Allocation de registres

(Ces transparents sont à visionner en mode plein écran)

Alexis Nasr
Franck Dary
Pacôme Perrotin

Compilation – L3 Informatique
Département Informatique et Interactions
Aix Marseille Université

Registres fictifs et registres réels I

- Le pré-assembleur définit un nombre illimité de registres **fictifs** $r_1, r_2, r_3, \ldots$ correspondant en général aux temporaires du code trois adresses.
- Mais le processeur ne dispose que d'un nombre limité de registres **réels**, en général inférieur au nombre de registre fictifs.
- Le but de l'allocation de registres est de désigner pour chaque registre fictif, un registre réel.
- Le principe de l'allocation consiste à assigner le même registre réel à des registres fictifs qui ne sont jamais vivants en même temps.

Registres fictifs et registres réels II

- Pour cela, on construit un **graphe d'interférence**, en s'aidant de la solution au graphe d'analyse (les ensembles *in* et *out*).
- Le graphe d'interférence est un graphe non orienté comportant autant de sommets qu'il y a de registres fictifs.
- Une arête (r_1, r_2) indique que les deux registres fictifs r_1 et r_2 ne peuvent être assignés à un même registre réel.

Registres fictifs et registres réels III

- On **colore** les sommets du graphe de sorte que :

  deux sommets connectés par une arête
  ne peuvent avoir la même couleur.

- Le nombre de couleurs (K) est égal au nombre de registres réels.
- S'il est possible de colorer le graphe avec K couleurs, alors le problème de l'allocation est résolu : on assigne à chaque registre fictif (chaque sommet) un registre réel (la couleur du sommet).
- S'il n'est pas possible de colorer le graphe avec K couleurs, alors il faudra utiliser la pile pour stocker certains registres fictifs.

Coloration par simplification

- La K-coloration de graphe est un problème NP-Complet, on ne connaît pas d'algorithme polynomial permettant de trouver une solution au problème.
- On utilise un algorithme d'approximation linéaire en quatre étapes :
  1. Construction du graphe d'interférence
  2. Simplification
  3. Débordement
  4. Sélection

Notations

- $G_A = (S_A, A_A)$ est le graphe d'analyse
  - les sommets S_A correspondent aux instructions du code du pré-assembleur
  - l'arc $(i_1, i_2) \in A_A$ indique que l'instruction i_2 peut suivre directement l'instruction i_1
  - $in(s)$, $s \in S_A$ sont les registres vivants en entrée du sommet s
  - $out(s)$, $s \in S_A$ sont les registres vivants en sortie du sommet s

Notations

- $G_A = (S_A, A_A)$ est le graphe d'analyse
  - les sommets S_A correspondent aux instructions du code du pré-assembleur
  - l'arc $(i_1, i_2) \in A_A$ indique que l'instruction i_2 peut suivre directement l'instruction i_1
  - $in(s),\ s \in S_A$ sont les registres vivants en entrée du sommet s
  - $out(s),\ s \in S_A$ sont les registres vivants en sortie du sommet s
- $G = (S, A)$ est le graphe d'interférence
  - les sommets S correspondent aux registres fictifs, il y en a R
  - $\mathcal{R} = \{1, \ldots R\}$ est l'ensemble des registres fictifs.
  - l'arête $(r_1, r_2) \in A$ indique que les registres fictifs r_1 et r_2 interfèrent (ils ne peuvent être assignés au même registre réel)

Notations

- $G_A = (S_A, A_A)$ est le graphe d'analyse
  - les sommets S_A correspondent aux instructions du code du pré-assembleur
  - l'arc $(i_1, i_2) \in A_A$ indique que l'instruction i_2 peut suivre directement l'instruction i_1
  - $in(s)$, $s \in S_A$ sont les registres vivants en entrée du sommet s
  - $out(s)$, $s \in S_A$ sont les registres vivants en sortie du sommet s
- $G = (S, A)$ est le graphe d'interférence
  - les sommets S correspondent aux registres fictifs, il y en a R
  - $\mathcal{R} = \{1, \dots R\}$ est l'ensemble des registres fictifs.
  - l'arête $(r_1, r_2) \in A$ indique que les registres fictifs r_1 et r_2 interfèrent (ils ne peuvent être assignés au même registre réel)
- K est le nombre de couleurs ou de registres réels.
  - $\mathcal{C} = \{1, \dots, K\}$ est l'ensemble des couleurs.
  - $\phi : \mathcal{R} \rightarrow \mathcal{C} \cup \{0\}$ est la fonction de pré-coloration, elle permet d'attribuer un registre réel à certains registres fictifs, avant l'allocation.

Construction du graphe d’interférence

Algorithm 1 Construction du graphe d’interférence $G = (S, A)$ à partir du graphe d’analyse $G_A = (S_A, A_A)$

$A \leftarrow \emptyset$
$S \leftarrow \{1, \ldots R\}$
for all $i \in S_A$ **do**
  for all $(r, r') \in in(s) \times in(s)$, $r \neq r'$ **do**
    $A \leftarrow A \cup (r, r')$
  end for
  for all $(r, r') \in out(s) \times out(s)$, $r \neq r'$ **do**
    $A \leftarrow A \cup (r, r')$
  end for
end for

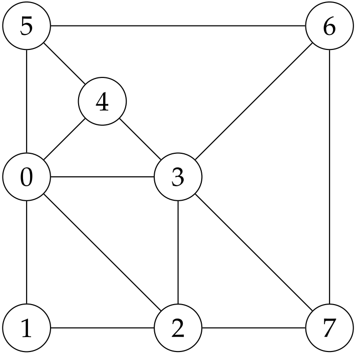
Simplification I

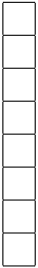
- Supposons que G possède un sommet s ayant moins de K voisins.
- Soit G' le graphe $G - \{s\}$: le graphe obtenu à partir de G en retirant le sommet s.
- S'il est possible de colorer G' avec K couleurs, alors, il est possible de colorer G car les voisins de s auront au plus $K - 1$ couleurs différentes, il reste donc une couleur disponible pour s.

Simplification II

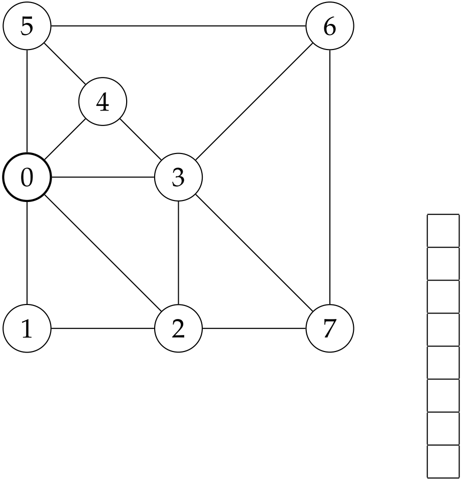
- Algorithme récursif : on enlève un sommet, on l'empile et on relance récursivement l'algorithme sur le nouveau graphe.
- **Idée clef** : l'élimination d'un sommet diminue le nombre de voisins d'autres sommets et augmente par conséquent les chances d'obtenir des sommets ayant moins de K voisins.
- Si on arrive ainsi à un graphe possédant un seul sommet, alors il est possible de colorer le graphe avec K couleurs.

