# Ricerca veloce di pattern comuni a due grafi

Luca Versari

Scuola Normale Superiore

*luca.versari@sns.it*

15 aprile 2016

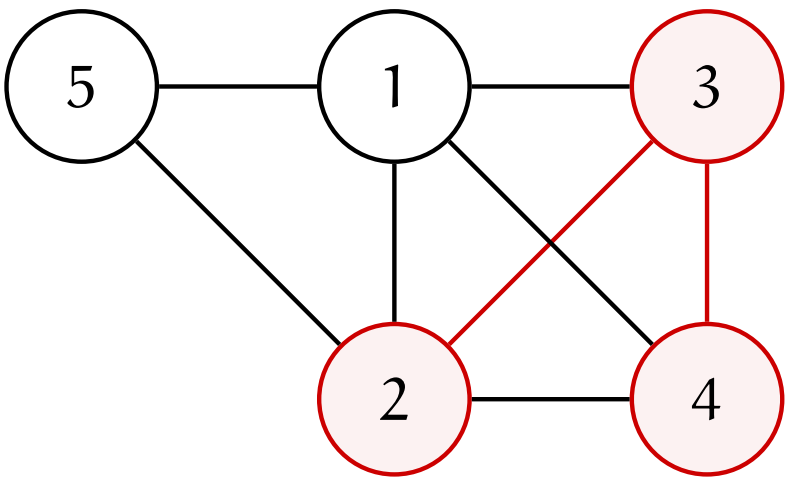
# Struttura della presentazione

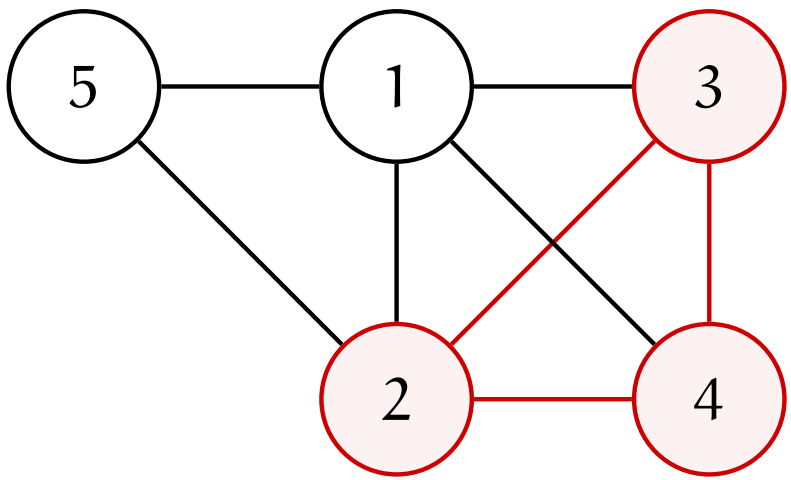
1. Introduzione
   - Definizioni
   - Problema
2. Algoritmi di risoluzione
   - Riduzione di Levi
   - Bron-Kerbosch
   - Miglioramenti
3. Risultati sperimentali
   - Risultati sperimentali

# Definizioni

## Definizione

Un grafo $G' = (V', E')$ si dice **sottografo** di $G = (V, E)$, e si indica con $G \subset G'$, se $V' \subset V$ e $E' \subset E$.
Un sottografo si dice **indotto da un insieme di vertici** $V'$, e si indica con $G < G'$, se $E' = (V' \times V') \cap E$.



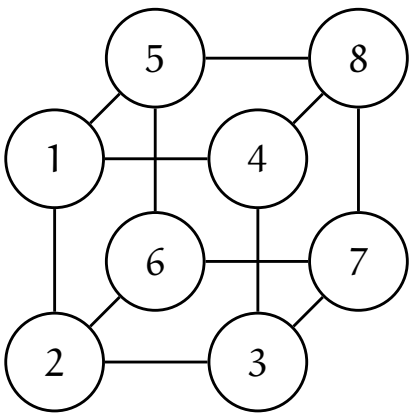
# Definizioni

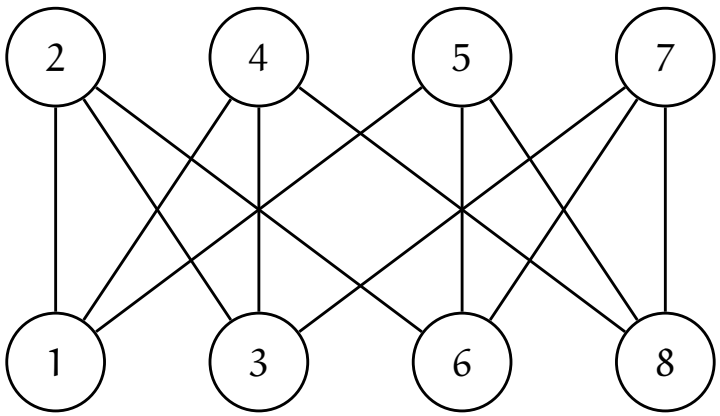
## Definizione

Dati due grafi $G = (V, E)$ e $G' = (V', E')$, una funzione bigettiva $I : V \to V'$ si dice **isomorfismo di grafi** se

$$(u, v) \in E \Leftrightarrow (I(u), I(v)) \in E'$$

Se $H < G$, $I_{|H}$ indica l'isomorfismo $I$ ristretto ai vertici di $H$.



# Definizioni

## Definizione

Si dice **etichettatura** di un grafo $G = (V, E)$ una funzione $l : V \to L$. Gli elementi di L sono detti etichette.
Si dice che un isomorfismo I **conserva l'etichettatura** se $l(I(v)) = l(v)\ \forall v \in V$.

## Definizione

Si definisce **cricca** in un grafo G un sottografo di G completo (ovvero che ha tutti gli archi possibili).
Inoltre si dice che un insieme di vertici è una cricca se il sottografo da essi indotto è una cricca.

# Una prima formulazione

Ci poniamo l'obiettivo di trovare le sottostrutture comuni a due grafi etichettati *sparsi*. Una delle possibili formulazioni formali del problema è la seguente:

**Isomorfismi massimali che conservano l'etichettatura (MLI)**

Dati due grafi $G$ e $H$ con le rispettive etichettature $l_G$ e $l_H$, elencare tutti gli isomorfismi I massimali (secondo l'ordinamento standard tra funzioni) nella famiglia degli isomorfismi

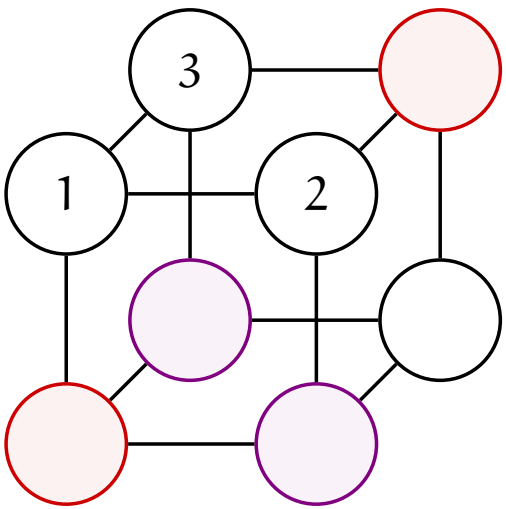
$$\mathcal{I}_{G,H} = \{I : G' \to H', G' < G, H' < H, l_H(I(v)) = l_G(v)\ \forall v \in G'\}$$

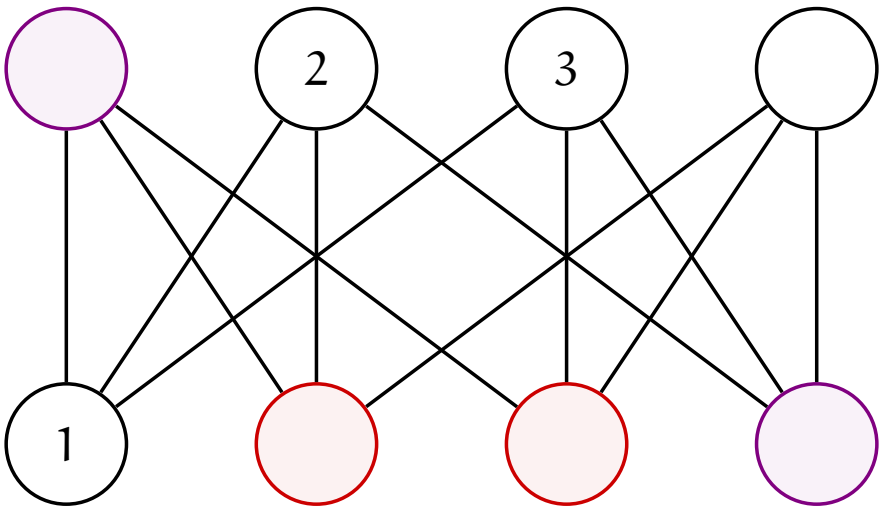
# Una formulazione più utile

**Problema MCLI**

Nella maggior parte delle applicazioni pratiche è sufficiente considerare due sottografi $G'$ e $H'$ che siano *connessi*: aggiungendo tale richiesta il numero di sottografi da trovare si riduce considerevolmente in molte situazioni.

# Un esempio di isomorfismo massimale connesso



# Prodotto modulare

## Definizione

Dati due grafi $G$ e $H$ con etichettatura $l_G$ e $l_H$, il loro **prodotto modulare** $G \cdot H$ è il grafo che ha:

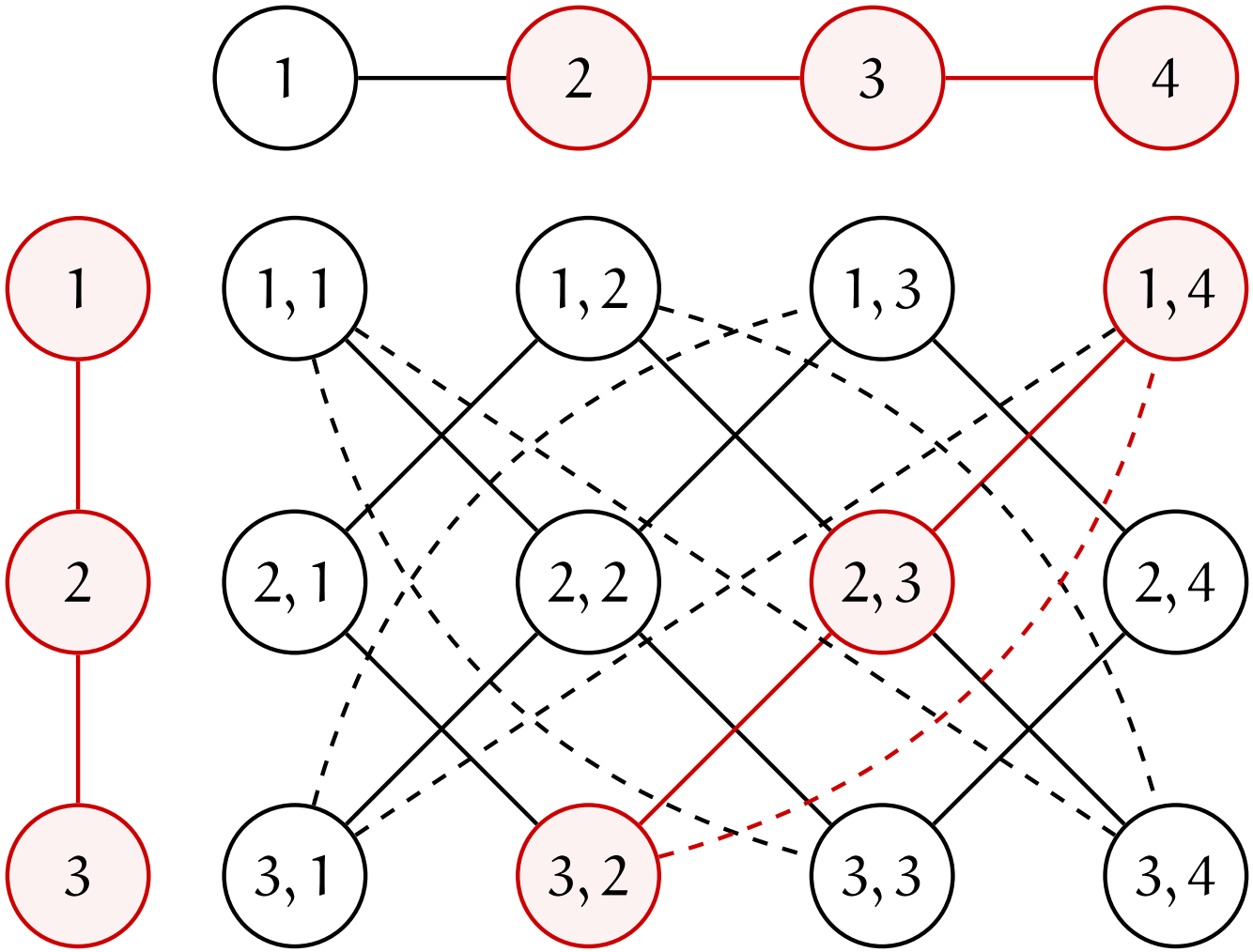
- per vertici l'insieme $\{(v, v') \in V \times V' : l_G(v) = l_H(v')\}$
- per archi tutti e soli gli archi tra due vertici $(u, u')$ e $(v, v')$ che soddisfano:
  - $(u, v) \in E$ e $(u', v') \in E'$ ("**archi neri**")
  - $u \neq u'$, $v \neq v'$, $(u, v) \notin E$ e $(u', v') \notin E'$ ("**archi bianchi**")

# Riduzione di Levi

**Teorema (Riduzione di Levi)**

*Esiste una bigezione tra l'insieme* $\mathcal{I}_{\mathsf{G},\mathsf{H}}$ *e l'insieme delle cricche del grafo* $\mathsf{G} \cdot \mathsf{H}$. *Inoltre, questa bigezione fa corrispondere isomorfismi massimali a cricche massimali e viceversa.*

# Riduzione di Levi



# Connessione

**Lemma**

*Un isomorfismo* $I : G' \to H'$ *è tale che* $G'$ *è connesso se e solo se il sottografo formato dai soli archi neri nella cricca* $C$ *a esso corrispondente in* $G \cdot H$ *è connesso.*

**Dimostrazione.**

($\Leftarrow$) Siano $u, v \in G'$. Poichè il sottografo di $C$ formato dai soli archi neri è connesso, esiste un cammino tra $(u, I(u))$ e $(v, I(v))$ formato da soli archi neri, diciamo $(v_1, I(v_1)), \ldots, (v_k, I(v_k))$. Allora i nodi $u = v_1, \ldots, v_k = v$ formano un cammino in $G'$.

# Connessione

## Lemma

*Un isomorfismo* $I : G' \to H'$ *è tale che* $G'$ *è connesso se e solo se il sottografo formato dai soli archi neri nella cricca* $C$ *a esso corrispondente in* $G \cdot H$ *è connesso.*

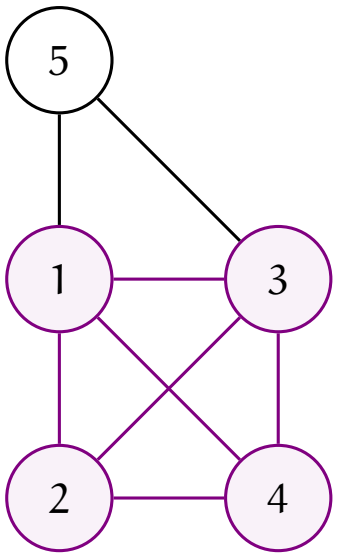
## Dimostrazione.

($\Rightarrow$) Siano $(u, I(u))$ e $(v, I(v))$ due nodi nella cricca $C$ di $G \cdot H$. Poich $u, v \in G'$ e $G'$ è connesso, esiste un cammino $u = v_1, \ldots, v_k = v$ in $G'$. Allora $I(v_i)$ e $I(v_{i+1})$ sono connessi da un arco, e quindi vi è un arco nero tra $(v_i, I(v_i))$ e $(v_{i+1}, I(v_{i+1}))$. Dunque $(v_1, I(v_1)), \ldots, (v_k, I(v_k))$ è un cammino di archi neri che congiunge $(u, I(u))$ e $(v, I(v))$. □

# Algoritmo di Bron-Kerbosch

**function** BronKerbosch($C, P, X$)
    **if** $P = \varnothing \wedge X = \varnothing$ **then**
        $C$ è una cricca massimale
    **end if**.
    **for all** $v \in P$ **do**
        BronKerbosch($C \cup \{v\}, P \cap N(v), X \cap N(v)$)
        $P \leftarrow P \setminus \{v\}$
        $X \leftarrow X \cup \{v\}$
    **end for**.
**end function**.

