Master2 Informatique
UE: Algorithmique Distribuées – 4TIN913U
Responsable : M. Gavoille
Date : 5 janvier 2022
Durée : 1h30
Documents (course and personnal notes) allowed

Answers can be written in French or in English. All "algorithms" involved in this exam are distributed algorithms in the LOCAL model.

Coloring Cycles with 3 Colors

Question 1. *Give the main features of the LOCAL model.*

Solution. 1. Synchronisme, 2. Identifiant, 3. Taille illimitée des messages

Recall that an *oriented* cycle is a cycle graph where each vertex is aware of its successor (and thus of its predecessor). This knowledge can be used to speedup coloring algorithm.

In the lecture, we described a fast 3-coloring algorithm for oriented cycles with n vertices running in at most $f(n)+c$ rounds, where $f(n)$ is some suitable fonction of n, and c is some small constant. For simplicity, let us denote this algorithm by A.

Question 2. *Describe the principle of algorithm* A*, i.e., the main steps, and specify what is* $f(n)$ *and the constant* c*.*

Solution. On calcule en $\lceil \log^*(n)/2 \rceil$ rondes une 6-coloration en répétant un double PosDiff entre sa couleur x et celles de ses voisins y, z. Puis on élimine successivement les couleurs de numéro $3, 4, 5$. Au final, cela prend $f(n)+c = \lceil \log^*(n)/2 \rceil + 3$ rondes.

The goal is to slightly improve algorithm A. Recall that, at some stage, algorithm A reduces the palette of colors (given as a range of integers) from $[0,6[= \{0,1,2,3,4,5\}$ to $[0,3[= \{0,1,2\}$.

Question 3. *Show that the colors* 4 *and* 5 *can be removed simultaneously from the initial palette* $[0,6[$ *in only one round. [Hint: Manage carefully the case where vertices of colors* 4 *and* 5 *are adjacent.] Does this round use the orientation of the cycle?*

Solution. Soit c la coloration. On calcule $X = c(N(u)) = \{c(u) : u \in N(u)\}$ en une ronde. Puis si $c(u) = 4$, $c'(u) = \text{FirstFree}(X)$ et si $c(u) = 5$, $c'(u) = \text{FirstFree}(X \cup [0,1[)$. Cette ronde n'utilise pas l'orientation. Preuve que cela marche. Soit v un voisin de u. Il faut montrer que $c'(u) \neq c'(u)$. Si $c(v) \notin \{4,5\}$, alors $c'(v) = c(v)$ et donc $c'(u) \neq c'(v)$ car $c'(u) \notin X$ qui contient $c(v)$. Supposons $c(u) = 4$ et $c(v) = 5$ (ou le contraire en échangeant le rôle de u et v). Comme X contient 5, $c'(u) < 2$. Et donc $c'(v) \neq c'(u)$ car $c'(v) \geqslant 2$ par construction.

Let us call this improved algorithm, algorithm B.

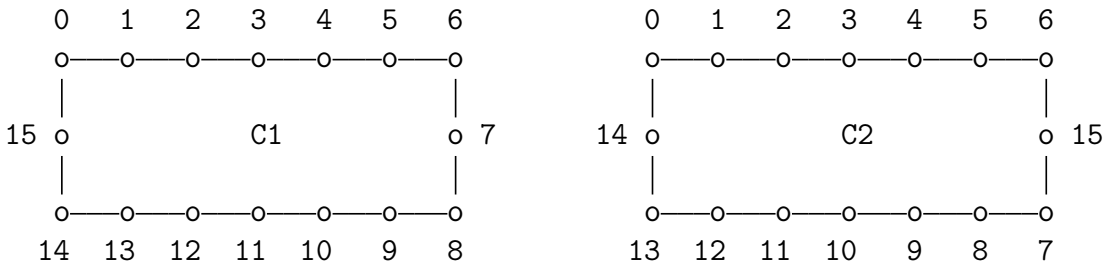
Question 4. *What is the maximum number of rounds of* B *if* $n \leqslant 65536 = 2^{16}$*? Justify.*

Solution. On a $2^{16} = 2^{2^{2^1}}$ et donc $\log^* 65536 = 4$. L'algorithme prend donc un temps de $\lceil \log^*(n)/2 \rceil + (3-1) = \lceil 4/2 \rceil + 2 = 4$ rondes.

