Gestion d'actions

Étude SR05

Hugo Jespierre, Adhavane Moudougannane,
Nathan Piteux, Virginie Tio

Contexte

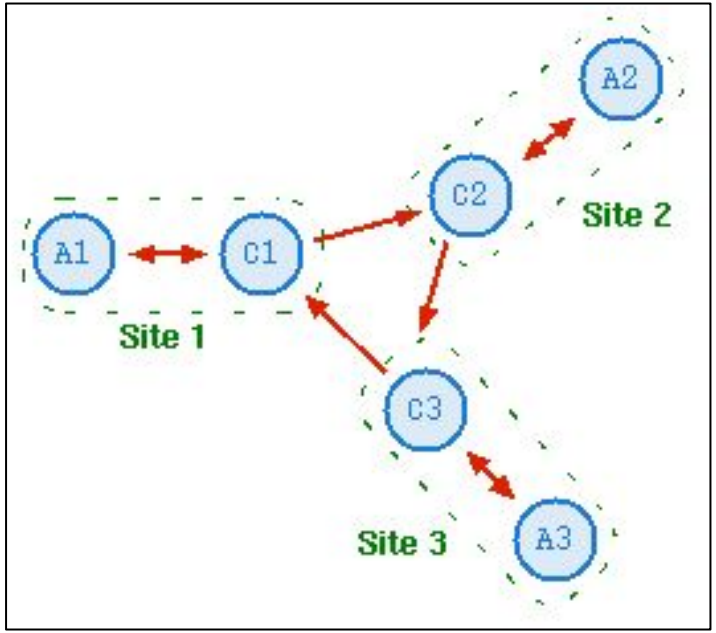
- Gestion d’un stock d’actions par une banque
- Donnée partagée : stock d’action à la banque
- Sites = clients
  - Exclusion mutuelle pour achat et vente
- Nombre de clients statique au fil du temps

Contexte

- Gestion d'un stock d'actions par une banque
- Donnée partagée : stock d'action à la banque
- Sites = clients
  - ➢ Exclusion mutuelle pour achat et vente
- Nombre de clients ~~statique~~ dynamique au fil du temps