# Esempio di esecuzione




$h = 4$

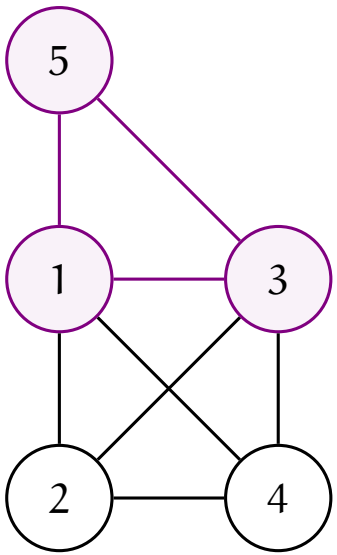
$h = 3$

$h = 2$

$h = 1$

$h = 0$

# Esempio di esecuzione




$h = 4$

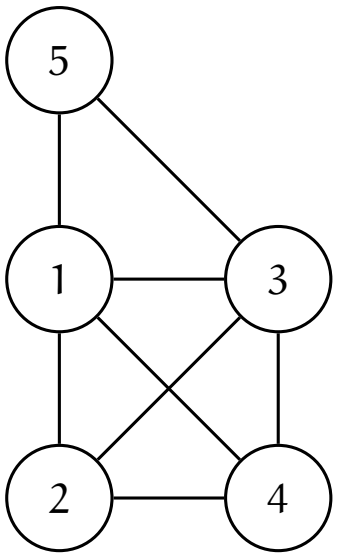
$h = 3$

$h = 2$

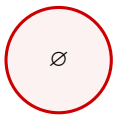
$h = 1$

$h = 0$

# Esempio di esecuzione




$h = 4$



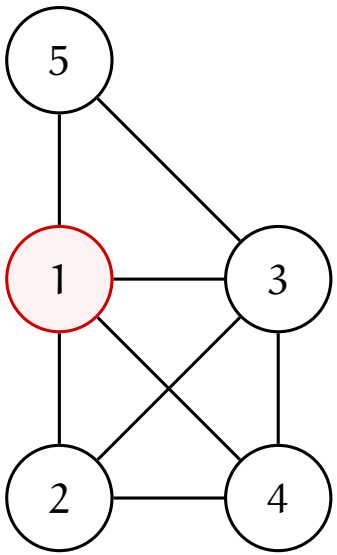

$h = 3$

$h = 2$

$h = 1$

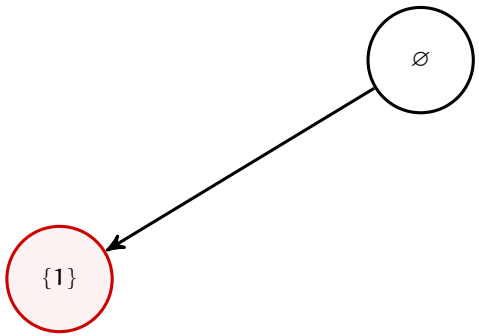
$h = 0$

# Esempio di esecuzione




$h = 4$

$h = 3$

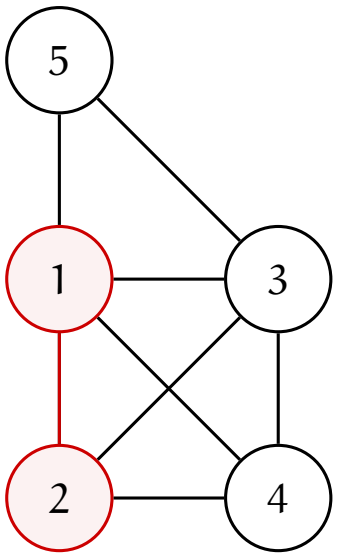


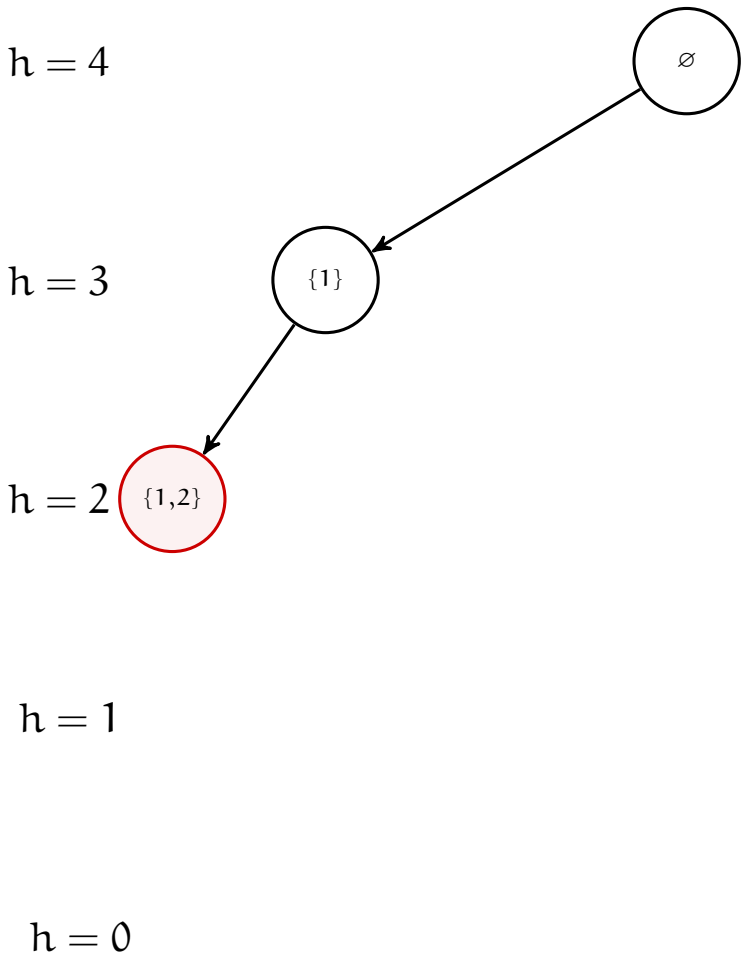

$h = 2$

$h = 1$

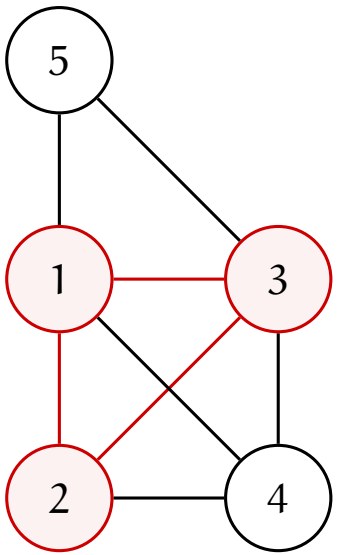
$h = 0$

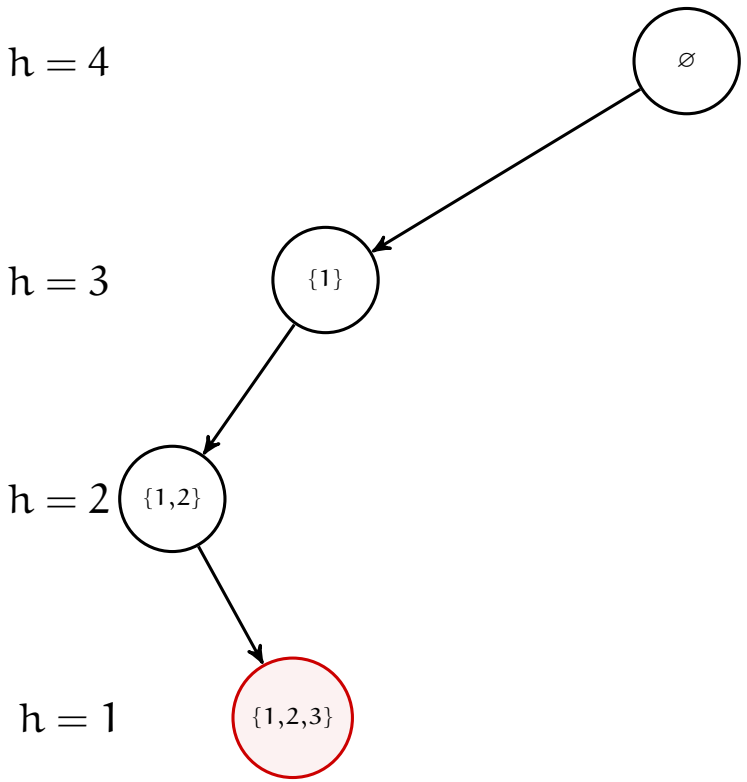
# Esempio di esecuzione



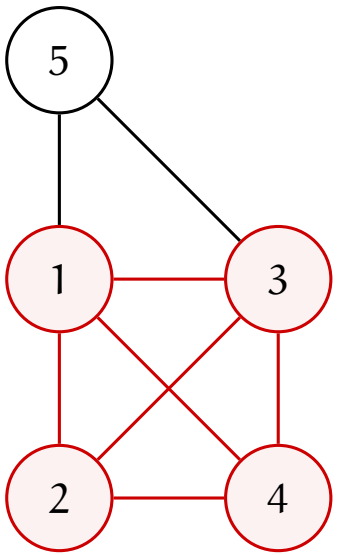
# Esempio di esecuzione

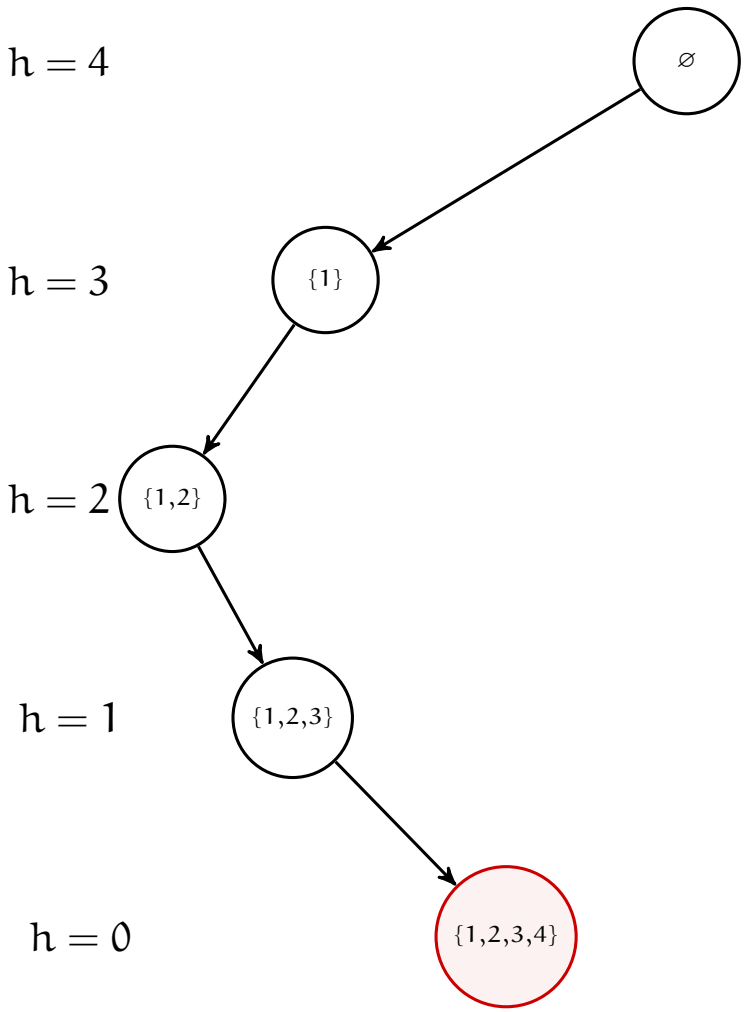


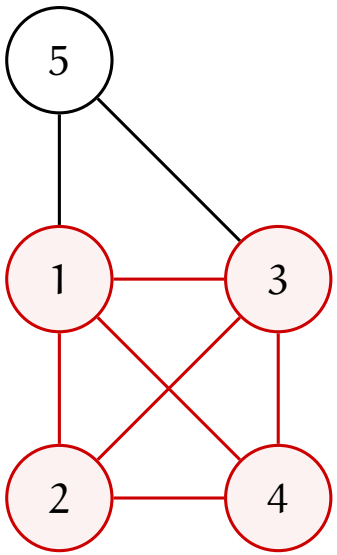

h = 0

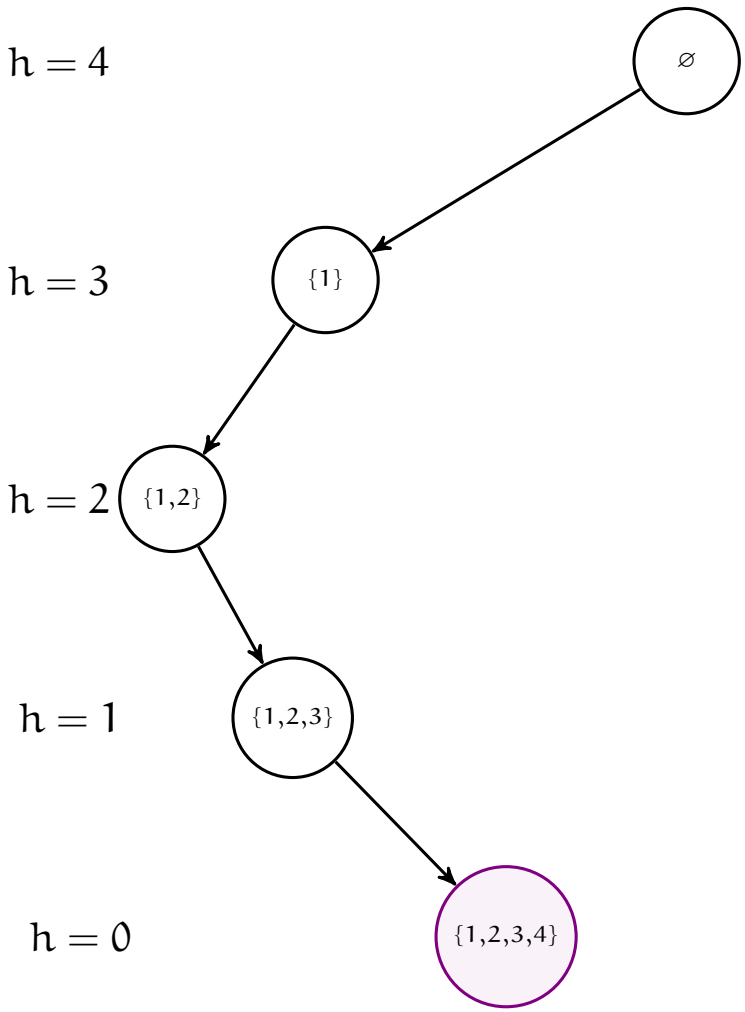
# Esempio di esecuzione



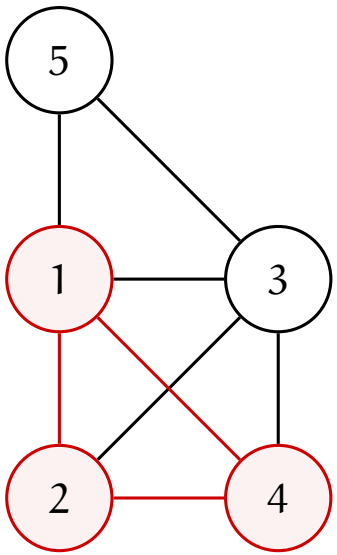



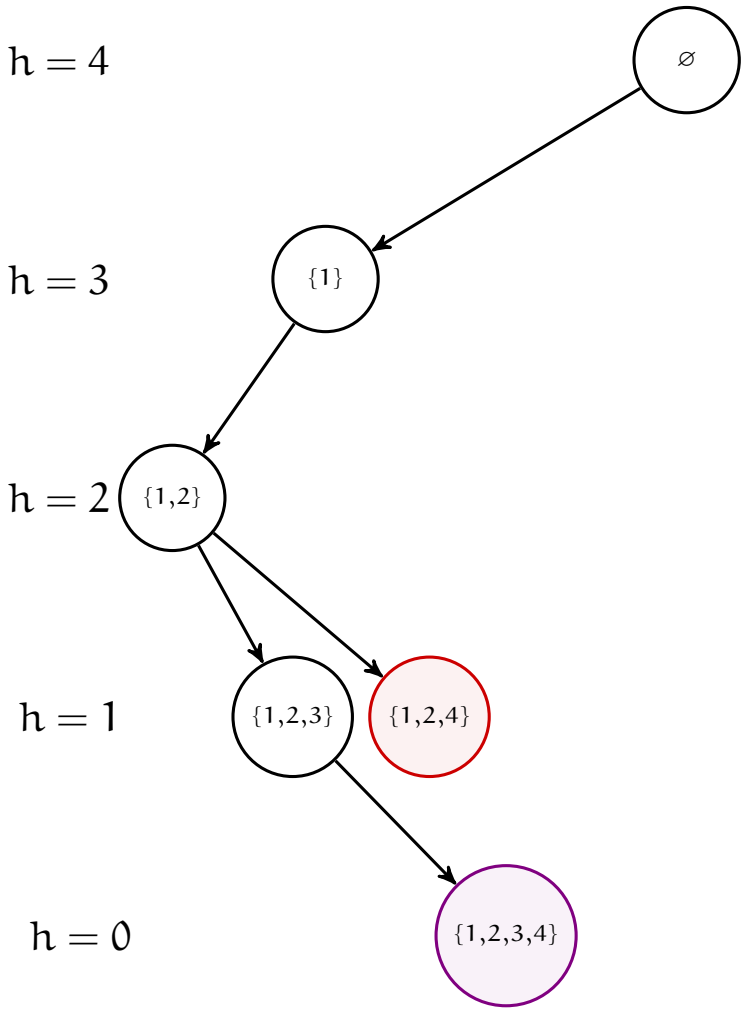
# Esempio di esecuzione



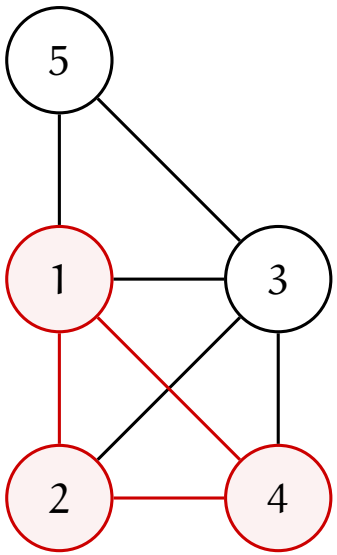



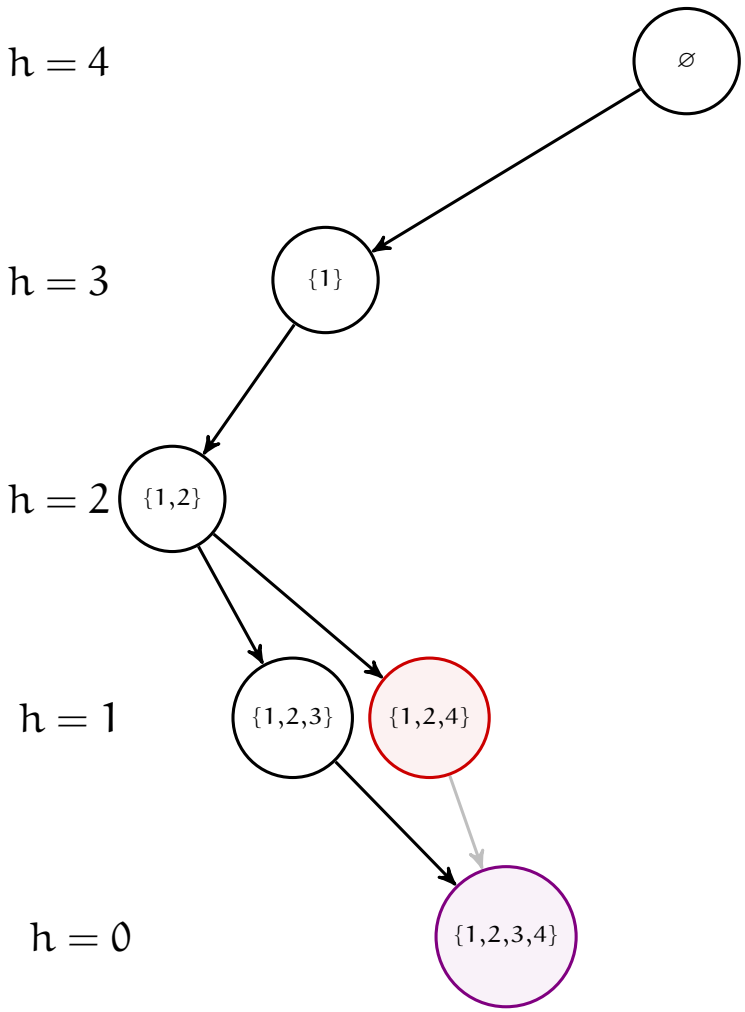
# Esempio di esecuzione



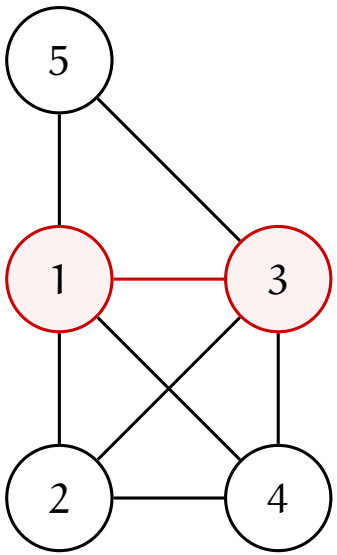



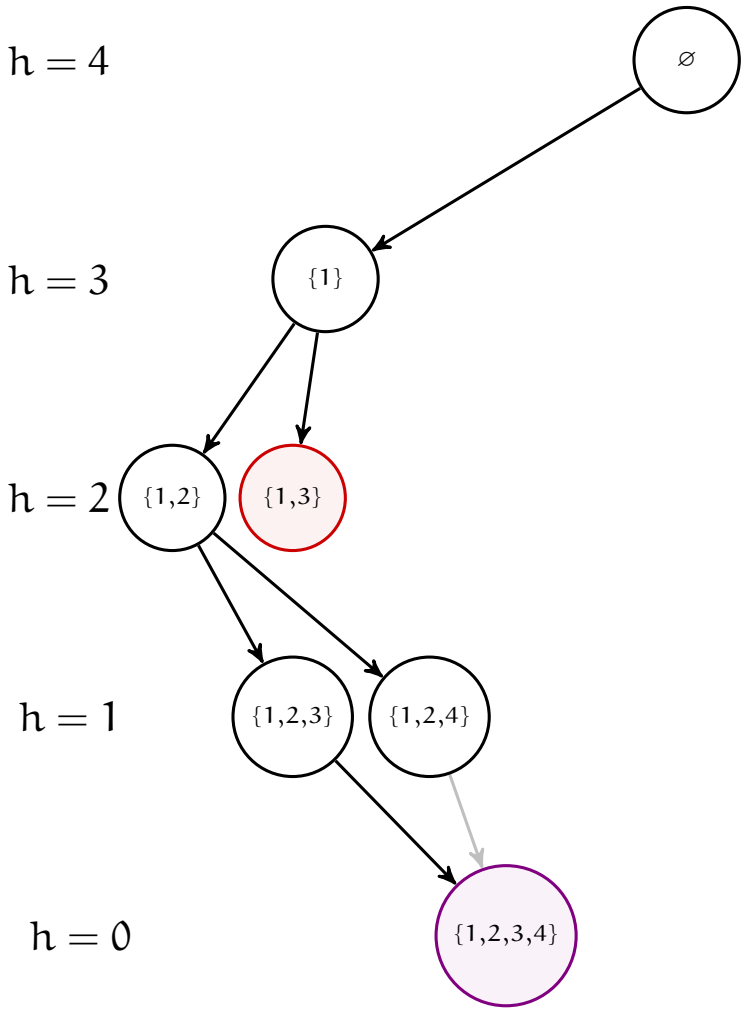
# Esempio di esecuzione



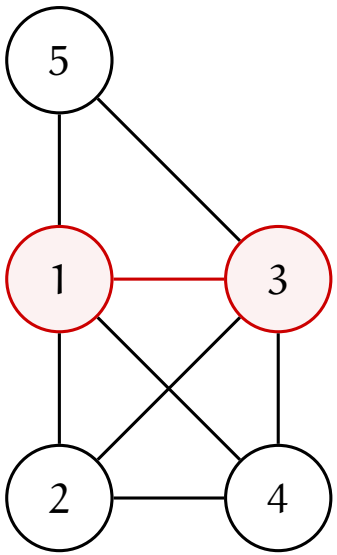



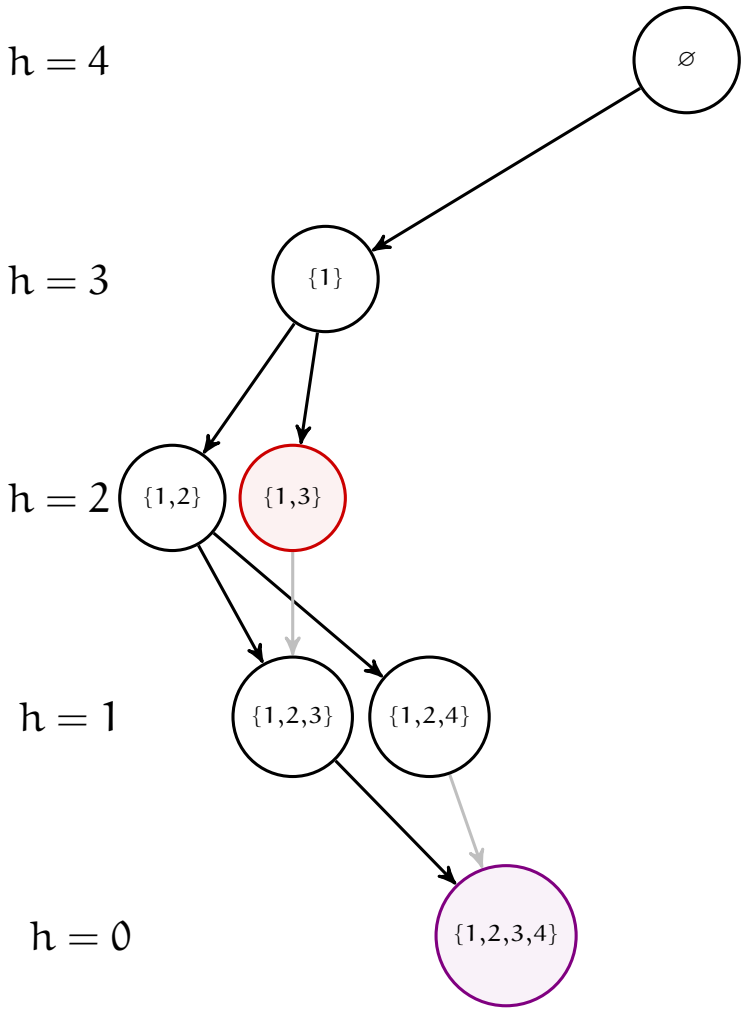
# Esempio di esecuzione



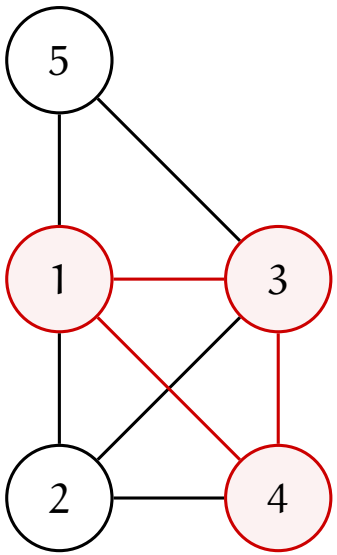



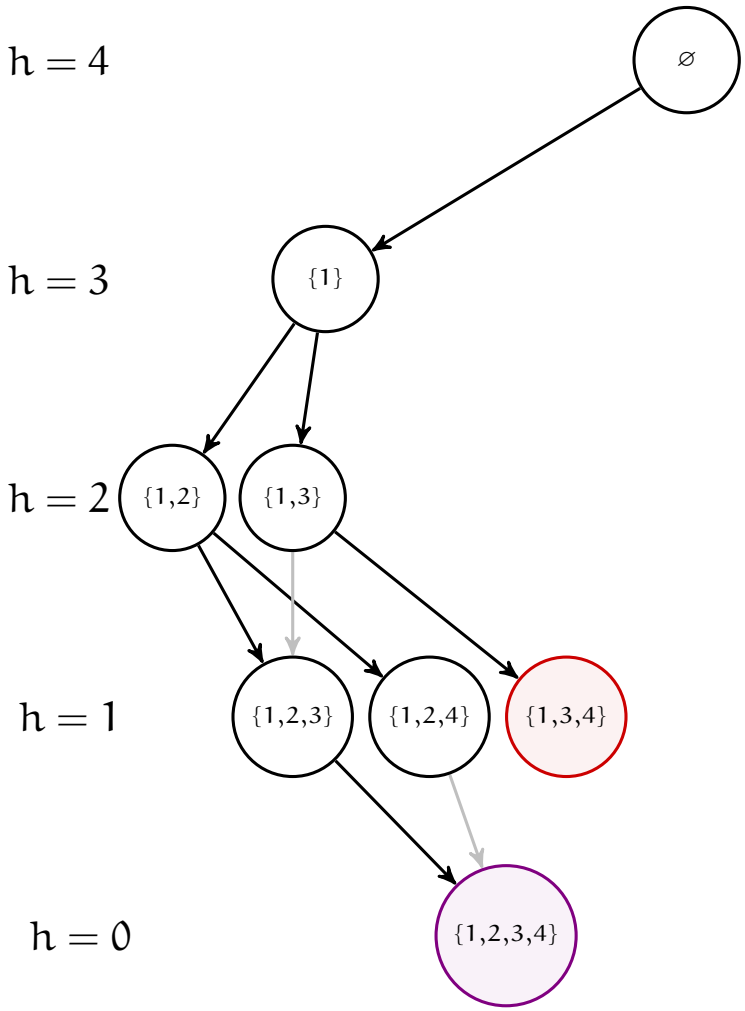
