

Election dans les anneaux avec identités

Jérémie Albert

Université Bordeaux 1

LaBRI

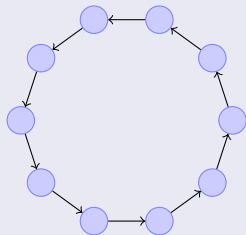
Sommaire

- 1 Definitions
 - Election
 - Anneau
 - Identité
- 2 Algorithme de LeLann (1977)
 - Algorithme
 - Exemple
 - Complexité
- 3 Algorithme de Chang et Roberts (1979)
 - Algorithme
 - Exemple 1
 - Exemple 2
 - Complexité
- 4 Algorithme de Franklin (1982)
 - Algorithme
 - Un premier exemple
 - Cas plus général
 - Complexité

Soit G un graphe connexe. En algorithmique distribuée, l'élection est le processus qui permet de distinguer un unique sommet u tel que $u \in G$ parmi l'ensemble des sommets du graphe G .

Anneau unidirectionnel

Exemple

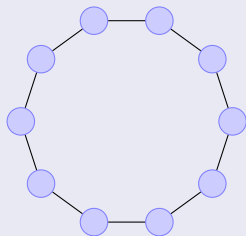


Actions possibles

- reception depuis le sommet précédent
- envoi au sommet suivant

Anneau bidirectionnel

Exemple



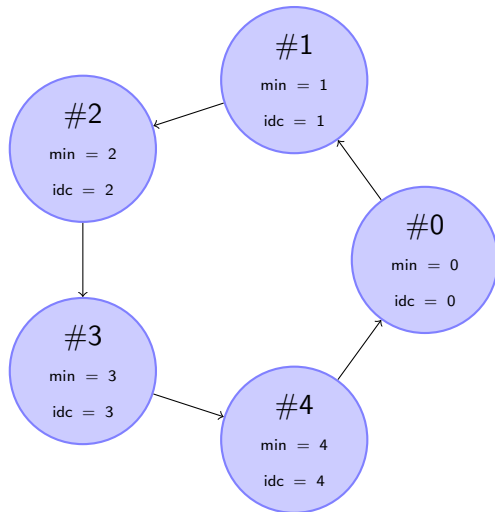
Actions possibles

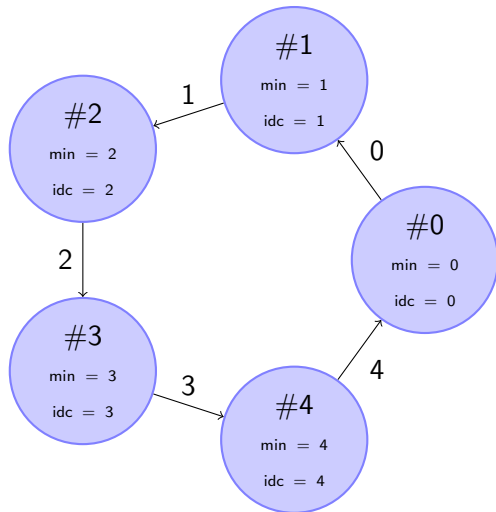
- reception depuis le sommet précédent
- envoi au sommet suivant
- reception depuis le sommet suivant
- envoi au sommet précédent

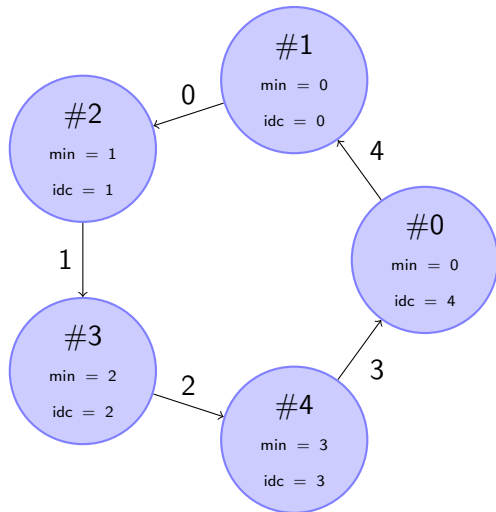
Propriétés

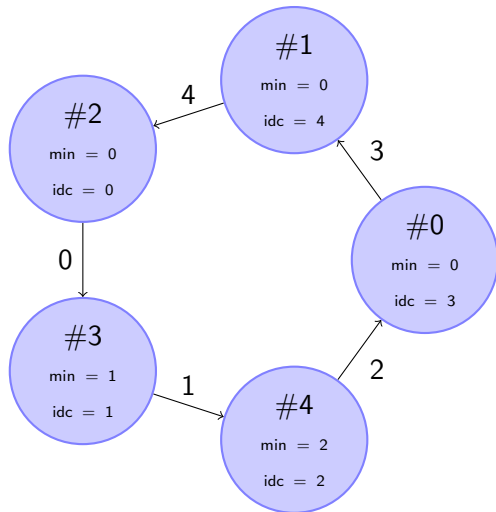
- unique
- il existe une relation d'ordre sur l'ensemble E des identités (reflexive, antisymétrique et transitive)

