Algoritmos Parametrizados

Métodos Aleatorizados

Lehilton Pedrosa

Segundo Semestre de 2016

Instituto de Computação – Unicamp

Roteiro

1. Métodos Aleatorizados
2. Conjunto de retroalimentação de Vértices
3. Codificação de cores ; caminho mais longo
4. Separação aleatória
5. Codificação cromática
6. Desaleatorização

Métodos Aleatorizados

Algoritmo Aleatorizado

Modelo probabilístico

Algoritmo Aleatorizado

Modelo probabilístico

- $P \subseteq \Sigma^*$ um problema

Algoritmo Aleatorizado

Modelo probabilístico

- $P \subseteq \Sigma^*$ um problema
- $r \geq 0$ o número de bits aleatórios

Algoritmo Aleatorizado

Modelo probabilístico

- $P \subseteq \Sigma^*$ um problema
- $r \geq 0$ o número de bits aleatórios

Seja um algoritmo ALG com parâmetros:

Algoritmo Aleatorizado

Modelo probabilístico

- $P \subseteq \Sigma^*$ um problema
- $r \geq 0$ o número de bits aleatórios

Seja um algoritmo ALG com parâmetros:

- $I \in \Sigma^*$

Algoritmo Aleatorizado

Modelo probabilístico

- $P \subseteq \Sigma^*$ um problema
- $r \geq 0$ o número de bits aleatórios

Seja um algoritmo ALG com parâmetros:

- $I \in \Sigma^*$
- $\omega \in \{0,1\}^r$

Algoritmo Aleatorizado

Modelo probabilístico

- $P \subseteq \Sigma^*$ um problema
- $r \geq 0$ o número de bits aleatórios

Seja um algoritmo ALG com parâmetros:

- $I \in \Sigma^*$
- $\omega \in \{0,1\}^r$

Dizemos que ALG responde R para I com probabilidade:

$$p := \frac{|\{\omega : Alg(I,\omega) = R\}|}{2^r}$$

Algoritmo Monte Carlo

Definição

Um algoritmo Monte Carlo com falsos negativos e probabilidade p é um algoritmo aleatorizado tal que:

Algoritmo Monte Carlo

Definição

Um algoritmo Monte Carlo com falsos negativos e probabilidade p é um algoritmo aleatorizado tal que:

1. Tem tempo de execução limitado

Algoritmo Monte Carlo

Definição

Um algoritmo Monte Carlo com falsos negativos e probabilidade p é um algoritmo aleatorizado tal que:

1. Tem tempo de execução limitado
2. Tem as seguintes probabilidades:

Algoritmo Monte Carlo

Definição

Um algoritmo Monte Carlo com falsos negativos e probabilidade p é um algoritmo aleatorizado tal que:

1. Tem tempo de execução limitado
2. Tem as seguintes probabilidades:

Entrada	Prob. da resposta correta
Instância não: $I \notin P$	1
Instância sim: $I \in P$	p

Prob. de encontrar a solução

INST	RESP	TIPO
SIM	SIM	Verd. Posit
SIM	NÃO	FALSO Negativo
NÃO	SIM	—
NÃO	NÃO	—

Amplificação de probabilidade

O algoritmo ALG′ amplifica a probabilidade de ALG:

Amplificação de probabilidade

O algoritmo ALG′ amplifica a probabilidade de ALG:

ALG′(I, t)

1. Para $i = 1, \ldots, t$:

Amplificação de probabilidade

O algoritmo ALG′ amplifica a probabilidade de ALG:

ALG′(I, t)

1. Para $i = 1, \ldots, t$: execute ALG(I)

Amplificação de probabilidade

O algoritmo ALG′ amplifica a probabilidade de ALG:

ALG′(I, t)

1. Para $i = 1, \ldots, t$: execute ALG(I)
2. Se todas execução de ALG responder não,

Amplificação de probabilidade

O algoritmo ALG′ amplifica a probabilidade de ALG:

ALG′(I, t)

1. Para $i = 1, \ldots, t$: execute ALG(I)
2. Se todas execução de ALG responder não, responda não.

Amplificação de probabilidade

O algoritmo ALG′ amplifica a probabilidade de ALG:

ALG′(I, t)

1. Para $i = 1, \ldots, t$: execute ALG(I)
2. Se todas execução de ALG responder não, responda não.
3. Senão,

Amplificação de probabilidade

O algoritmo ALG′ amplifica a probabilidade de ALG:

ALG′(I, t)

1. Para $i = 1, \ldots, t$: execute ALG(I)
2. Se todas execução de ALG responder não, responda não.
3. Senão, responda sim.

Amplificação de probabilidade

O algoritmo ALG′ amplifica a probabilidade de ALG:

ALG′(I, t)

1. Para $i = 1, \ldots, t$: execute ALG(I)
2. Se todas execução de ALG responder não, responda não.
3. Senão, responda sim.

O algoritmo ALG′ tem probabilidade $p' = e^{-pt}$.

Amplificação de probabilidade

O algoritmo ALG′ amplifica a probabilidade de ALG:

ALG′(I, t)

1. Para $i = 1, \ldots, t$: execute ALG(I)
2. Se todas execução de ALG responder não, responda não.
3. Senão, responda sim.

O algoritmo ALG′ tem probabilidade $p' = e^{-pt}$.

Observação para FPT:

Amplificação de probabilidade

O algoritmo ALG′ amplifica a probabilidade de ALG:

ALG′(I, t)

1. Para $i = 1, \ldots, t$: execute ALG(I)
2. Se todas execução de ALG responder não, responda não.
3. Senão, responda sim.

O algoritmo ALG′ tem probabilidade $p' = e^{-pt}$.

Observação para FPT:

- Se $t = \lceil \frac{1}{p} \rceil$,

$p = \frac{1}{n} \Rightarrow e^{-pt} = e^{-1}$

$t = \lceil \frac{1}{p} \rceil$

Amplificação de probabilidade

O algoritmo ALG′ amplifica a probabilidade de ALG:

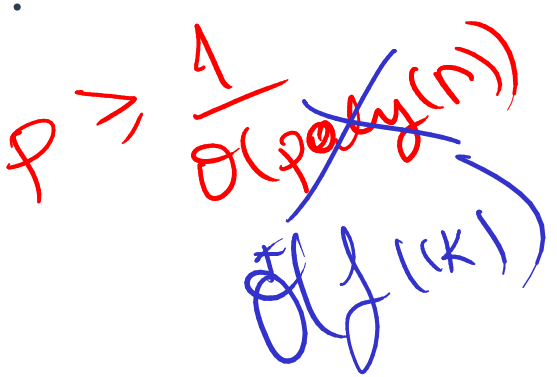
ALG′(I, t)

1. Para $i = 1, \ldots, t$: execute ALG(I)
2. Se todas execução de ALG responder não, responda não.
3. Senão, responda sim.

O algoritmo ALG′ tem probabilidade $p' = e^{-pt}$.

Observação para FPT:

- Se $t = \lceil \frac{1}{p} \rceil$, então p' é constante



Amplificação de probabilidade

O algoritmo ALG′ amplifica a probabilidade de ALG:

ALG′(I, t)

1. Para $i = 1, \ldots, t$: execute ALG(I)
2. Se todas execução de ALG responder não, responda não.
3. Senão, responda sim.

O algoritmo ALG′ tem probabilidade $p' = 1 - e^{-pt}$.

Observação para FPT:

- Se $t = \lceil \frac{1}{p} \rceil$, então p' é constante
- Se ALG tem $p = 1/\mathcal{O}^*(f(k))$ e tempo $\mathcal{O}^*(g(k))$,

Amplificação de probabilidade

O algoritmo ALG′ amplifica a probabilidade de ALG:

ALG′(I, t)

1. Para $i = 1, \ldots, t$: execute ALG(I)
2. Se todas execução de ALG responder não, responda não.
3. Senão, responda sim.

O algoritmo ALG′ tem probabilidade $p' = e^{-pt}$.

$p' = 1 - e^{-pt} = 1 - \frac{1}{e}$

Observação para FPT:

- Se $t = \lceil \frac{1}{p} \rceil$, então p' é constante
- Se ALG tem $p = 1/\mathcal{O}^*(f(k))$ e tempo $\mathcal{O}^*(g(k))$, então ALG' tem p' constante e tempo $\mathcal{O}^*(f(k)g(k))$.

Conjunto de retroalimentação de Vértices

Dado G, encontra $F \subseteq V(G)$ tq $G - X$ não tem ciclos

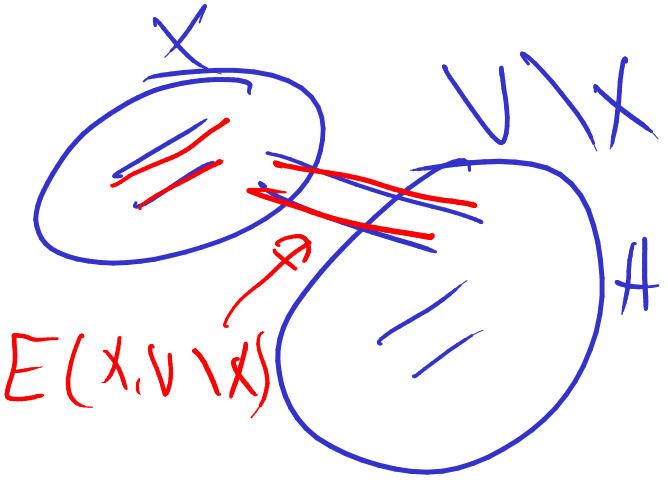
Conjunto de retroalimentação de Vértices

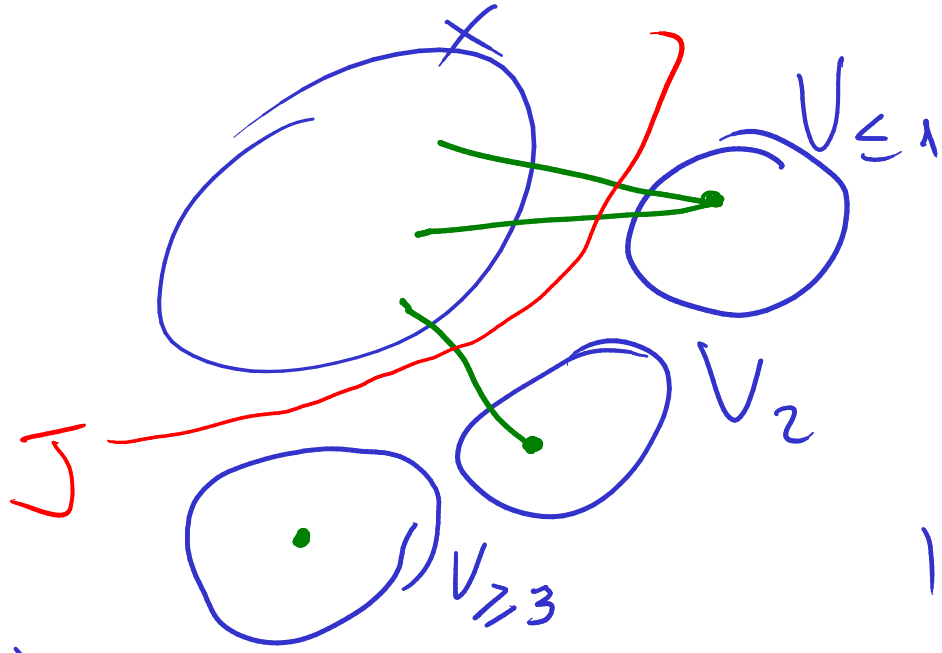
Lema

Se G é multigrafo com $d(v) \geq 3$ para $v \in V(G)$ e X é conjunto de retroalimentação de vértices, então mais da metade das arestas incidem em X.

Seja X ejnt.-cjrctmo. Def $H := G - X$

Como todo aresta em $E(X, V \setminus X)$ tem extremo em X, então é suficiente $J := E(X, V \setminus X) > E(H)$.

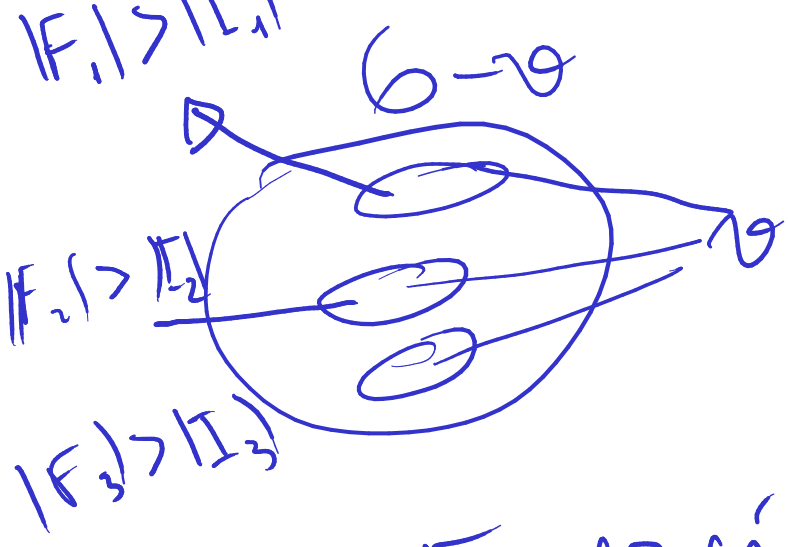




- Seja $V_{\leq 1}$ os vértices v tq $d_H(v) \leq 1$ etc.