Simplification ($K = 4$)



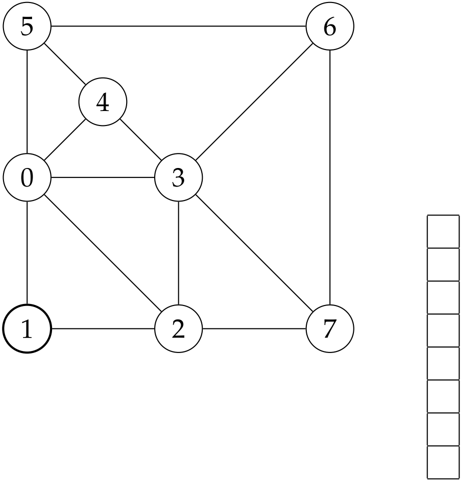


Simplification ($K = 4$)



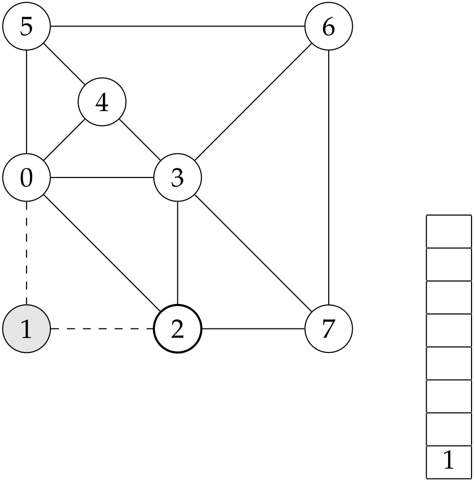
Le sommet 0 possède cinq voisins, on ne peut pas l’enlever

Simplification ($K = 4$)



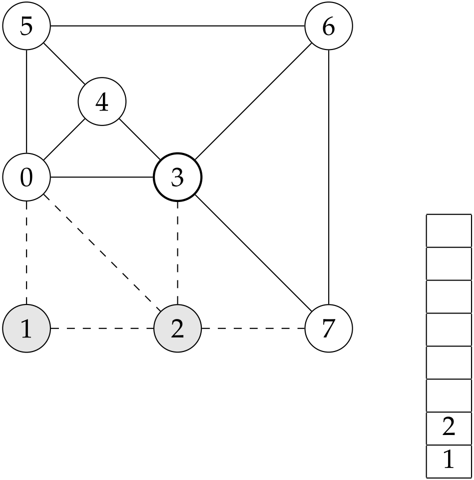
Le sommet 1 possède deux voisins, on peut l'enlever

Simplification ($K = 4$)



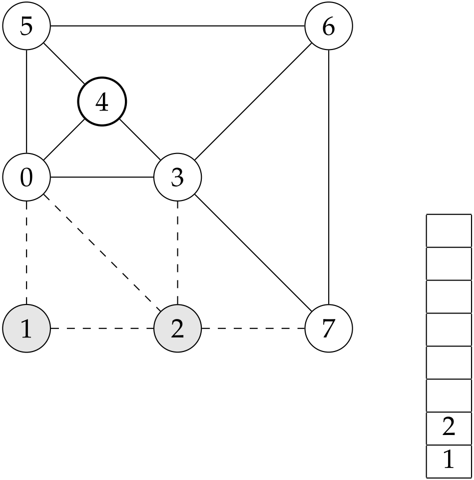
Le sommet 2 possède trois voisins, on peut l'enlever

Simplification ($K = 4$)



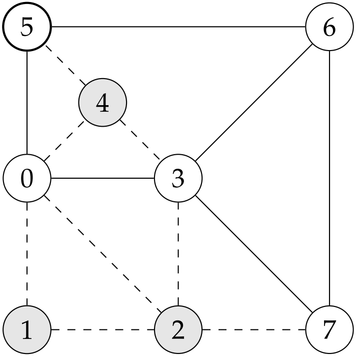
Le sommet 3 possède quatre voisins, on ne peut pas l'enlever

Simplification ($K = 4$)



Le sommet 4 possède trois voisins, on peut l'enlever

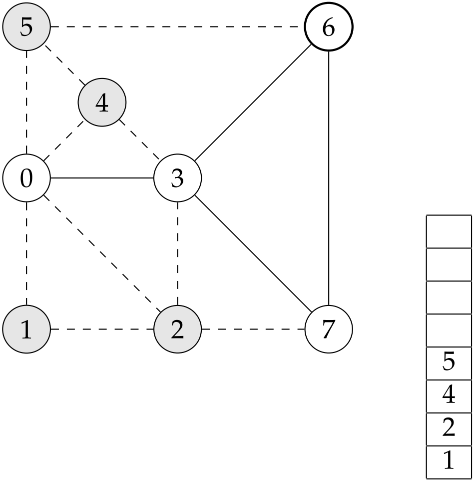
Simplification ($K = 4$)



4
2
1

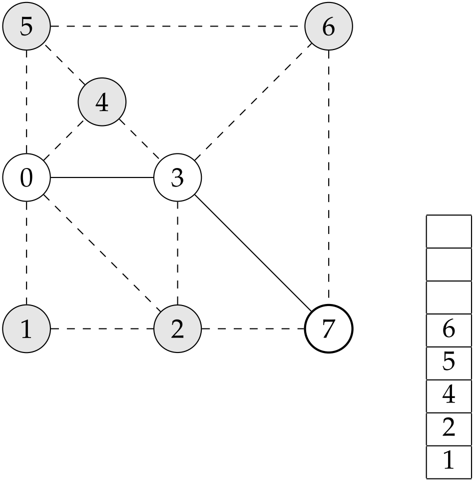
Le sommet 5 possède deux voisins, on peut l'enlever

Simplification ($K = 4$)



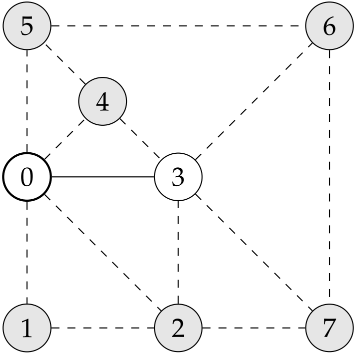
Le sommet 6 possède deux voisins, on peut l'enlever

Simplification ($K = 4$)



Le sommet 7 possède un voisin, on peut l'enlever

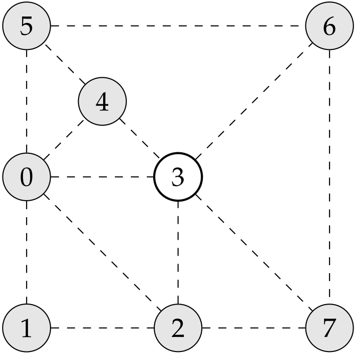
Simplification ($K = 4$)



7
6
5
4
2
1

Le sommet 0 possède un voisin, on peut l'enlever

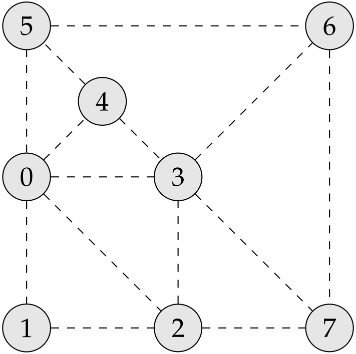
Simplification ($K = 4$)



0
7
6
5
4
2
1

Le sommet 3 n’a pas de voisins, on peut l’enlever

Simplification ($K = 4$)



3
0
7
6
5
4
2
1

Le graphe peut être coloré avec 4 couleurs !

Algorithme de simplification

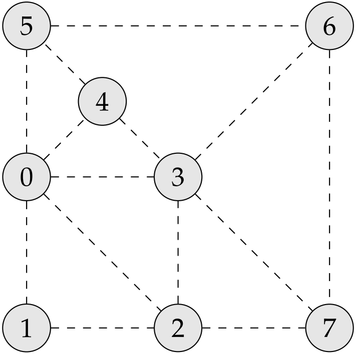
Algorithm 2 Simplification du graphe $G = (S, A)$

```
pile ← ∅
modif ← vrai
while taille(pile) ≠ R et modif = vrai do
    modif ← faux
    for all s ∈ S do
        if nb_voisins(s) < K then
            empile(s)
            S ← S − {s}
            modif ← vrai
        end if
    end for
end while
```

Sélection

- Si à l'issue de l'algorithme de simplification, la taille de la pile vaut R (le nombre de registres fictifs) alors on est sûr qu'une solution existe.
- Dans ce cas, il suffit de dépiler les sommets et de leur assigner une couleur.
- On est sûr qu'une couleur sera disponible pour tout sommet dépilé.