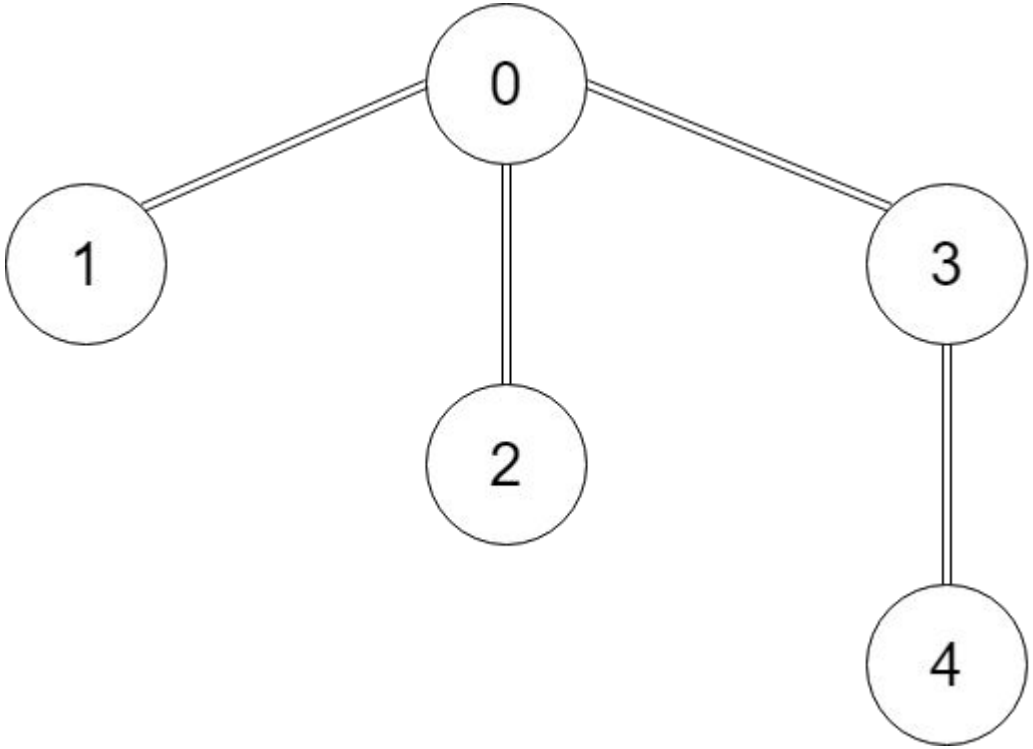
Architecture : avant

- Anneau avec une application de contrôle
- Programmes : **application** et **contrôle**



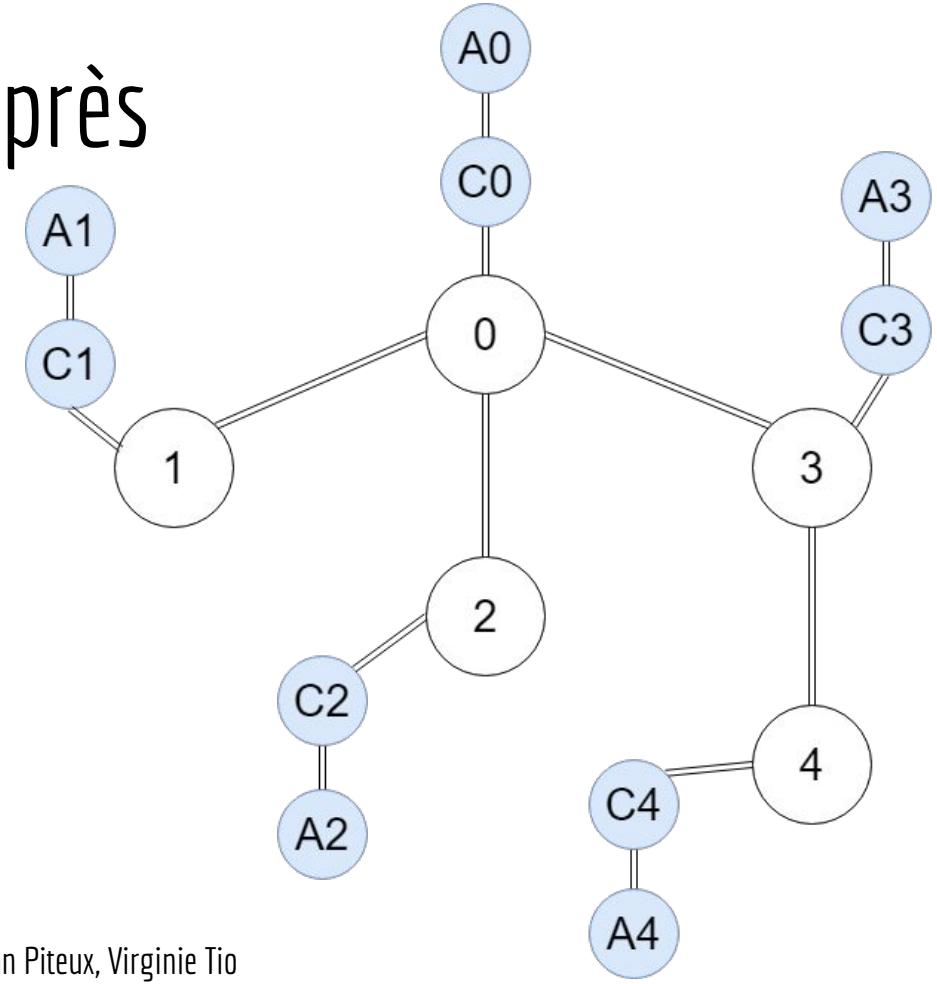
Hugo Jespierre, Adhavane Moudougannane, Nathan Piteux, Virginie Tio