- Temos

$$|J| \geq |V_1| \cdot 2 + |V_2| \cdot 1$$



- Mas como H é fecunda temos $|V_1| > |V_{\geq 3}|$

$$\Rightarrow |J| \geq |V_1| + |V_2| + |V_3|$$
$$\geq |E[H]| + 1$$
$$> |E[H]|.$$

Exercício!

FPT aleatorizado

Teorema

Dados G e k, existe algoritmo polinomial que

FPT aleatorizado

Teorema

Dados G e k, existe algoritmo polinomial que

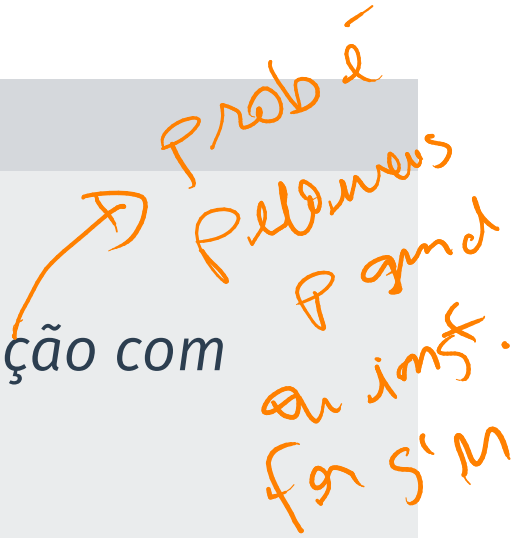
1. *ou encontra um conjunto de retroalimentação com tamanho k;*

FPT aleatorizado

Teorema

Dados G e k, existe algoritmo polinomial que

1. *ou encontra um conjunto de retroalimentação com tamanho k;*
2. *ou reponde não.*



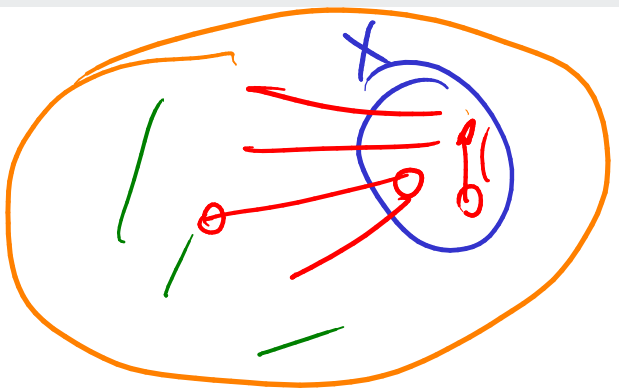
FPT aleatorizado

Teorema

Dados G e k, existe algoritmo polinomial que

1. *ou encontra um conjunto de retroalimentação com tamanho k;*
2. *ou reponde não.*

Se G tiver conjunto de retroalimentação de tamanho k, então a probabilidade de sucesso é pelo menos 4^{-k}.



Algoritmo: (G, K)

1) Aplique as reduções $\{VS.\{1,\dots,5\}$
2) Obtenha uma inst. reduzida (G', K')
   (seja X_0 os vértices obrigatórios)
3) Se G' é vazio, devolva X_0.
4) Escolha $e \in E(G')$ uniformemente
5) Escolha um extremo v de e com prob. $1/2$.
6) Seja $G'' \leftarrow G' - v$; $K'' \leftarrow K' - 1$
7) Chame o alg. recursivamente com (G'', K'')
   (obtenha X'')
8) Devolva $X_0 \cup \{v\} \cup X''$.

Análise: INDUÇÃO. Seja X^* uma solução com $|X^*| \leq k$

BASE: $k = 0$, OK

PASSO: Seja $k > 0$.

→ Se $\bar{m}$ tiver arestas OK

→ Senão 1º) $p(v \in X^*) \geq \frac{1}{4}$.

2º) $p(X'' = X^* \setminus \{v\}) \geq \frac{1}{4^{k-1}}$

por indução.

Daí, $p(X = X^*) = p(v \in X^*) \cdot$

$p(X'' = X^* \setminus \{v\} \mid v \in X^*)$ ∎

FPT aleatorizado

Corolário

Existe um algoritmo aleatorizado que, em tempo $4^k n^{\mathcal{O}(1)}$ encontra um conjunto de retroalimentação com tamanho k de G, ou reporta falha. Se a instância for sim, então devolve um conjunto com probabilidade constante.

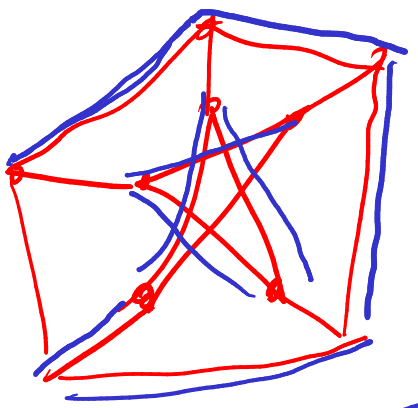
Prova: executa o algoritmo anterior 4^k vezes.

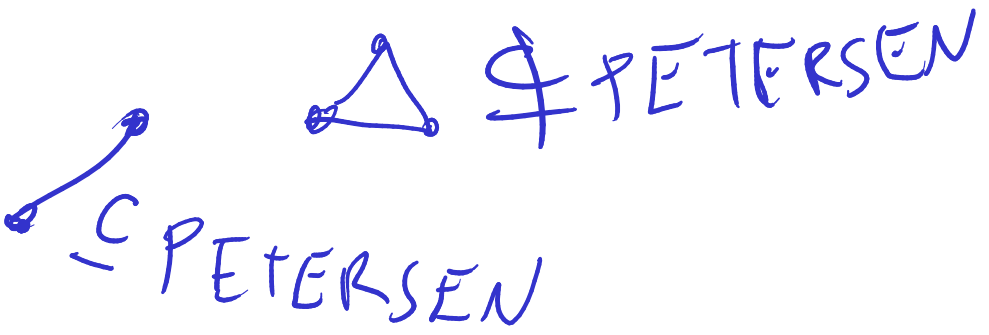
Codificação de cores

Isomorfismo de subgrafo

Isomorfismo de subgrafo

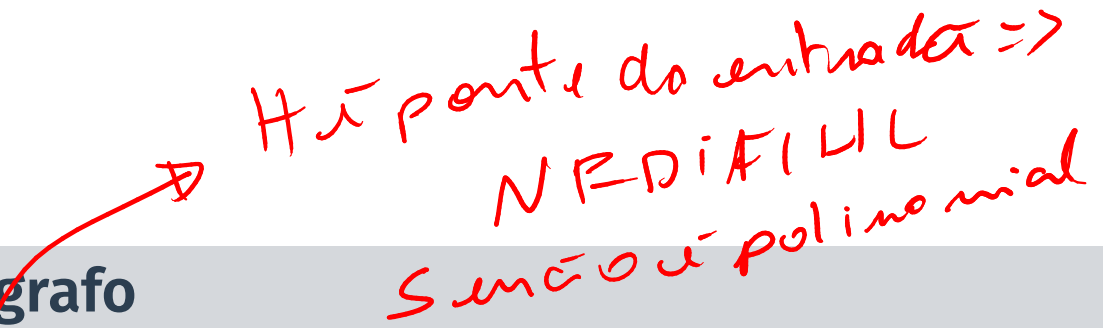
Dados grafos G e H, H é subgrafo de G?





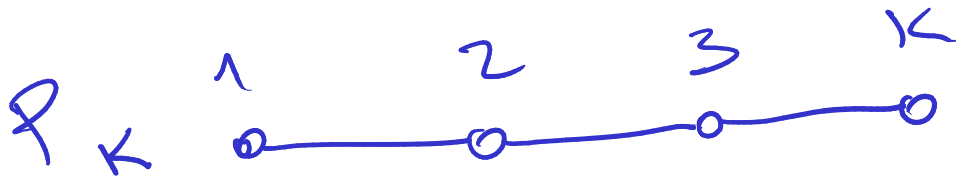
Tem P_3 subgrafo;
$P_4, P_5, P_6, P_7, P_8, P_9, P_{10}$

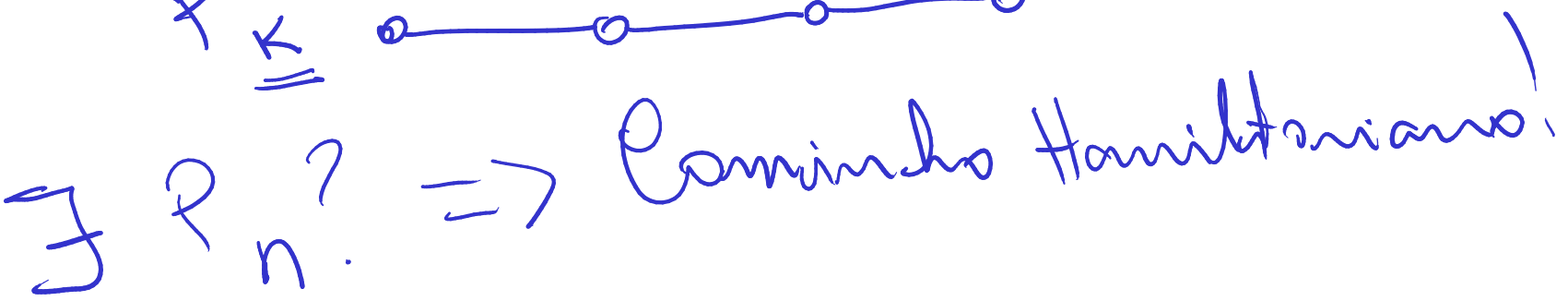
Isomorfismo de subgrafo



Isomorfismo de subgrafo
Dados grafos G e H, H é subgrafo de G?

Problema do Caminho mais longo: caso particular quando H é um caminho P_k, com k vértices.

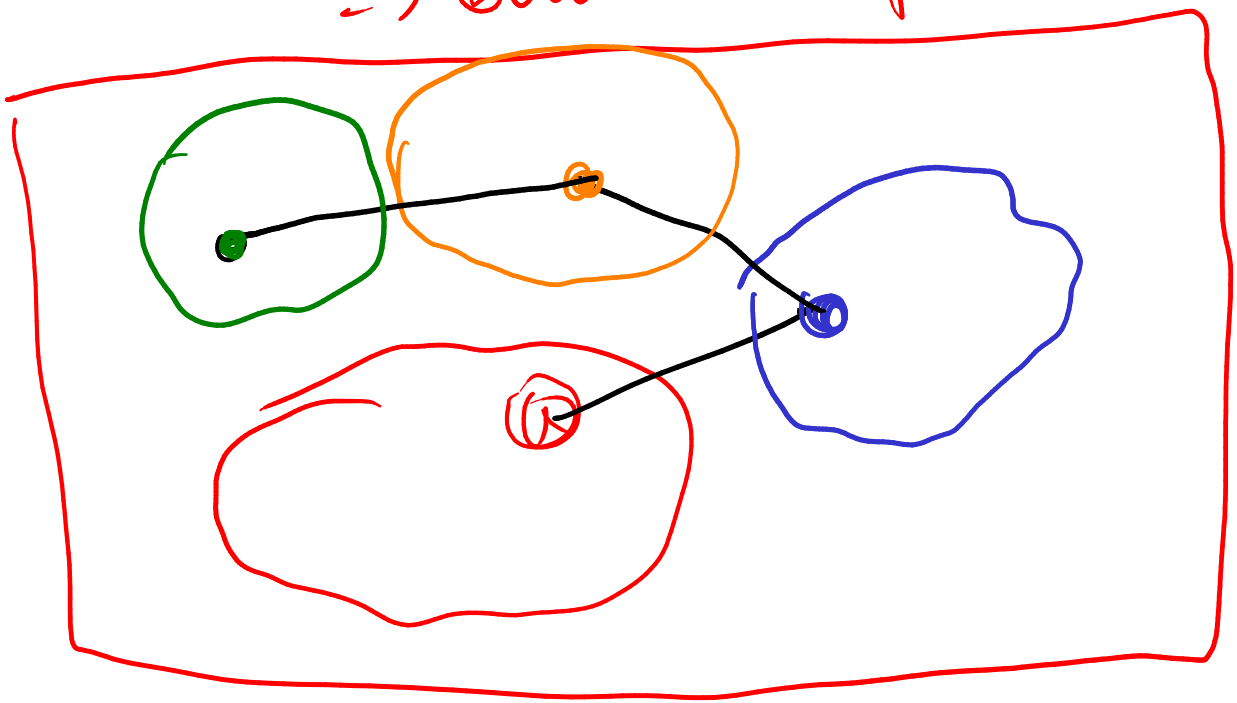




Algoritmo natural:

- Marco os vértices visitados.
- Visitamos no máximo K

=> Quantas possibilidades? $\binom{n}{k}$

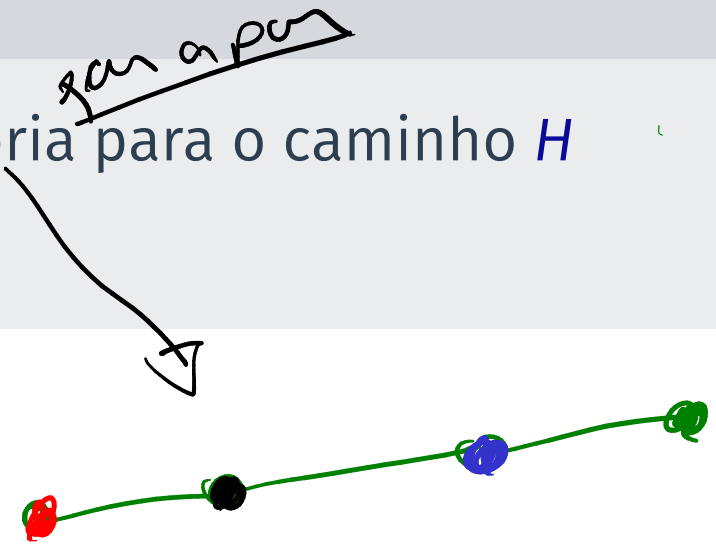


=> onts possib?

Codificação de Cores

Ideia

- damos uma coloração própria para o caminho H



Codificação de Cores

Ideia

- damos uma coloração própria para o caminho H
- adivinhamos as cores no grafo G

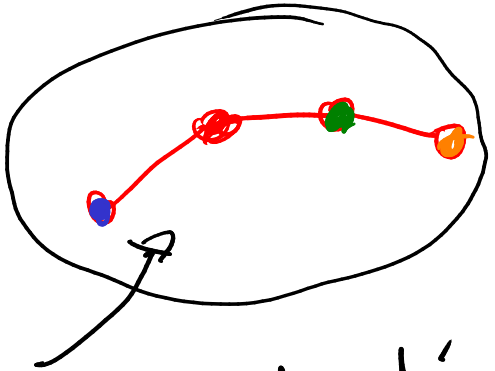
Codificação de Cores

Ideia

- damos uma coloração própria para o caminho H
- adivinhamos as cores no grafo G

Flexibilidade da aleatorização:

- como se testássemos todas possibilidades



Codificação de Cores

Ideia

- damos uma coloração própria para o caminho H
- adivinhamos as cores no grafo G

Flexibilidade da aleatorização:

- como se testássemos todas possibilidades
- mas com tempo de execução para somente uma

Codificação de Cores

Ideia

- damos uma coloração própria para o caminho H
- adivinhamos as cores no grafo G

Flexibilidade da aleatorização:

- como se testássemos todas possibilidades
- mas com tempo de execução para somente uma

Dificuldades:

Codificação de Cores

Ideia

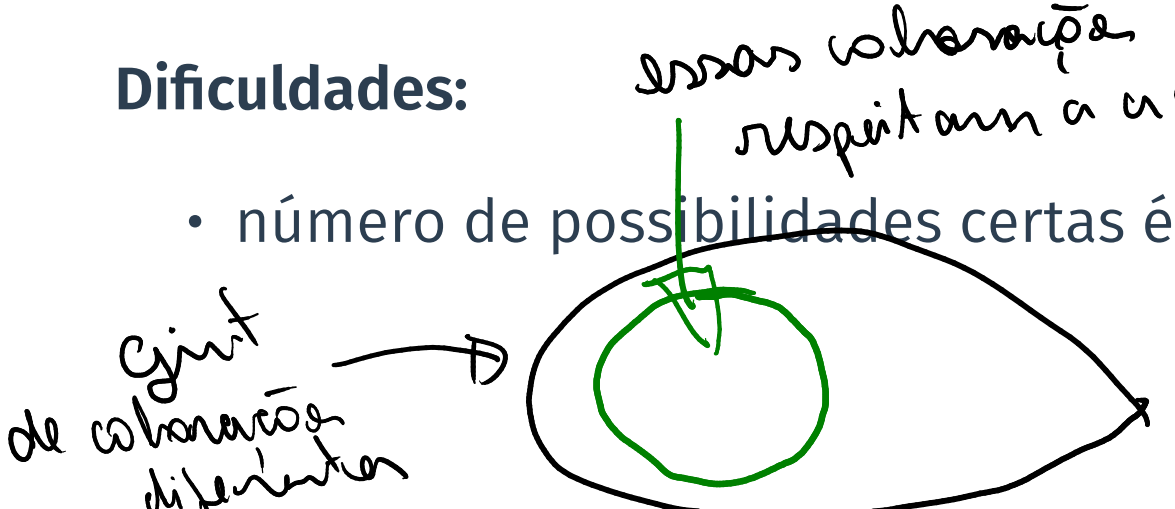
- damos uma coloração própria para o caminho H
- adivinhamos as cores no grafo G

Flexibilidade da aleatorização:

- como se testássemos todas possibilidades
- mas com tempo de execução para somente uma

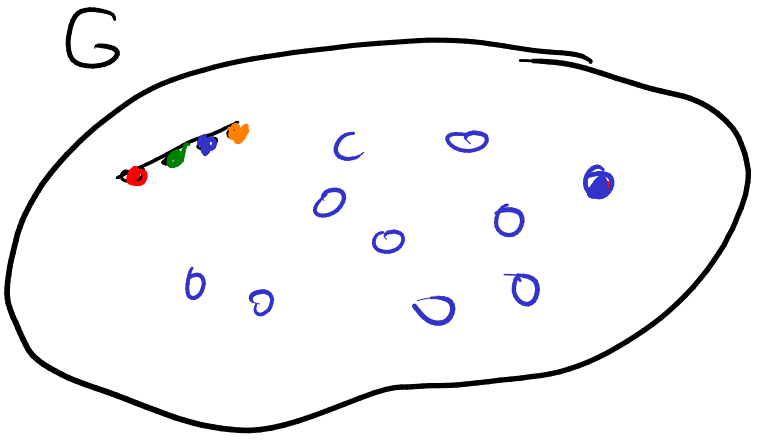
Dificuldades:

- número de possibilidades certas é "grande"



Contando o número de possibilidades

Se k for pequeno comparado a n, então precisamos acertar as cores de poucos vértices!



Contando o número de possibilidades

Se k for pequeno comparado a n, então precisamos acertar as cores de poucos vértices!

Lema

Seja U um conjunto de n elementos e $X \subseteq U$ com $|X| = k$.

Contando o número de possibilidades

Se k for pequeno comparado a n, então precisamos acertar as cores de poucos vértices!

Lema

Seja U um conjunto de n elementos e $X \subseteq U$ com $|X| = k$. Seja $\chi : U \to [k]$ uma coloração de U, escolhida aleatória e uniformemente.

Contando o número de possibilidades

Se k for pequeno comparado a n, então precisamos acertar as cores de poucos vértices!

Lema

Seja U um conjunto de n elementos e $X \subseteq U$ com $|X| = k$. Seja $\chi : U \to [k]$ uma coloração de U, escolhida aleatória e uniformemente.

Então a probabilidade de que χ induz uma coloração própria de X é pelo menos e^{-k}.

para cada

colorações que induzem coloração própria para cada de X:

$$(\underbrace{x_1, x_2, \ldots, x_k}_{k\ k-1\ \ldots\ 1}, \underbrace{-, -, -, -, -}_{k\ k\ k\ k\ k\ k})$$

$$= k! \cdot k^{n-k}$$

$$\# \text{colorações} = k^n \implies \frac{k! \cdot k^{n-k}}{k^n} = \frac{k!}{k^k} \geq \frac{(k/e)^k}{k^k} = e^{-k}$$

Encontrando um caminho colorido

Nos concentramos no seguinte problema:

- Dados G e uma coloração de $V(G)$ com k cores;

Encontrando um caminho colorido

Nos concentramos no seguinte problema:

- Dados G e uma coloração de $V(G)$ com k cores;
- Queremos um caminho de G com as k cores

Encontrando um caminho colorido

Nos concentramos no seguinte problema:

- Dados G e uma coloração de $V(G)$ com k cores;
- Queremos um caminho de G com as k cores

Ideia

- Sem cores:

Encontrando um caminho colorido

Nos concentramos no seguinte problema:

- Dados G e uma coloração de $V(G)$ com k cores;
- Queremos um caminho de G com as k cores

Ideia

- Sem cores:
  - guardamos que vértices foram visitados: tempo $\binom{n}{k}$

Encontrando um caminho colorido

Nos concentramos no seguinte problema:

- Dados G e uma coloração de $V(G)$ com k cores;
- Queremos um caminho de G com as k cores

Ideia

- Sem cores:
  - guardamos que vértices foram visitados: tempo $\binom{n}{k}$
- Com cores:

Encontrando um caminho colorido

Nos concentramos no seguinte problema:

- Dados G e uma coloração de $V(G)$ com k cores;
- Queremos um caminho de G com as k cores

Ideia

- Sem cores:
  - guardamos que vértices foram visitados: tempo $\binom{n}{k}$
- Com cores:
  - precisamos guardar que cores foram visitadas 2^k

Encontrando um caminho colorido

Nos concentramos no seguinte problema:

- Dados G e uma coloração de $V(G)$ com k cores;
- Queremos um caminho de G com as k cores

Ideia

- Sem cores:
  - guardamos que vértices foram visitados: tempo $\binom{n}{k}$
- Com cores:
  - precisamos guardar que cores foram visitadas

Lema

Seja $\chi : V(G) \to [k]$.

Encontrando um caminho colorido

Nos concentramos no seguinte problema:

- Dados G e uma coloração de $V(G)$ com k cores;
- Queremos um caminho de G com as k cores

Ideia

- Sem cores:
  - guardamos que vértices foram visitados: tempo $\binom{n}{k}$
- Com cores:
  - precisamos guardar que cores foram visitadas

Lema

Seja $\chi : V(G) \to [k]$. Existe algoritmo que verifica em tempo $2^k n^{\mathcal{O}(1)}$ se G contém um caminho com k vértices e k cores; se sim, também devolve o caminho.

Seja $V_1, V_2, \ldots V_k$ a partição induzida por k.

Subproblema:

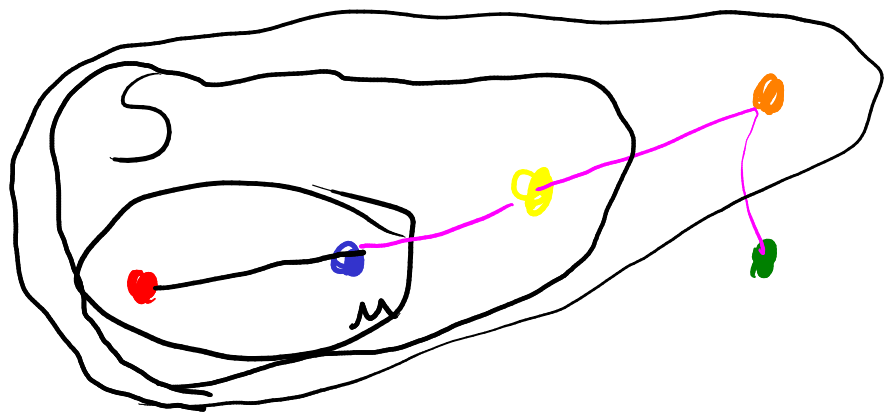
Seja $S \subseteq [k]$

$u \in \bigcup_{i \in S} V_i$

Exemplo:

$S = \{\text{verm, azul}\}$

u é um vértice verm. ou azul.



Existe um caminho que passa pelas cores de S e termina em u?

$\Rightarrow$ PATH(S, u).

Caso base: $S = \{i\} \Rightarrow PATH(S, u) = T \ \forall v \in V_i$

$PATH(S, u) = F \ \forall v \notin V_i$

Caso geral: quando ou que dou certo

$$PATH(S, v) = \begin{cases} \bigvee \{PATH(S \setminus \{\chi(v)\}, u) : uv \in E\} \\ \quad \text{se } \chi(v) \in S \\ F \quad \rightarrow \text{c. c.} \end{cases}$$

$\Rightarrow$ tamanho da tabela: $2^k \cdot n$.