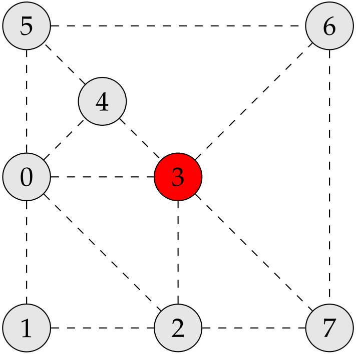
Sélection ($K = 4$)



3
0
7
6
5
4
2
1

On dépile 3 et on lui donne une couleur

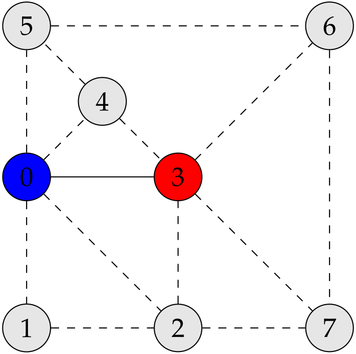
Sélection ($K = 4$)



0
7
6
5
4
2
1

On dépile 0

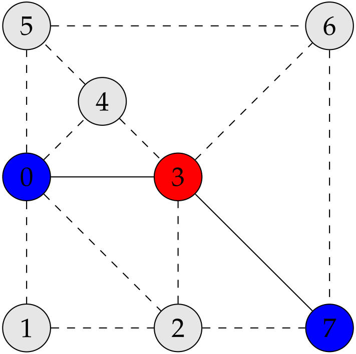
Sélection ($K = 4$)



7
6
5
4
2
1

On dépile 7

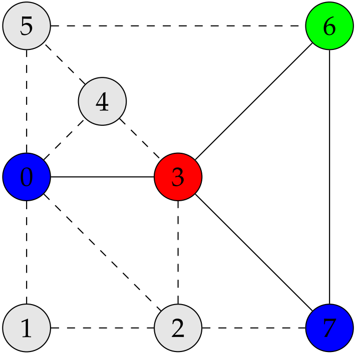
Sélection ($K = 4$)



6
5
4
2
1

On dépile 6

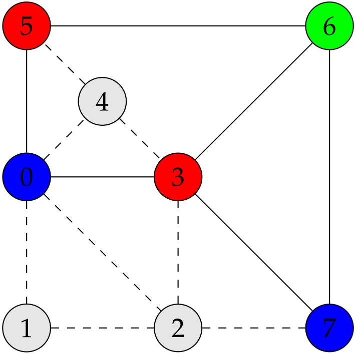
Sélection ($K = 4$)

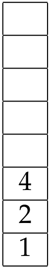


5
4
2
1

On dépile 5

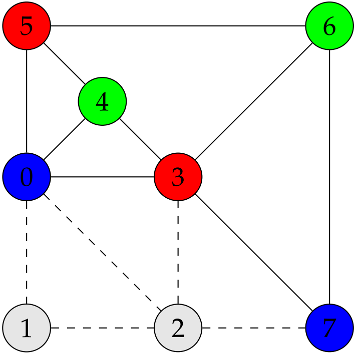
Sélection ($K = 4$)

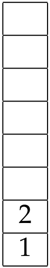




On dépile 4

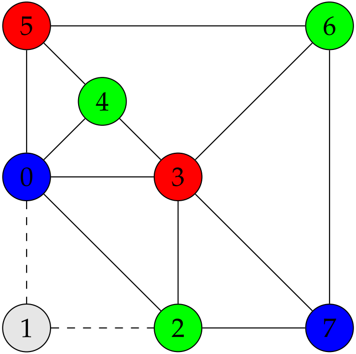
Sélection ($K = 4$)

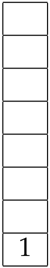




On dépile 2

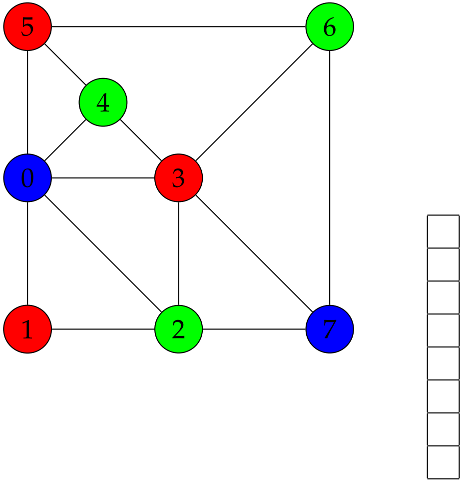
Sélection ($K = 4$)





On dépile 1

Sélection ($K = 4$)



C'est fini ! trois couleurs ont suffi

Algorithme de sélection

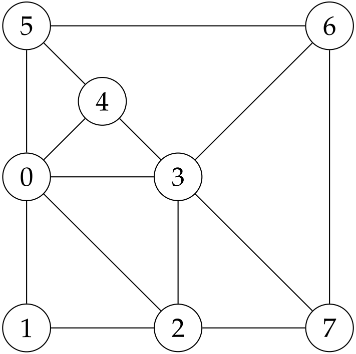
Algorithm 3 Sélection d'une couleur pour les sommets du graphe $G = (S, A)$
avec la pile produite par l'algorithme de Simplification

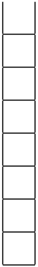
$\forall s \in S$ $couleur[s] \leftarrow 0$
while taille(pile) $\neq 0$ **do**
    $s \leftarrow$ depile
    $C \leftarrow couleurs\_voisins(s)$
    if $|C| \neq K$ **then**
        $couleur[s] \leftarrow choisis\_couleur(\mathcal{C} - C)$
    end if
end while

Débordement

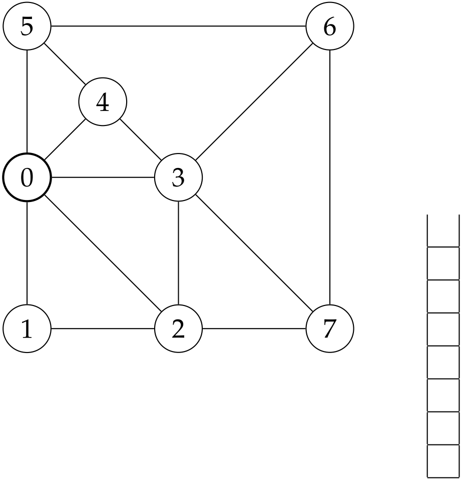
- Ca ne marche pas toujours !
- Il est possible qu'à l'issue de l'algorithme de simplification, il n'y ait plus de sommets ayant moins que K voisins.
- On empile quand même un sommet ayant K voisins ou plus, en espérant que parmi ses voisins, plusieurs aient la même couleur.
- Illustration sur le même exemple, mais avec $K = 3$.

Simplification ($K = 3$)



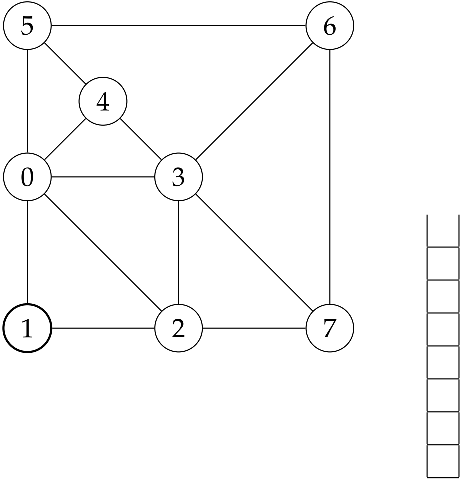


Simplification ($K = 3$)



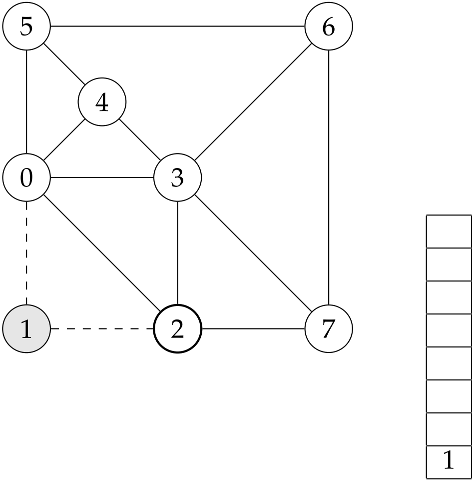
Le sommet 0 possède cinq voisins, on ne peut pas l'enlever

Simplification ($K = 3$)



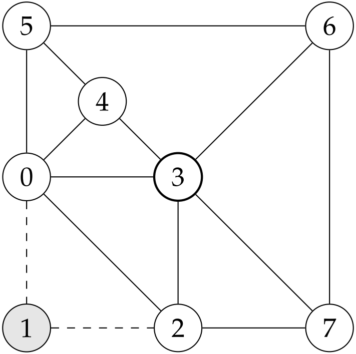
Le sommet 1 possède deux voisins, on peut l’enlever

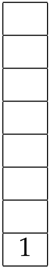
Simplification ($K = 3$)