Architecture : après



Hugo Jespierre, Adhavane Moudougannane, Nathan Piteux, Virginie Tio

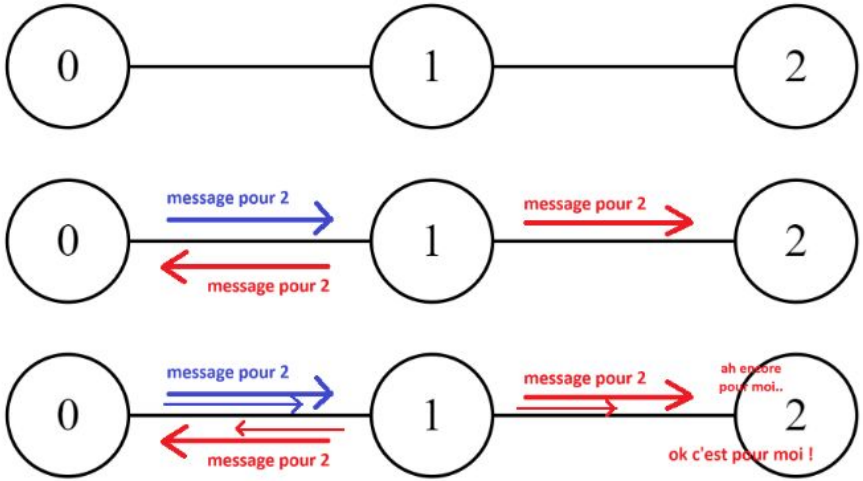
Architecture : après

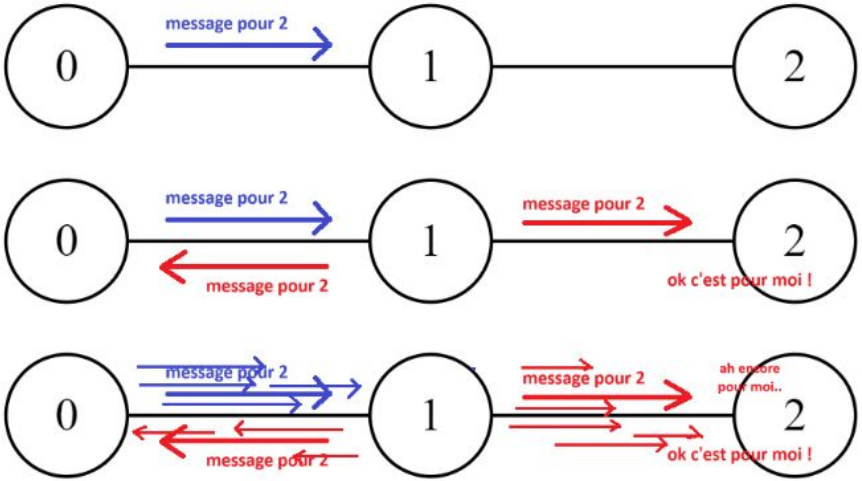


Hugo Jespierre, Adhavane Moudougannane, Nathan Piteux, Virginie Tio

Diffusion de messages

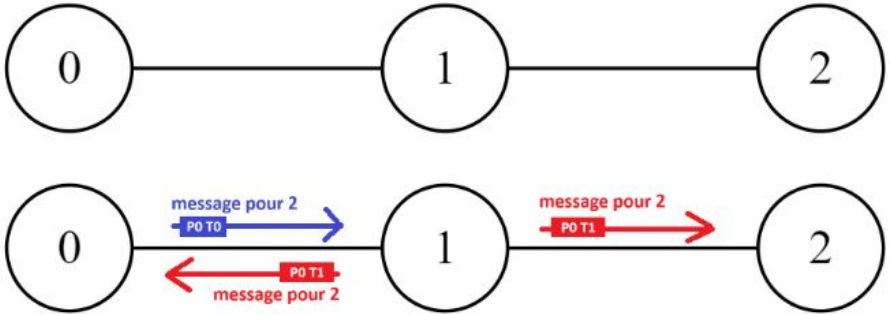
Messages qui bouclent à l'infini sans un réseau bidirectionnel

