TD introduction à MPI

Travail à rendre par mail (pierre.lermusiaux@univ-lorraine.fr) le 18/04, au plus tard.

1 Algorithme de coloration dans un anneau

début
  $c(u) \longleftarrow Id(u)$
  pour $i = 1$ **à** N **faire**
    Envoie $c(u)$ **au** voisin de droite
    Recoit $c(v)$ **du** voisin de gauche v
    $k \longleftarrow$ plus petit indice $t.q.\ c_k(v) \neq c_k(u)$
    $c(u) \longleftarrow c_k(u) : \bar{k}$
  fin
  pour $i = K$ **à** 4 **faire**
    Envoie $c(u)$ **aux** voisins de u
    Recoit $c(v)$ **du** voisin de gauche v
    Recoit $c(w)$ **du** voisin de droite w
    si $c(u) = i$ **alors**
      $c(u) \longleftarrow min(\{1, 2, 3\} \setminus \{c(v), c(w)\})$
    fin
  fin
fin

Algorithme 1 : Algorithme de Cole et Vishkin pour la 3-coloration distribuée pour un sommet u dans un anneau

Correction :

— La coloration initiale est propre par unicité de l'identifiant de chaque noeud.

— On suppose maintenant que la coloration est propre après i itérations et on montre qu'elle est propre après la $(i+1)$-ème itération. On note c la coloration après i itération et on prouve pour tout noeud u que sa couleur est distincte de celle de son voisin de gauche v. Les couleurs respectives des noeuds u et v après la $(i+1)$-ème itération sont $(c_k(u), \bar{k})$ et $(c_{k'}(v), \bar{k'})$ avec $c_k(u) \neq c_k(v)$. Si $k = k'$, les 2 couleurs sont donc distinctes, et sinon, $k \neq k'$, les couleurs sont également distinctes.

Terminaison :

— Si la coloration est encodé en L bits avant la i-ème itération, tout indice $k \in [0, L-1]$ de la coloration est encodable en $\lceil log_2(L) \rceil$ bits, donc $c(u)$, après la i-ème itération, est encodé en $\lceil log_2(L) \rceil + 1$ bits.

— On peut remarquer que pour $L \geq 4$, $\lceil log_2(L) \rceil + 1 < L$, donc après un certain nombre d'itérations, on atteint un minimum en $L = 3$.

— On note $f(x) = \lceil log_2(x) \rceil + 1$, N est le plus petit entier i tel qu'en itérant i fois f depuis $L = \lceil log_2(n) \rceil$, on obtienne une valeure ≤ 3. Après N itérations, on a donc $K = 2^3 = 8$ couleurs.

2 Implémentation

Implémentez l'algorithme en MPJ. Chaque noeud du graphe sera modélisé par une tâche du programme MPI. Lancer le programme avec n tâches résolvera donc la 3-coloration d'un anneau de taille n.

(Note : le nombre de bits de la coloration à la i-ème itération est la taille hypothétique du message des messages échangés)

Vous pourrez utiliser le code fourni ci-dessous par encoder un entier en un tableau de booléens et inversement :

```java
//encodage de l'entier i en un tableau de booleen minimal
public static boolean[] fromInt(int i) {
    return fromInt(i, (int) Math.floor(Math.log(i)/Math.log(2)) +1);
}

//encodage d'un entier b en un tableau de k booleen
public static boolean[] fromInt(int b, int k) {
    boolean [] flags = new boolean[k];
    for (int j = 0; j < k; j++) {
        flags[j] = (b & (1 << j)) != 0;
    }
    return flags;
}

//decodage d'un tableau de booleen bt en un entier
public static int toInt(boolean[] bt) {
    int i = 0;
    for (int j = 0; j < bt.length; j++) {
```

```
        if (bt[j]) i |= (1 << j);
    }
    return i;
}
```

3 Optimisation

Implémentez la version optimisée :

début
- $c(u) \longleftarrow Id(u)$
- **pour** $i = 1$ **à** N **faire**
  - **Envoie** $c(u)$ **aux** voisins de u
  - **Recoit** $c(v)$ **du** voisin de gauche v
  - **Recoit** $c(w)$ **du** voisin de droite w
  - $k \longleftarrow$ plus petit indice $t.q.$ $c_k(v) \neq c_k(u)$
  - $k' \longleftarrow$ plus petit indice $t.q.$ $c_{k'}(w) \neq c_{k'}(u)$
  - $c'(u) \longleftarrow c_k(u) : \bar{k}$
  - $c'(w) \longleftarrow c_{k'}(w) : \bar{k'}$
  - $k \longleftarrow$ plus petit indice $t.q.$ $c'_k(w) \neq c'_k(u)$
  - $c(u) \longleftarrow c'_k(u) : \bar{k}$
- **fin**
- **pour** $i = K$ **à** 4 **faire**
  - **Envoie** $c(u)$ **aux** voisins de u
  - **Recoit** $c(v)$ **du** voisin de gauche v
  - **Recoit** $c(w)$ **du** voisin de droite w
  - **si** $c(u) = i$ **alors**
    - $c(u) \longleftarrow min(\{1, 2, 3\} \setminus \{c(v), c(w)\})$
  - **fin**
- **fin**

fin

Algorithme 2 : Algorithme optimisé de Cole et Vishkin pour la 3-coloration distribuée pour un sommet u dans un anneau

(Note : à chaque itération de la première boucle, on simule une étape du premier algorithme en plus, le nombre de bits de la coloration est donc réduit 2 fois, et, par conséquent, N est réduit d'autant.)