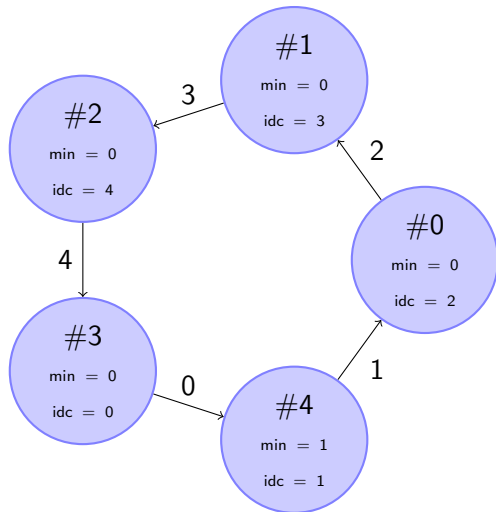
```
debut
  min := mon id ;
  id courant := mon id ;
  repeter
    envoyer <id courant> à suivant ;
    recevoir <id courant> de précédent ;
    si id courant < min alors
      min := id courant ;
  jusqu'à id courant == mon id;
  si min == mon id alors
    statut := élu ;
  sinon
    statut := non-élu ;
fin
```

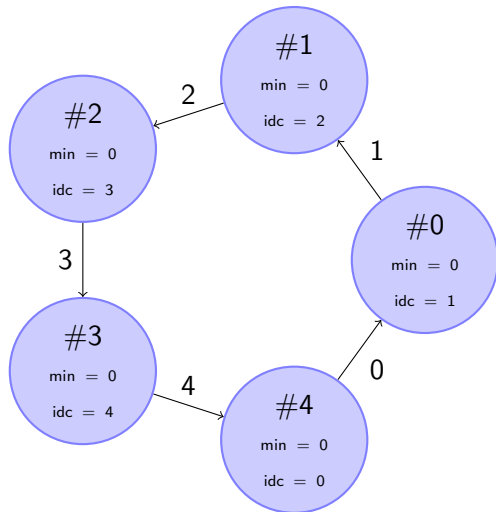


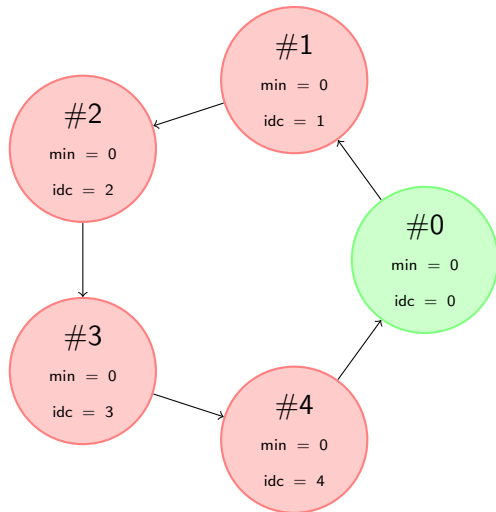








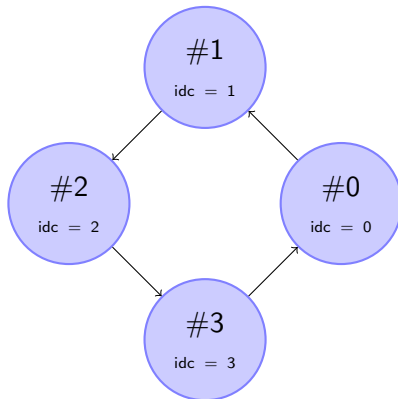




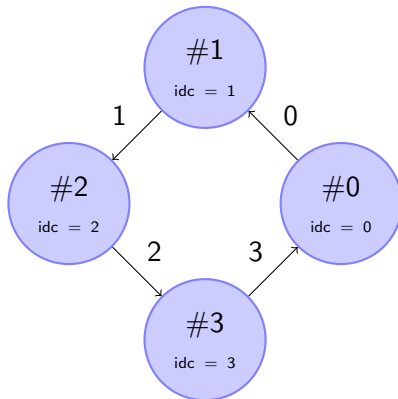
	nombre de messages	temps
au mieux	$O(n^2)$	$O(n)$
au pire	$O(n^2)$	$O(n)$
en moyenne	$O(n^2)$	$O(n)$

```
debut
  envoyer <mon id> à suivant ;
  repeter
    recevoir m de precedent ;
    si m < mon id alors
      statut := non-elu ;
      envoyer m à suivant ;
  jusqu'à id courant == mon id ou m == <fin>;
  envoyer <fin> à suivant ;
  si id courant == mon id alors
    statut := élu ;
    recevoir <fin> de précédent ;
fin
```