Coloring Even Cycles with 2 Colors

In this part we are interesting in 2-coloring cycles with an even number of vertices. We will assume that all cycles are oriented, even if we did not say it explicitly.

Consider two cycles, C_1 and C_2, each with $n = 16$ vertices, and with the following ID distributions:



Question 5. *Design a 2-coloring algorithm specific for C_1 that runs in 0 round. Same question for C_2.*

SOLUTION. Pour C_1, $c(u) = \text{ID}(u)\%2$. Pour C_2, si $\text{ID}(u) \in [7, 14]$, $c(u) = 1 - \text{ID}(u)\%2$, sinon $c(u) = \text{ID}(u)\%2$.

In the lecture, we proved a lower bound of $(\frac{1}{2}\log^* n) - 1$ rounds for any 4-coloring algorithm for cycles with n vertices.

Question 6. *Noting that a 2-coloring is a particular 4-coloring, what is the minimum number of rounds implied by this above lower bound applied to 2-coloring for cycles of $n = 16$ vertices? Justify.*

SOLUTION. $\log^* 16 = \log^* 2^{2^2} = 3$. Il faut donc au moins $\lceil 3/2 - 1 \rceil = 1$ ronde.

Question 7. *Is the result of Question 6 in contradiction with the algorithms of Question 5? Justify.*

SOLUTION. Non, car la borne inférieure s'applique pour la coloration de tout cycle à n sommets, c'est-à-dire tout cycle avec toute distribution d'ID dans $[0, n[$.

Question 8. *Design a fast 2-coloring algorithm that applies to both C_1 and C_2. Specify its number of rounds. [Hint: Detect if a vertex lives in C_1 or C_2.]*

SOLUTION. Dans C_1 on remarque que les sommets 7 et 15 sont à distance 8, donc chaque boule de rayon 4 doit contenir 7 ou 15 (et même le chemin menant à l'un de ces sommets). Pour qu'un sommet u soit dans C_1, il faut avoir dans $B(u, 4)$, soit l'arête $0 - 15$, soit l'arête $14 - 15$, soit l'arête $6 - 7$. Sinon le sommet u est dans C_2 puisque ces arêtes n'existent pas dans C_2. Ainsi 4 rondes de communications sont suffisantes pour que chaque sommet u calcule $B(u, 4)$. On teste alors si u est dans C_1 ou C_2. Ensuite on exécute le bon algorithme qui coûte 0 ronde. Au total cela prend 4 rondes.

In the lecture, we showed that any k-coloring algorithm running in t rounds can be seen as a mapping from t-views to $[0, k[$. Recall that a t-view of a vertex u is the sequence of all the IDs that u contains in its ball of radius t, i.e., the IDs that u can see after t rounds of communication. For a cycle with n vertices, this is a sequence of $2t+1$ unique integers taken from $[0, n[$. For instance, in C_1 the 3-view of vertex with ID 4 is the sequence $(1, 2, 3, \boxed{4}, 5, 6, 7)$ assuming a clockwise orientation.

Question 9. *Show that any 2-coloring algorithm for both C_1 and C_2 requires at least 4 rounds. [Hint: Consider the 3-views for vertices with IDs 3 and 10 for instance.]*

SOLUTION. Dans C_1 et C_2 les sommets 3 et 10 ont les mêmes vues et pourtant sont à des distances resp. impaire et paire. Donc quelque soit le mapping de ces vues vers les couleurs 0 ou 1, cela sera incorrect pour au moins l'un des deux graphes car si la distance est paire les sommets doivent être de la même couleur, et opposée sinon.