# Esempio di esecuzione



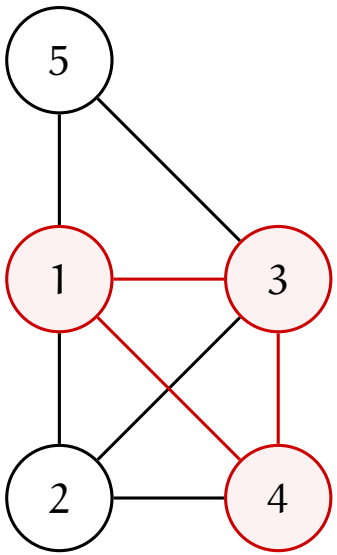



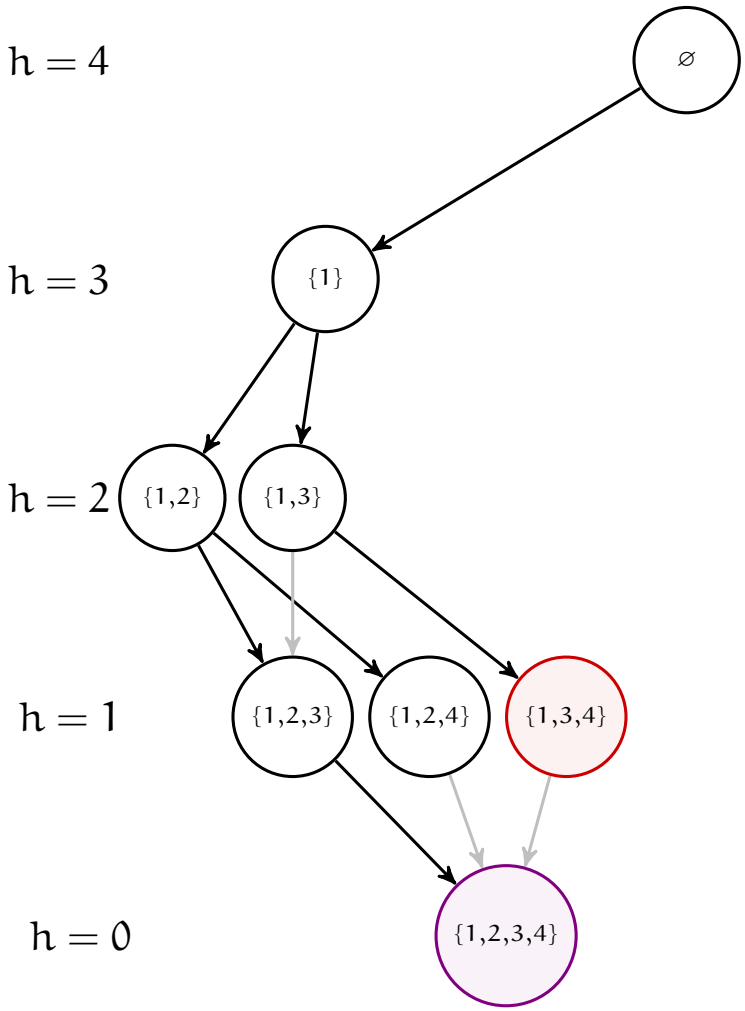
# Esempio di esecuzione



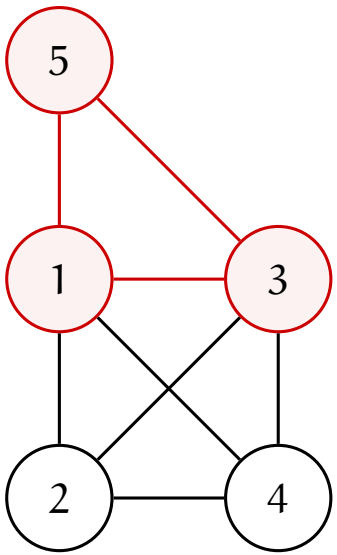



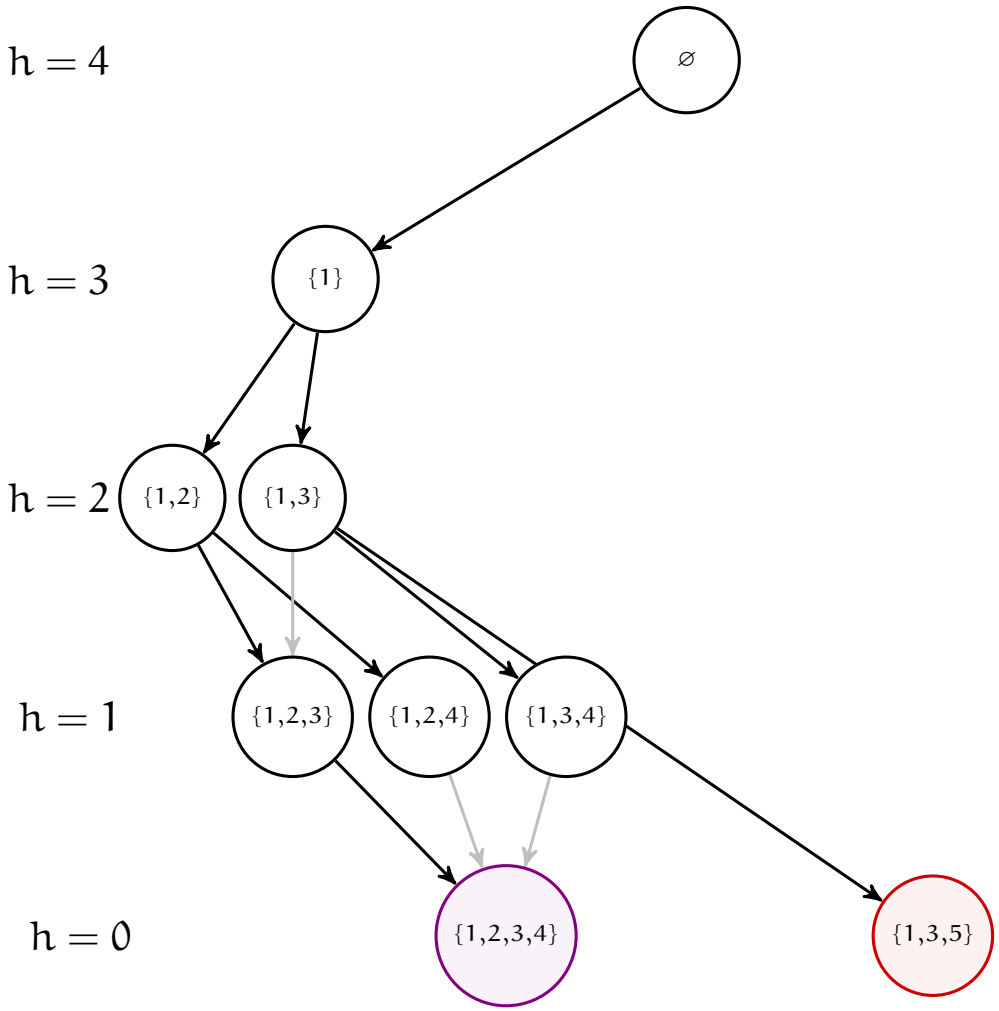
# Esempio di esecuzione



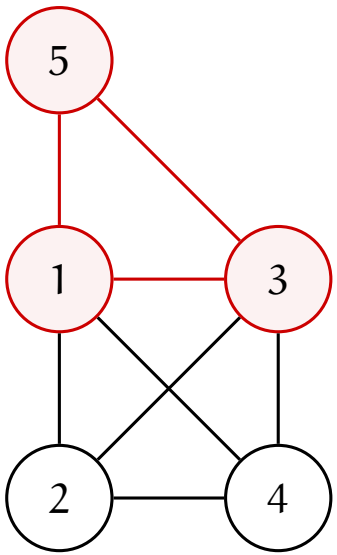



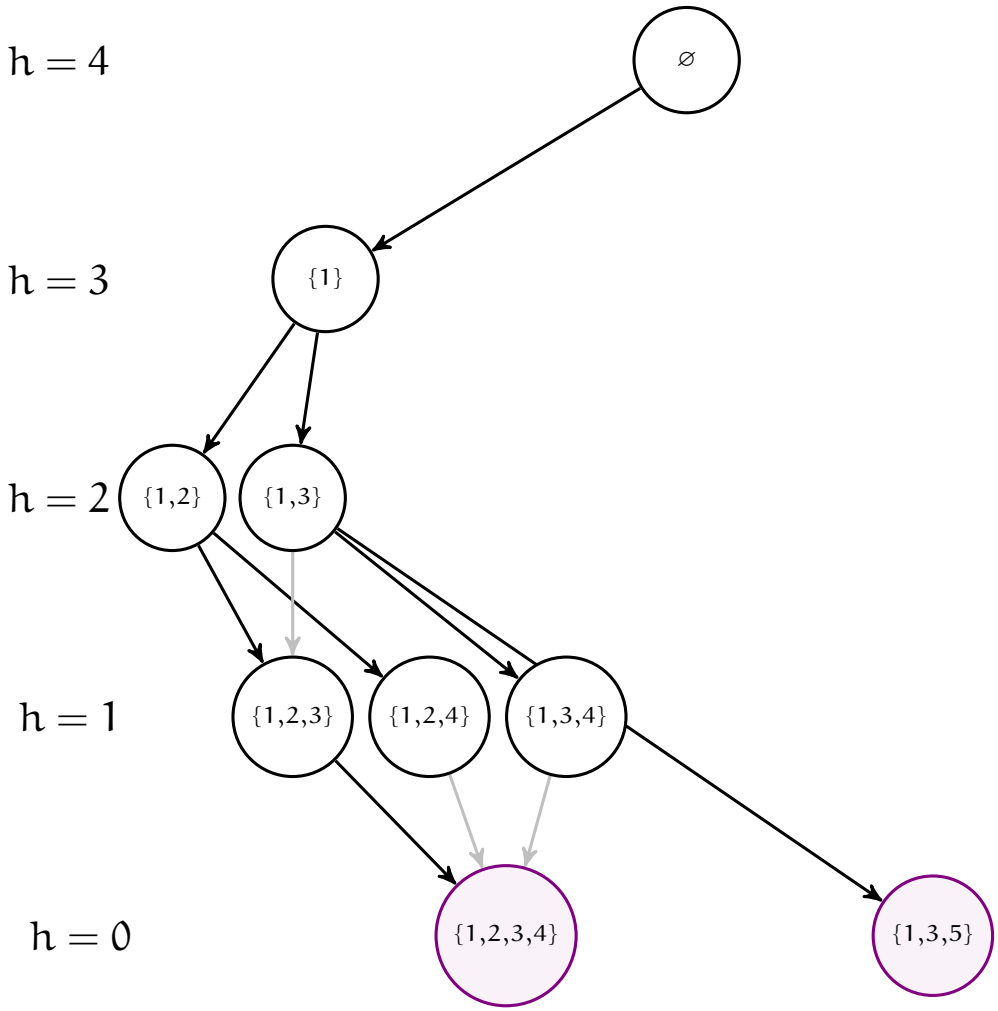
# Esempio di esecuzione



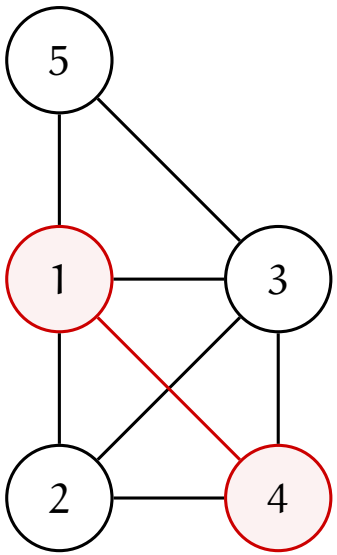



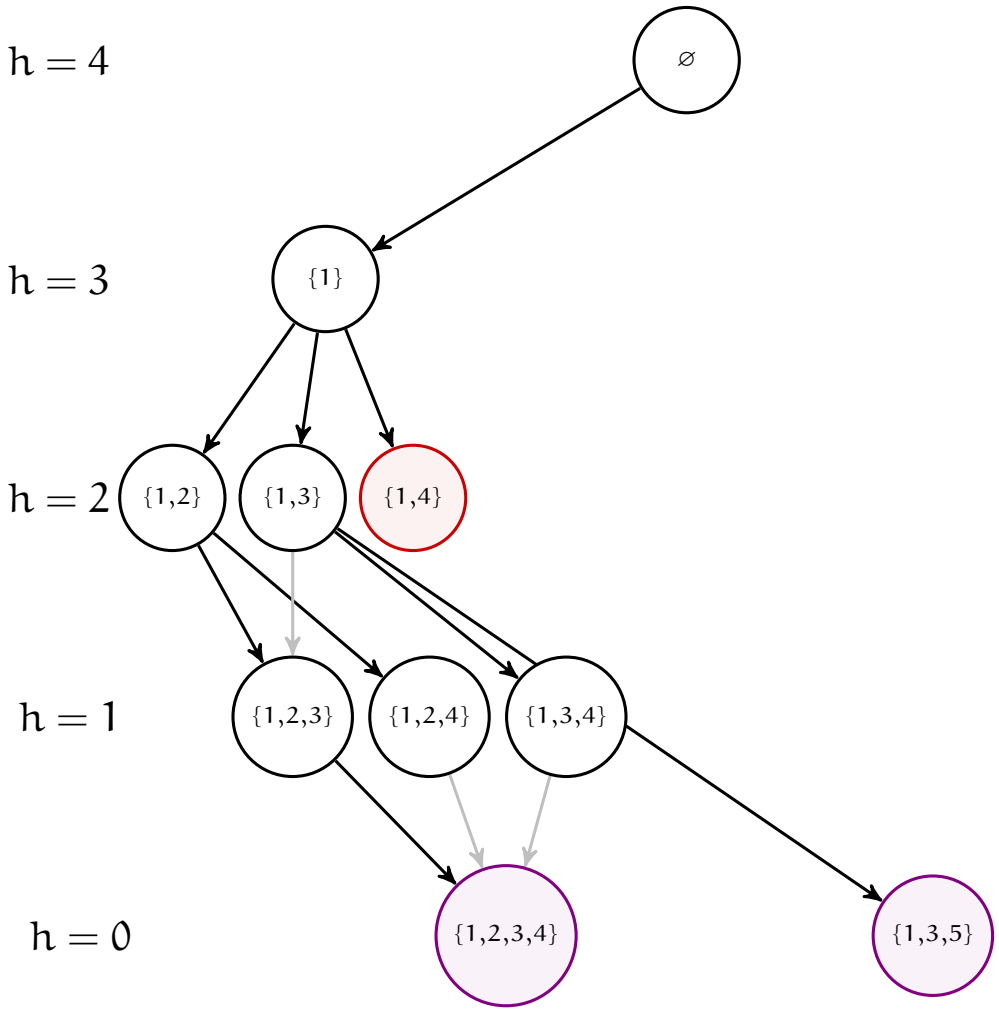
# Esempio di esecuzione



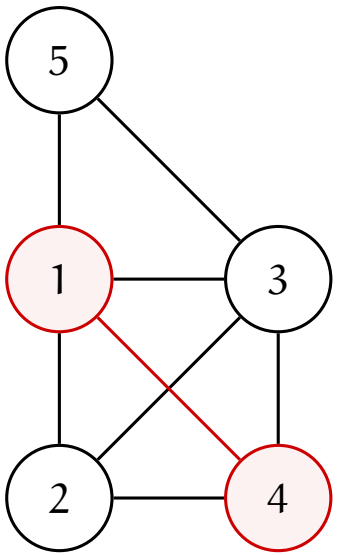



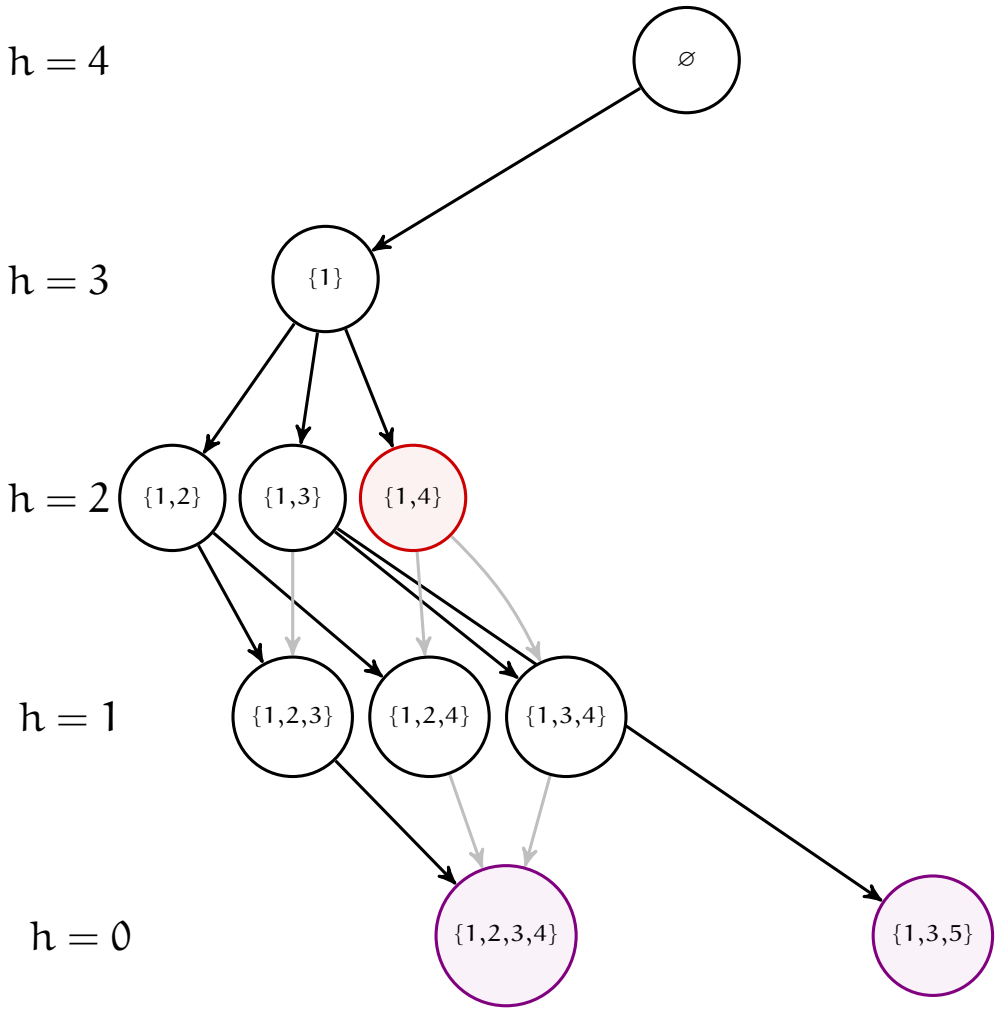
# Esempio di esecuzione



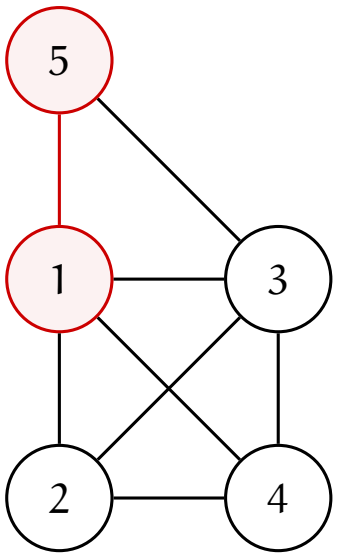



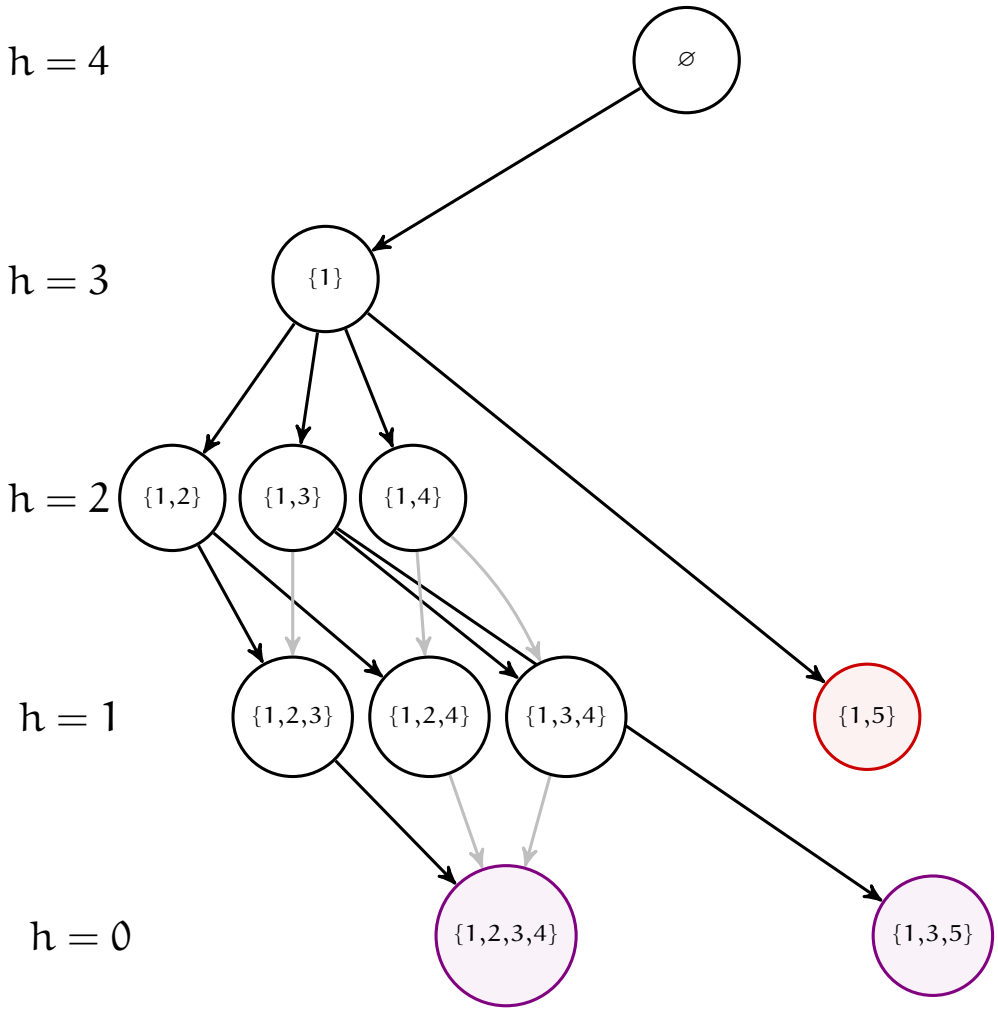
# Esempio di esecuzione



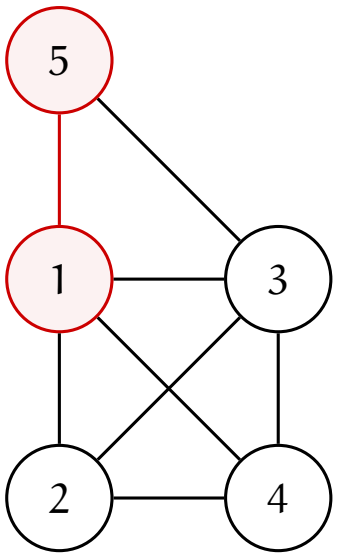



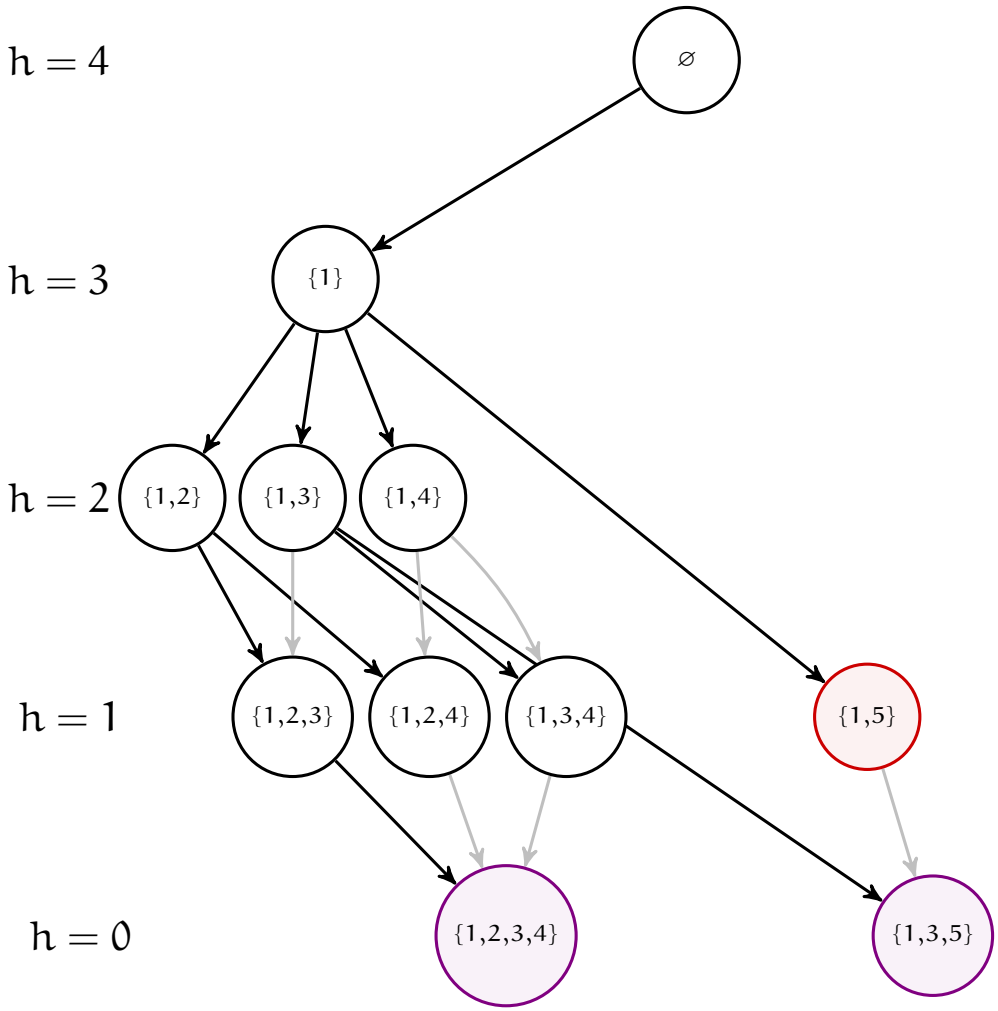
# Esempio di esecuzione



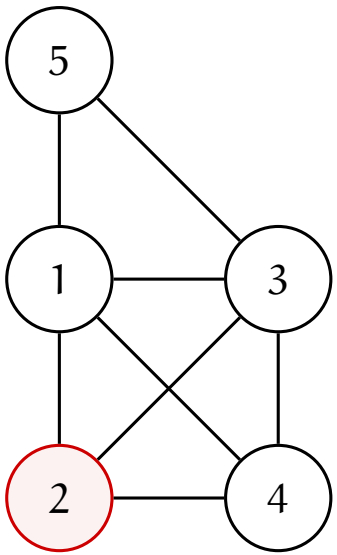



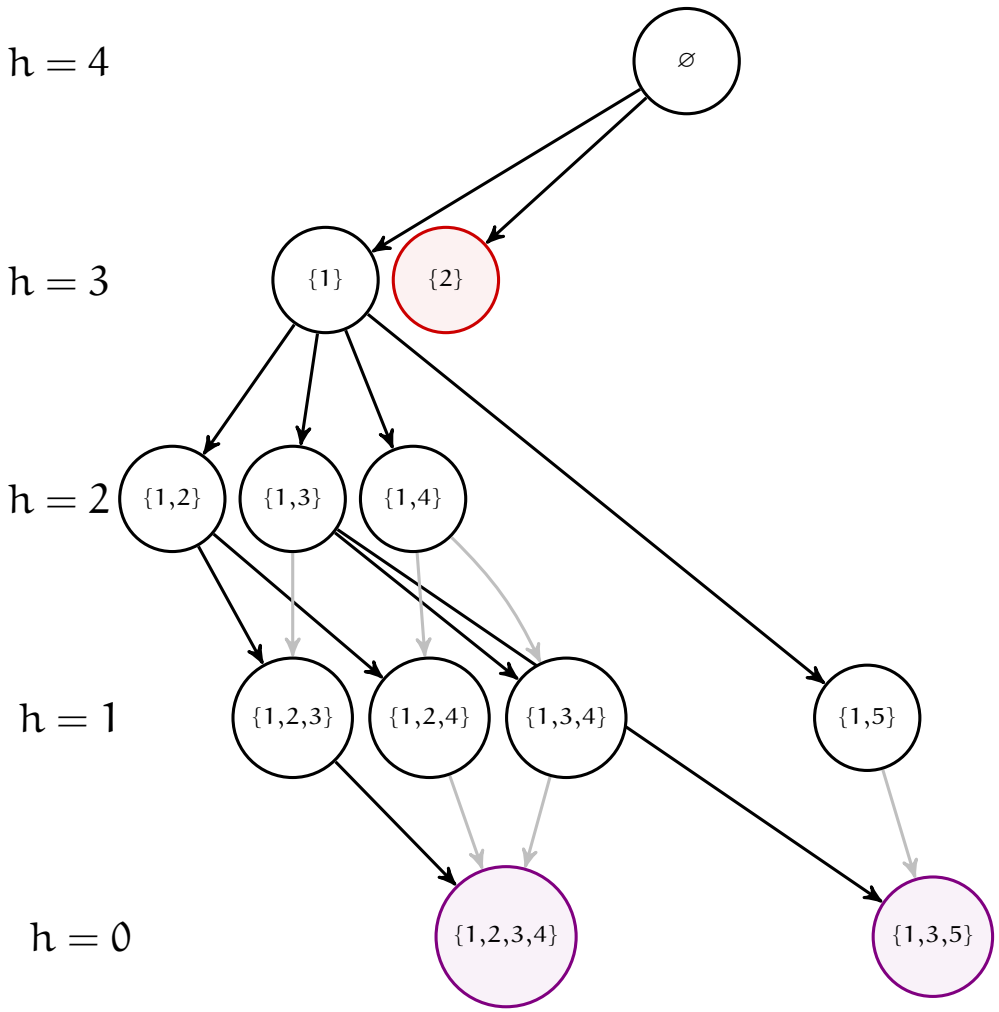
# Esempio di esecuzione



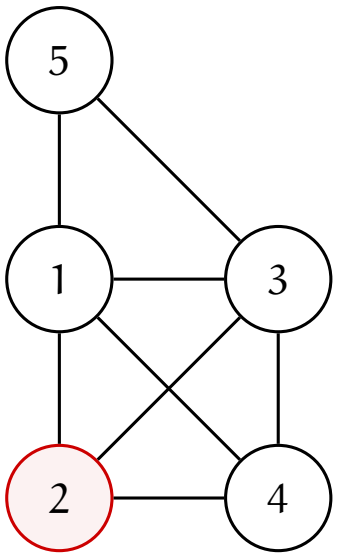



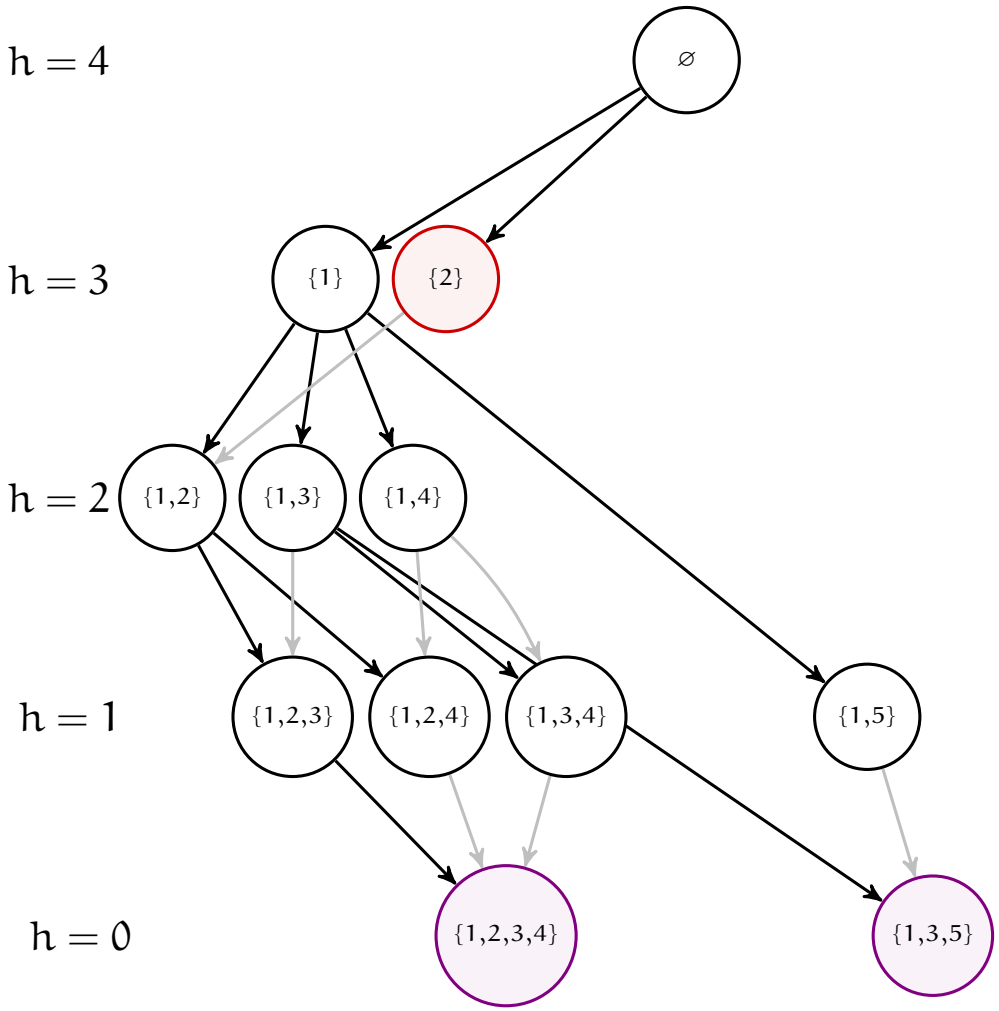
# Esempio di esecuzione



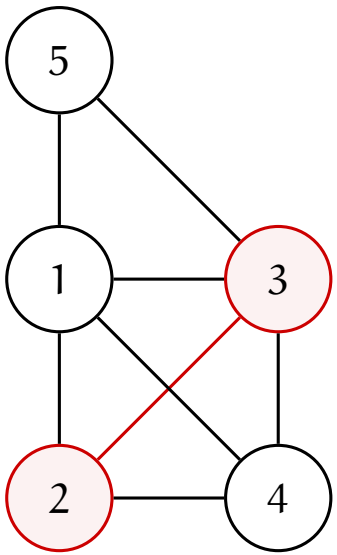



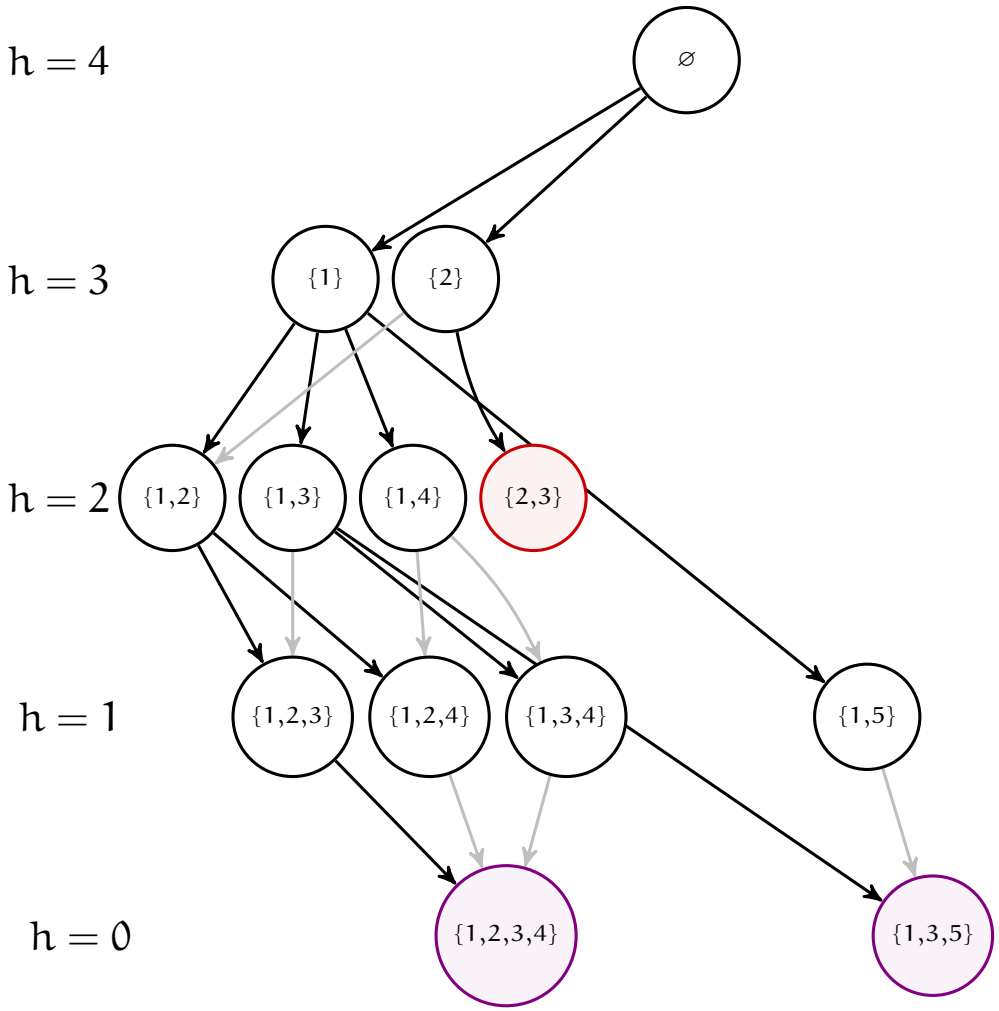
# Esempio di esecuzione



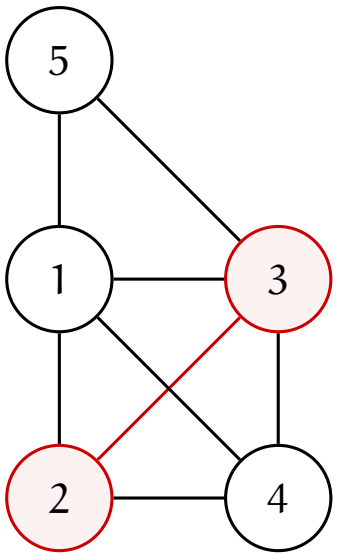



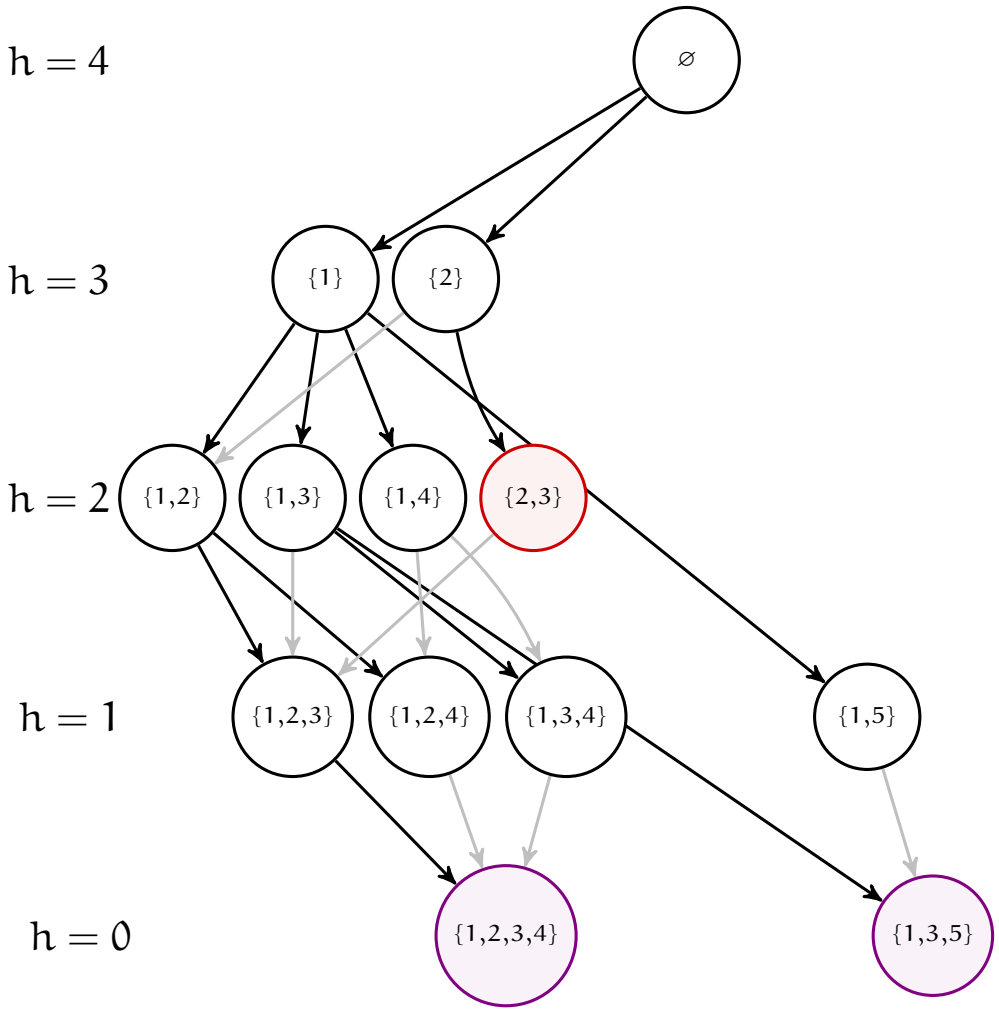
# Esempio di esecuzione



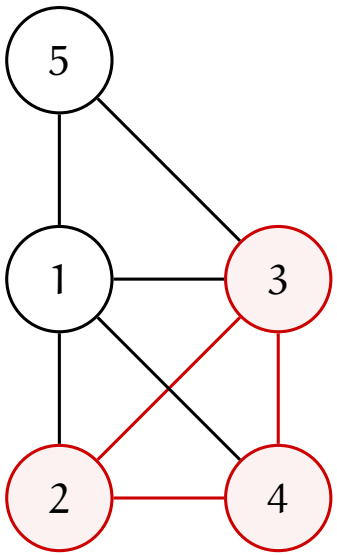



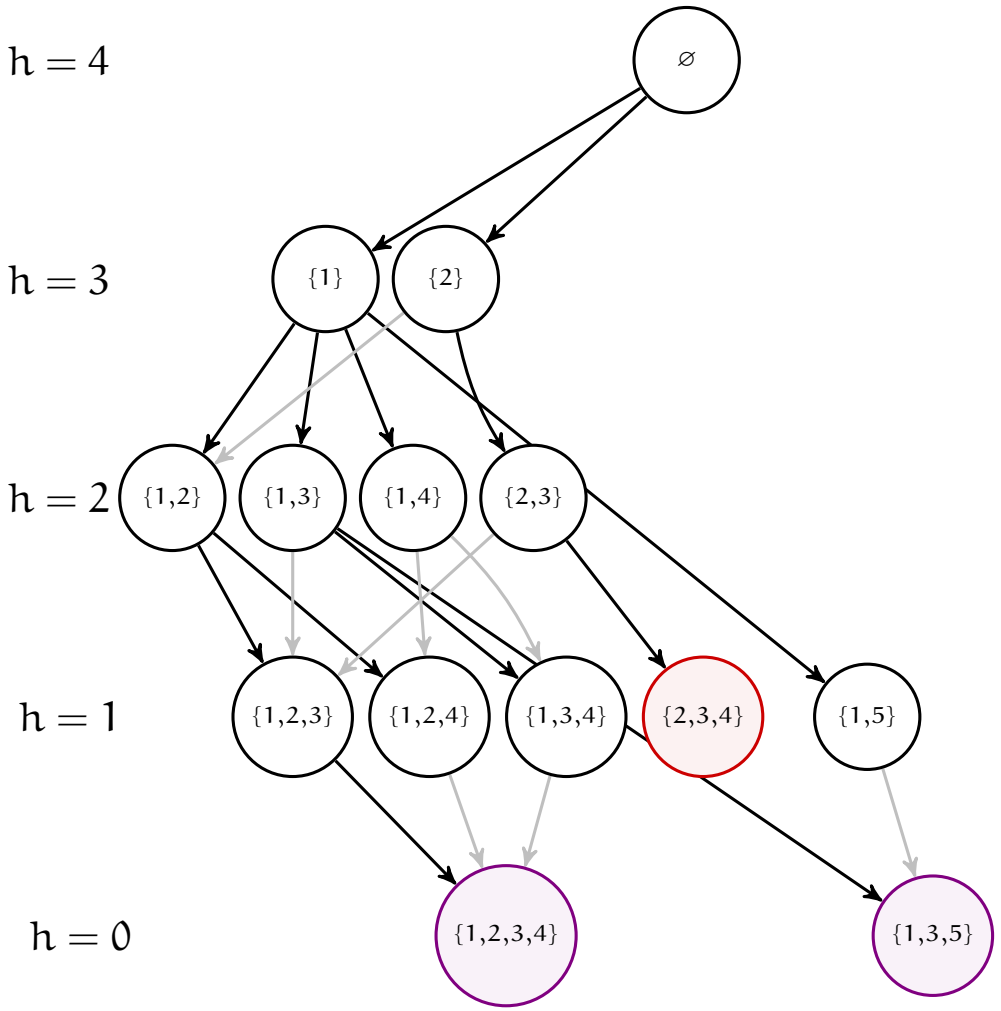
# Esempio di esecuzione



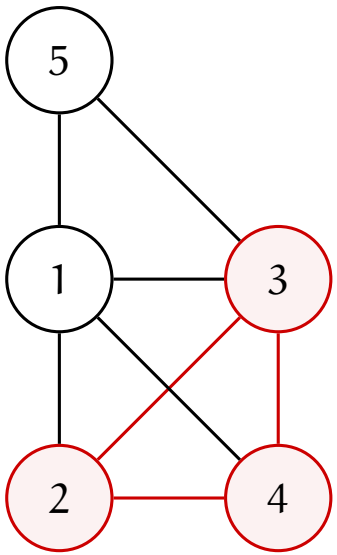



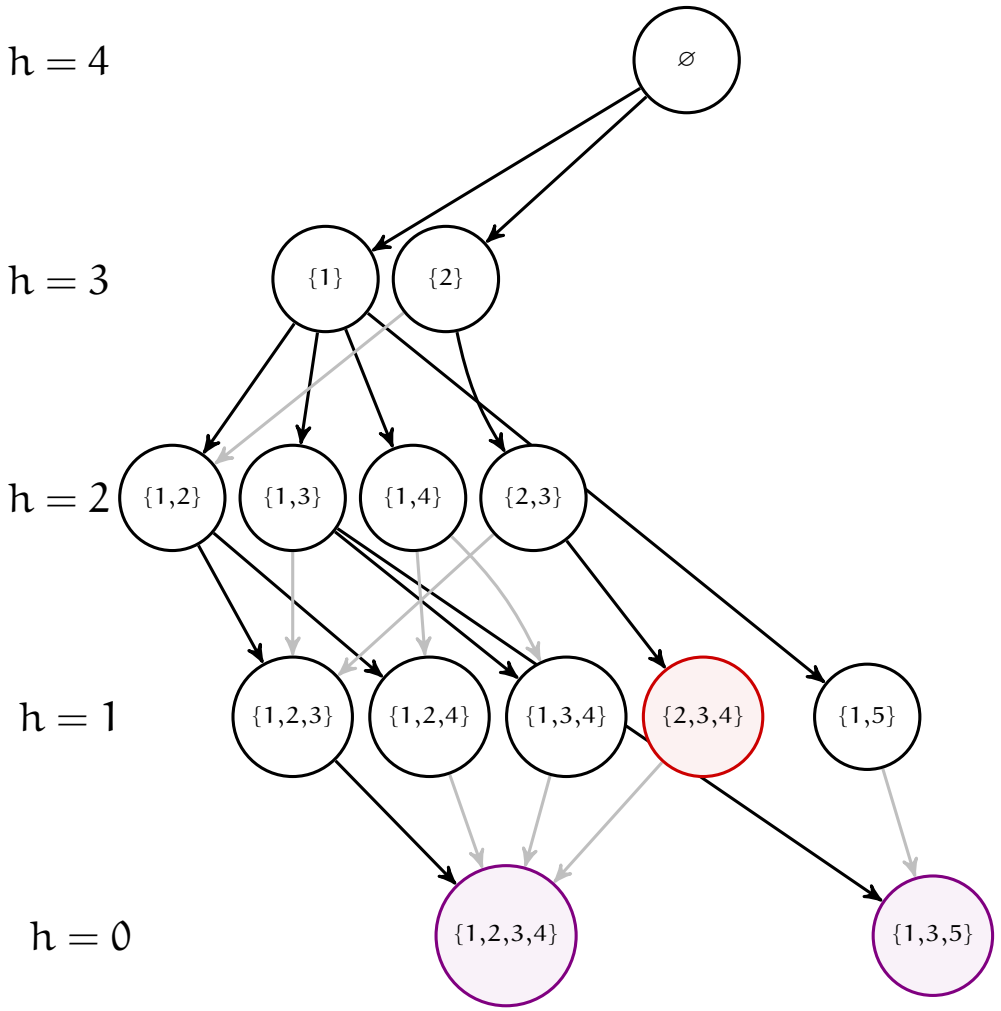