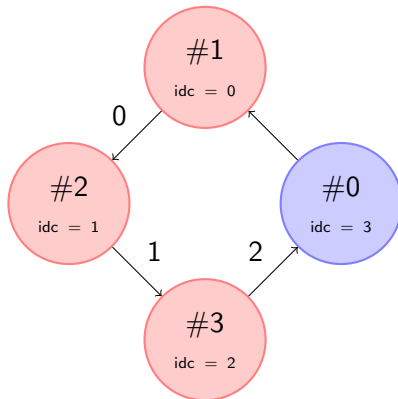
Exemple 1



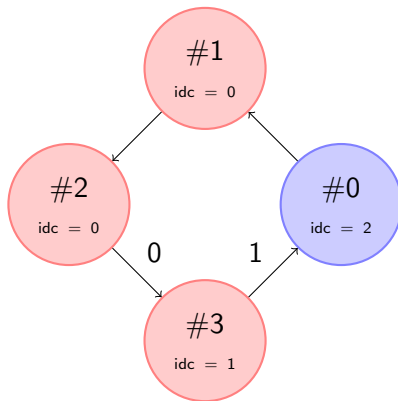
Exemple 1



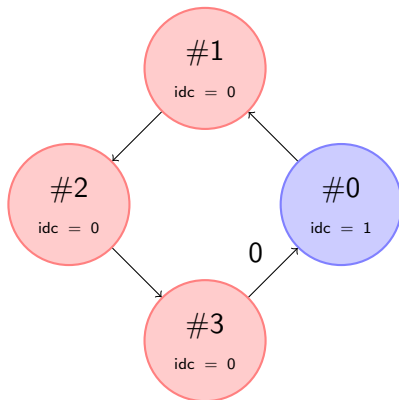
Exemple 1



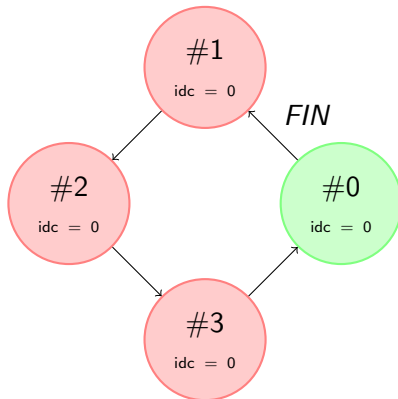
Exemple 1



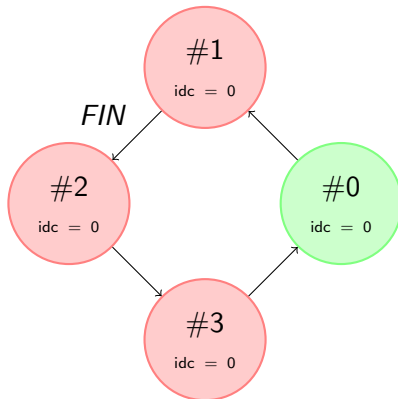
Exemple 1



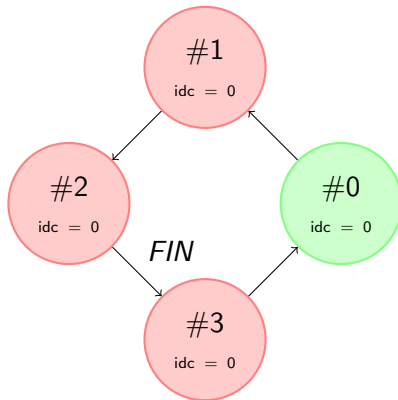
Exemple 1



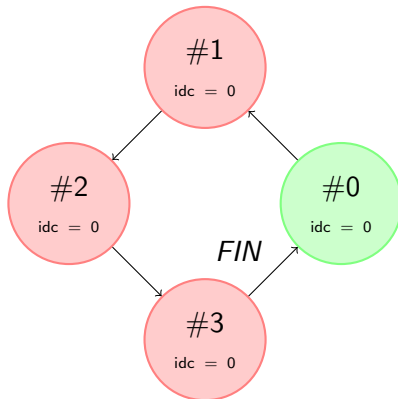
Exemple 1



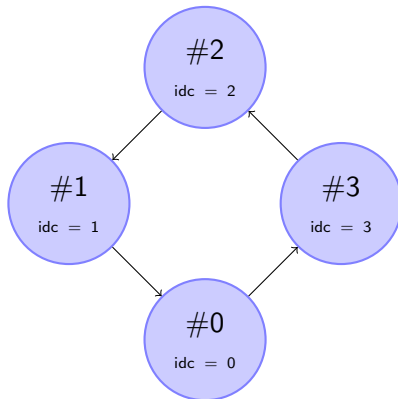
Exemple 1



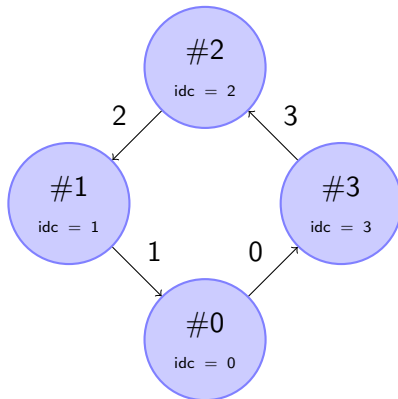
Exemple 1



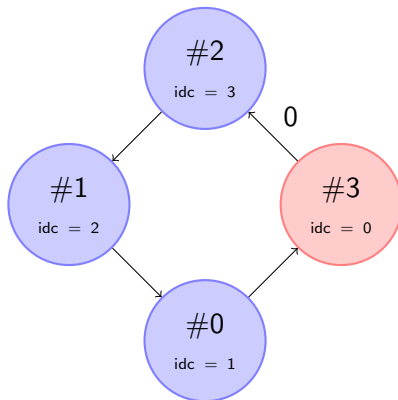
Exemple 2



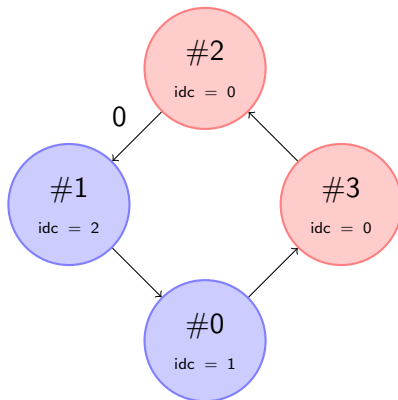
Exemple 2