□

Algoritmo para Caminho mais longo

Teorema

Existe um algoritmo aleatorizado que, dada uma instância do Caminho mais longo (G, k), em tempo $(2e)^k n^{\mathcal{O}(1)}$, ou reporta falha, ou encontra um caminho de tamanho pelo menos k. Se a instância for sim, então ele encontra uma solução com probabilidade constante.

Prova: Algoritmo:
1) Colora G aleatoriamente
2) Executa P.P.
3) Repita e^k vezes.

Separação aleatória

Separação aleatória

Ideia

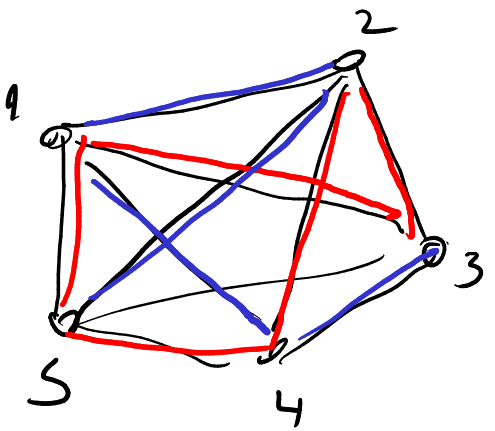
Colorimos arestas de forma que:

Separação aleatória

Ideia

Colorimos arestas de forma que:

- arestas das solução sejam vermelhas
- arestas adjacentes seja azuis



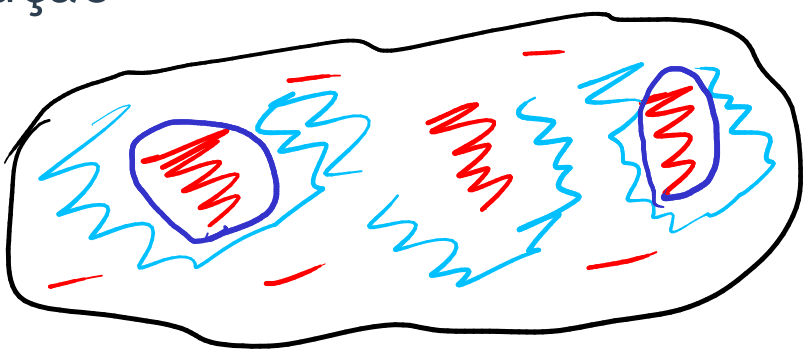
Separação aleatória

Ideia

Colorimos arestas de forma que:

- arestas das solução sejam vermelhas
- arestas adjacentes seja azuis

$\rightarrow$ componentes conexas vermelhas são “componentes” da solução



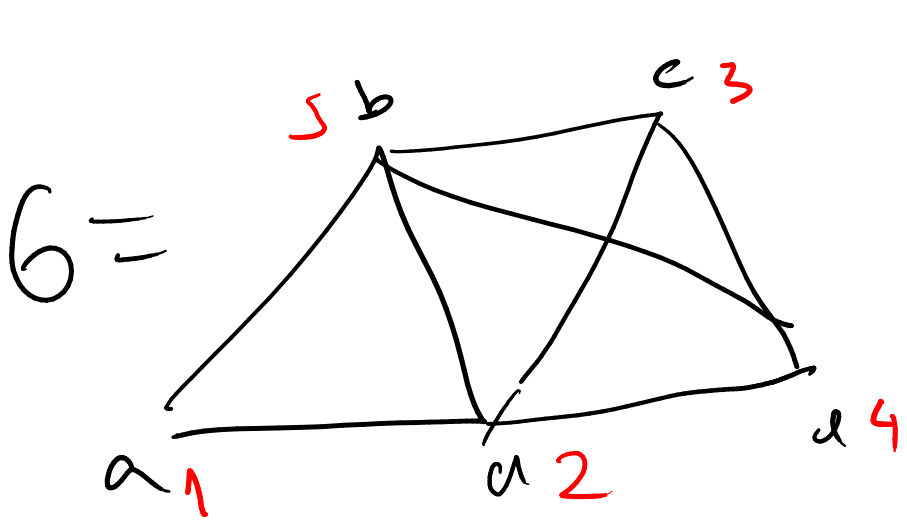
Isomorfismo de subgrafo

Seja (G, H) uma instância sim de Isomorfismo de subgrafo:

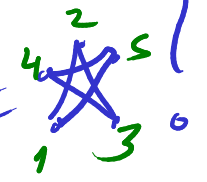
Isomorfismo de subgrafo

Seja (G, H) uma instância sim de Isomorfismo de subgrafo:

- i.e., existe $\hat{H} \subseteq G$ tal que $\hat{H} \cong H$



Existe $H = \Delta$?
SIM

Existe $H =$  ?
SIM

Existe $H = K_5$?
NÃO

Isomorfismo de subgrafo

Seja (G, H) uma instância sim de Isomorfismo de subgrafo:

- i.e., existe $\hat{H} \subseteq G$ tal que $\hat{H} \cong H$

Coloração independente: pintamos cada aresta de G de vermelho ou azul, com probabilidade $1/2$ para cada cor.

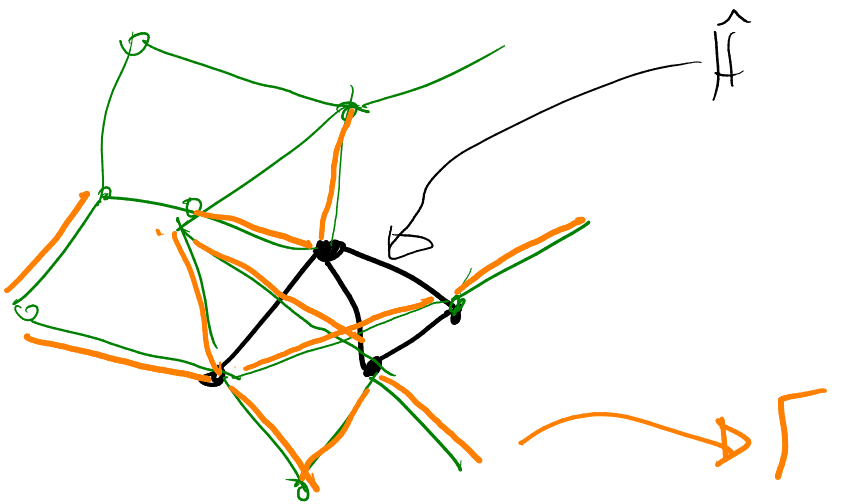
Isomorfismo de subgrafo

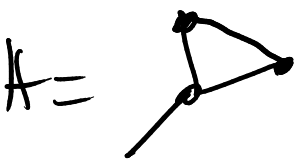
Seja (G, H) uma instância sim de Isomorfismo de subgrafo:

- i.e., existe $\hat{H} \subseteq G$ tal que $\hat{H} \cong H$

Coloração independente: pintamos cada aresta de G de vermelho ou azul, com probabilidade $1/2$ para cada cor.

Seja Γ o conjunto das arestas adjacentes de $\hat{H}$.





Isomorfismo de subgrafo

Seja (G, H) uma instância sim de Isomorfismo de subgrafo:

- i.e., existe $\hat{H} \subseteq G$ tal que $\hat{H} \cong H$

Coloração independente: pintamos cada aresta de G de vermelho ou azul, com probabilidade $1/2$ para cada cor.

Seja Γ o conjunto das arestas adjacentes de H.

Coloração bem-sucedida

1. $\hat{H}$ é vermelho
2. Γ é azul

Isomorfismo de subgrafo

Seja (G, H) uma instância sim de Isomorfismo de subgrafo:

- i.e., existe $\hat{H} \subseteq G$ tal que $\hat{H} \cong H$

Coloração independente: pintamos cada aresta de G de vermelho ou azul, com probabilidade $1/2$ para cada cor.

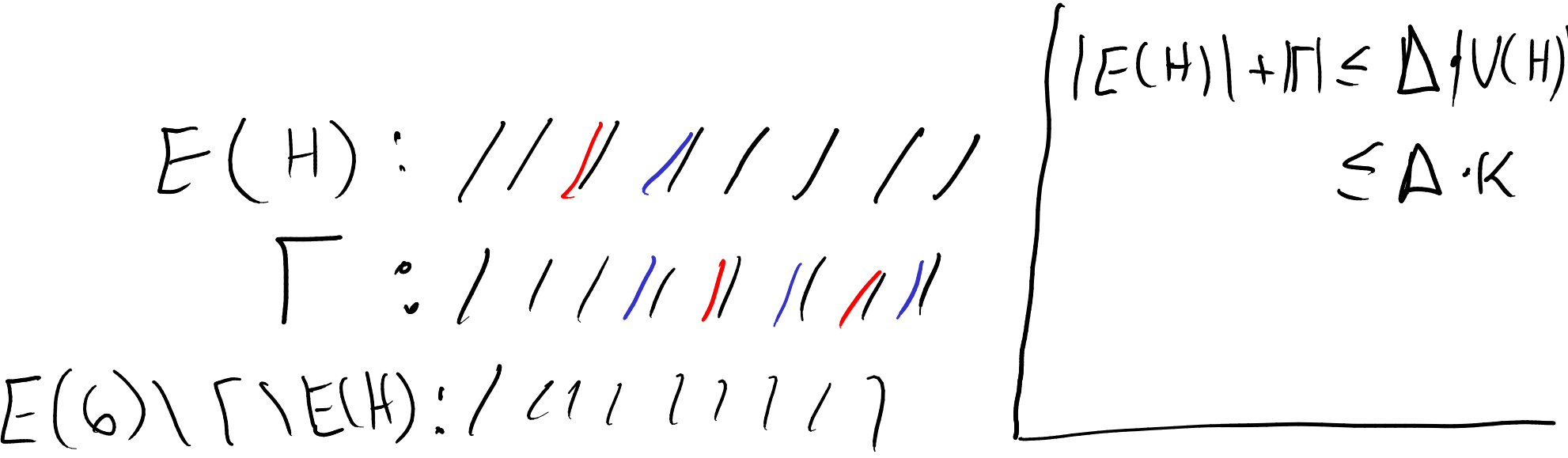
Seja Γ o conjunto das arestas adjacentes de H.

Coloração bem-sucedida

1. H é vermelho
2. Γ é azul

Lema

A probabilidade de obter uma coloração independente bem-sucedida é pelo meno $1/2^{dk}$.



1) $Pr(E(H) \text{ sin marcar}) = \frac{1}{2^{|E(H)|}}$

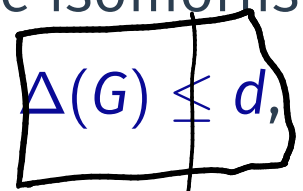
2) $Pr(\Gamma \text{ sin azul}) = \frac{1}{2^{|\Gamma|}}$

$Pr(\text{col. sin bur. suc}) = \frac{1}{2^{|E(H)|}} \cdot \frac{1}{2^{|\Gamma|}} \geq \frac{1}{2^{\Delta K}}$

□

Algoritmo para isomorfismo

- acreditamos que isomorfismo não é **FPT**
- caso particular: $\Delta(G) \leq d$, para algum d fixo



→ generaliza clique:

Clique = Isomorfismo de subgrafo com $H = K_k$.

Parametrizamos por $d + k$!

Algoritmo para isomorfismo

- acreditamos que isomorfismo não é **FPT**
- caso particular: $\Delta(G) \leq d$, para algum d fixo

Teorema

Existe um algoritmo que dados (G, H), com $\Delta(G) \leq d$, ou reporta falha ou decide que existe um subgrafo de G isomorfo a H que executa em tempo $2^{dk} k! n^{\mathcal{O}(1)}$.

Algoritmo para isomorfismo

- acreditamos que isomorfismo não é **FPT**
- caso particular: $\Delta(G) \leq d$, para algum d fixo

Teorema

Existe um algoritmo que dados (G, H), com $\Delta(G) \leq d$, ou reporta falha ou decide que existe um subgrafo de G isomorfo a H que executa em tempo $2^{dk} k! n^{\mathcal{O}(1)}$.

Se a reposta for sim, então ele devolve o subgrafo com probabilidade constante.