Le sommet 2 possède trois voisins, on ne peut pas l'enlever

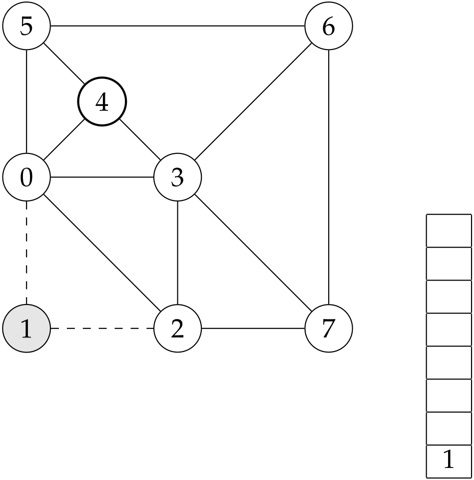
Simplification ($K = 3$)





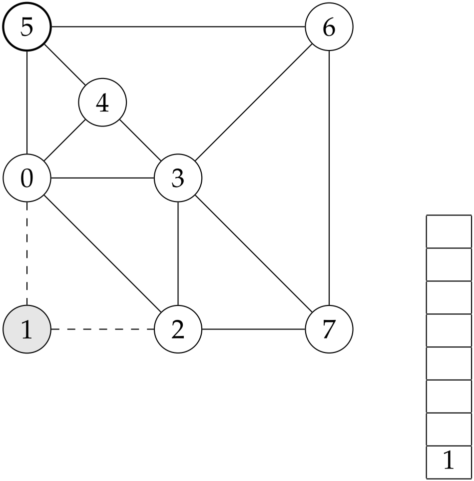
Le sommet 3 possède cinq voisins, on ne peut pas l'enlever

Simplification ($K = 3$)



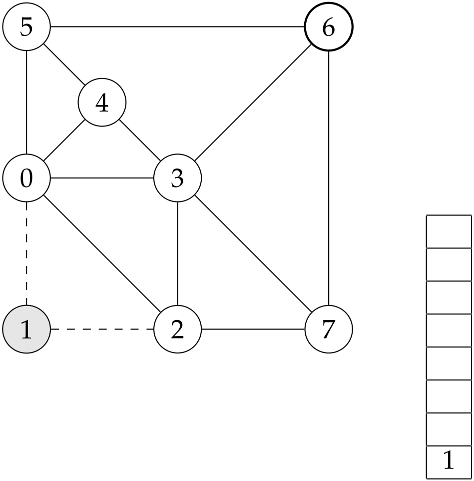
Le sommet 4 possède trois voisins, on ne peut pas l’enlever

Simplification ($K = 3$)



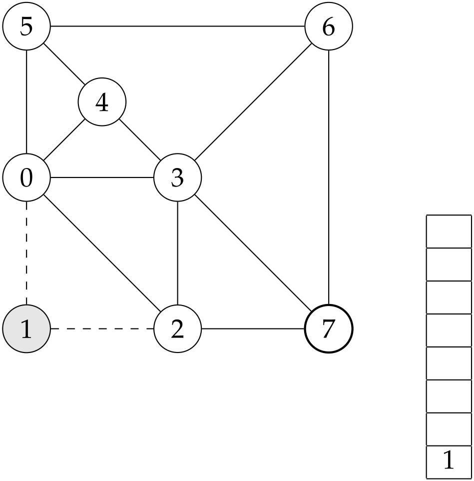
Le sommet 5 possède trois voisins, on ne peut pas l'enlever

Simplification ($K = 3$)



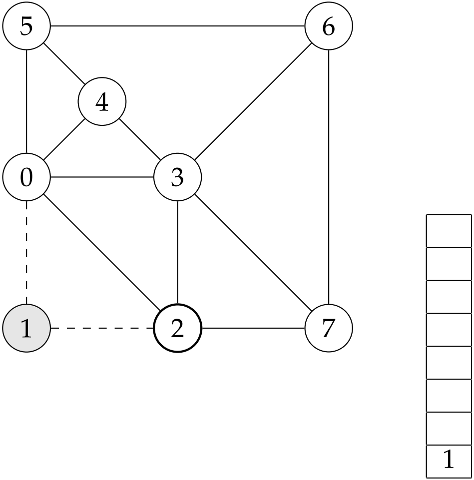
Le sommet 6 possède trois voisins, on ne peut pas l'enlever

Simplification ($K = 3$)



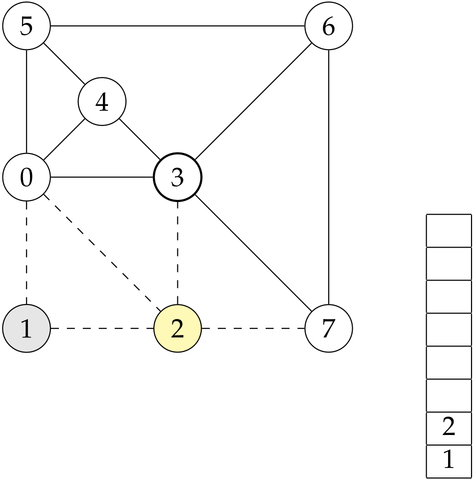
Le sommet 7 possède trois voisins, on ne peut pas l'enlever

Simplification ($K = 3$)



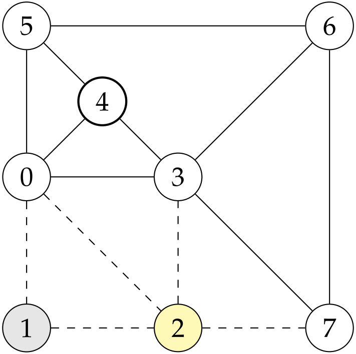
On est bloqué, on décide d'empiler 2 et on le marque comme débordement possible.

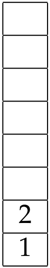
Simplification ($K = 3$)



Le sommet 3 possède quatre voisins, on ne peut pas l’enlever

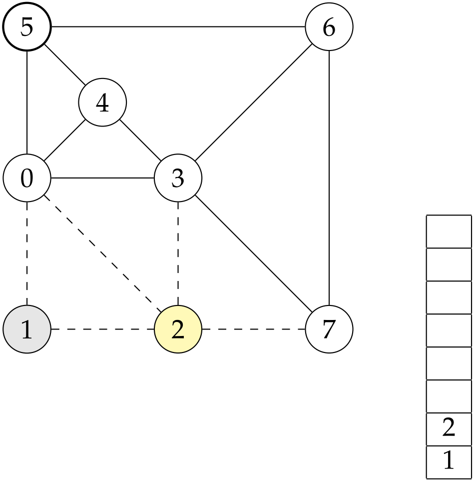
Simplification ($K = 3$)





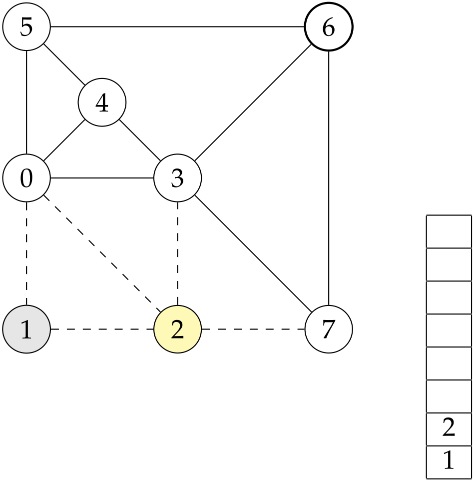
Le sommet 4 possède trois voisins, on ne peut pas l'enlever

Simplification ($K = 3$)



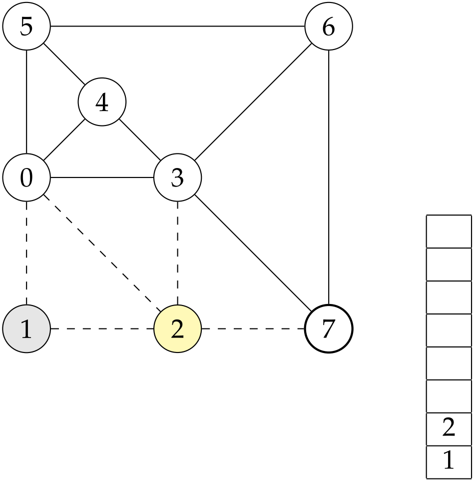
Le sommet 5 possède trois voisins, on ne peut pas l'enlever

Simplification ($K = 3$)



