11 Plus courts chemins en nombre d'arcs

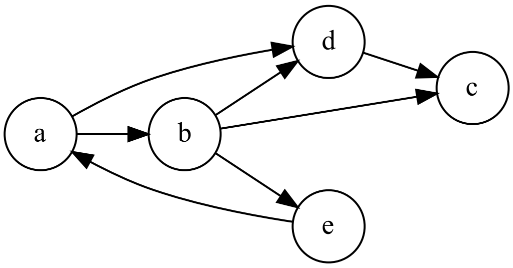

FIGURE 1 – Graphe orienté G

11.1 La question reformulée en termes de graphe est la suivante : existe-il un chemin orienté depuis a vers tous les autres sommets ? La réponse est oui :

— Accès à b par (a, b)

— Accès à c par (a, b) puis (b, c)

— Accès à d par (a, b) puis (b, d)

— Accès à e par (a, b) puis (b, e)

11.2 Le parcours en largeur est adapté pour calculer les distances dans un graphe non pondéré. Ici, tous les arcs ont le même poids, donc on peut considérer que le graphe est non pondéré.

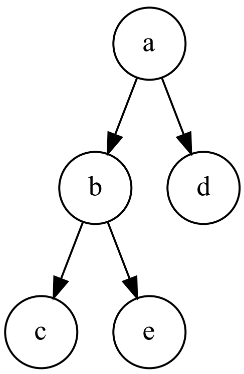

FIGURE 2 – Avec un parcours en largeur du graphe G, on voit que b et d sont à une distance 1, c et e à une distance 2.

Algorithme 1 : $ParcoursLargeur(G, s)$

$L \leftarrow [\,]$
pour $x \in S$ **faire**
  $visite[x] \leftarrow 0$
fin
pour $x \in S$ **faire**
  # Rq : les x qui vérifient cette condition sont les points de regénération.
  si $visite[x] = 0$ **alors**
    $ParcoursLargeurAux(x, G, visite, L)$
  fin
fin

Algorithme 2 : $ParcoursLargeurAux(G, x, visite, L)$

$F \leftarrow$ file vide
$Enfiler(F, x)$
tant que F *non vide* **faire**
  $y \leftarrow Defiler(F)$
  $Ajouter(L, y)$
  $visite[y] \leftarrow 1$
  pour $z \in voisins(y)$ **faire**
    $Enfiler(F, z)$
  fin
fin

Pour calculer les distances, on remplace le tableau *visite* par un tableau *dist* qui stocke les distances à a.

Algorithme 3 : $ParcoursLargeur'(G, s)$

$L \leftarrow [\,]$
pour $x \in S$ **faire**
  $dist[x] \leftarrow -1$
fin
pour $x \in S$ **faire**
  si $dist[x] = -1$ **alors**
    $ParcoursLargeurAux'(x, G, dist, L)$
  fin
fin

Algorithme 4 : $ParcoursLargeurAux'(G, x, dist, L)$

$F \leftarrow$ file vide
$Enfiler(F, x)$
$dist[x] \leftarrow 0$
tant que F *non vide* **faire**
  $y \leftarrow Defiler(F)$
  $Ajouter(L, y)$
  pour $z \in voisins(y)$ **faire**
    si $dist[z] = -1$ **alors**
      $dist[z] \leftarrow dist[y] + 1$
      $Enfiler(F, z)$
    fin
  fin
fin