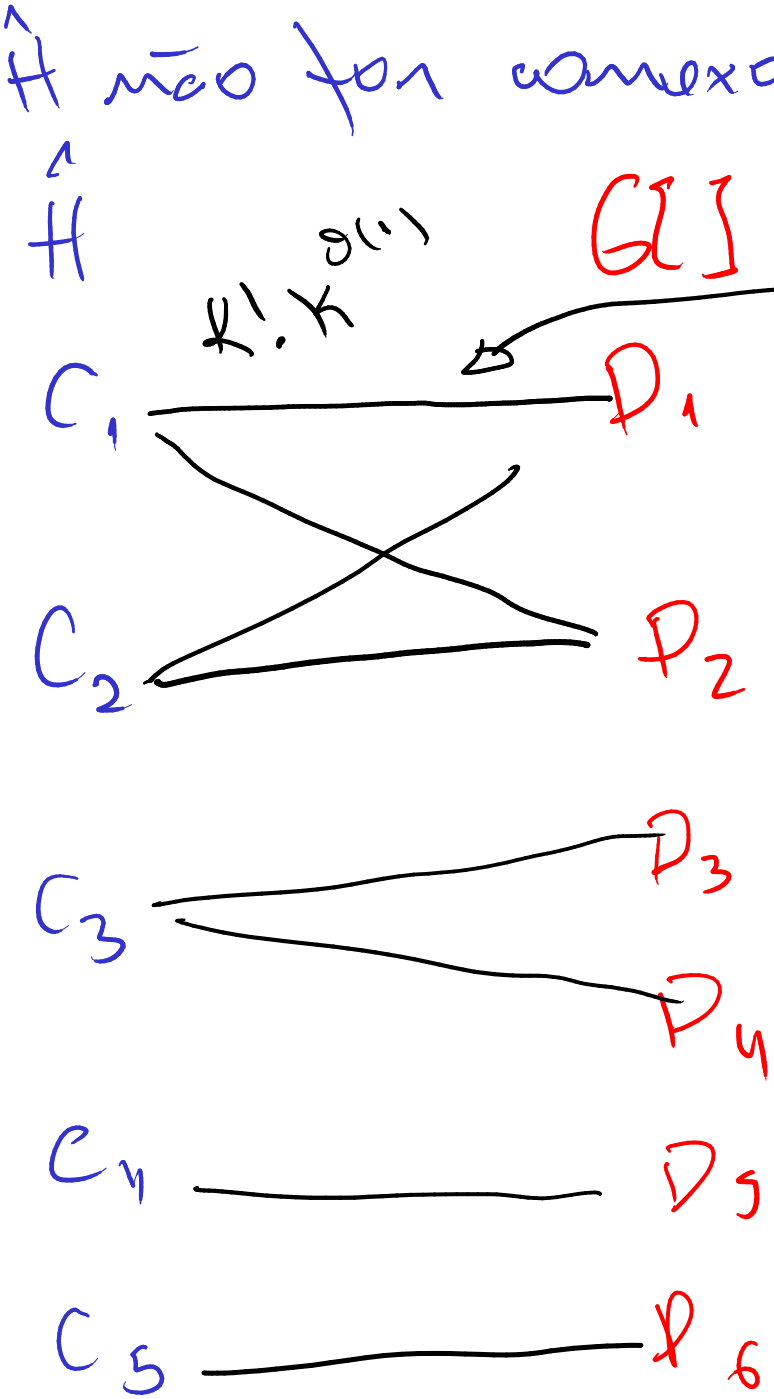
Prova:

1) Obtenha uma colaboração ind. de aresta.

A) Suponha que H é conexo

- Se ainda for sim, com prob. $\geq 1/2^{kd}$
  $\hat{H}$ é mandado e T é agud.
- Daí, $\hat{H}$ é uma comp. conexa de $G[\text{arestas mandadas}]$
- Comparamos com cada comp. C de $G[\,]$:
  - Se $|V(C)| \neq |V(H)|$ ou $|E(C)| \neq |E(H)|$, testo próximo.
  - Senão, testo por força bruta: $k! \cdot k^{O(1)}$

B) Se $\hat{H}$ não for conexo.



adiciona aresta se $C_1 \cong D_1$

a) Constrói o grafo bipartido em tempo:

$\leq K \cdot n$ passos

$\times$

$k! \cdot k^{O(1)}$ por par

$= k! \cdot k^{O(1)}$.

b) Verifica se existe matching que cobre $A \cup B$

Codificação cromática

Problema do d-agrupamento

Problema de edição de arestas:

Problema do d-agrupamento

Problema de edição de arestas:

- queremos adicionar ou remover arestas

Problema do d-agrupamento

Problema de edição de arestas:

- queremos adicionar ou remover arestas
- obter um novo grafo de um certa classe

Problema do d-agrupamento

Problema de edição de arestas:

- queremos adicionar ou remover arestas
- obter um novo grafo de um certa classe

Termos:

- $A \subseteq \binom{V(G)}{2}$ é um conjunto de adjacências

Problema do d-agrupamento

Problema de edição de arestas:

- queremos adicionar ou remover arestas
- obter um novo grafo de um certa classe

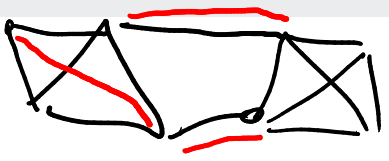
Termos:

- $A \subseteq \binom{V(G)}{2}$ é um conjunto de adjacências
- um d-agrupamento é a união disjunta de d cliques

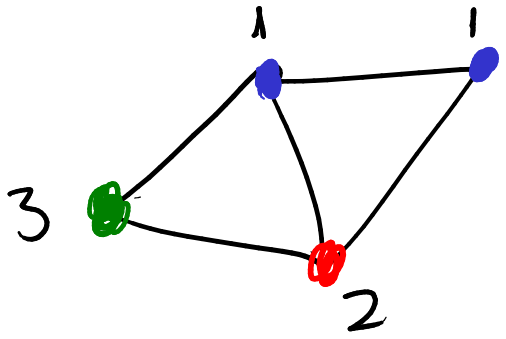
Problema do d-agrupamento de grafo

Dado um grafo G e um inteiro k, existe um conjunto de adjacências A, com $|A| \leq k$, tal que o grafo $(V(G), E(G) \triangle A)$ é um d-agrupamento?

$d = 2$ $k = 3$ $G =$



Codificação cromática

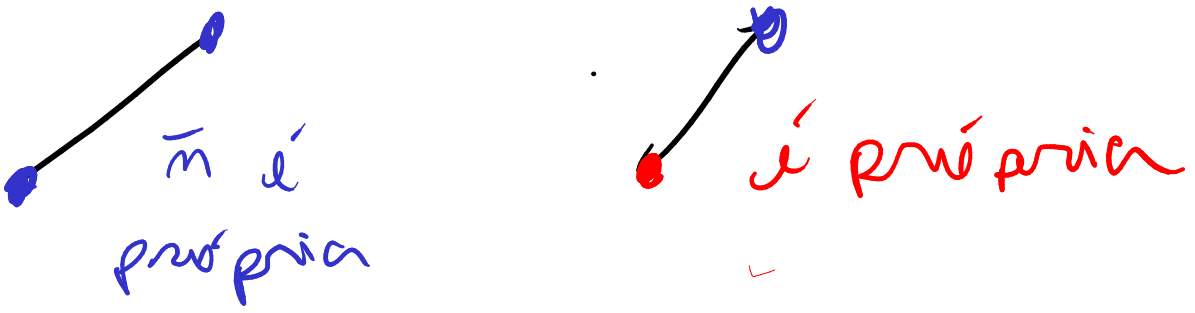


- seja uma coloração de vértices $\chi : V \to [q]$

Codificação cromática

- seja uma coloração de vértices $\chi : V \to [q]$

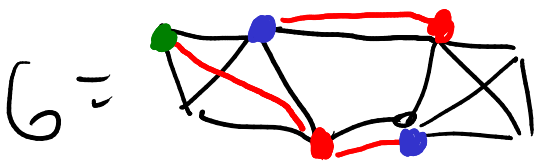
$\forall \{u, v\} \in \binom{V(G)}{2}$ é colorida propriamente se $\chi(u) \neq \chi(v)$



Codificação cromática

- seja uma coloração de vértices $\chi : V \to [q]$
- $\binom{uv \in V(G)}{2}$ é colorida propriamente se $\chi(u) \neq \chi(v)$)

Codificação cromática: solução A é colorida propriamente



Obtendo uma coloração própria

Lema

Seja $q = \lceil \sqrt{8k} \rceil$ e G um grafo com k arestas cujos vértices foram coloridos aleatoriamente com q cores.

Obtendo uma coloração própria

Lema

Seja $q = \lceil \sqrt{8k} \rceil$ e G um grafo com k arestas cujos vértices foram coloridos aleatoriamente com q cores.

Então a probabilidade de $E(G)$ ser colorida propriamente é pelo menos $2^{-\sqrt{k/2}}$.

Prova

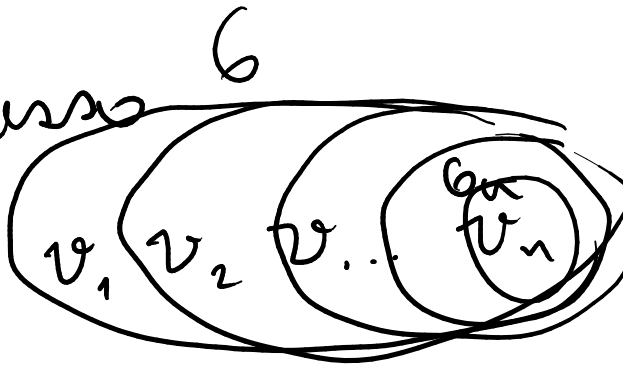
Considere o seguinte processo

1) $G_0 = G$

2) Para cada $i = 1 \dots n$:

a) Escolha v_i de G_{i-1} com menor grau

b) $G_i = G_i - v_i$



Temos: • $d_i \leq |V(G_{i-1})| - 1$ (grafo G_{i-1} é simples)

• $2K = 2|E(G)| \geq 2|E(G_{i-1})| = \sum_{v \in V(G_{i-1})} d_v \geq |V(G_{i-1})| \cdot d_i$

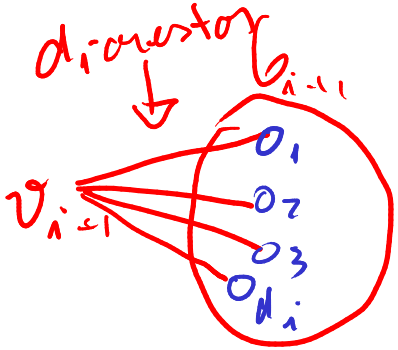
$\Rightarrow 2K \geq (d_i + 1)\, d_i \geq d_i^2 \Rightarrow d_i \leq \sqrt{2K}$

"

• Vamos colorir $V(G)$ em ordem $v_n, v_{n-1}, \ldots, v_1$.

• Cada vértice recebe uma cor entre $q = \lceil \sqrt{8K} \rceil$

Def: Seja P_i a probabilidade de que G_i é colorido propriamente.



Agora: 1) $P_n(P_n) = 1$

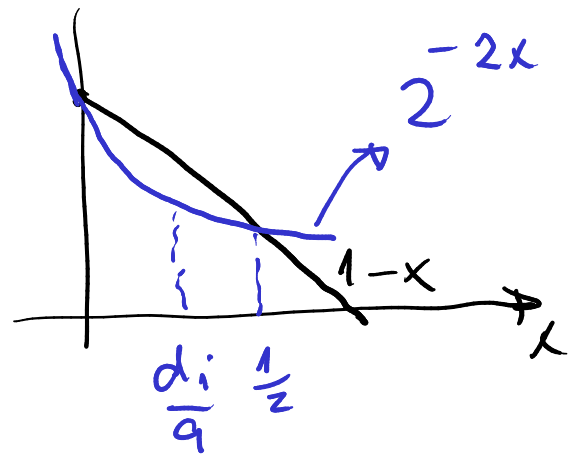
2) $P_r(P_i \mid P_{i+1}) \geq \frac{q - d_i}{q}$