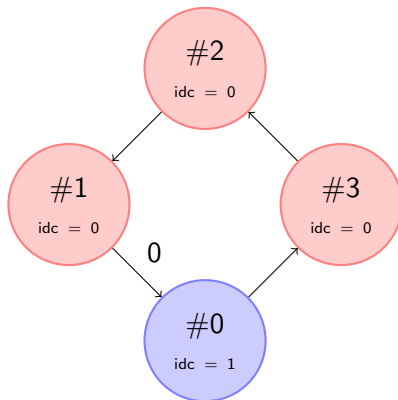
Exemple 2



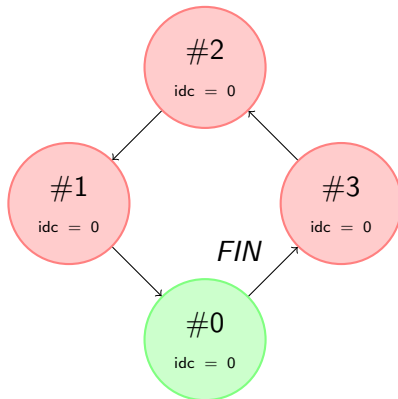
Exemple 2



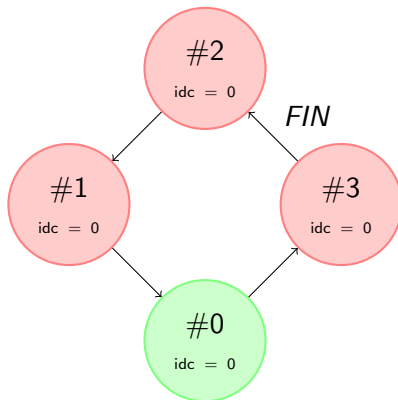
Exemple 2



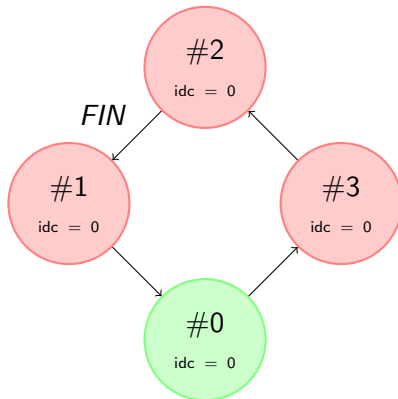
Exemple 2



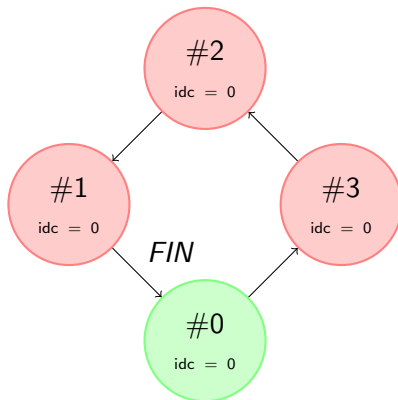
Exemple 2



Exemple 2



Exemple 2



	nombre de messages	temps
au mieux	$O(n)$	$O(n)$
au pire	$O(n^2)$	$O(n)$
en moyenne	$O(n * \log(n))$	$O(n)$

```
debut
```

```
  statut := actif ;
```

```
  fini := faux ;
```

```
  tant que "pas" fini faire
```

```
    si statut == actif alors
```

```
      envoyer <mon id> à suivant ;
```

```
      recevoir <id prec> de precedent ;