# Esempio di esecuzione



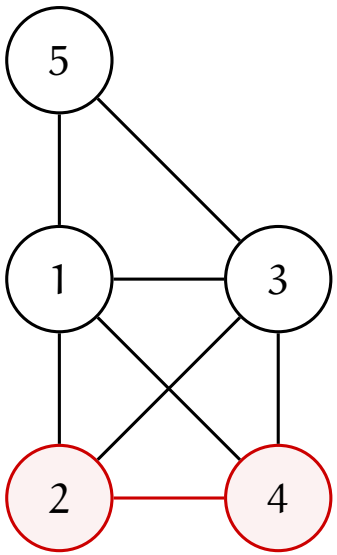



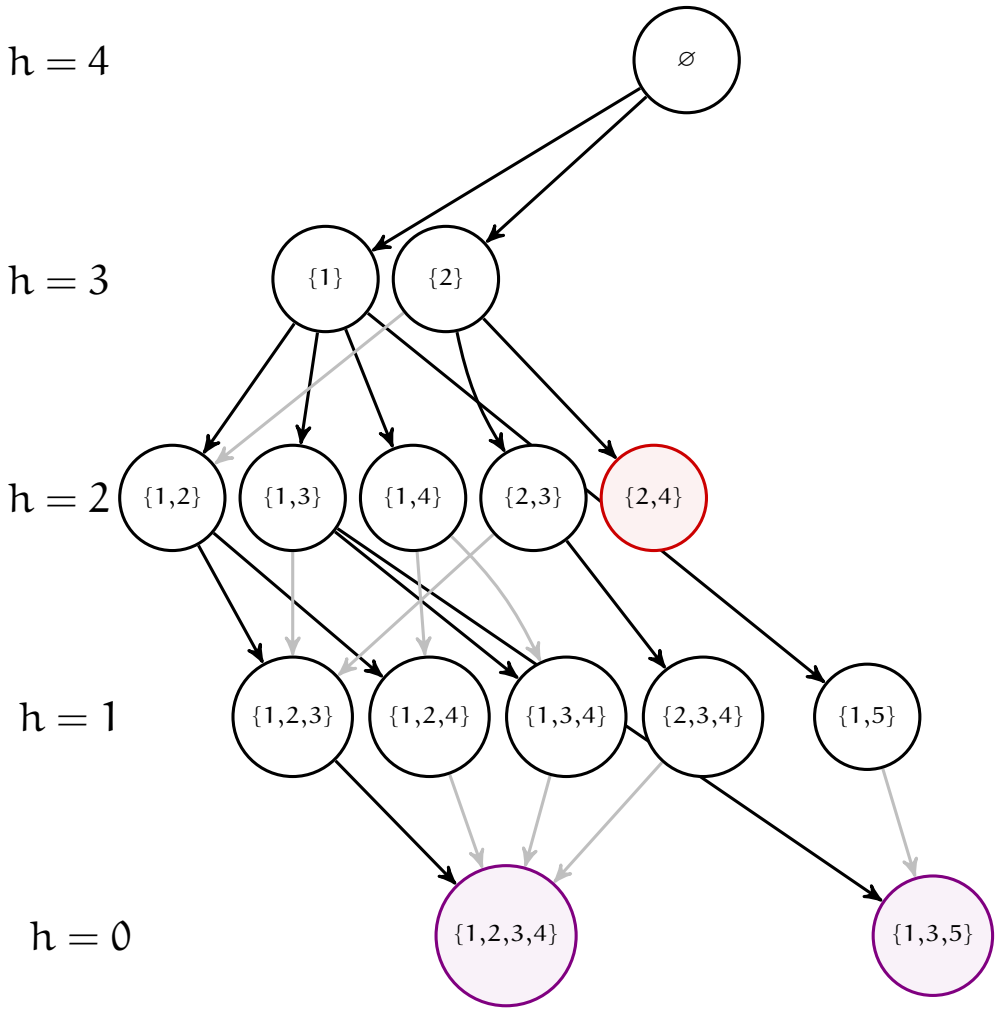
# Esempio di esecuzione



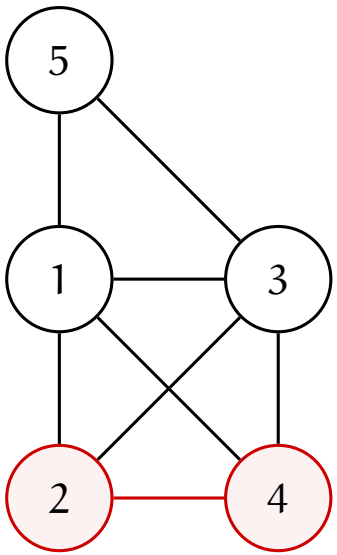



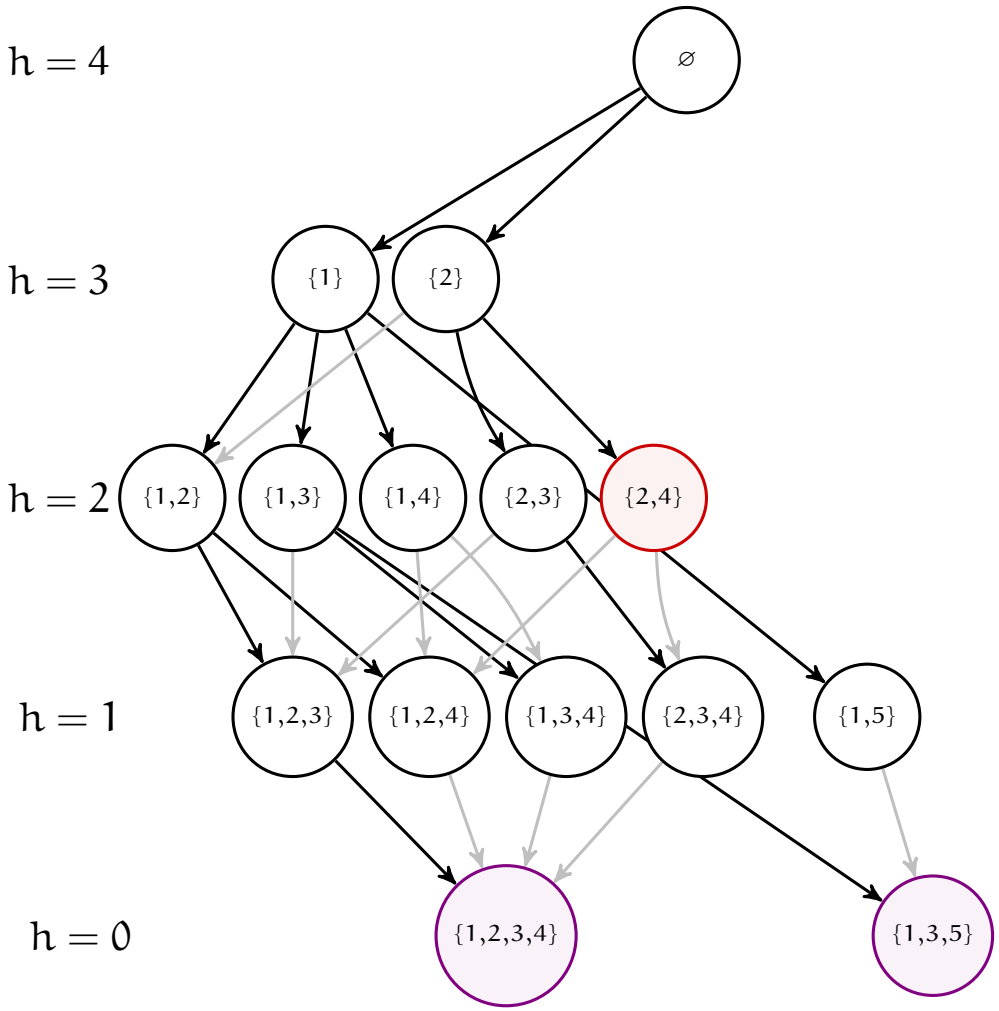
# Esempio di esecuzione



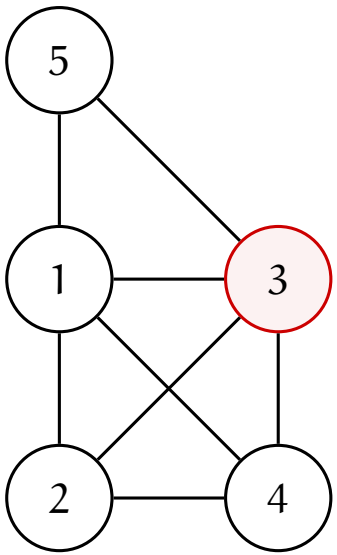



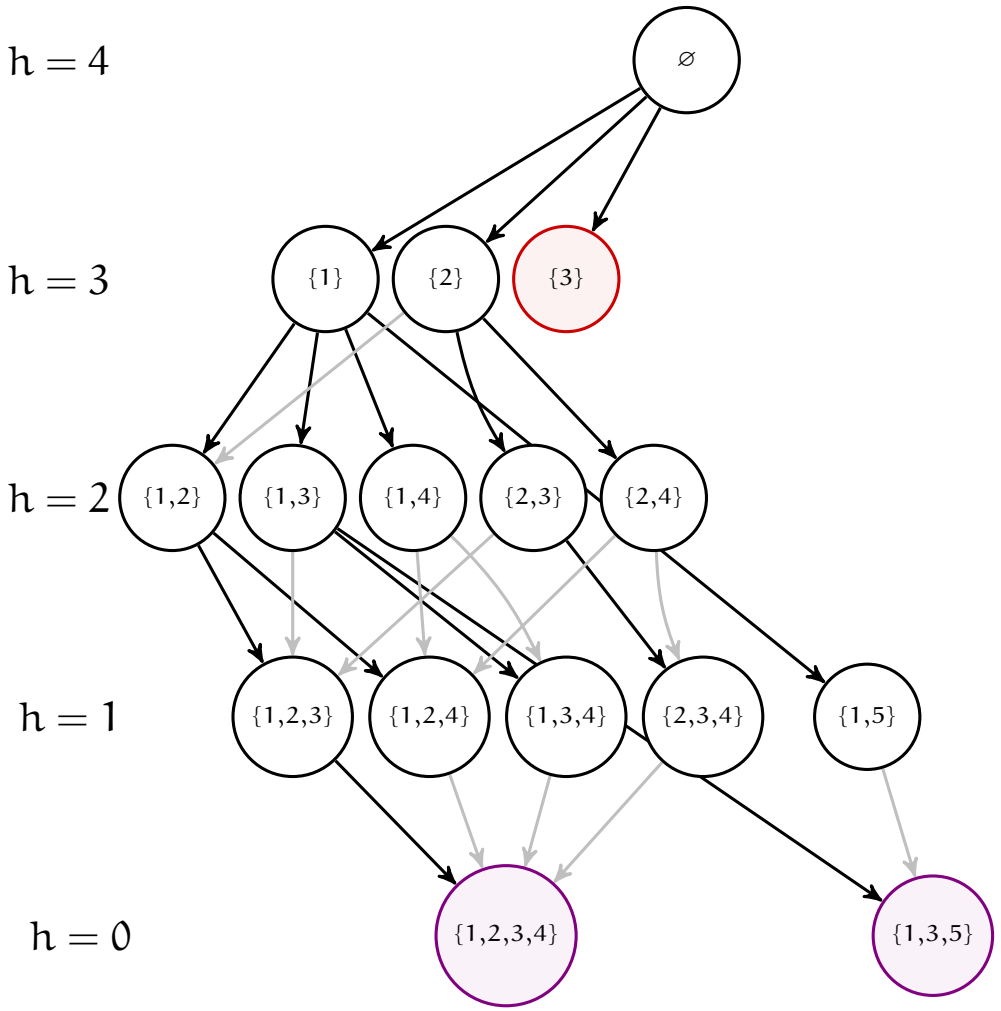
# Esempio di esecuzione



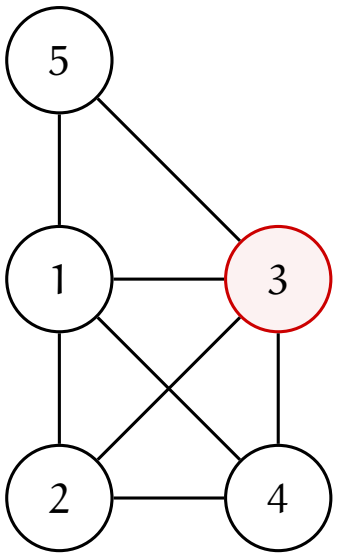



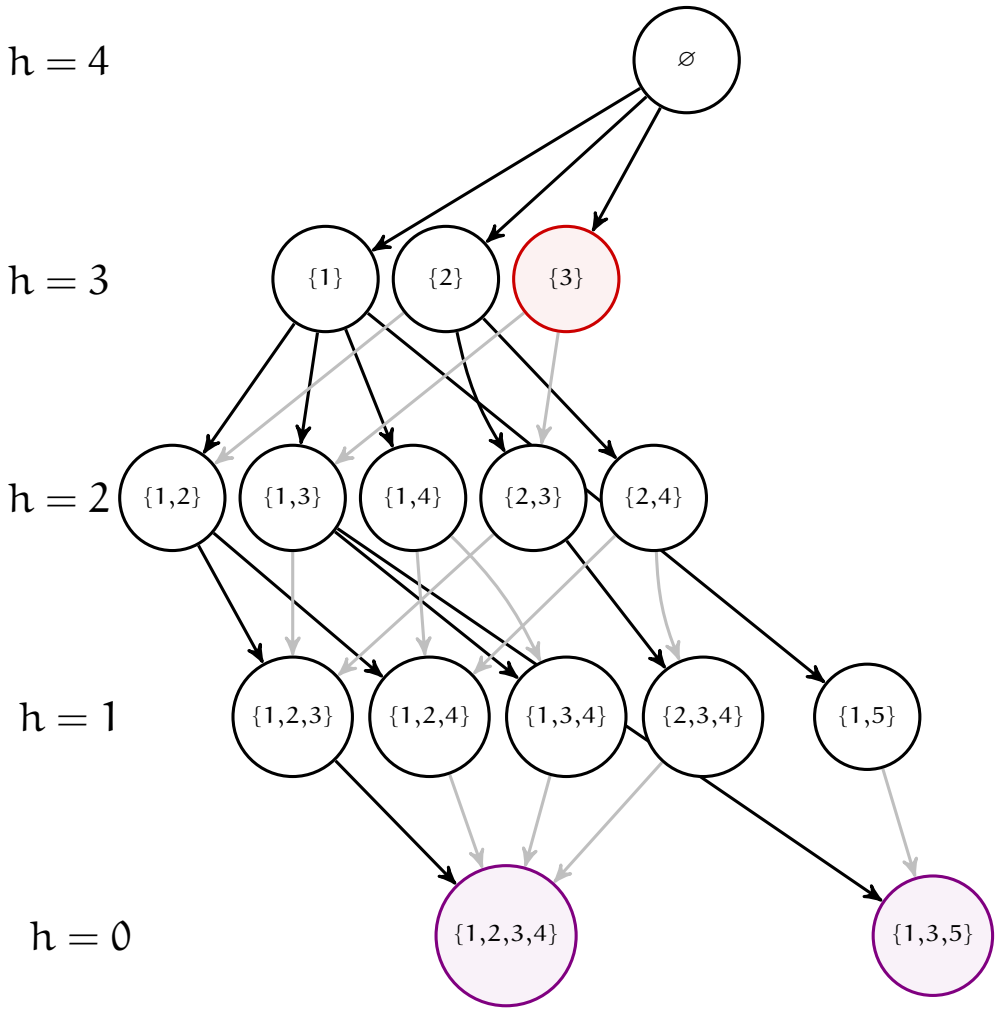
# Esempio di esecuzione



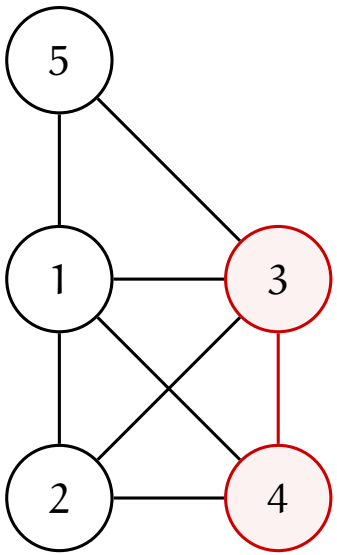



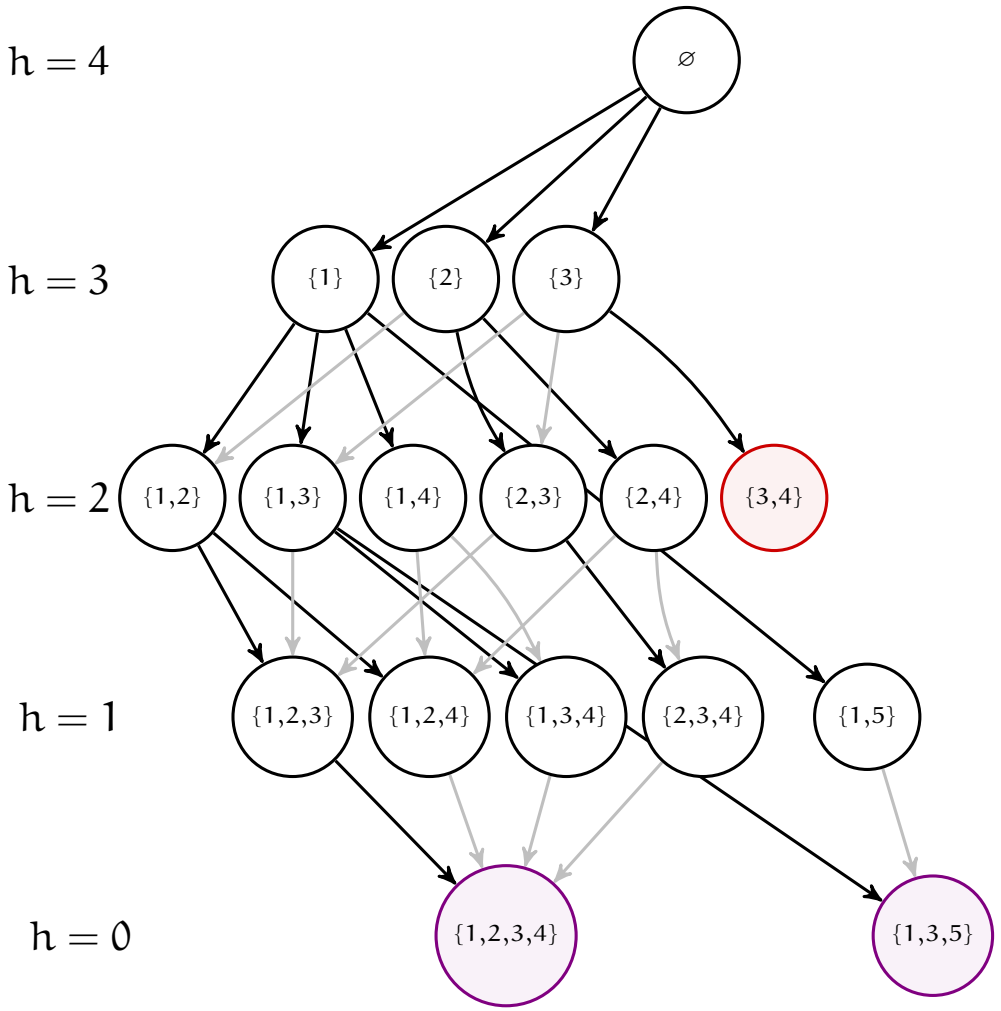
# Esempio di esecuzione



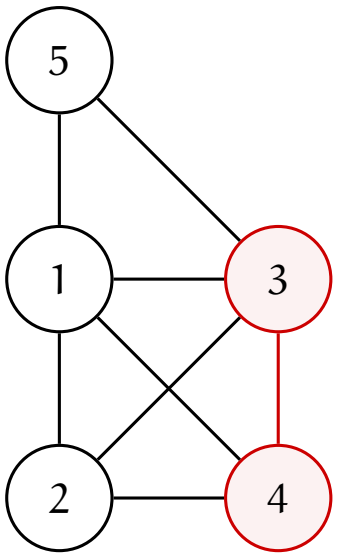



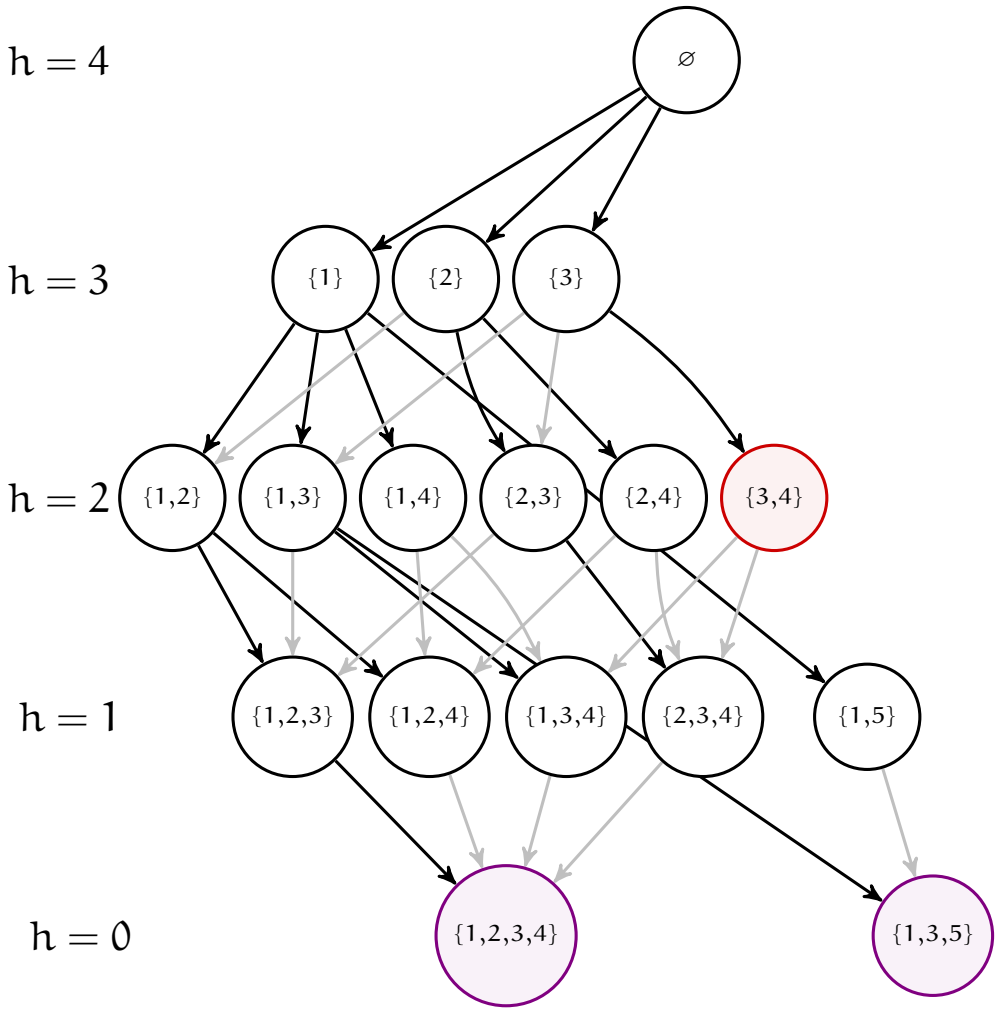
# Esempio di esecuzione



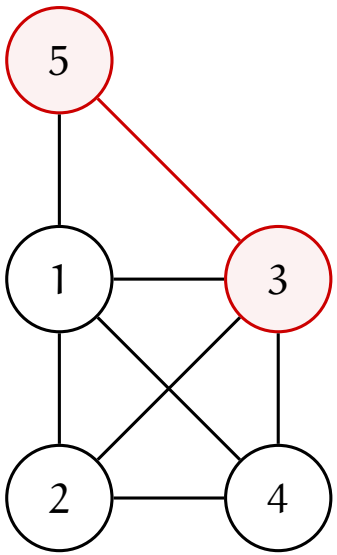



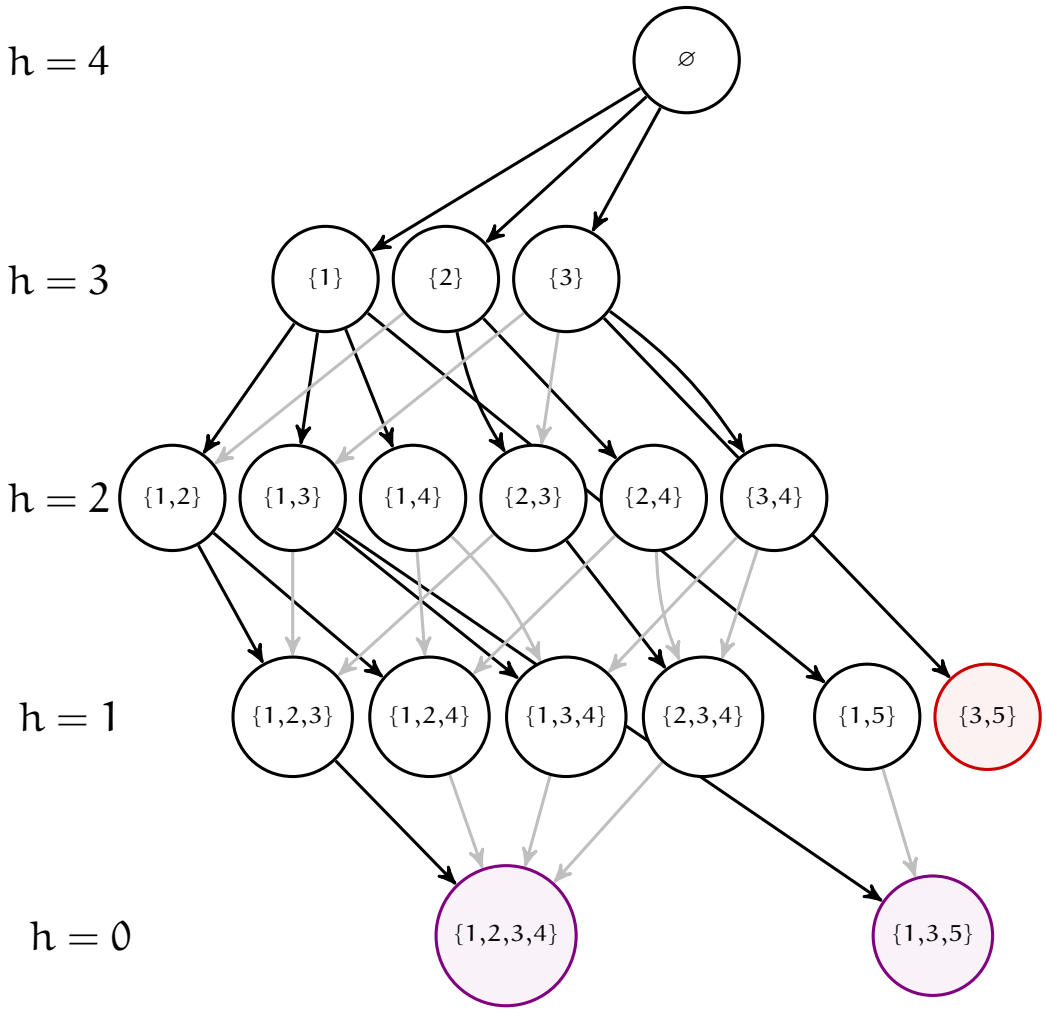
# Esempio di esecuzione



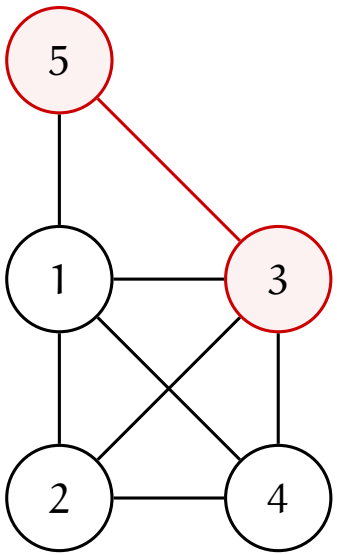



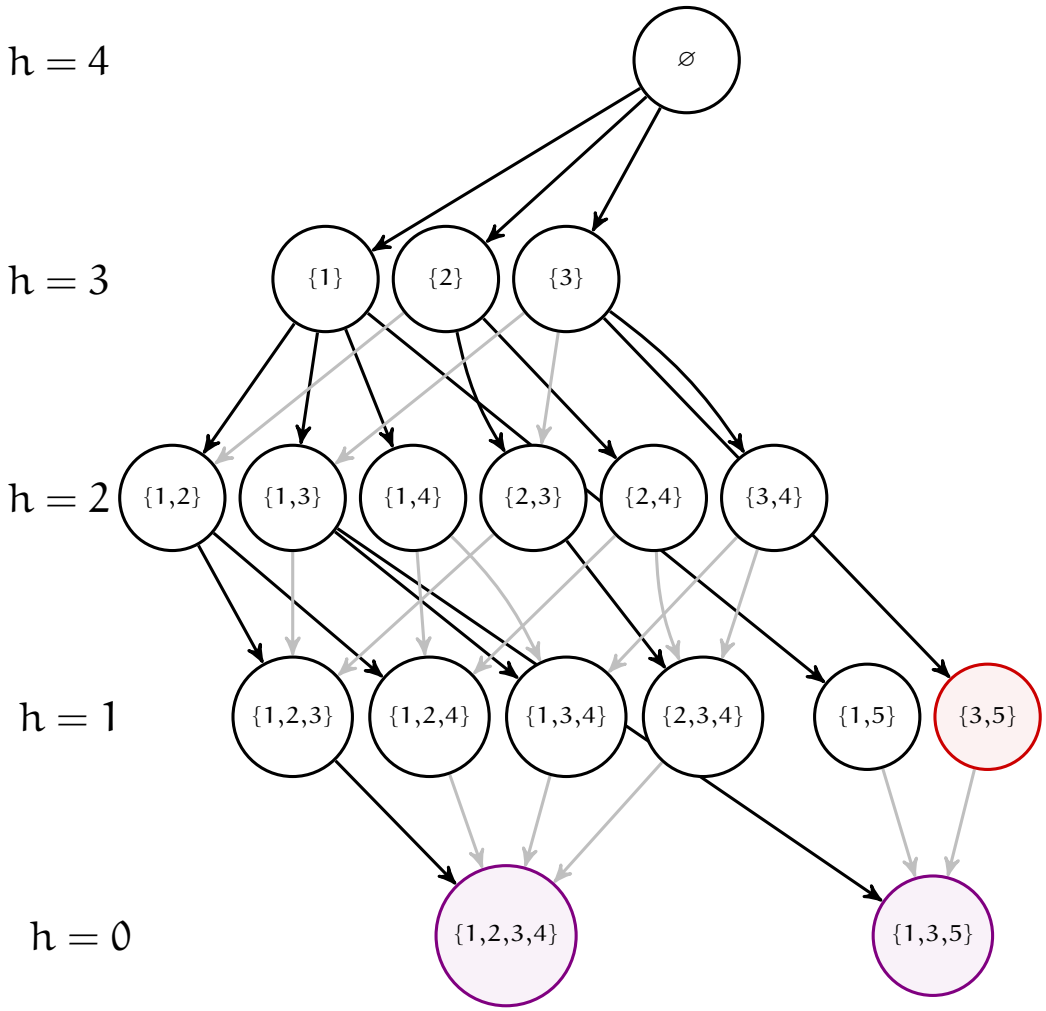
# Esempio di esecuzione



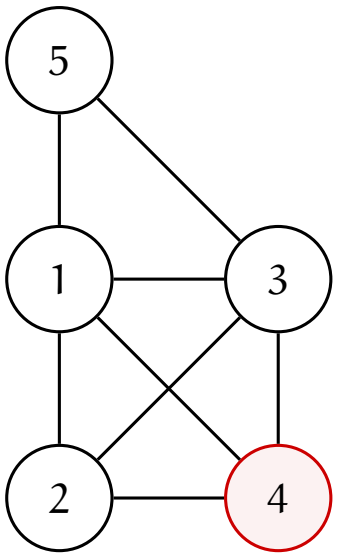



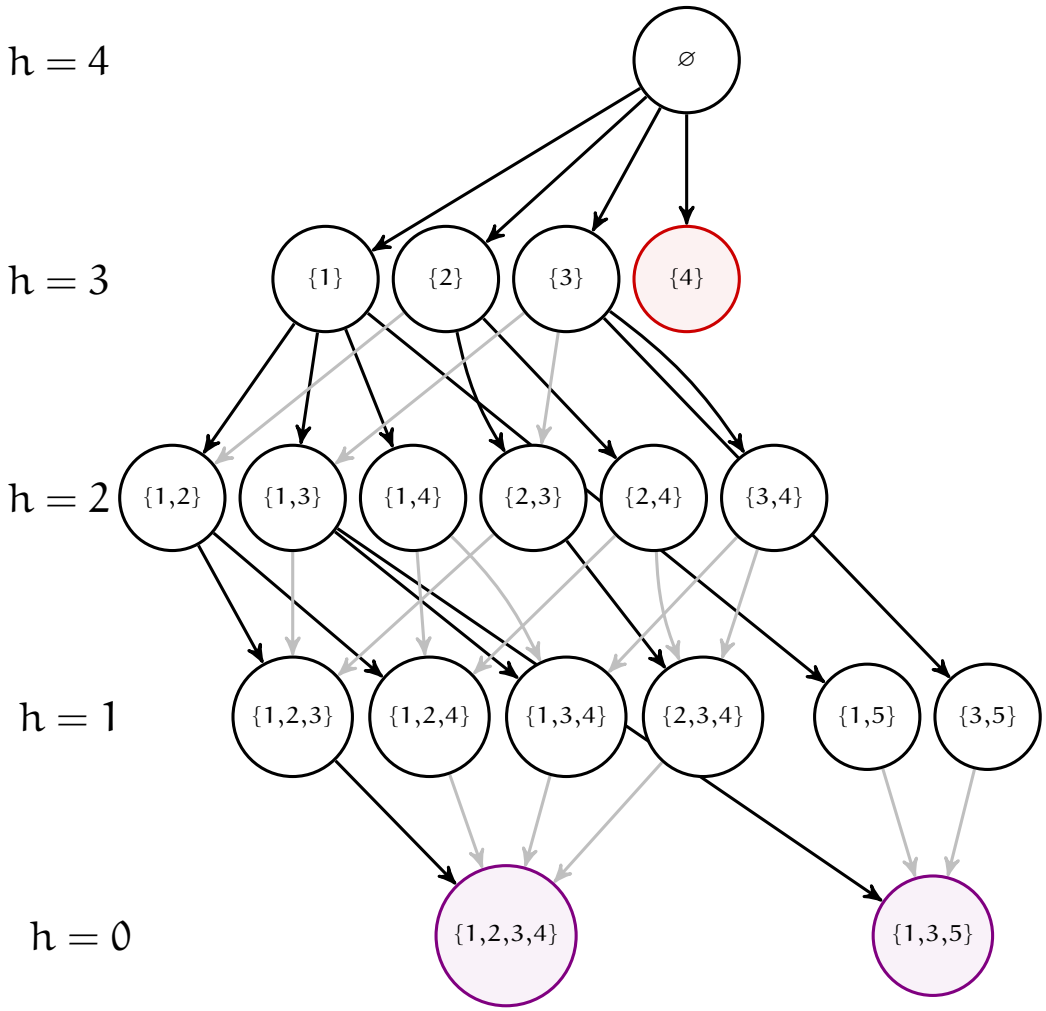
# Esempio di esecuzione



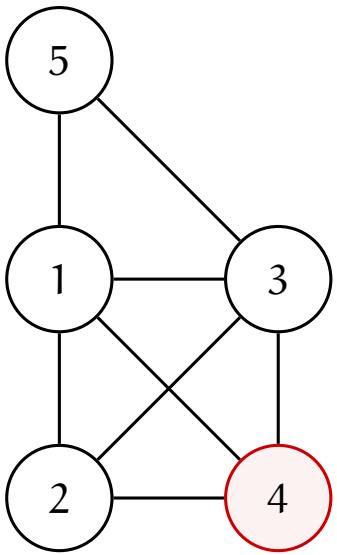



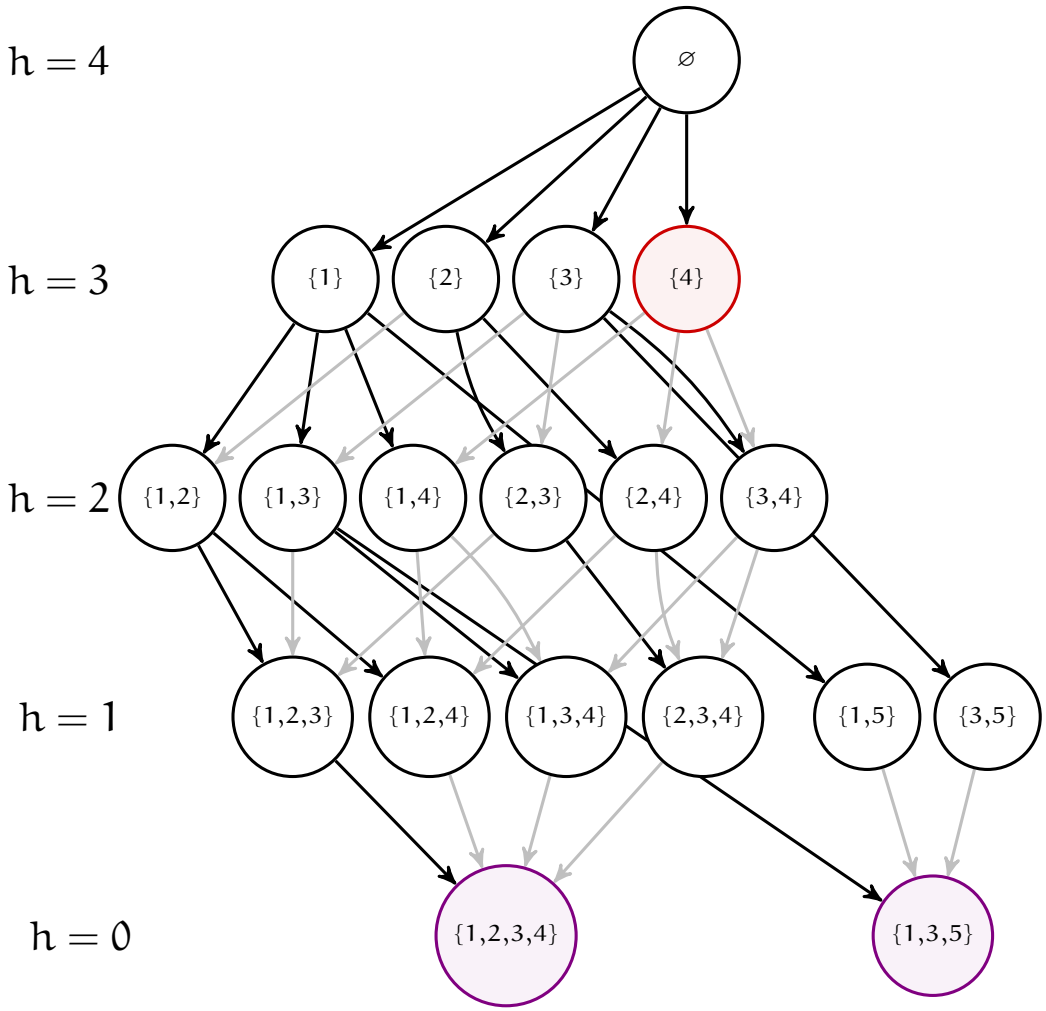
# Esempio di esecuzione



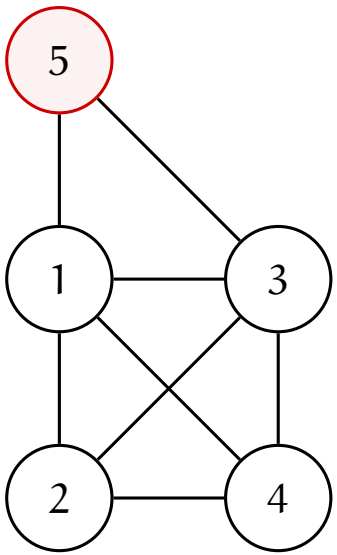



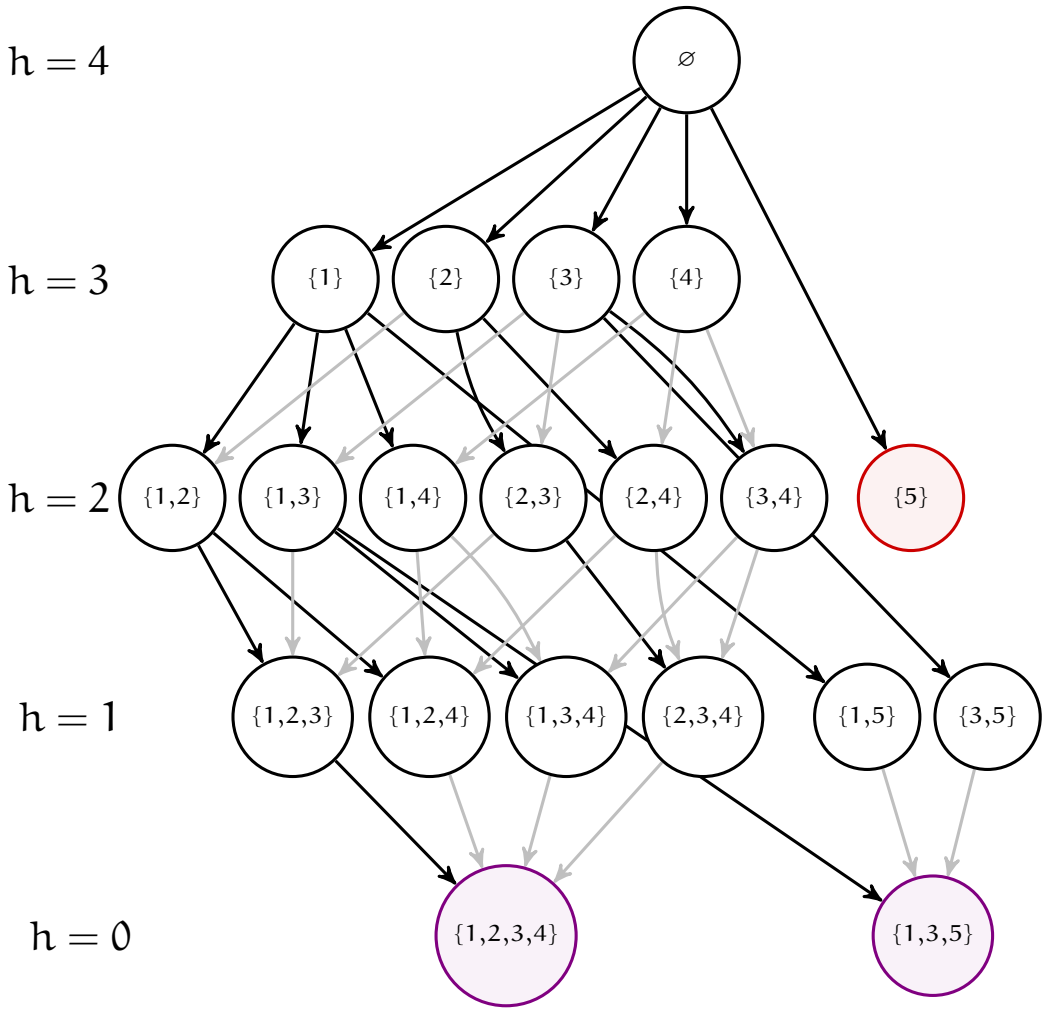
# Esempio di esecuzione



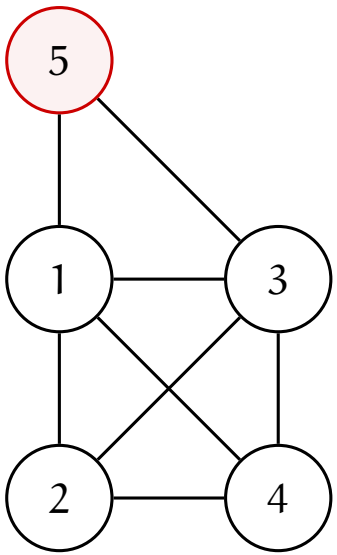



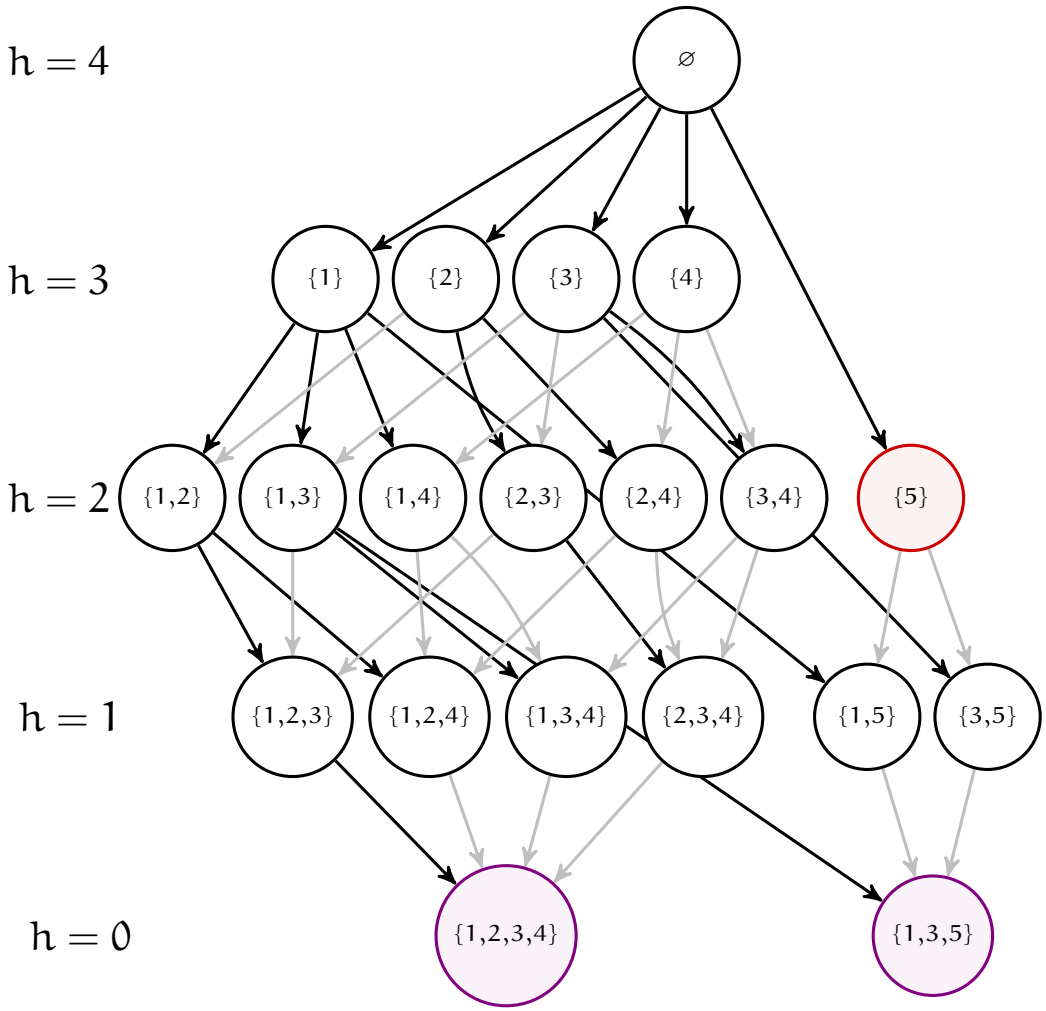
# Esempio di esecuzione



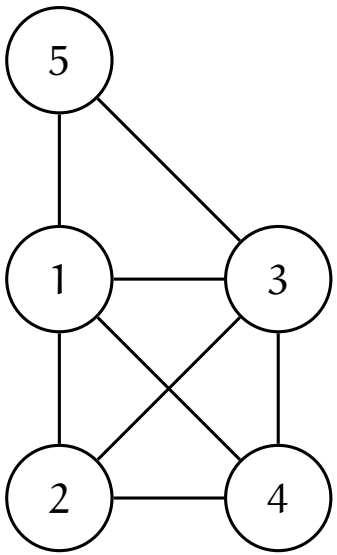



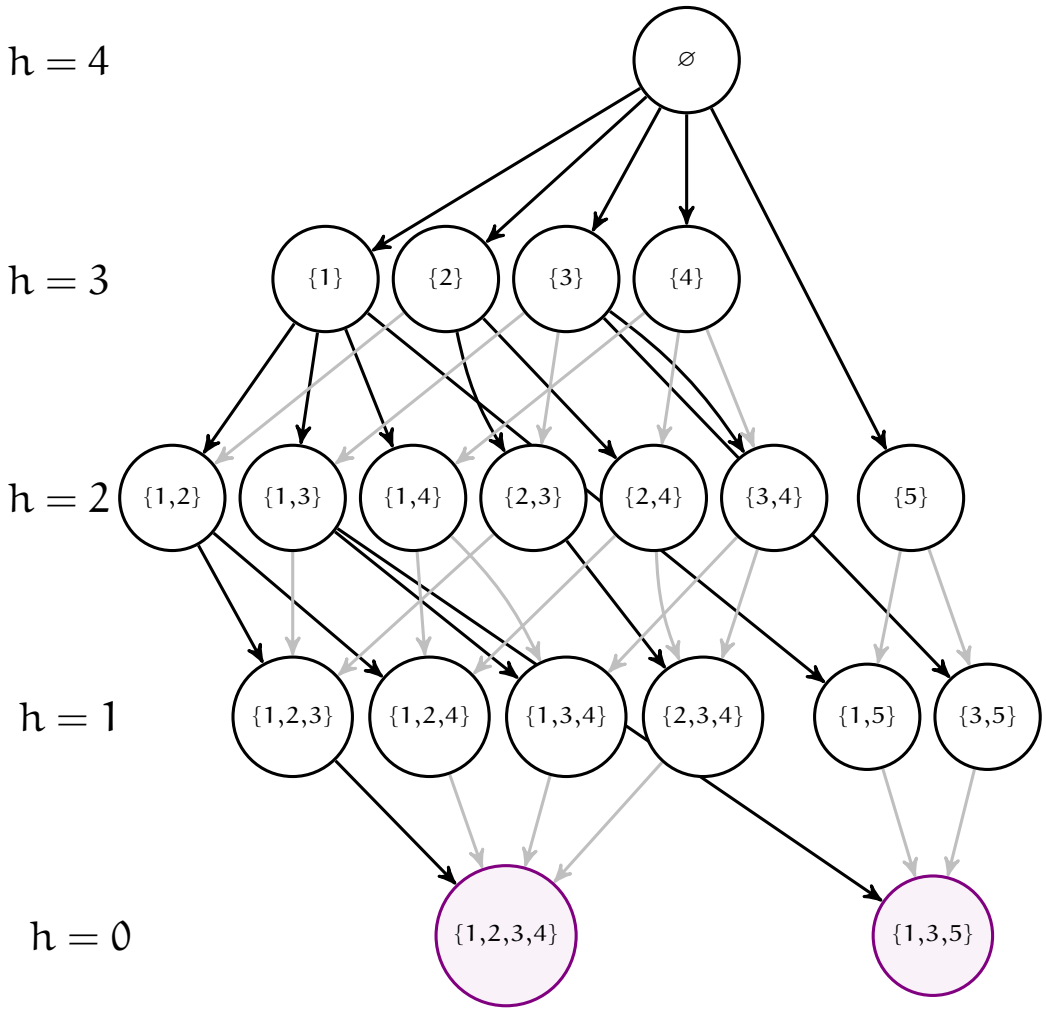