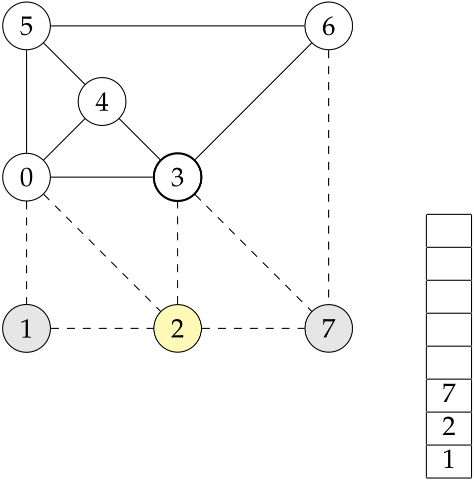
Le sommet 6 possède trois voisins, on ne peut pas l’enlever

Simplification ($K = 3$)



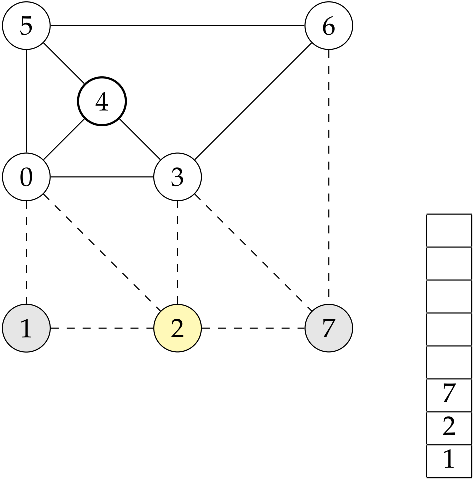
Le sommet 7 possède deux voisins, on peut l'enlever

Simplification ($K = 3$)



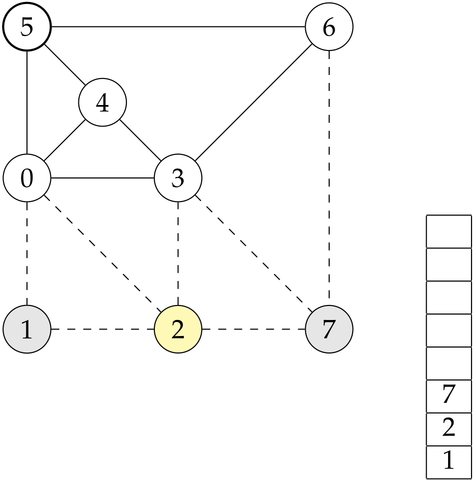
Le sommet 3 possède trois voisins, on ne peut pas l'enlever

Simplification ($K = 3$)



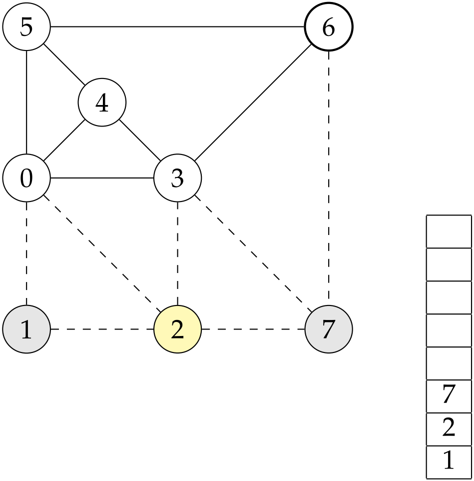
Le sommet 4 possède trois voisins, on ne peut pas l'enlever

Simplification ($K = 3$)



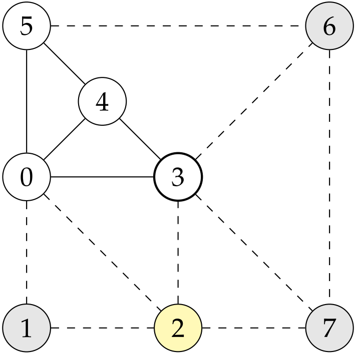
Le sommet 5 possède trois voisins, on ne peut pas l'enlever

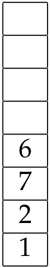
Simplification ($K = 3$)



Le sommet 6 possède deux voisins, on peut l'enlever

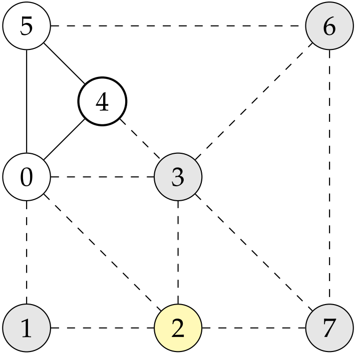
Simplification ($K = 3$)





Le sommet 3 possède deux voisins, on peut l'enlever

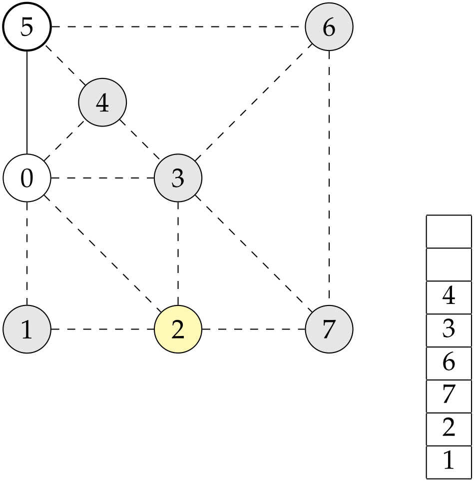
Simplification ($K = 3$)



3
6
7
2
1

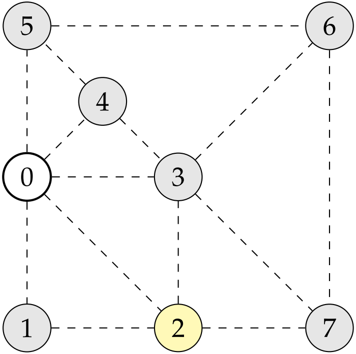
Le sommet 4 possède deux voisins, on peut l'enlever

Simplification ($K = 3$)



Le sommet 5 possède un voisin, on peut l'enlever

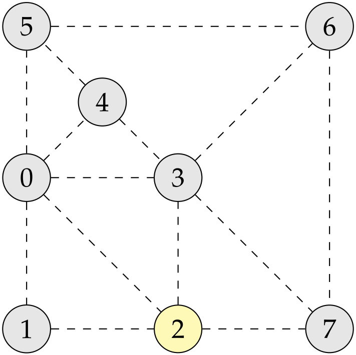
Simplification ($K = 3$)



5
4
3
6
7
2
1

Le sommet 0 n'a pas de voisin, on peut l'enlever

Simplification ($K = 3$)

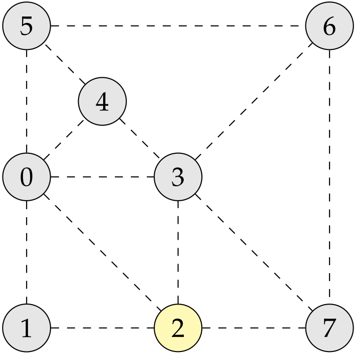


0
5
4
3
6
7
2
1

On a fini d'empiler, mais on n'a pas la garantie que l'on pourra attribuer une couleur à chaque sommet !

On espère que les voisins de 2 ($\{0, 1, 3, 7\}$) n'aient pas besoin de 3 couleurs.