Question 10. *Show that, more generally, any 2-coloring algorithm for cycles of n vertices requires $\Omega(n)$ rounds. [Hint: Consider some cycles with $n \geqslant 4(t+1)$ vertices, and argue about the t-views of some specific vertices.]*

SOLUTION. On prend t le plus grand entier tel que $n \geqslant (4t+2)+2$, soit $t = \Omega(n)$. On peut alors construire deux cycles chacun ayant deux t-voisinages disjoints, puisque cela fait $2{\cdot}(2t{+}1) = 4t{+}2 < n$ sommets. Il reste deux sommets (au moins) permettant de placer le centre des t-vues soit à distance pair ou impair l'un de l'autre, ce qui mène à une constradiction pour une 2-coloration. Il faut donc $> t = \Omega(n)$ rondes.

Fast Palette Reduction

The goal of this part is to design a $(\Delta+1)$-coloring for any graph of maximum degree Δ. We have seen in the lecture such an algorithm running in $\log^* n + 2^{O(\Delta)}$ rounds. We will focus on the second phase of the algorithm, the palette reduction, that takes about $2^{O(\Delta)}$ rounds to decrease the palette from $[0, 3^\Delta[$ to $[0, \Delta]$.

Question 11. *Give an accurate bound on the number of rounds it takes to perform this palette reduction?*

SOLUTION. Cela prend $3^\Delta - (\Delta+1)$ rondes.

In order to perform a faster palette reduction, we will remove several colors in a single rounds, as done in Question 3. Assume that after the first phase, we have a coloring c with palette $[0, k[$ and that each vertex see at most d different colors in its neighborwood. In other words, $d = \max_u |c(N(u))|$. For simplicity we assume that $k = sd + s$ for some integer $s \geqslant 2$.

The goal is to reduce the intial palette from $[0, k[= [0, sd+s[$ to $[0, sd[$ by removing the s largest colors of $[0, k[$. For this, every vertex u with a color in $[sd, sd+s[$ applies in parallel a FIRSTFREE$(X \cup I_i)$, where $X = c(N(u))$ and I_i is some subinterval of $[0, sd[$ depending on the color $i = c(u)$ of u and whose aim is to avoid collisions. Indeed, two neighbors of different colors in $[sd, sd+s[$ will apply in parallel FIRSTFREE.

To make more concrete this technique, consider the following example where $k = 9$, $d = 2$ and $s = 3$. The initial palette is $[0, k[= [0, sd+s[= [0, 3\cdot 2+3[= [0, 9[$ and the goal is to reduce it to $[0, 6[$ by removing the colors $6, 7, 8$ in a single round. These latter colors are called *special*. Consider a vertex u with a special color. The round of communication consists in exchanging the color of u with its neighbors. Let $X = c(N(u))$ be the set of colors of u's neighbors. If X does not contain any special color, then u can recolor with FIRSTFREE(X) since none of its neighbors are concerned with a recoloring. Otherwise, split the palette $[0, k[= [0, sd+s[$ into s subintervals of length d followed by the s special colors. Namely, split $[0, 9[$ into $[0, 1] \cup [2, 3] \cup [4, 5] \cup [6, 7, 8]$. Then, the recoloring is as follows: if $c(u) = 6$, then u tries to recolor in $[0, 1]$ using a FIRSTFREE(X); if $c(u) = 7$, then u tries to recolor in $[2, 3]$ using a FIRSTFREE$(X \cup [0, 2[)$; and if $c(u) = 8$, then u tries to recolor in $[4, 5]$ using a FIRSTFREE$(X \cup [0, 4[)$. In other words, u tries to recolor in the subinterval corresponding to the rank of its special color. The rank of the special color of u is precisely $c(u) - sd$. Note that intervals of length d suffice. Indeed, if two neighbors are of special colors, then they can see only $d-1$ non-special colors each or less (instead of d). And thus FIRSTFREE will necessarily find a free color in an interval of length $(d-1)+1 = d$.