# Esempio di esecuzione



# Esempio di esecuzione



# Correttezza

**Teorema**

*La funzione* BronKerbosch *chiamata con* $\mathsf{C} = \varnothing$, $\mathsf{P} = \mathsf{V}$, $\mathsf{X} = \varnothing$ *visita una e una sola volta tutte le cricche di* $\mathsf{G}$.

**Lemma**

*Quando una chiamata a* BronKerbosch$(\mathsf{C}, \mathsf{P}, \mathsf{X})$ *termina, tutte le cricche che contengono* $\mathsf{C}$ *e che non contengono elementi di* $\mathsf{X}$ *sono state esaminate esattamente una volta. Inoltre nessuna cricca contenente elementi di* $\mathsf{X}$ *viene esaminata.*

# Correttezza

**Dimostrazione.**

Dimostriamo il lemma per induzione su $h(C)$.

**Passo base:** $h(C) = 0$.

$C$ non può essere estesa $\Leftrightarrow$ $C$ è massimale. Allora $P$ e $X$ sono vuoti. Quindi non vengono effettuate altre chiamate a BronKerbosch e la cricca $C$ è stata esaminata esattamente una volta.

# Correttezza

**Dimostrazione.**

**Passo induttivo.**

Consideriamo l'ordine $v_1, \ldots, v_k$ in cui vengono esaminati i nodi in $P$. L'$i$-esima chiamata di BronKerbosch visita $C' = C \cup \{v_i\}$, con $P' = (P \setminus \{v_1, \ldots, v_{i-1}\}) \cap N(v_i)$ e $X' = (X \cup \{v_1, \ldots, v_{i-1}\}) \cap N(v_i)$, quindi visita tutte le cricche contenenti $C \cup \{v_i\}$ che non contengono i nodi in $X$ o i nodi $v_1, \ldots, v_{i-1}$. Quindi le cricche contenute in $C$ sono visitate esattamente una volta. Inoltre, tutte le cricche contenute in $C \cup \{v_i\}$ e contenenti un nodo di $X$ contengono un nodo di $X \cap N(v_i)$ e non sono visitate dall'$i$-esima chiamata di BronKerbosch. □