```

```
      envoyer <mon id> à precedent ;
```

```
      recevoir <id suiv> de suivant ;
```

```
      si id prec < mon id ou id suiv < mon id alors
```

```
        statut := non-elu ;
```

```
      si id prec == mon id et id suiv == mon id alors
```

```
        statut := élu ;
```

```
        fini := vrai ;
```

```
  sinon
```

```
    recevoir <id prec> de precedent ;
```

```
    envoyer <id prec> à suivant ;
```

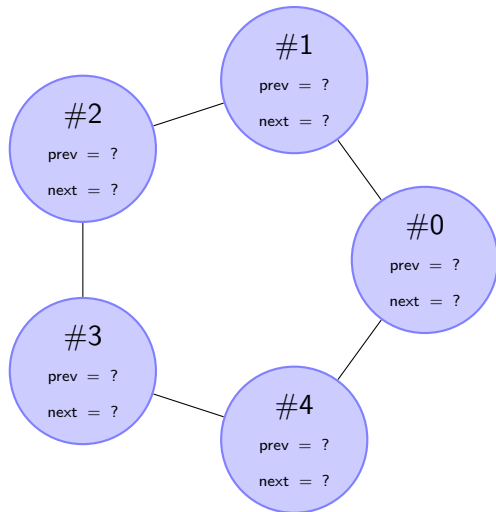
```
    recevoir <id suiv> de suivant ;
```

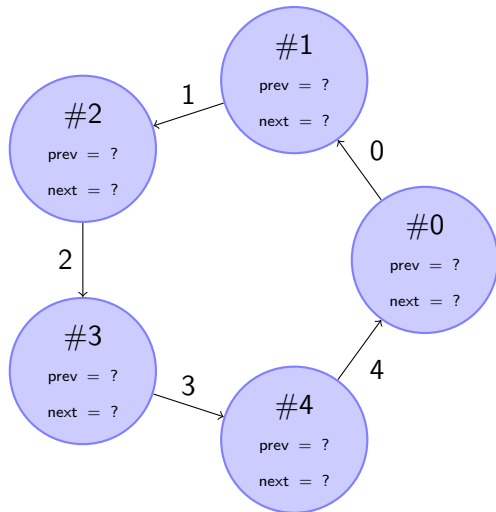
```
    envoyer <id suiv> à précédent ;
```