Question 12. *Formalized the above algorithm that reduces the palette from $[0, sd+s[$ to $[0, sd[$ in one round.*

SOLUTION.

```
NEWPULSE
SEND(v,c(u)) pour tout voisin v
```

```
X := U_v RECEIVE(v)
Si c(u) \in [sd,sd+s[
   Si X intersecte [sd,sd+s[, alors X := X \cup [0,(c(u)-sd)d[.
   c(u) := \FF(X)
```

In the remaining, we fix $d = \Delta$. We will apply iteratively the Fast Palette Reduction (FPR) as in Question 12, possibly with a different parameter s, until we get a palette of 2Δ colors or less. From that point the FPR does not apply anymore and a $O(\Delta)$-round classical palette reduction is needed to get the wanted $(\Delta+1)$-coloring.

More precisely, let $s_0, s_1, \ldots, s_t$ be some reals such that: (1) The initial palette is $[0, k[= [0, s_0\Delta[$; (2) At step $i \in \{1, \ldots, t\}$, whenever FPR is applied, the palette reduces from $[0, s_{i-1}\Delta[= [0, s_i\Delta + s_i[$ to $[0, s_i\Delta[$.

In the remaining, we will negligate the fact that some s_i are not necessarily integers[1].

Question 13. *Express s_i as fonction of i, Δ and k.*

SOLUTION. Il faut $k = s_0\Delta$, d'où $s_0 = k/\Delta$. Pour $i > 0$, il faut $s_{i-1}\Delta = s_i(\Delta+1)$ soit $s_i = (\Delta/(\Delta+1)) \cdot s_{i-1}$. Soit $c = \Delta/(\Delta+1)$. Donc $s_i = c \cdot s_{i-1} = c^2 \cdot s_{i-2} = c^j \cdot s_{i-j} = c^i \cdot s_0 = c^i \cdot (k/\Delta)$.

Question 14. *Find the smallest number t of FPR steps so that the final palette is $[0, 2\Delta[$ or smaller. Overall, what is the complexity of the resulting new $(\Delta+1)$-coloring algorithm? [Hint: Assume known the inequality $1/(x+1) < \ln(1+1/x) < 1/x$ for every $x > 0$.]*

SOLUTION. On cherche le plus petit t tq $s_t \leqslant 2$. Il faudra penser à rajouter $O(\Delta)$ rondes supplémentaires pour passer de 2Δ à $\Delta+1$ couleurs. On a $s_t = c^t s_0 \leqslant 2$. C'est équivalent à $c^t \leqslant 2/s_0$ ou encore $(1/c)^t \geqslant s_0/2 \Leftrightarrow (1+1/\Delta)^t \geqslant s_0/2 \Leftarrow e^{t/(\Delta+1)} > s_0/2 \Leftrightarrow t/(\Delta+1) > \ln(s_0/2) \Leftrightarrow t > (\Delta+1)\ln(k/(2\Delta)) = (\Delta+1)\ln(2^{O(\Delta)}) = \Theta(\Delta^2)$. Donc le plus petit nombre t recherché est en $\Theta(\Delta^2)$, ce qui implique bien que $s_t \leqslant 2$ comme requis. Au total $\lceil t \rceil + O(\Delta) = O(\Delta^2)$ rondes.

In a celebrate paper of 1992, Nathan Linial showed how to find a $O(\Delta^2)$-coloring in $O(\log^* n)$ rounds.

Question 15. *Using this Linial's precoloring, and combining with Question 14, design an even faster $(\Delta+1)$-coloring algorithm. What is its complexity?*

SOLUTION. Utiliser la coloration de Linial, puis faire la FPR. Le nombre de rondes est donc $O(\log^* n) + t$ avec $t = O(\Delta \ln(k/\Delta)) = O(\Delta \ln \Delta)$.

FIN.

[1] Otherwise we will have to consider $\lfloor s_i \rfloor$ instead of s_i, which is awkward and does not affect anway the asymptotic results.