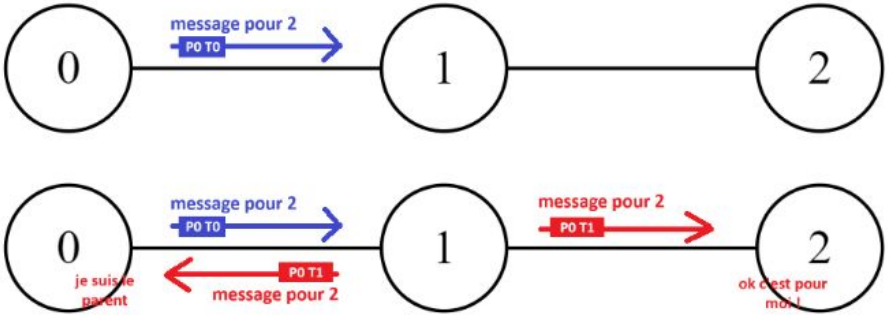




Diffusion de messages

Système du parent/transmetteur pour ignorer les messages rebonds





Contrôleur NET

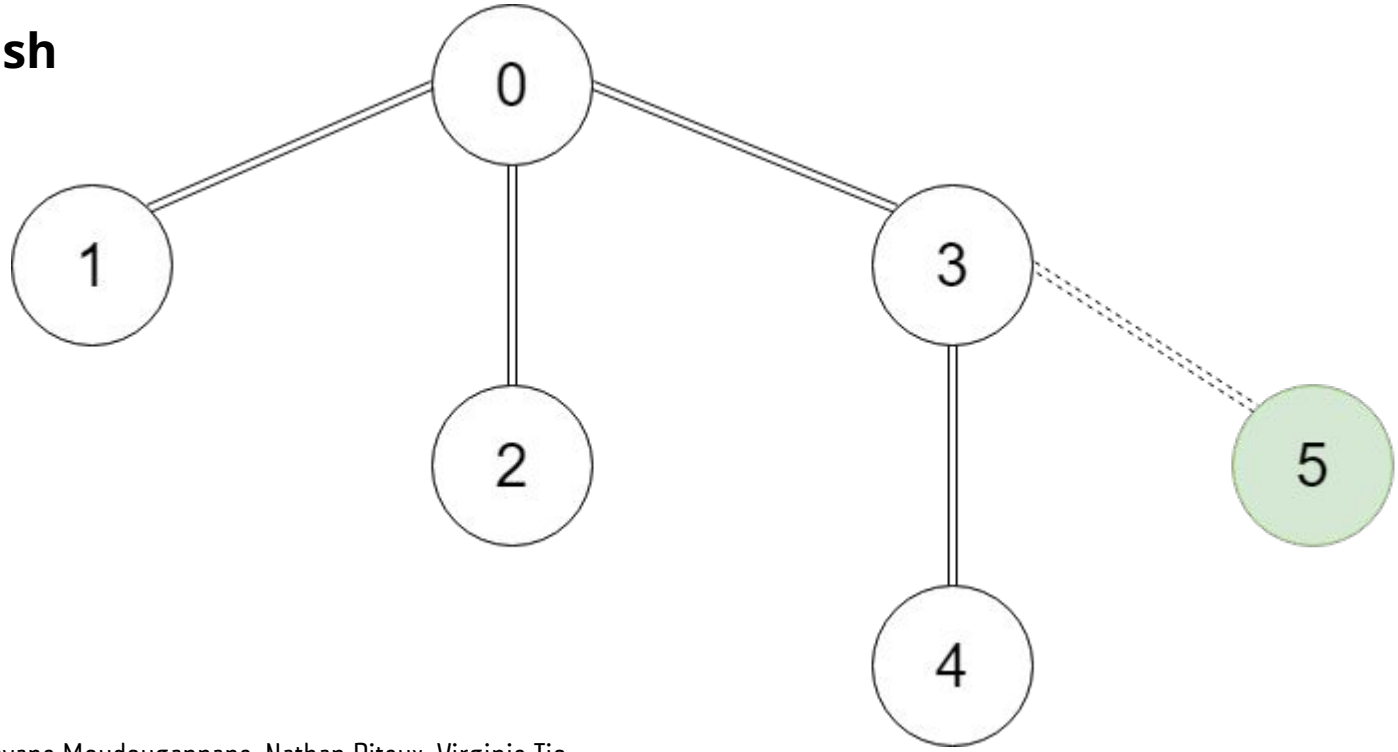
- Gestion de l’élection pour :
  - Ajout d’un site
  - Départ d’un site