# Variante di Koch per cricche connesse con archi neri

**function** KOCH($C, RP, UP, RX, UX$)
  **if** $RP = \varnothing \wedge RX = \varnothing$ **then**
    $C$ è una cricca connessa massimale
  **end if.**
  **for all** $v \in RP$ **do**
    $RP' \leftarrow (RP \cup (UP \cap N_b(v))) \cap N(v)$
    $RX' \leftarrow (RX \cup (UX \cap N_b(v))) \cap N(v)$
    $UP' \leftarrow UP \cap N_w(v)$
    $UX' \leftarrow UX \cap N_w(v)$
    KOCH$(C \cup \{v\}, RP', UP', RX', UX')$
    $RP \leftarrow RP \setminus \{v\}$
    $RX \leftarrow RX \cup \{v\}$
  **end for.**
**end function.**

# Correttezza di Koch

**Teorema**

*La funzione* KOCH, *chiamata* $n$ *volte con* $C = \{v_i\}$, $RP = N_b(v_i) \setminus V_{i-1}$, $UP = N_w(v_i) \setminus V_{i-1}$, $RX = V_{i-1} \cap N_b(v_i)$, $UX = V_{i-1} \cap N_w(v_i)$ *rispettivamente, visita una e una sola volta tutte le cricche connesse del grafo* $G$.

**Lemma**

*Quando una chiamata a* KOCH$(C, RP, UP, RX, UX)$ *termina, tutte le cricche connesse che contengono* $C$ *e che non contengono elementi di* $RX \cup UX$ *sono state esaminate esattamente una volta. Inoltre nessuna cricca contenente elementi di* $RX \cup UX$ *viene esaminata.*

# Costo computazionale

## Costo di Bron-Kerbosh

Il tempo di esecuzione dell'algoritmo di Bron-Kerbosh è $O(\Delta|\mathcal{C}|)$, dove $\mathcal{C}$ indica l'insieme di tutte le cricche di $G$, e la memoria utilizzata è $O(\Delta^2)$.