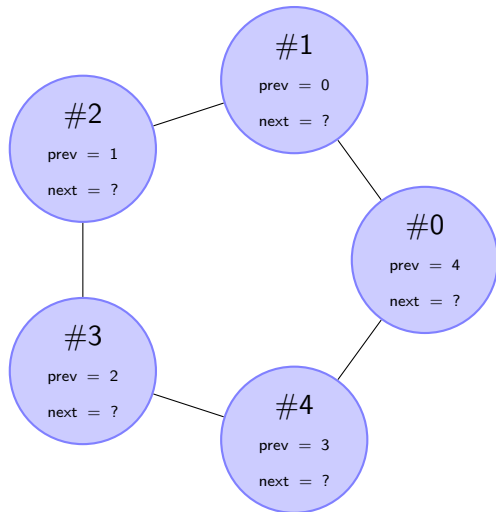
```
    si id prec == id suiv alors
```

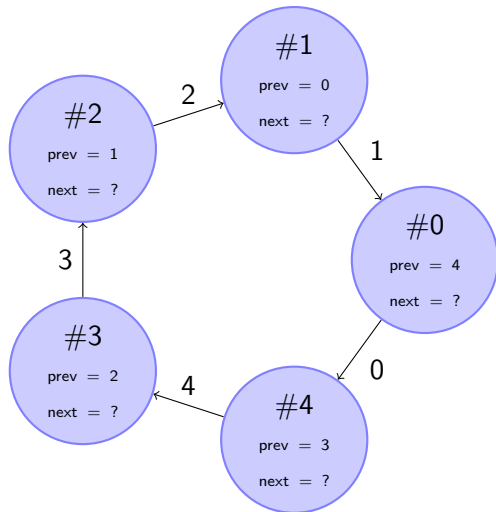
```
      fini := vrai ;
```

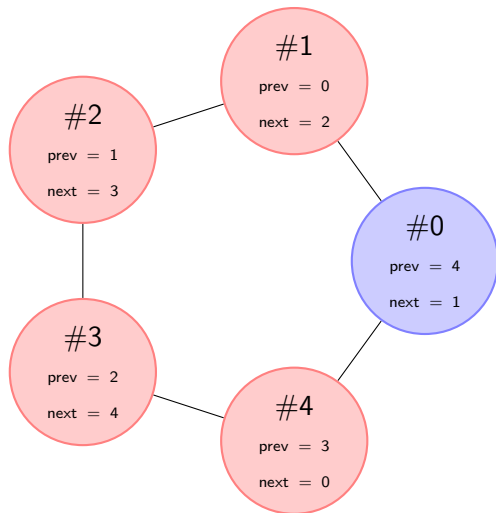
```
fin
```

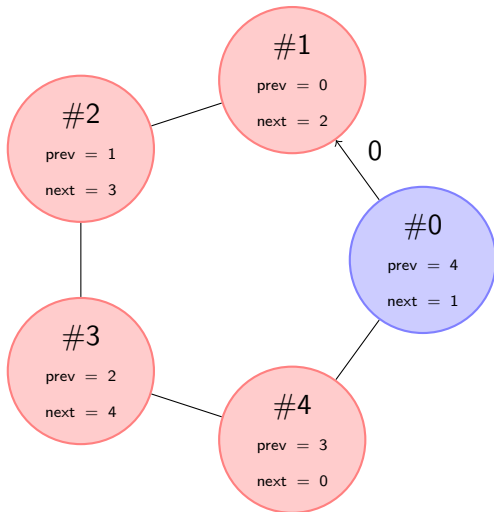


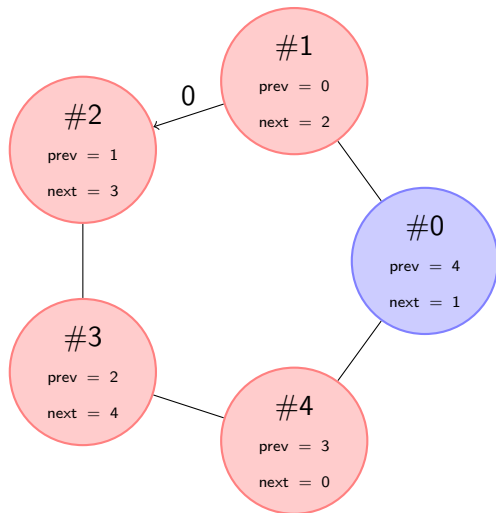


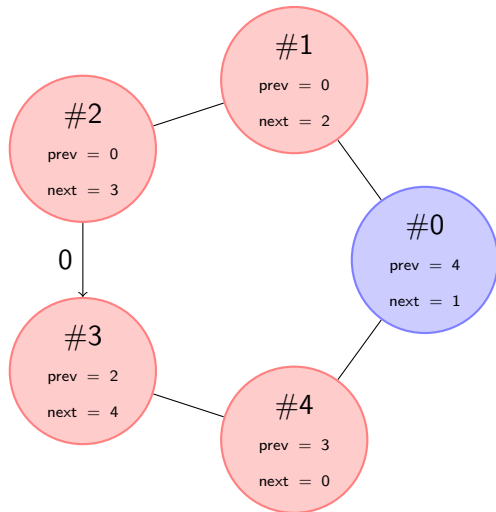