$= 1 - \frac{d_i}{q} \geq 2^{-2d_i/q}$

pois $\frac{d_i}{q} \leq \frac{\sqrt{2K}}{\sqrt{8K}} = \frac{1}{2}$

Tenho $q = \lceil \sqrt{8K} \rceil$

mas $d_i \leq \sqrt{2K}$

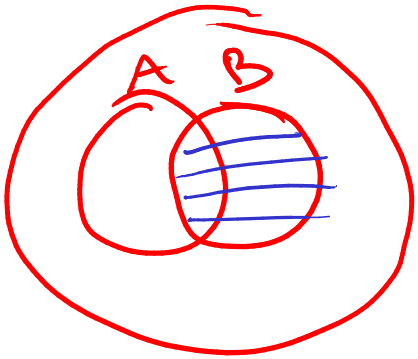


Portanto $P_r(\text{ser colorido propriamente}) =$

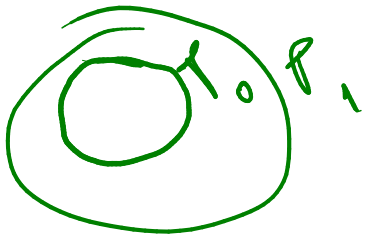
$$P_r(P_0) = P_r(P_0 \mid P_1) \cdot P_r(P_1)$$

$$= P_r(P_0 \mid P_1) \cdot P_r(P_1 \mid P_2) \cdot P_r(P_3)$$

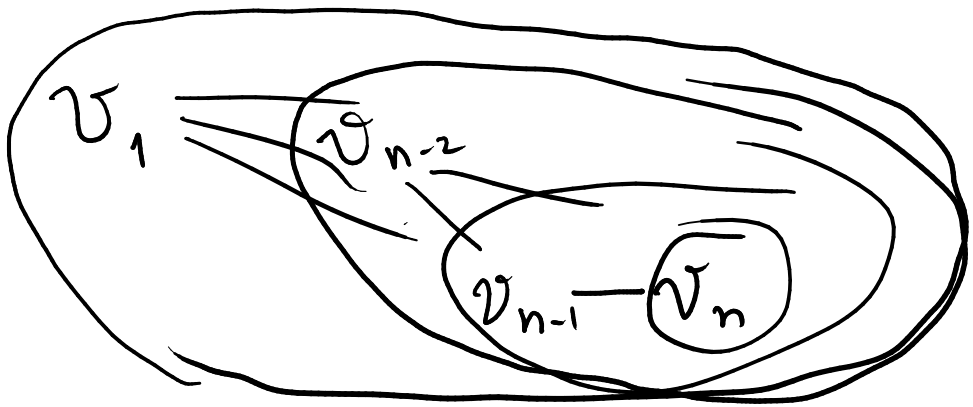
$$= P_r(P_n) \cdot \prod_{i=1}^{n} P_r(P_{i-1} \mid P_i)$$



$$P_r(A \mid B) = \frac{P_r(A \cap B)}{P_r(B)}$$



$$\geq 1 \cdot \prod_{i=1}^{n} 2^{-\frac{2d_i}{q}} = 2^{-\sum_{i=1}^{n} \frac{2d_i}{q}}$$



$$= 2^{-\frac{2K}{\sqrt{8K}}} = 2^{-\sqrt{K/2}}.$$

□

Encontrando uma solução colorida

Lema

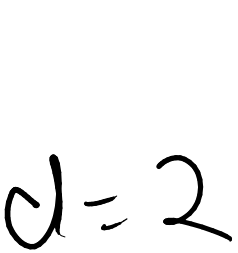
Seja G um grafo e $\chi : V(G) \to [q]$. Seja V_i os vértices com cor i.

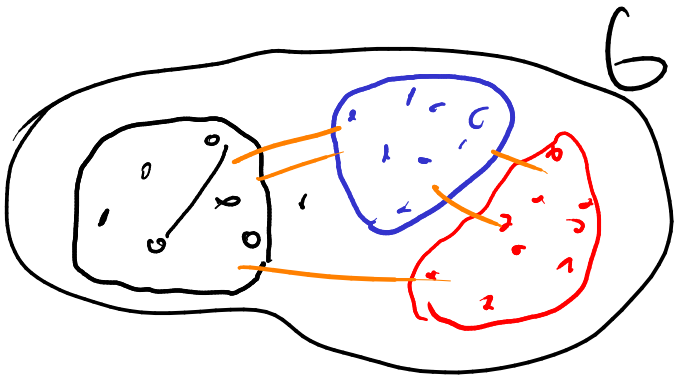
Encontrando uma solução colorida

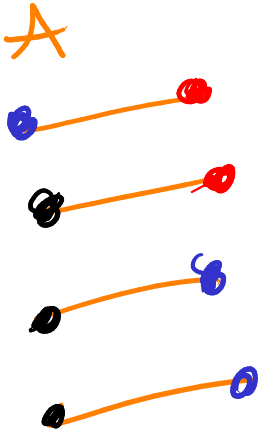
Lema

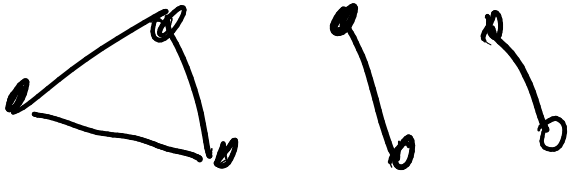
Seja G um grafo e $\chi : V(G) \to [q]$. Seja V_i os vértices com cor i.

Se existe uma solução A colorida propriamente por χ, então, para todo i, $G[V_i]$ é um l-agrupamento com $l \leq d$.









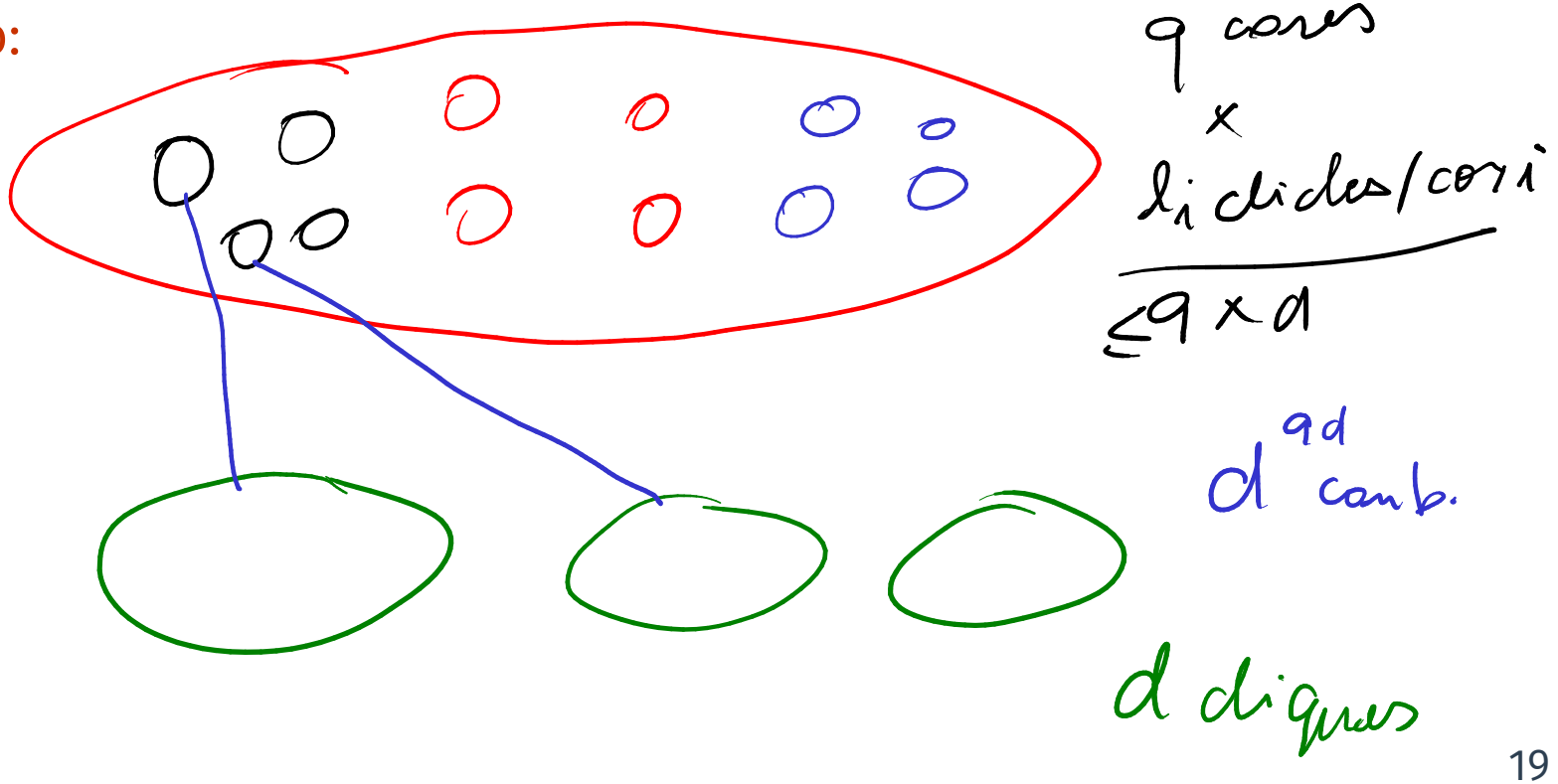
Encontrando uma solução colorida

Lema

Seja G um grafo e $\chi : V(G) \to [q]$. Seja V_i os vértices com cor i.

Se existe uma solução A colorida propriamente por χ, então, para todo i, $G[V_i]$ é um l-agrupamento com $l \leq d$.

Algoritmo:



Combinando

Lema

Dada uma instância (G, k) de d-agrupamento com coloração $\chi : V(G) \to [\lceil \sqrt{8k} \rceil]$, podemos decidir se há uma solução colorida propriamente em tempo $2^{\mathcal{O}(d \log d \sqrt{k})}$.

Combinando

Lema

Dada uma instância (G, k) de d-agrupamento com coloração $\chi : V(G) \to [\lceil\sqrt{8k}\rceil]$, podemos decidir se há uma solução colorida propriamente em tempo $2^{\mathcal{O}(d \log d\sqrt{k})} \cdot n^{\mathcal{O}(1)}$

Teorema

Existe um algoritmo aleatorizado que, dada uma instância (G, k) do d-agrupamento, em tempo $2^{\mathcal{O}(d\sqrt{k})} n^{\mathcal{O}(1)}$ ou reporta falha, ou encontra uma solução. Além disso, dada uma instância sim, ele retorna uma solução com probabilidade constante.

Desaleatorização

Desaleatorizando

Famílias pseudoaleatórias

Queremos colorações $\chi : [n] \to [k]$ boas

Desaleatorizando

Famílias pseudoaleatórias

Queremos colorações $\chi : [n] \to [k]$ boas

A invés de obter χ aleatório, retiramos de uma família $\mathcal{F}$:

Desaleatorizando

Famílias pseudoaleatórias

Queremos colorações $\chi : [n] \to [k]$ boas

A invés de obter χ aleatório, retiramos de uma família $\mathcal{F}$:

- $\mathcal{F}$ contém pelo menos uma coloração boa
- $\mathcal{F}$ é pequena $\Rightarrow$ $f(k)\, n^{O(1)}$

Desaleatorizando

Famílias pseudoaleatórias

Queremos colorações $\chi : [n] \to [k]$ boas

A invés de obter χ aleatório, retiramos de uma família $\mathcal{F}$:

- $\mathcal{F}$ contém pelo menos uma coloração boa
- $\mathcal{F}$ é pequena

Vamos usar algumas construções prontas:

Desaleatorizando

Famílias pseudoaleatórias

Queremos colorações $\chi : [n] \to [k]$ boas

A invés de obter χ aleatório, retiramos de uma família $\mathcal{F}$:

- $\mathcal{F}$ contém pelo menos uma coloração boa
- $\mathcal{F}$ é pequena

Vamos usar algumas construções prontas: separadores, conjuntos universais, hashs perfeitos e famílias independentes,

Separadores

Definição

Um (n, k, l)-splitter é uma família $\mathcal{F}$ de funções $f : [n] \to [l]$.