## Costo di Koch

Il tempo di esecuzione dell'algoritmo di Koch è $O(\Delta|\mathcal{C}^c|)$, dove $\mathcal{C}^c$ indica l'insieme di tutte le cricche connesse di $G$, e la memoria utilizzata è $O(\Delta^2)$.

# Costo computazionale per MCLI

## Costo di Koch

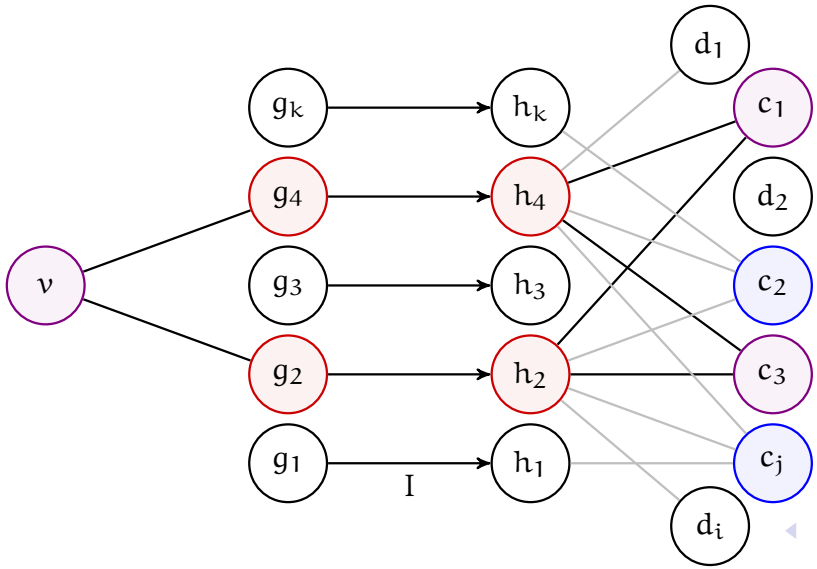
Detto $L = \sum_{i=1}^{k} |l_G^{-1}(i)||l_H^{-1}(i)|$, il tempo di esecuzione dell'algoritmo di Koch esplicito è $O(L|\mathcal{J}|)$ e la memoria occupata è $O(L|V|)$.

# Idea principale

$$N = \{v \in V : I_G \cap N_G(v) \neq \emptyset\}$$

$$C_v = \bigcap_{n \in N_G(v) \cap I_G} N_H(I(n)) \setminus I_H$$

$$RP \cup RX = \bigcup_{v \in N} \{v\} \times \{c \in C_v : I^{-1}(N_H(c) \cap I_H) = N_G(v) \cap I_G\}$$



# Algoritmo di Koch implicito

**function** KochImplicit($I, N, X$)
  maximal $\leftarrow$ **true**
  **for all** $v \in N$ **do**
    $C_v \leftarrow \bigcap_{n \in N_G(v) \cap I_G} N_H(I(n)) \setminus I_H$
    $C_v \leftarrow \{c \in C_v : I^{-1}(N_H(c) \cap I_H) = N_G(v) \cap I_G\}$
    **if** $C_v \neq \varnothing$ **then**
      maximal $\leftarrow$ **false**
    **end if.**
    **if** $v \notin X$ **then**
      **for all** $c \in C_v$ **do**
        KochImplicit$(I \cup \{(v, c)\}, N \cup N_G(v) \setminus \{v\}, X)$
      **end for.**
    **end if.**
    $X \leftarrow X \cup \{v\}$
  **end for.**
  **if** maximal **then**
    I è un isomorfismo massimale