- Messages de types `AJOUT`, `SUPPRESSION`, `ACCEPTATION_AJOUT`, et `PREVENTION_VOISINS`

- Appel des scripts shell

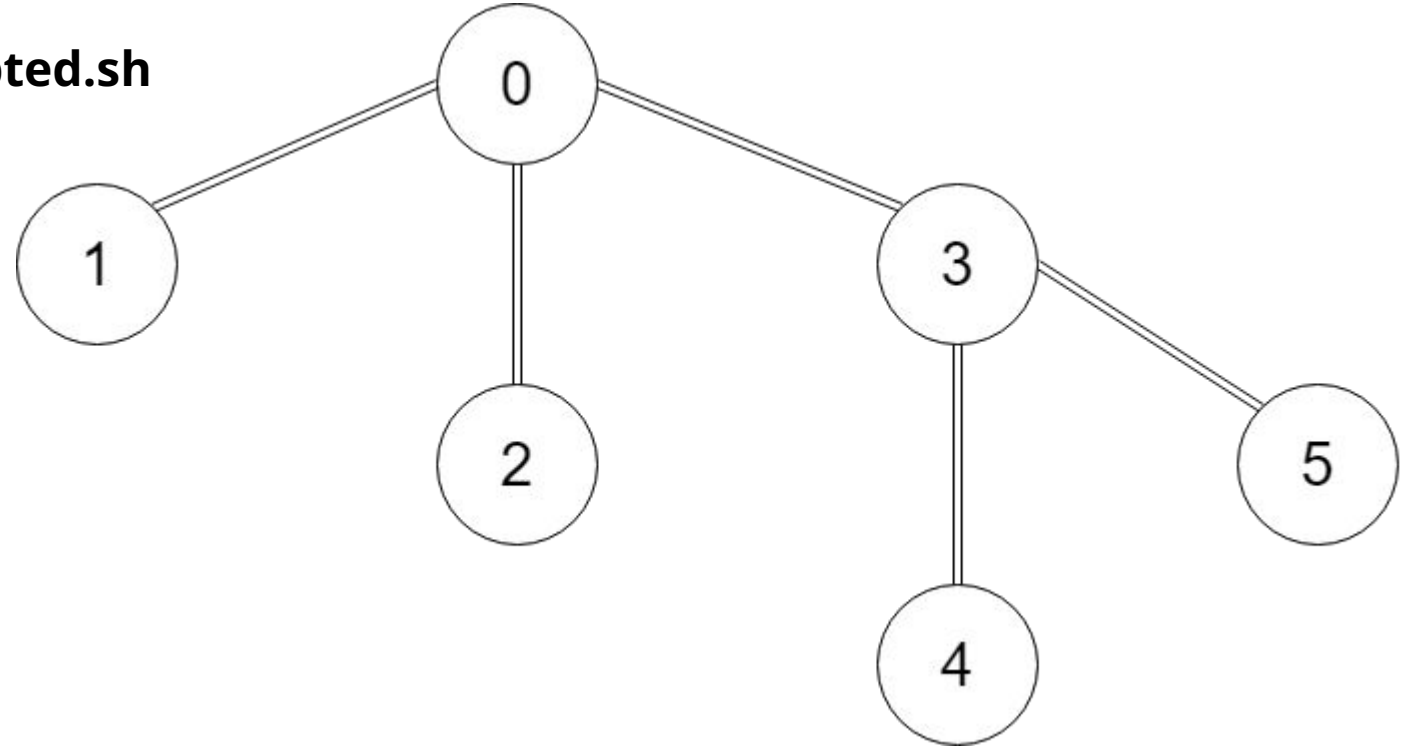
Ajout d'un site : demande

addsite.sh



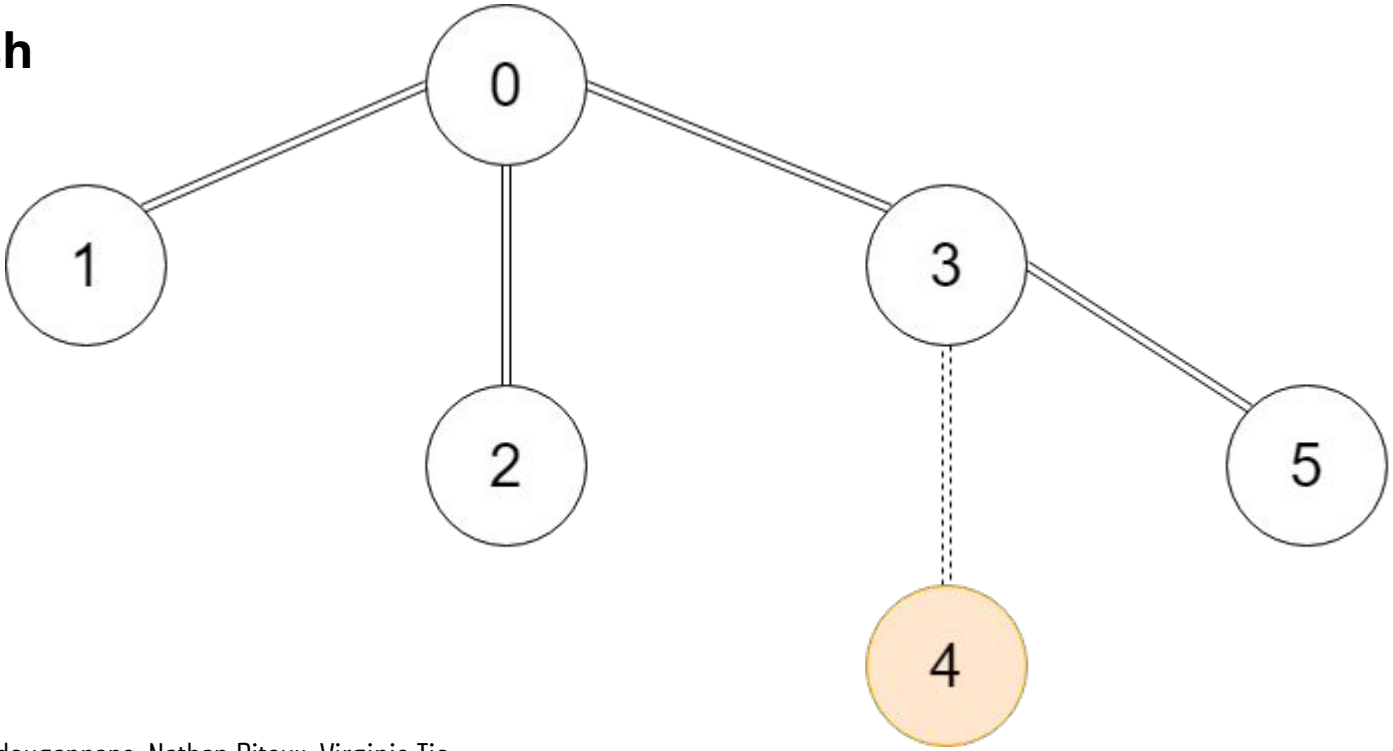
Ajout d'un site : accepté

addsite_accepted.sh

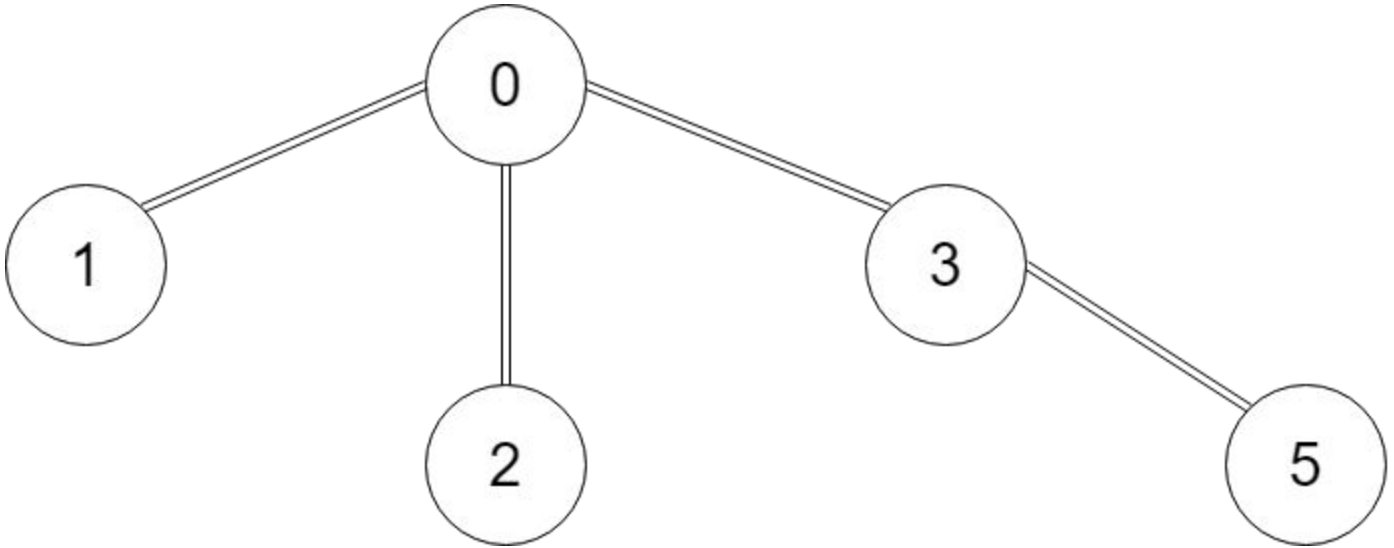


Départ d'un site feuille : demande

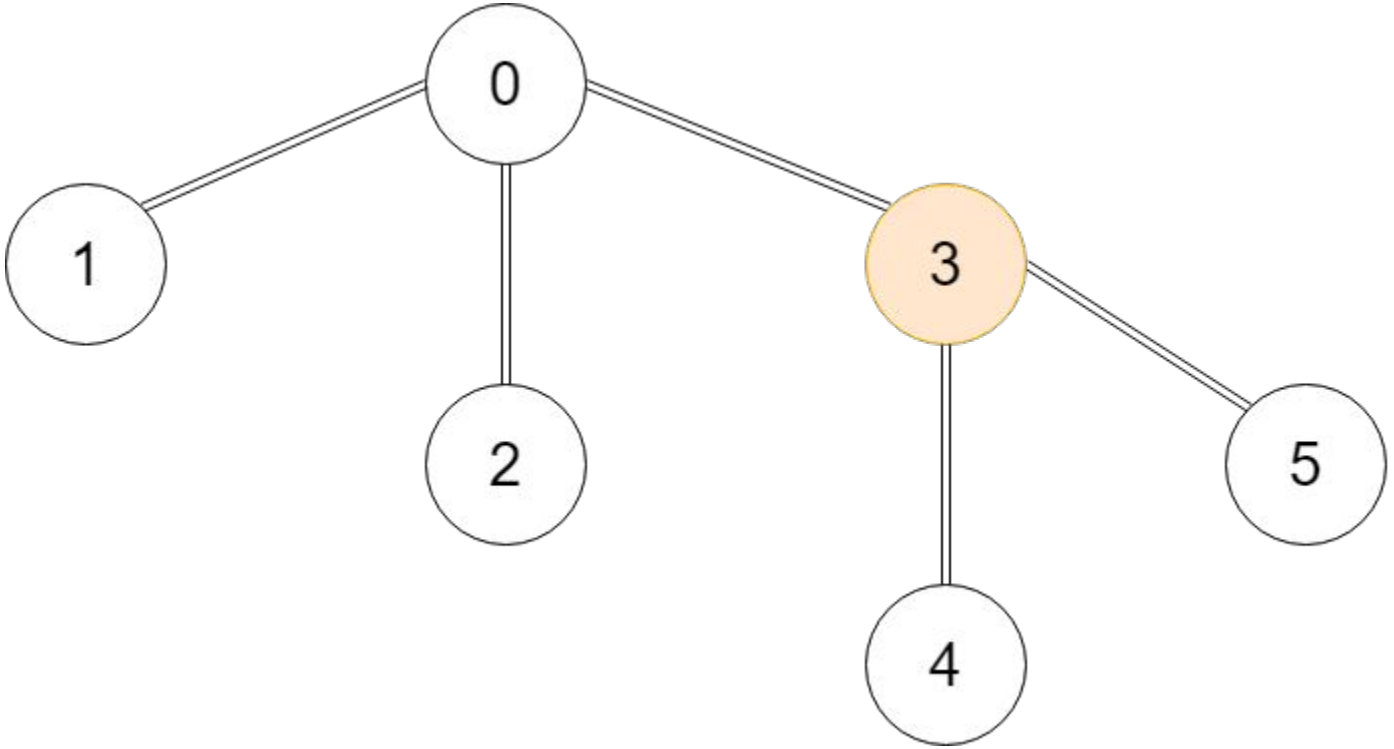
removesite.sh



Départ d'un site feuille : accepté



Départ d'un site intermédiaire



