HLIN608 - Algorithmique du texte - TD SSP

Antoine AFFLATET et Jérémie ROUX (L3 Groupe C)

2019 - 2020

Problème de décision SSP (*Shortest Superstring Problem*)

Entrée : Un ensemble de mots $\mathcal{F} = \{F_1,...,F_n\}$, et un entier strictement positif K.
Question : Existe-t-il une superséquence S de $\mathcal{F}$ de longueur $\leq K$ telle que chaque mot de $\mathcal{F}$ est un sous-mot de la superséquence S ?

Exercice 1

On admet *ssp-optimisation*($\mathcal{F}$) une fonction qui en temps polynomial renvoie la longueur de la plus petite superséquence S telle que chaque mot de $\mathcal{F}$ est un sous-mot de S (algorithme polynomial pour le problème d'optimisation).

Algorithme 1 : *ssp-decision*($\mathcal{F}$: ensemble de mots, K : entier) : booléen

```
si ssp-optimisation(F) > K alors
    renvoyer faux;
fin
renvoyer vrai;
```

L'**Algorithme 1** est en temps polynomial $O(1)+O(ssp-optimisation)$ soit $O(ssp-optimisation)$ où $O(ssp-optimisation)$ est la complexité de ssp-optimisation (polynomial). Ainsi le problème d'optimisation est "au moins aussi difficile" que le problème de décision qui lui est associé.

Remarque :

On admet maintenant *ssp-decision*($\mathcal{F}$,K) une fonction qui en temps polynomial renvoie vrai s'il existe une superséquence S de $\mathcal{F}$ de longueur $\leq K$ telle que chaque mot de $\mathcal{F}$ est un sous-mot de la superséquence S, faux sinon (algorithme polynomial pour le problème de décision).

Algorithme 2 : *ssp-optimisation*($\mathcal{F}$: ensemble de mots) : entier

```
Variables : min, max, mid : entier;
min ← 0;
max ← 0;
mid ← 0;
pour chaque mot m de F faire
    max ← max + longueur(m);
fin
tant que min >= max faire
    mid ← (max − min)/2;
    si ssp-decision(F,mid) alors
        max ← mid;
    sinon
        min ← mid + 1;
    fin
fin
renvoyer max;
```

L'**Algorithme 2** étant en temps polynomial $o(n + log(m) * O(\textit{ssp-decision}))$ où n est le nombre de mots dans F, m la somme des tailles des mots de F et $O(\textit{ssp-decision})$ la complexité de *ssp-decision* (polynomial). Ainsi, le problème d'optimisation est "au moins aussi facile" que le problème de décision.

Exercice 2

Algorithme 3 : *recherche-naive*(m : mot, S: superséquence) : booléen

```
Variables : resultat : booléen; i, j : entier;
resultat ← faux;
i ← 0;
tant que i < taille(S) et ! resultat faire
    j ← 0;
    tant que j < taille(m) et m[j] = S[i + j] faire
        j ← j + 1;
    fin
    si j = taille(m) alors
        resultat ← vrai;
    fin
    i ← i + 1;
fin
renvoyer resultat;
```

L'algorithme *recherche-naïve* (**Algorithme 3**) est en $o(s * n)$ où s est est la taille de S (superséquence) et n la taille du mot m; soit $O(n^2)$.

Algorithme 4 : *verif-ssp*($\mathcal{F}$: ensemble de mots, n : nombre de mots, S : superséquence) : booléen

Variables : trouve : booléen; i : entier;
$trouve \leftarrow vrai$;
$i \leftarrow 0$;
tant que $trouve$ *et* $i < n$ **faire**
    $trouve \leftarrow$ *recherche-naive*($\mathcal{F}[i]$,S);
    $i \leftarrow i + 1$;
fin
renvoyer $trouve$;

L'algorithme *verif-ssp* (**Algorithme 4**) est en $o(n * O(\text{recherche-naïve}))$ soit $O(n^3)$ où n est le nombre de mots dans F. Ainsi, cet algorithme polynomial vérifie si pour une instance donnée du problème et une superchaîne, cette chaîne satisfait le problème de décision.

Exercice 3

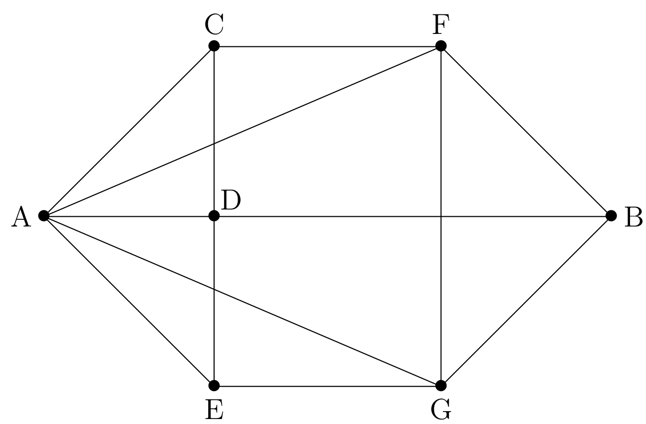

Fig. 1 – *Graphe non-orienté de départ*

Problème de décision VERTEX COVER

Entrée : Un graphe $G = (V,E)$ et un entier strictement positif k.
Question : Existe-t-il une couverture des sommets de G, notée C, de taille $\leq k$?