  **end if.**
**end function.**

# Costo computazionale per MCLI

## Costo di Koch

Detto $L = \sum_{i=1}^{k} |l_G^{-1}(i)||l_H^{-1}(i)|$, il tempo di esecuzione dell'algoritmo di Koch esplicito è $O(L|\mathcal{I}|)$ e la memoria occupata è $O(L|V|)$.

## Costo di Koch implicito

Il tempo di esecuzione dell'algoritmo di Koch implicito è $O\left(\Delta^3 \sum_{I \in \mathcal{I}} |I|\right)$ e la memoria occupata è $O(\Delta|V|)$.

# Risultati sperimentali

| | N | M | k | Δ | Koch | KochI |
|---|---|---|---|---|---|---|
| long_path | 100 | 99 | 5 | 2 | 0.7s | 0.01s |
| very_long_path | 300 | 299 | 10 | 2 | 12.6s | 0.01s |
| small_random | 50 | 50 | 2 | 5 | 1.8s | 0.04s |
| random | 100 | 100 | 4 | 7 | 11.6s | 0.10s |
| mb_random | 200 | 200 | 5 | 7 | 30s | 0.07s |
| big_random | 500 | 500 | 15 | 7 | 41s | 0.03s |
| huge_random | 5000 | 10000 | 40 | 14 | 46h | 0.77s |
| few_colors | 500 | 500 | 5 | 7 | 2h | 1.99s |
| medium_colors | 500 | 500 | 50 | 7 | 4.8s | 0.01s |
| medium_colors_huge | 5000 | 10000 | 500 | 14 | 40m | 0.09s |
| many_colors | 500 | 500 | 100 | 7 | 1.1s | 0.01s |
| many_colors_huge | 5000 | 5000 | 1000 | 9 | 10m | 0.06s |
| many_colors_very_huge | 40000 | 100000 | 9999 | 17 | $>$ 100h | 1.5s |
| clique | 13 | 78 | 4 | 12 | 0.8s | 0.8s |
| small_clique | 8 | 28 | 1 | 7 | 1.9s | 1.5s |
| small_dense | 80 | 1200 | 80 | 58 | 0.03s | 0.1s |
| dense | 100 | 2000 | 100 | 70 | 0.05s | 0.4s |
| big_dense | 150 | 4500 | 150 | 114 | 0.53s | 14s |