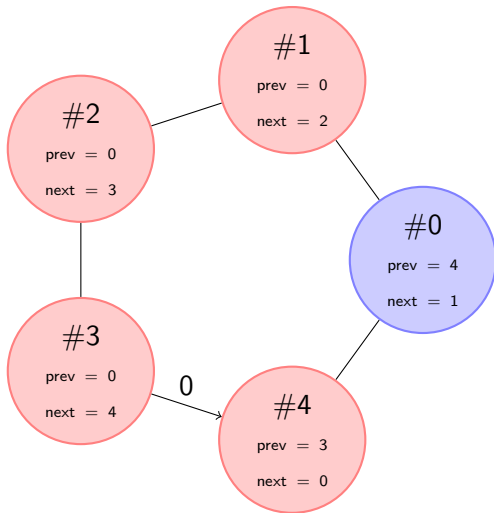


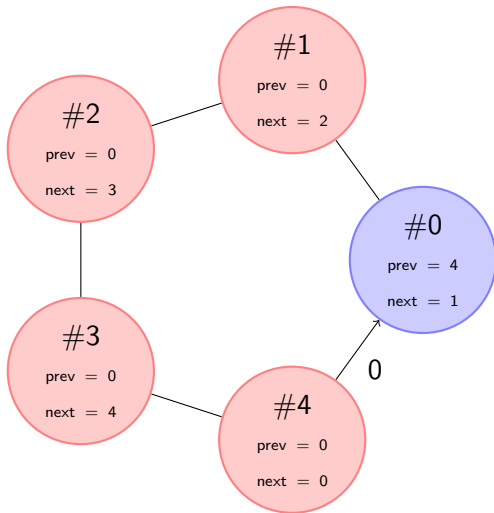


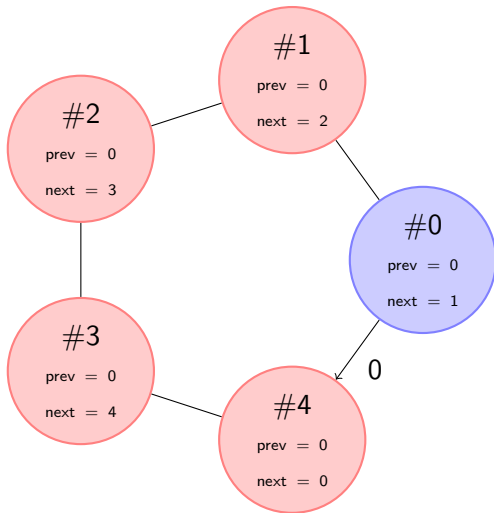


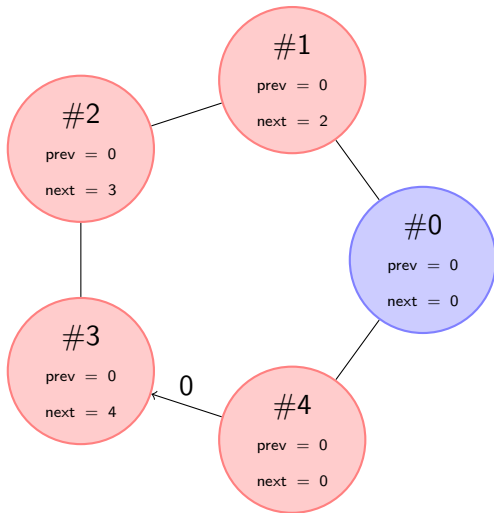


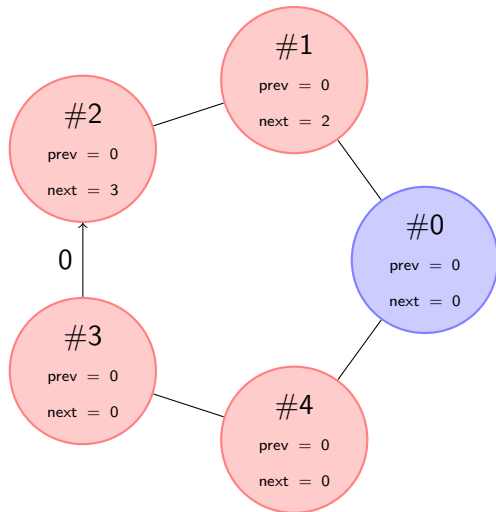


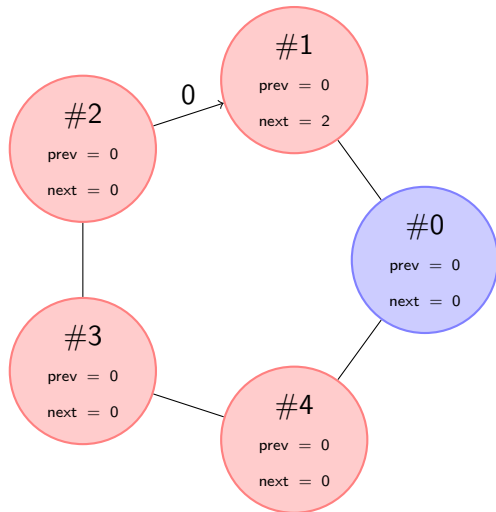


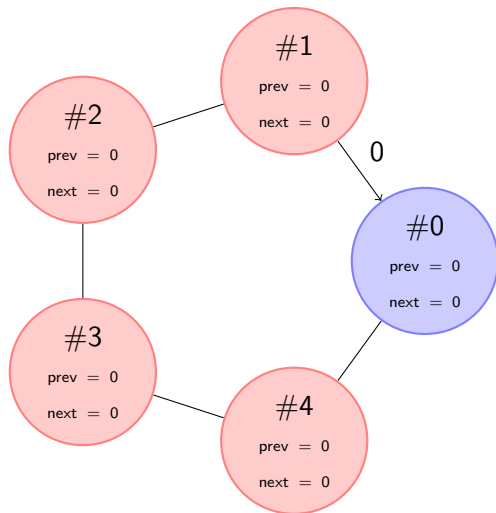