Exercice 4

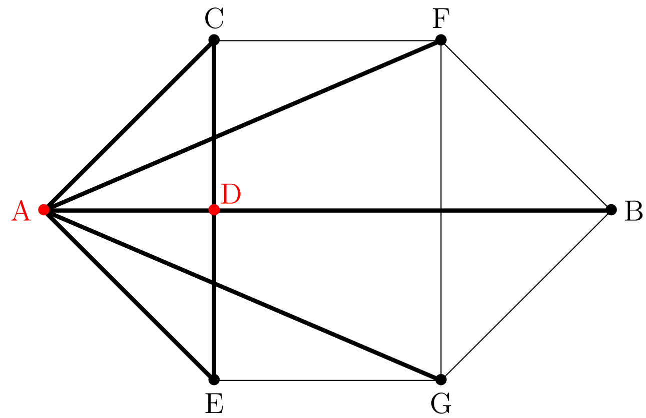

FIG. 2 – *Exemple d'une couverture minimale pour le graphe de la* FIG. 1 *(taille 2)*

Algorithme 5 : *VERTEX-COVER*$(G = (V,E)$: graphe, k : entier) : booléen

---

$\mathcal{F} \leftarrow \{\}$;
pour chaque *arête* $\{a,b\}$ *de* E **faire**
  $\mathcal{F} \leftarrow \mathcal{F} \cup \{abab\} \cup \{baba\}$;
fin

`/* Calcul de la longueur` H `de la superchaine` S `du problème SSP pour laquelle le graphe` G `admet un vertex-cover de taille` k`, on montrera plus tard que cette dernière vaut 4*|E|+k */`

$H \leftarrow longueurSuperchaine(E,k)$;

renvoyer *ssp-decision*$(\mathcal{F},H)$;

En admettant que l'on puisse calculer *ssp-decision* en un temps polynomial et que *longueurSuperchaine* est en $O(1)$, l'**Algorithme 5** montre une réduction du problème du VERTEX COVER vers le problème SSP (décisionnel) en un temps polynomial $O(n) + O(1) + O(ssp)$. Ainsi la complexité de cet algorithme dépend de la complexité de *ssp-decision* (la transformation étant en temps polynomial O(n)).

Exercice 5

L'alphabet considéré est l'ensemble des sommets de la FIG. 1, soit :

$$\Sigma = V = \{a,b,c,d,e,f,g\}$$

On étudie le graphe de la FIG. 2 qui est un VERTEX COVER du graphe de la FIG. 1 :

Arêtes	**Ensemble de chaînes associées**
$\{a,c\}$	$\{acac,caca\}$
$\{a,f\}$	$\{afaf,fafa\}$
$\{a,d\}$	$\{adad,dada\}$
$\{a,g\}$	$\{agag,gaga\}$
$\{a,e\}$	$\{aeae,eaea\}$
$\{d,c\}$	$\{dcdc,cdcd\}$
$\{d,e\}$	$\{dede,eded\}$
$\{d,b\}$	$\{dbdb,bdbd\}$

On a donc $\mathcal{F}$ qui est l'union des chaines associées aux arêtes du VERTEX COVER, soit :

$$\mathcal{F} = \{acac,caca,afaf,fafa,adad,dada,agag,gaga,aeae,eaea,dcdc,cdcd,dede,eded,dbdb,bdbd\}$$

Exercice 6

Soit $m = |E|$ le nombre d'arêtes du graphe du VERTEX COVER, démontrons que :

$$G \text{ a une couverture des sommets de taille } k$$
$$\iff$$
$$\text{L'instance transformée de } G \text{ admet une superchaine de taille } 4m + k.$$

Intuition avec le VERTEX COVER de la FIG. 2 :

$$\mathcal{F} = \{acac, caca, afaf, fafa, adad, dada, agag, gaga, aeae, eaea, dcdc, cdcd, dede, eded, dbdb, bdbd\}$$

Par simple concaténation des éléments de $\mathcal{F}$, on obtient une chaîne de longueur $L = 2 * 4 * |E| = 2 * 4 * 8 = 64$.

Pour chaque arête, on peut associer les deux chaînes entre-elles. On a donc :

$$\mathcal{F}_1 = \{acaca, afafa, adada, agaga, aeaea, dcdcd, deded, dbdbd\}$$

Par simple concaténation des éléments de $\mathcal{F}_1$, on obtient une chaîne de longueur $L_1 = 5 * |E| = 5 * 8 = 40$.

On peut maintenant regrouper les chaînes qui ont la même première (et dernière) lettre ou dont leur autre forme équivalente respecte cette règle ({ababa} $\iff$ {babab}). On a donc :

$$\mathcal{F}_2 = \{acaca, afafa, adada, agaga, aeaea\} \cup \{dcdcd, deded, dbdbd\}$$

Chaque mot de $\mathcal{F}_2$ a 5 lettres. Si on concatène tous les mots au sein de chaque groupe, on obtient 4 lettres par arête initiale + une lettre représentant le sommet commun à toutes ces arêtes. On a donc $4 * |E|$ lettres + une par groupe avec k groupes, soit $4 * |E| + k$ lettres en tout.

Ainsi, on concatène les chaînes qui ont les mêmes premières et dernières lettres. On obtient donc :

$$\mathcal{F}_3 = \{acacafafadadagagaeaea, dcdcdededbdbd\}$$

Il ne reste plus qu'à concaténer les mots restants pour obtenir la plus petite superchaine possible ayant comme sous-mots l'ensemble des mots de départ :

$$S = acacafafadadagagaeaeadcdcdededbdbd$$

$$longueur(S) = 34 = 4 * 8 + 2 = 4 * m + k = 4 * |E| + k$$

On a donc démontré que G a une couverture des sommets de taille k si et seulement si l'instance transformée de G admet une superchaine de taille $4 * m + k$ (avec $m = |E|$).