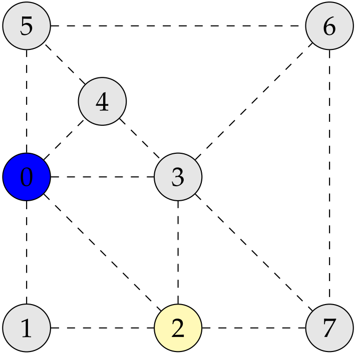
Exemple $K = 3$



0
5
4
3
6
7
2
1

On dépile 0

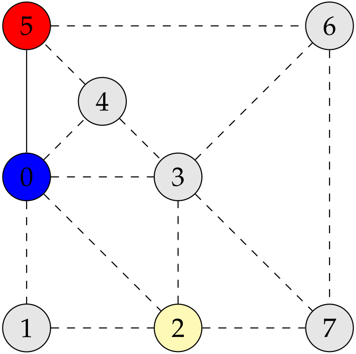
Exemple $K = 3$

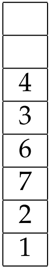


5
4
3
6
7
2
1

On dépile 5

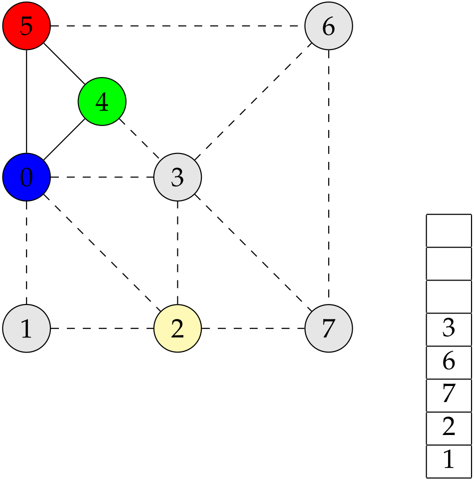
Exemple $K = 3$





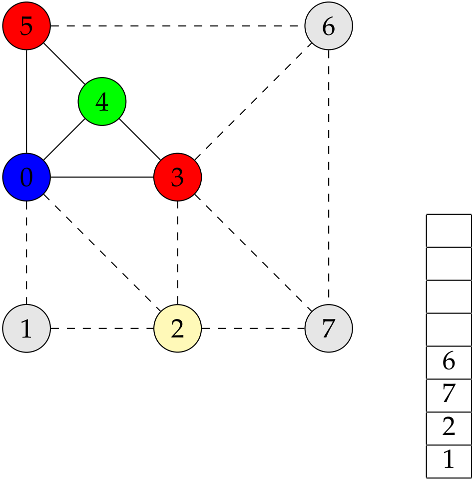
On dépile 4

Exemple $K = 3$



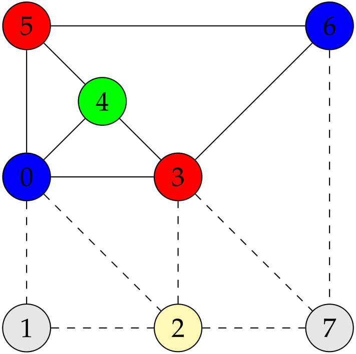
On dépile 3

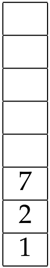
Exemple $K = 3$



On dépile 6

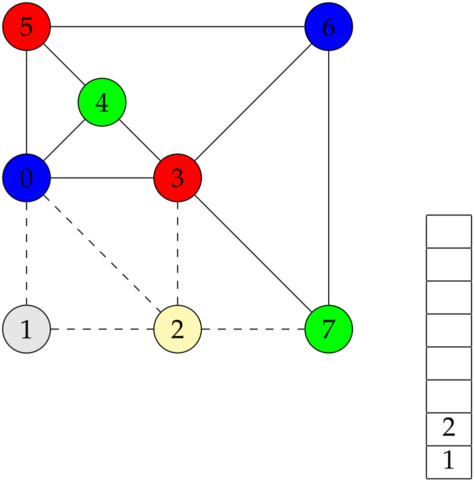
Exemple $K = 3$





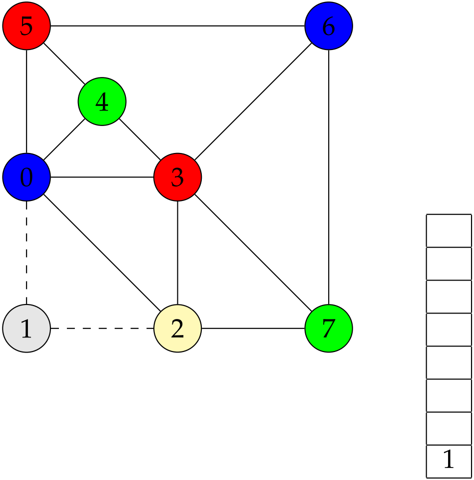
On dépile 7

Exemple $K = 3$



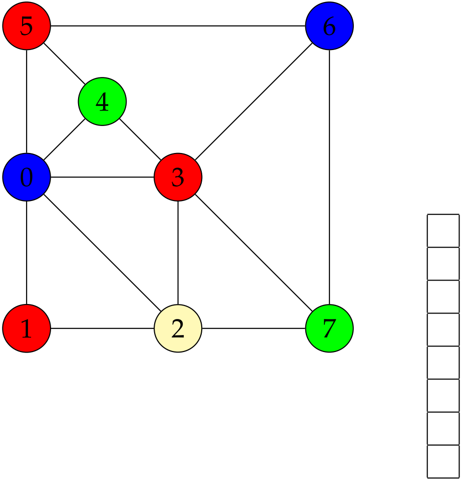
On dépile 2, mais on ne peut pas lui attribuer de couleur !
Les trois couleurs sont déjà utilisées par ses voisins.

Exemple $K = 3$



On dépile 1

Exemple $K = 3$



Une solution existe, mais l'algorithme ne l'a pas trouvé !

Algorithme de débordement

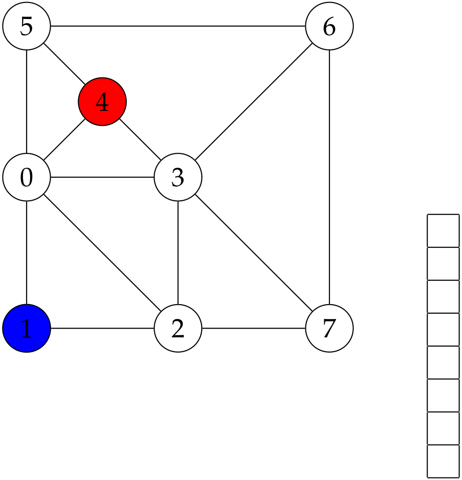
Algorithm 4 Débordement

$deborde \leftarrow \emptyset$
while taille(pile) $\neq R$ **do**
  $s \leftarrow choisis\_sommet$
  $empile(s)$
  $S \leftarrow S - \{s\}$
  $deborde = deborde \cup \{s\}$
  Simplifie
end while

Sommets pré-colorés

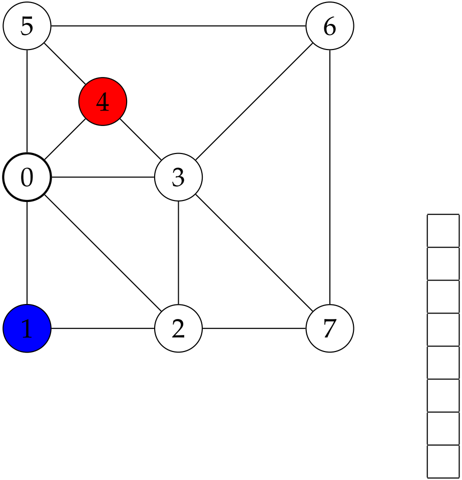
- Le compilateur peut pré-colorer certains registre fictifs : leur attribuer directement un registre réel.
- On définit une fonction $\phi : S \rightarrow \mathcal{C} \cup \{0\}$ qui associe à tout sommet du graphe (S, A) une couleur choisie dans l'ensemble $\mathcal{C} = \{1 \ldots K\}$
- Un sommet s n'est pas pré-coloré si $\phi(s) = 0$
- On adapte les deux algorithmes de simplification et de sélection de manière à prendre en compte la pré-coloration.
- **Idée clef** : un sommet pré-coloré ne peut être mis dans la pile, il a déjà une couleur.