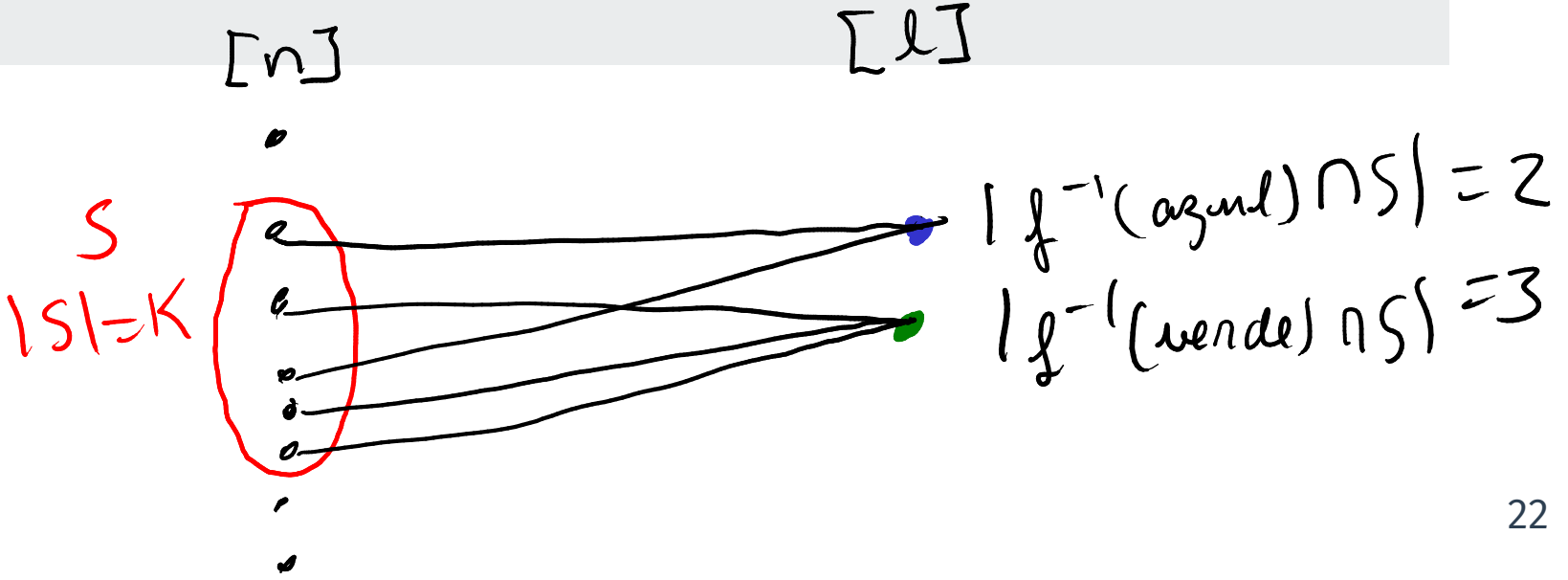
Separadores

Definição

Um (n, k, l)-splitter é uma família $\mathcal{F}$ de funções $f : [n] \to [l]$.

Para todo $S \subseteq [n]$, como $|S| = k$, existe $f \in \mathcal{F}$ tal que:

$$-1 \leq |f^{-1}(j) \cap S| - |f^{-1}(j') \cap S| \leq 1. \quad \forall j, j' \in [l]$$



Separadores

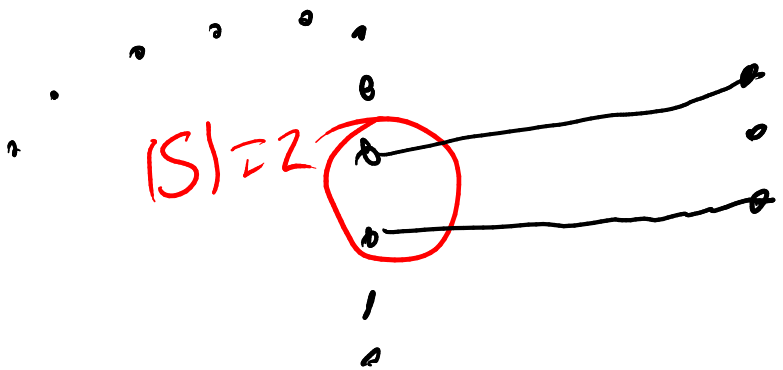
Definição

Um (n, k, l)-splitter é uma família $\mathcal{F}$ de funções $f : [n] \to [l]$.

Para todo $S \subseteq [n]$, como $|S| = k$, existe $f \in \mathcal{F}$ tal que:

$$-1 \leq |f^{-1}(j) \cap S| - |f^{-1}(j') \cap S| \leq 1.$$

Splitters injetores: Se $l \geq k$, então f induz uma função injetora em S.



Ferramenta básica

Teorema

Para $n, k \geq 1$, existe algoritmo que:

Ferramenta básica

Teorema

Para $n, k \geq 1$, existe algoritmo que:

- *obtém um (n, k, k^2)-splitter de tamanho $k^{\mathcal{O}(1)} \log n$*

Ferramenta básica

Teorema

Para $n, k \geq 1$, existe algoritmo que:

- *obtém um (n, k, k^2)-splitter de tamanho $k^{\mathcal{O}(1)} \log n$*
- *tem tempo de execução $k^{\mathcal{O}(1)} n \log n$.*

Hash perfeito

Um (n, k, k)-splitter é chamado de (n, k)-hash perfeito.

Teorema

Para $n, k \geq 1$, existe algoritmo que:

Hash perfeito

Um (n, k, k)-splitter é chamado de (n, k)-hash perfeito.

Teorema

Para $n, k \geq 1$, existe algoritmo que:

- *obtém um (n, k)-hash perfeito de tamanho $e^k k^{\mathcal{O}(\log k)} \log n$*

Hash perfeito

Sketch: • Seja f_1 um splitter (n, K, K^2) de tam $\approx K^{\log n}$ $\quad \theta(1)$

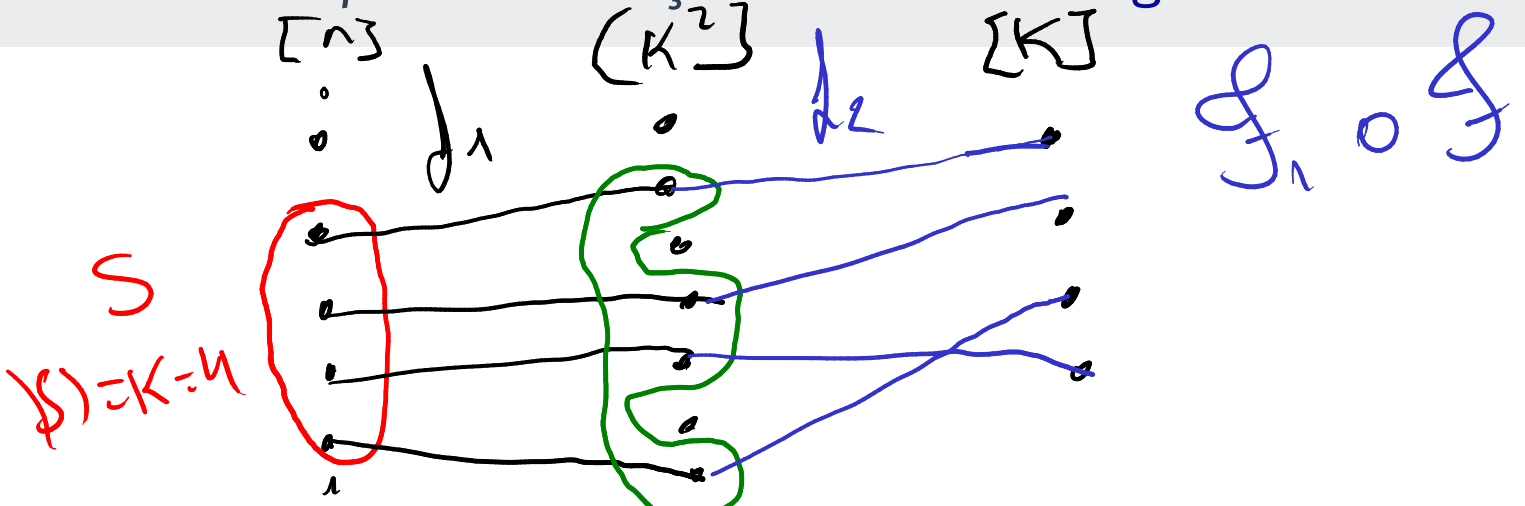
• Seja f_2 um hash (K^2, K) de tam $\approx e^K$

Um (n, k, k)-splitter é chamado de (n, k)-hash perfeito.

Teorema

Para $n, k \geq 1$, existe algoritmo que:

- *obtém um (n, k)-hash perfeito de tamanho $e^k k^{\mathcal{O}(\log k)} \log n$*
- *tem tempo de execução $e^k k^{\mathcal{O}(\log k)} n \log n$.*



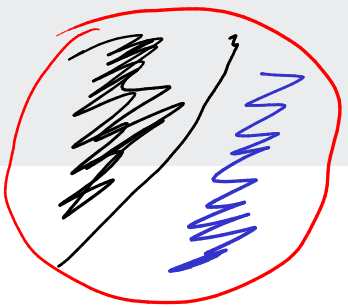
Conjunto Universal

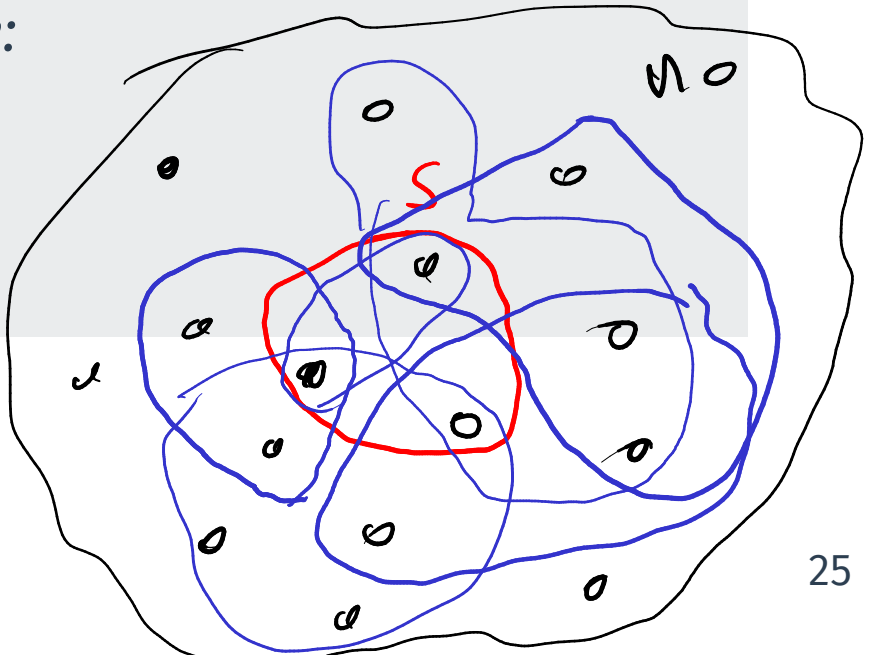
Definição

Um (n, k)-universal set é uma família $\mathcal{U}$ de subconjuntos de $[n]$ tal que para cada $S \subseteq [n]$ de tamanho k, a família $\{A \cap S : A \in \mathcal{U}\} = 2^{[k]}$.

Teorema

Para $n, k \geq 1$, existe algoritmo que:





Conjunto Universal

Definição

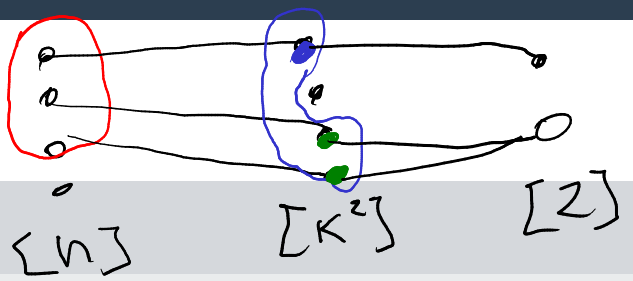
Um (n, k)-universal set é uma família $\mathcal{U}$ de subconjuntos de $[n]$ tal que para cada $S \subseteq [n]$ de tamanho ki, a família $\{A \cap S : A \in \mathcal{U}\} = 2^{[k]}$.

Teorema

Para $n, k \geq 1$, existe algoritmo que:

- *obtém um (n, k)-universal set de tamanho $2^k k^{\mathcal{O}(\log k)} \log n$*

Conjunto Universal



Definição

Um (n, k)-universal set é uma família $\mathcal{U}$ de subconjuntos de $[n]$ tal que para cada $S \subseteq [n]$ de tamanho ki, a família $\{A \cap S : A \in \mathcal{U}\} = 2^{[k]}$.

