing parameter by $t$)

Try to extend $S$ to a feedback vertex set of size $\leq k$

(By running $4^k n^{O(1)}$ time algorithm on $(G-S, k-t)$)
Analysis

Success probability: \( \frac{\binom{k}{t}}{\binom{n}{t}} \)

Running time for constant success probability:
\[
\frac{\binom{n}{t}}{\binom{k}{t}} 4^{k-t}
\]

Given \( n \) and \( k \) pick \( t \) so that running time is minimized:
\[
\min_{t \leq k} \frac{\binom{n}{t}}{\binom{k}{t}} 4^{k-t}
\]

Running time:
\[
4^{k-t}n^{O(1)}
\]

Given \( n \), consider the worst \( k \):
\[
\max_{k \leq n} \min_{t \leq k} \frac{\binom{n}{t}}{\binom{k}{t}} 4^{k-t}
\]
Analysis, cont’d

**Lemma:** For all $c > 1$, \( \max_{k \leq n} \min_{t \leq k} \frac{n^k}{k^t} c^{k-t} \leq O((2 - \frac{1}{c})^n) \)

**Corollary:** Feedback Vertex Set in \( O((2 - \frac{1}{4})^n) = O(1.75^n) \) time.
Generalizing

Nothing in the algorithm / analysis was specific to FVS!

Look for a set of size $k$ in a universe of size $n$.

If we can **extend** a set of size $t$ to a solution of size $k$ in time $c^{k-t}n^{O(1)}$ then

We can **find** a solution in time $O((2 - \frac{1}{c})^n)$

Can be fully derandomized.
Techniques

- Chromatic Coding
- Random Separation
- Color Coding
- Picking Random Solution Vertices
- Mod 2 Counting + Isolation
Thank you!