Simplification ($K = 4$)



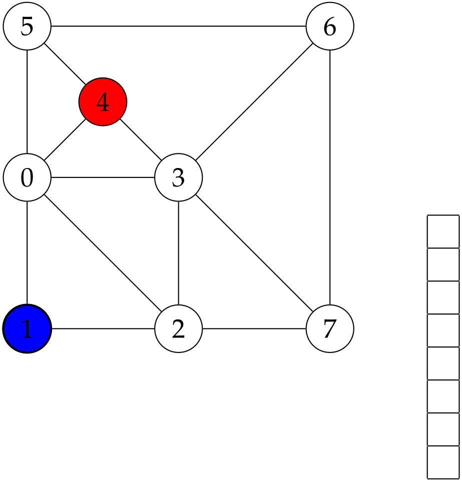
Les sommets 1 et 4 sont pré-colorés

Simplification ($K = 4$)



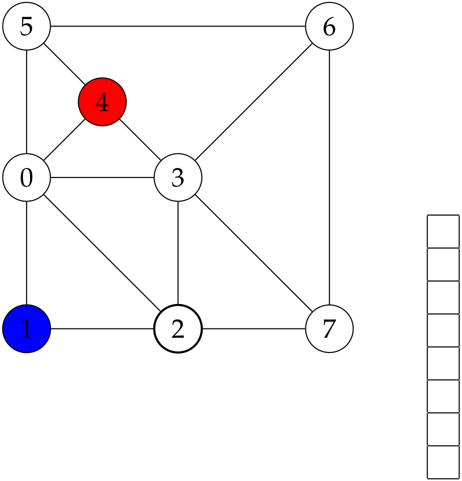
Le sommet 0 possède 5 voisins, on ne peut pas l'enlever

Simplification ($K = 4$)



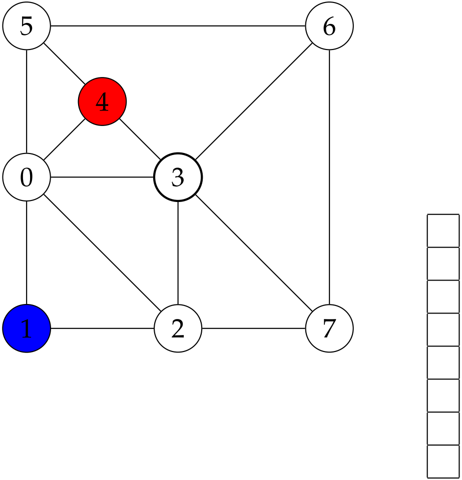
Le sommet 1 est déjà coloré, on ne peut pas l'enlever

Simplification ($K = 4$)



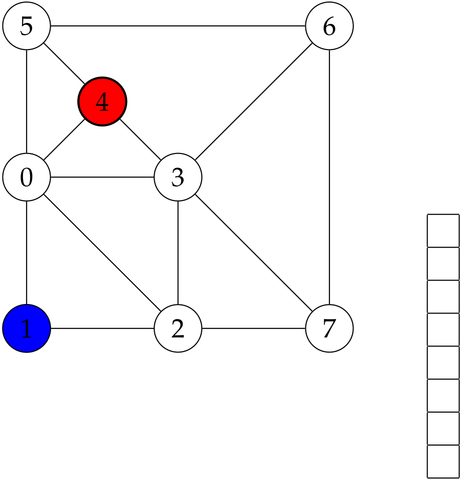
Le sommet 2 possède 4 voisins, on ne peut pas l'enlever

Simplification ($K = 4$)



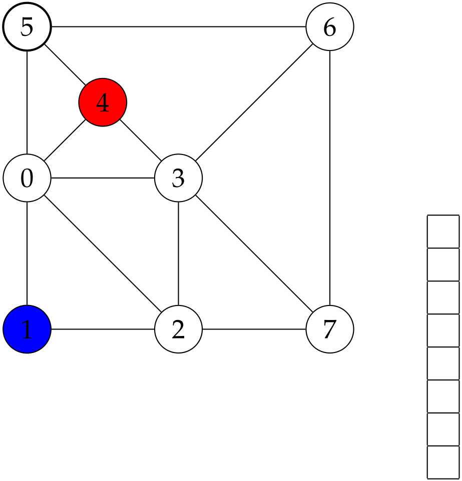
Le sommet 3 possède 5 voisins, on ne peut pas l'enlever

Simplification ($K = 4$)



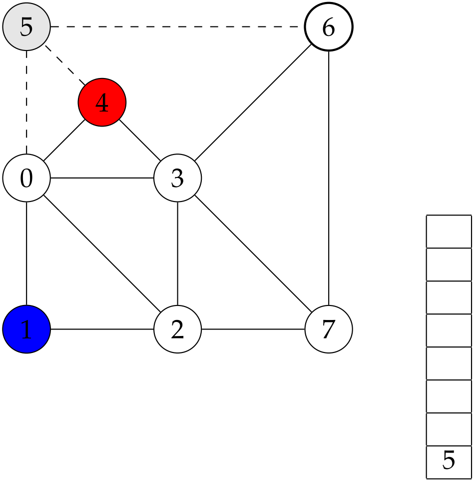
Le sommet 4 est déjà coloré, on ne peut pas l'enlever

Simplification ($K = 4$)



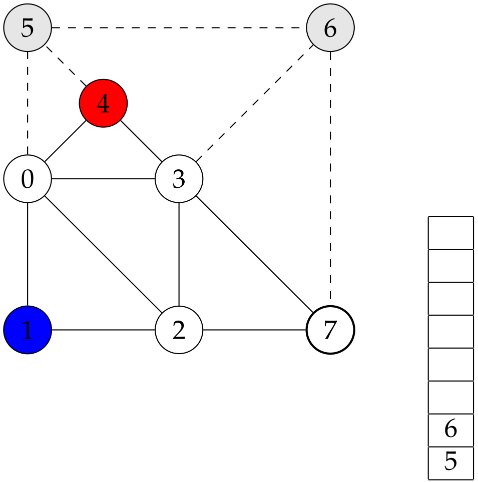
Le sommet 5 possède trois voisins, on l'enlève

Simplification ($K = 4$)



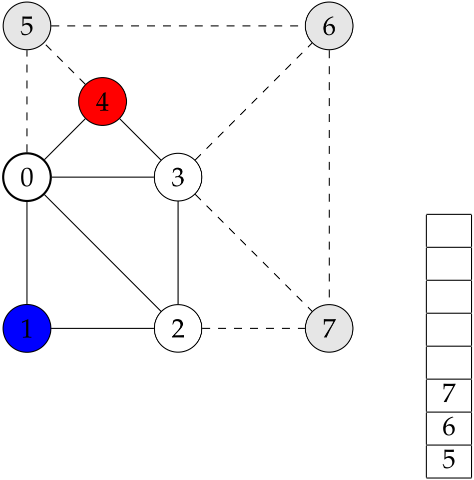
Le sommet 6 possède deux voisins, on l'enlève

Simplification ($K = 4$)



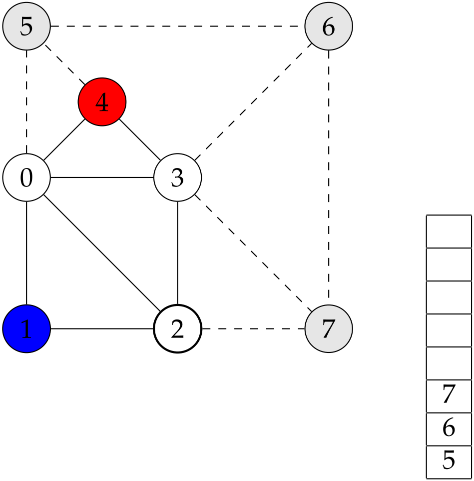
Le sommet 7 possède deux voisins, on l'enlève

Simplification ($K = 4$)



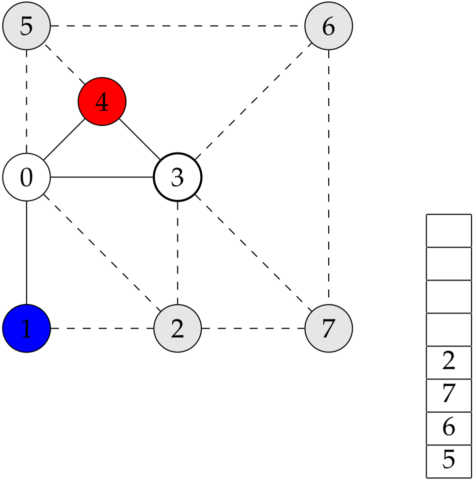
Le sommet 0 possède quatre voisins, on ne peut pas l'enlever

Simplification ($K = 4$)



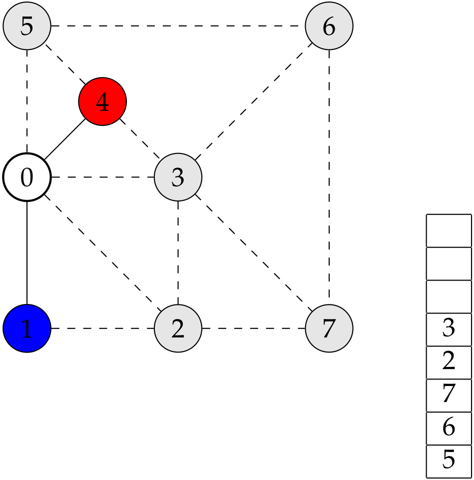
Le sommet 2 possède trois voisins, on l'enlève