Teorema

Para $n, k \geq 1$, existe algoritmo que:

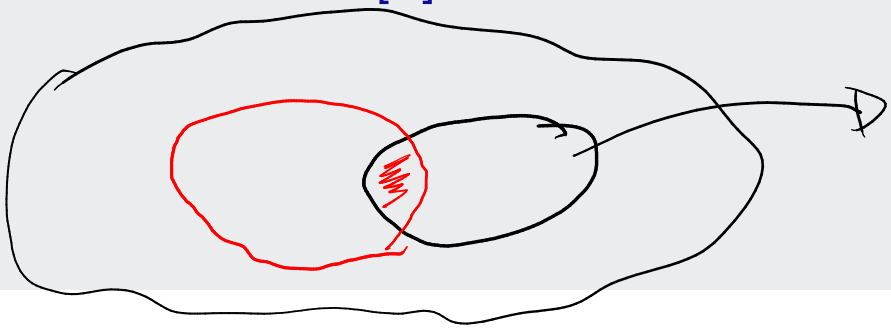
- *obtém um (n, k)-universal set de tamanho $2^k k^{\mathcal{O}(\log k)} \log n$*
- *tem tempo de execução $2^k k^{\mathcal{O}(\log k)} n \log n$.*

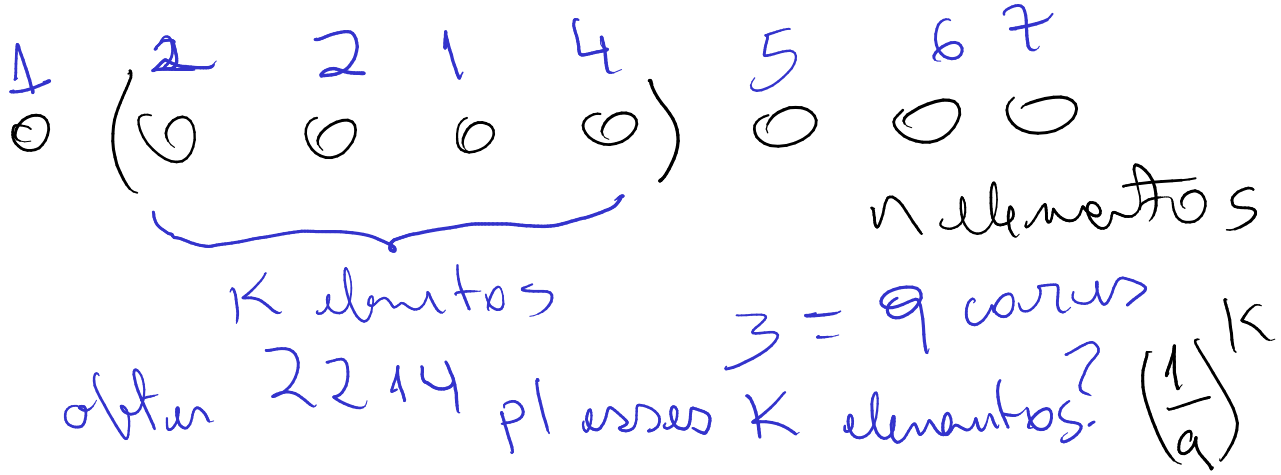
Combina (n, K, K^2)-splitter com (K^2, K)-universal set

Conjuntos independentes k a k

Definição

Uma família $H_{n,k,q}$ de funções $f : [n] \to [q]$ é chamada de espaço amostral independente k a k se, para todos conjuntos de k elementos de $[n]$, $i_1 \leq \cdots \leq i_k$ e todo $\alpha \in [q]^k$, valer





Conjuntos independentes k a k

Definição

Uma família $H_{n,k,q}$ de funções $f : [n] \to [q]$ é chamada de espaço amostral independente k a k se, para todos conjuntos de k elementos de $[n]$, $i_1 \leq \cdots \leq i_k$ e todo $\alpha \in [q]^k$, valer

$$Pr((f(i_1), \ldots, f(i_k)) = \alpha) = 1/q^k,$$

Conjuntos independentes k a k

Definição

Uma família $H_{n,k,q}$ de funções $f : [n] \to [q]$ é chamada de espaço amostral independente k a k se, para todos conjuntos de k elementos de $[n]$, $i_1 \leq \cdots \leq i_k$ e todo $\alpha \in [q]^k$, valer

$$Pr((f(i_1), \ldots, f(i_k)) = \alpha) = 1/q^k,$$

onde $f \in H_{n,k,q}$ é escolhida uniformemente.

Conjuntos independentes k a k

Definição

Uma família $H_{n,k,q}$ de funções $f : [n] \to [q]$ é chamada de espaço amostral independente k a k se, para todos conjuntos de k elementos de $[n]$, $i_1 \leq \cdots \leq i_k$ e todo $\alpha \in [q]^k$, valer

$$Pr((f(i_1), \ldots, f(i_k)) = \alpha) = 1/q^k,$$

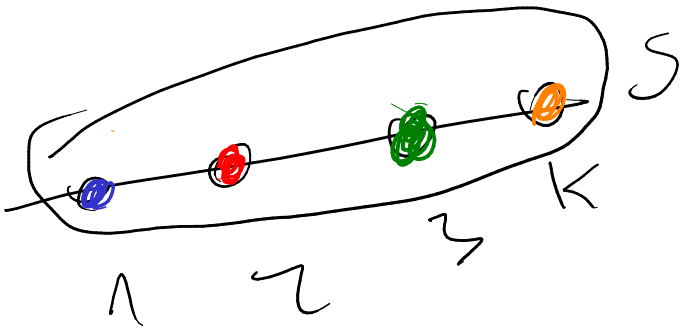
onde $f \in H_{n,k,q}$ é escolhida uniformemente.

Teorema

Existe algoritmo que obtém $H_{n,k,q}$ de tamanho $\mathcal{O}(n^k)$ em tempo linear na saída.

Desaleatorizando Caminho mais longo

Ideia: ao invés de usar uma coloração aleatória, testamos todas colorações de um (n, k)-hash perfeito



- não conheço S, mas $S \subseteq V$ e $|S| \leq K$.
- uma coloração boa é uma col. de V injetora em S.

Desaleatorizando Caminho mais longo

Ideia: ao invés de usar uma coloração aleatória, testamos todas colorações de um (n, k)-hash perfeito

Teorema

Caminho mais longo pode ser resolvido em tempo $(2e)^k k^{\mathcal{O}(\log k)} n^{\mathcal{O}(1)}$.

1) Obtenho um (n,k)-hash de tam $e^k k^{O(\log k)} \log n$

2) Se a instância for sim então existe uma coloração boa no hash. Basta testar todas e usar o bom algoritmo para cada em tempo $2^k n^{O(1)}$. □

Desaleatorizando d-agrupamento

Definição

Uma (n, k, q)-coloring family é uma família $\mathcal{F}$ de funções de $[n]$ em $[q]$ com a seguinte propriedade:

Desaleatorizando d-agrupamento

Definição

Uma (n, k, q)-coloring family é uma família $\mathcal{F}$ de funções de $[n]$ em $[q]$ com a seguinte propriedade:

- para todo grafo $G = (V, E)$ com $V = [n]$ e $|E| \leq k$, existe $f \in \mathcal{F}$ que colore E propriamente.

Desaleatorizando d-agrupamento

Definição

Uma (n, k, q)-coloring family é uma família $\mathcal{F}$ de funções de $[n]$ em $[q]$ com a seguinte propriedade:

- para todo grafo $G = (V, E)$ com $V = [n]$ e $|E| \leq k$, existe $f \in \mathcal{F}$ que colore E propriamente.

Objetivo: encontrar uma (n, k, q)-coloring family pequena

Desaleatorizando d-agrupamento

Definição

Uma (n, k, q)-coloring family é uma família $\mathcal{F}$ de funções de $[n]$ em $[q]$ com a seguinte propriedade:

- para todo grafo $G = (V, E)$ com $V = [n]$ e $|E| \leq k$, existe $f \in \mathcal{F}$ que colore E propriamente.

Objetivo: encontrar uma (n, k, q)-coloring family pequena

Ideia: Compor um (n, k, k^2)-splitter com uma (k^2, k, q)-coloring family

Desaleatorizando d-agrupamento

Definição

Uma (n, k, q)-coloring family é uma família $\mathcal{F}$ de funções de $[n]$ em $[q]$ com a seguinte propriedade:

- para todo grafo $G = (V, E)$ com $V = [n]$ e $|E| \leq k$, existe $f \in \mathcal{F}$ que colore E propriamente.

Objetivo: encontrar uma (n, k, q)-coloring family pequena

Ideia: Compor um (n, k, k^2)-splitter com uma (k^2, k, q)-coloring family

- cada aresta de G deve ser colorida propriamente

Desaleatorizando d-agrupamento

Definição

Uma (n, k, q)-coloring family é uma família $\mathcal{F}$ de funções de $[n]$ em $[q]$ com a seguinte propriedade:

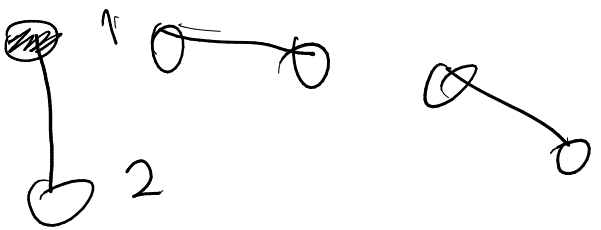
- para todo grafo $G = (V, E)$ com $V = [n]$ e $|E| \leq k$, existe $f \in \mathcal{F}$ que colore E propriamente.

Objetivo: encontrar uma (n, k, q)-coloring family pequena

Ideia: ~~Compor um~~ (n, k, k^2)-splitter com uma (k^2, k, q)-coloring family

- cada aresta de G deve ser colorida propriamente
- basta que pares de vértices sejam coloridos independentemente

com cores distintas



Uma família de coloração explícita para k^2

Lema

Para $k \geq 1$, exite algoritmo que:

- *obtém $(k^2, k, 2\lceil\sqrt{k}\rceil)$-coloring family de tamanho $2^{\mathcal{O}(\sqrt{k}\log k)}$;*

Uma família de coloração explícita para k^2

Lema

Para $k \geq 1$, exite algoritmo que:

- *obtém $(k^2, k, 2\lceil\sqrt{k}\rceil)$-coloring family de tamanho $2^{\mathcal{O}(\sqrt{k}\log k)}$;*
- *executa em tempo linear na saída.*

- Seja $q = \lceil\sqrt{k}\rceil$
- Obtenha uma 2-wise independent family $\mathcal{G}, H_{k^2,2,q}$
- Temos $|\mathcal{G}| \leq (k^2)^2 = \mathcal{O}(k^4)$
- Fixe $g \in \mathcal{G}$ e $T \subseteq [k^2]$ $T = |q|$
- Seja $T = \{i_1, i_2, \ldots, i_q\}$
- Defina $f_{g,T} : [k^2] \to [2q]$ tq:

$f_{gT}(i) = j + q$ se $i \in T$ e $i = i_j$ e

$f_{gT}(i) = g(i)$

Seja $\mathcal{F}$ o conj. de funções f_{gT} $\forall g \subset T$

Temos $|\mathcal{F}| = |G| \cdot \binom{K^2}{q} = O(K^4 \cdot (K^2)^q)$

$= 2^{O(\sqrt{K} \log K)}$

Queremos $\forall G = (V, E)$, $V = \{K^2\}$ e $|E| = K$

existe $f \in \mathcal{F}$ tq E é col. propriamente.

Seja o seguinte processo aleatório:

1) Escolha $g \in G$ aleatoriamente

2) Suponha que apenas $E' \subseteq E$ não
deixam colorir propriamente por g

3) Se $|E'| \leq q$

a) Escolha $T \subseteq V$, $|T| = q$ e T cobre E'

b) Escolha f_{gT}. Claramente,
f_{gT} colore propriamente E.

4) Se $|E'| > q$, falhou.

Alg $\mathbb{E}[|E'|] \leq \frac{k}{q} = \frac{k}{\lceil \sqrt{k} \rceil} \leq \sqrt{k} \leq \lceil \sqrt{k} \rceil = q$

$\Rightarrow \exists q' \leq q$ tq $Pr(|E'| = q') > 0$.

Prova:

Seja $uv \in E$. $Pr(uv \in E') =$

$= Pr(g(u) = g(v))$

$= \sum_{i=1}^{q} Pr(g(v) = g(v) = i)$

$= \sum_{i=1}^{q} 1/q^2 = 1/q$

$\mathbb{E}[|E'|] = \mathbb{E}\left[\sum_{uv \in E} \mathbb{1}(uv \in E')\right] = \sum_{uv \in E} \mathbb{E}[\mathbb{1}_{uv \in E'}] = \sum_{uv \in E} Pr(uv \in E')$

$= \sum_{uv \in E} 1/q = k \cdot \frac{1}{q}$ □

Combinando

Teorema

Para $n, k \geq 1$, exite algoritmo que:

- *obtém $(n, k, 2\lceil\sqrt{k}\rceil)$-coloring family de tamanho $2^{\mathcal{O}(\sqrt{k}\log k)}\log n$;*

Combinando

Teorema

Para $n, k \geq 1$, exite algoritmo que:

- *obtém $(n, k, 2\lceil\sqrt{k}\rceil)$-coloring family de tamanho $2^{\mathcal{O}(\sqrt{k}\log k)}\log n$;*
- *executa em tempo $2^{\mathcal{O}(\sqrt{k}\log k)} n \log n$*

Combinando

Teorema

Para $n, k \geq 1$, exite algoritmo que:

- *obtém $(n, k, 2\lceil\sqrt{k}\rceil)$-coloring family de tamanho $2^{\mathcal{O}(\sqrt{k}\log k)}\log n$;*
- *executa em tempo $2^{\mathcal{O}(\sqrt{k}\log k)} n \log n$*

Teorema

Existe algoritmo para d-agrupamento em tempo $2^{\mathcal{O}(\sqrt{k}(d+\log k))} n^{\mathcal{O}(1)}$.