Election

- Algorithme d'élection par extinction de vagues
- Élection du site candidat qui a le plus petit identifiant

1. Initialisation des sites

```
func NewNet(id int, nbNeighbours int, vecteur []int) *Net {
        return &Net{
                Id:               id,
                Active:           true,
                Vi:               vecteur,
                SitetoAdd:        -1,
                Parent:           -1,
                TypeElection:     false,
                NbVoisinsAttendus: nbNeighbours,
                Elu:              int(^uint(0) >> 1),
```

Election

2. Initialisation des sites

    Types de demandes : `DEMANDE_ADMISSION` et `DEMANDE_DEPART`

3. Propagation de la vague (message `ELECTION_BLEU`)

- Site candidat : lance l'élection → envoi de `ELECTION_BLEU` aux voisins
- Réception message :

Si $id_{candidat} < elu_{site}$: mise à jour elu_{site} et envoi de `ELECTION_BLEU` aux voisins

Sinon : message ignoré

Election

4. Remontée de la vague (Message `ELECTION_ROUGE`)

Réception message `ELECTION_BLEU` et le site n'attend plus de messages des voisins :

→ envoi de `ELECTION_ROUGE` au parent

Les messages `ELECTION_ROUGE` remontent vers l'initiateur de l'élection.

Election

5. Fin de l'élection

Initiateur : Réception message `ELECTION_ROUGE` → permission accordée

- Ajout de site : propagation de `ACCEPTATION_AJOUT` et `AJOUT`
- Départ de site : propagation de `SUPPRESSION` et `PREVENIR_VOISINS`
- Mise à jour des horloges vectorielles

Démonstration