Simplification ($K = 4$)



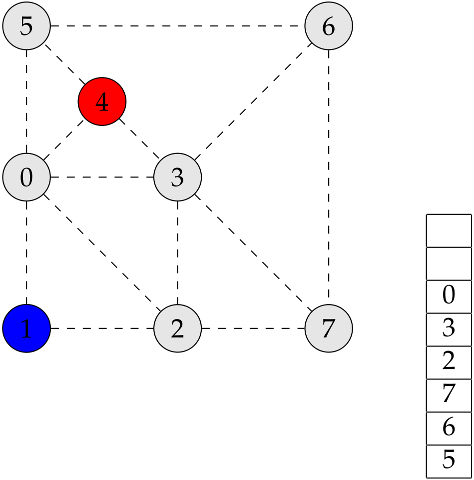
Le sommet 3 possède deux voisins, on l'enlève

Simplification ($K = 4$)



Le sommet 0 possède deux voisins, on l'enlève

Simplification ($K = 4$)



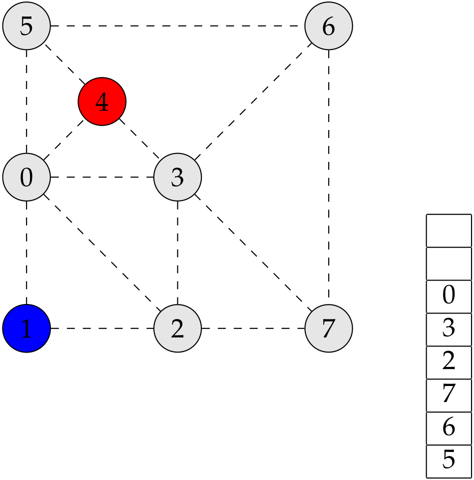
Tous les sommets non pré-colorés sont dans la pile

Algorithme de simplification avec pré-coloration

Algorithm 5 Simplification du graphe $G = (S, A)$
ϕ est la fonction de pré-coloration

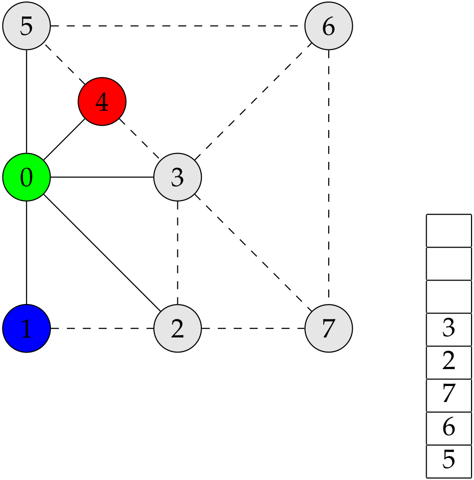
```
pile ← ∅
N ← R− nombre de sommets pré-colorés
modif ← vrai
while taille(pile) ≠ N et modif = vrai do
    modif ← faux
    for all s ∈ S do
        if nb_voisins(s) < K et φ(s) = 0 then
            empile(s)
            S ← S − {s}
            modif ← vrai
        end if
    end for
end while
```

Sélection ($K = 4$)



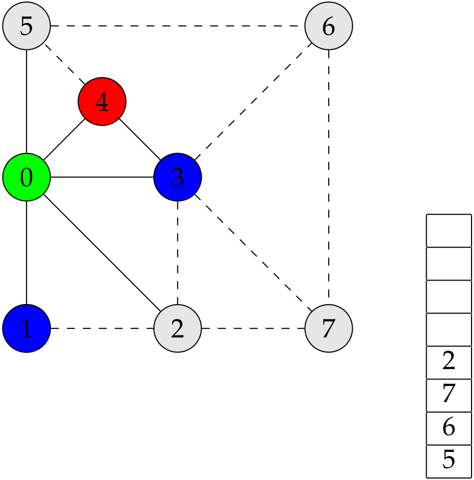
On dépile le sommet 0, on le colore en vert

Sélection ($K = 4$)



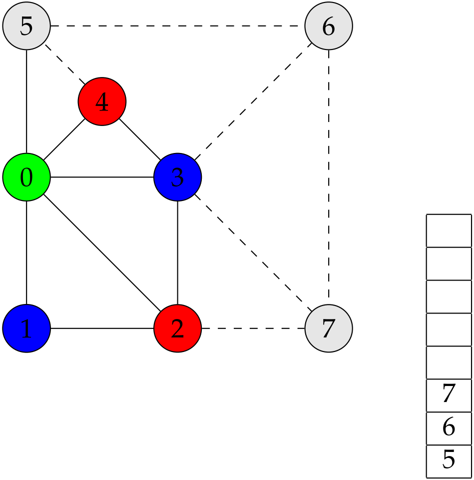
On dépile le sommet 3, on le colore en bleu

Sélection ($K = 4$)



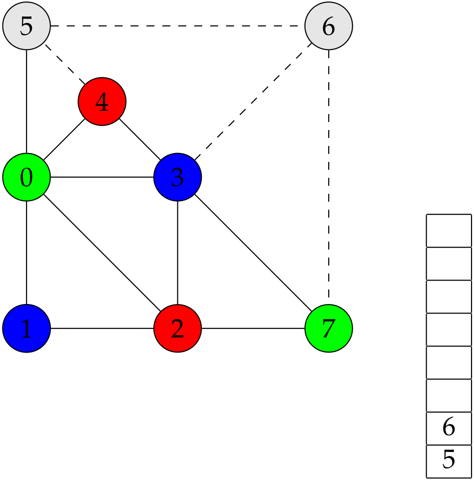
On dépile le sommet 2, on le colore en rouge

Sélection ($K = 4$)



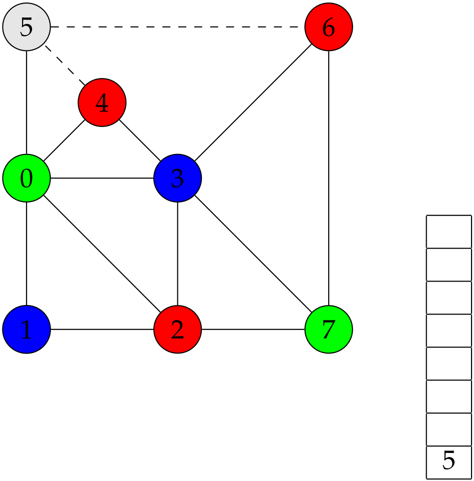
On dépile le sommet 7, on le colore en vert

Sélection ($K = 4$)



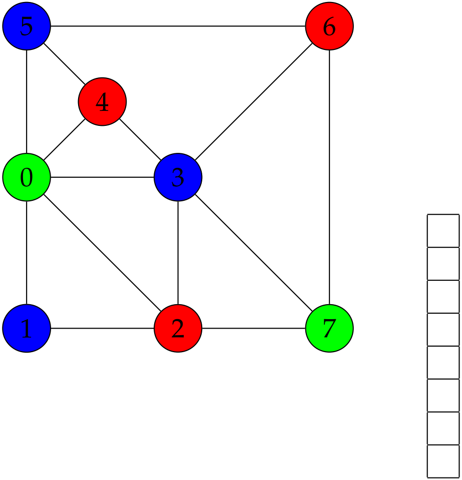
On dépile le sommet 6, on le colore en rouge

Sélection ($K = 4$)



On dépile le sommet 5, on le colore en bleu

Sélection ($K = 4$)



C'est fini !

Algorithme de sélection avec pré-coloration des sommets

Algorithm 6 Sélection d'une couleur pour les sommets du graphe $G = (S, A)$
avec la pile produite par l'algorithme de Simplification

$\forall s \in S$ $couleur[s] \leftarrow \phi(s)$
while taille(pile) $\neq 0$ **do**
    $s \leftarrow$ depile
    $C \leftarrow couleurs\_voisins(s)$
    if $|C| \neq K$ **then**
        $couleur[s] \leftarrow choisis\_couleur(\mathcal{C} - C)$
    end if
end while

Algorithme général de coloration

Algorithm 7 Coloration des sommets du graphe graphe $G = (S, A)$

1: $Construction$ (algorithme 1)
2: $Simplification$ (algorithme 5)
3: $Débordement$ (algorithme 4)
4: $Sélection$ (algorithme 6)