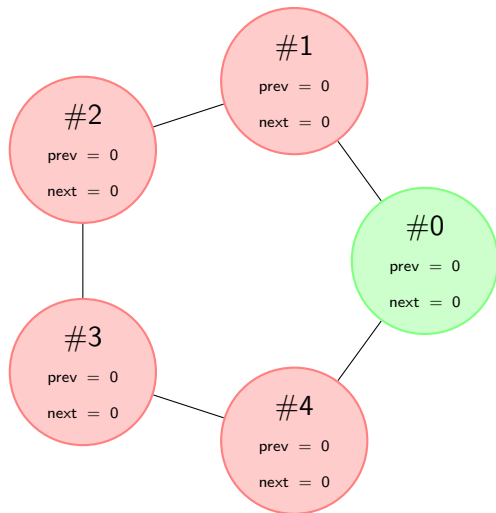


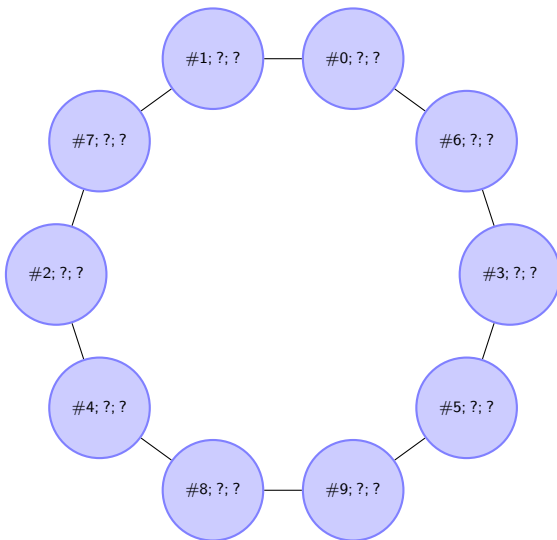


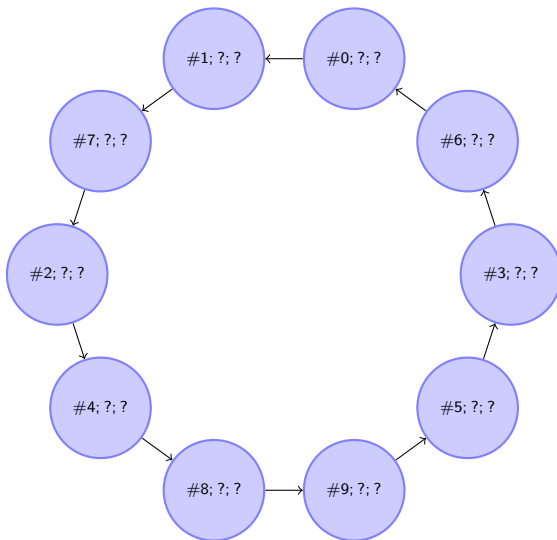


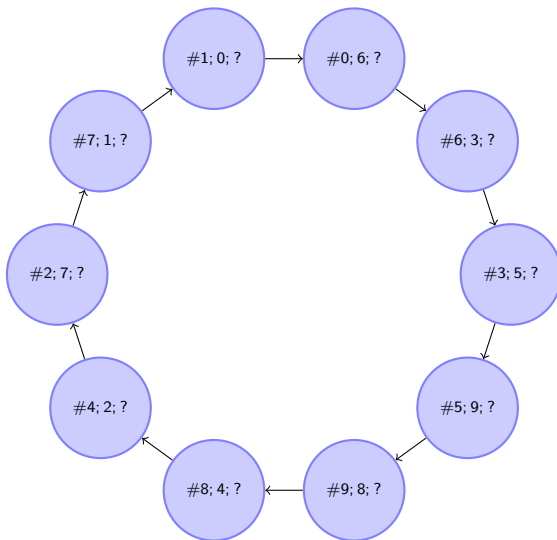


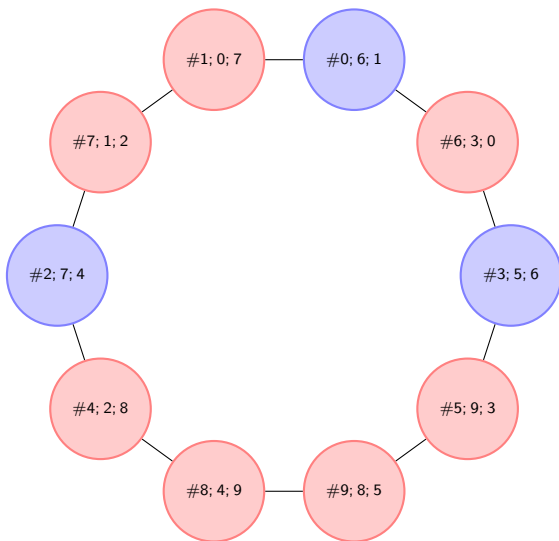












	nombre de messages	temps
au mieux	$O(n)$	$O(n)$
au pire	$O(n * \log(n))$	$O(n)$
en moyenne	$O(n * \log(n))$	$O(n)$