Le trente-sixième chapitre de *Madame Bovary*

Votre programme est à rendre par courier électronique à `Luc.Maranget@inria.fr` avant le 11 janvier 2008, 23h59.

Trop tard : Solution.

Questions posées avant le 11 janvier et mes réponses disponibles ici.

Objectif

Suivre la technique « chaîne de Markov » décrite lors de l'amphi 04 pour faire écrire par un ordinateur un chapitre plausible de *Madame Bovary* de Gustave Flaubert (qu'il nous pardonne, notre œuvre n'arrivera pas à la cheville de la sienne).

Ce qu'il faut faire

Écrire le programme `Markov` qui produit, à la demande, des chapitres dans le style de *Madame Bovary*.

Pour un entier $n \geq 1$ donné, le texte produit est tel que :

- Toutes les sous-suites de $n+1$ mots sont aussi des sous-suites de $n+1$ mots du texte de *Madame Bovary*.
- En outre, les n premiers mots sont aussi les n premiers mots d'un chapitre de *Madame Bovary*.
- Et les n derniers mots sont aussi les n derniers mots d'un chapitre de *Madame Bovary*.

L'entier n est donné en argument de la ligne de commande. Ainsi, la commande

```
% java Markov 3
```

produit un texte plutôt étrange, par exemple[1] :

> Un soir que la fenêtre était ouverte, et que, assise au bord, elle venait de s'échapper dans le jardin. Emma, tout exprès, avait retiré la clef de sol, et s'occupait volontiers de littérature après son dîner, quand il ne jouait pas aux cartes. M. Homais le considérait pour son instruction ; madame Homais l'affectionnait pour sa complaisance, car souvent il accompagnait au jardin les petits Homais, marmots toujours barbouillés, fort mal élevés et quelque peu lymphatiques, comme leur mère. Ils avaient pour les soigner, outre la bonne, Justin, l'élève en pharmacie, un arrière-cousin de M. Homais et peut-être les pesanteurs du déjeuner, restait indécis et comme sous la fascination du pharmacien qui répétait :
>
> [...]

[1] Le texte est légèrement modifié pour des raisons de présentation typographique.

> – Oui, du capharnaüm ! La clef qui enferme les acides avec les alcalis caustiques ! Avoir été prendre une bassine de réserve ! Une bassine à couvercle ! et dont jamais peut-être je ne me sens pas en mon assiette ; il faudra même un de ces hommes d'ardeurs taciturnes qui travaillent la nuit dans les livres, et portent enfin, à soixante ans, quand vient l'âge des rhumatismes, une brochette de croix, sur leur habit noir, mal fait. Elle aurait voulu ne rien entendre, ne rien voir, afin de ne pas payer, d'emprunter, de souscrire des billets, puis de renouveler ces billets, qui s'enflaient à chaque échéance nouvelle, elle avait fini par préparer au sieur Lheureux était encore si longue, qu'il n'y fallait pas songer. D'ailleurs, imaginant qu'elle y mettait de la délicatesse, Charles insista davantage ; si bien qu'elle demeurait fort embarrassée dans sa velléité de sacrifice, lorsque l'apothicaire vint à propos lui fournir une occasion.

Le texte ci-dessus est de toute évidence dérivé de l'œuvre de Flaubert. Dérivé est d'ailleurs le mot juste, on perçoit un certain flottement dans l'écriture.

Plus sérieusement, on notera par exemple, que le chapitre numéro 28 commence par « *Un soir que [Charles l'écoutait,...]* ». La jolie sentence « `Emma, tout exprès, avait retiré la clef de sol, [...]` » n'est pas de Flaubert, mais Flaubert a écrit :

- *Emma, tout exprès, avait retiré la clef de la barrière, que Charles crut perdue.* Une manœuvre d'Emma pour faciliter ses rencontres avec Rodolphe, chap. 19[2].
- *Puis il possédait des talents, il peignait à l'aquarelle, savait lire la clef de sol, et s'occupait volontiers de littérature après son dîner, quand il ne jouait pas aux cartes.* Une description de Léon, le clerc de notaire, chap. 12.

Il est donc établi que « `retiré la clef de` » et « `la clef de sol,` » sont deux suites de $n+1 = 4$ mots qui permettent de relier la clef de la barrière à la clef de sol.

Algorithme

Ce qu'est exactement un mot est expliqué (si on peut dire) plus loin comme étant défini par la classe des lecteurs de mots fournie. Disons pour le moment qu'il existe, dans les textes fournis, un mot un peu spécial `<PAR>` qui indique les sauts de paragraphes. Il est également commode de considérer deux mots supplémentaires `<START>` et `<END>` qui marquent le début et la fin d'un chapitre. Notez que ces mots ne sont pas présents dans les textes fournis, et n'ont pas besoin de l'être.

L'algorithme emploie une table de hachage dont les clés sont des suites de n mots et les valeurs des multi-ensembles (ensembles avec répétitions possibles des éléments) de mots. Il se décompose en deux phases, construction de la table et production du texte.

Construction de la table

1. Pour chaque chapitre,
   (a) Soit P préfixe, noté w_1, w_2, ..., w_{n-1}, w_n et valant initialement `<START>`, `<START>`, ..., `<START>`, `<START>`.
   (b) Pour chaque mot w du chapitre en cours,
      i. Ajouter w aux mots associés à P,
      ii. Le préfixe P devient w_2, w_3, ... w_n, w.
   (c) Ajouter `<END>` aux mots associés à P.

[2] À la relire, je trouve cette phrase impressionnante : deux scènes en quinze mots.

Production du texte

1. Soit P un préfixe, valant initialement `<START>`, `<START>`, ..., `<START>`, `<START>`.
2. Boucle,
   (a) Choisir w parmi les mots associés à P, selon une loi uniforme.
   (b) Si w est `<END>`, alors terminer.
   (c) Afficher w.
   (d) Le préfixe P devient w_2, w_3, ... w_n, w.

Sans chercher à produire une présentation parfaite, l'affichage devra être lisible (espaces entre les mots, sauts de lignes avant et après le mot `<PAR>`).

Difficultés prévisibles

Un « *multi-ensemble* » de mots est facilement représenté soit comme une liste soit comme un tableau. Sans que cela soit un jugement définitif, on peut dire que la liste est plus appropriée dans la phase de construction, tandis que le tableau est plus approprié dans la phase de production du texte (pour choisir le $n+1$-ième mot, cf. le cours.) Une solution simple est de convertir les listes en tableaux entre les deux phases de l'algorithme.

On peut iterer sur les associations d'une `HashMap<K,V>` de la bibliothèque de la manière suivante :

```
HashMap<K,V> t = ...
for (K k : t.keySet()) {
  V v = t.get(k) ;
  I[k, v]
}
```

Le code ci-dessus exécute $I[k, v]$ pour toutes les associations $(k \to v)$ contenues dans la table `t`. Les derniers transparents de l'amphi 04 donnent quelques explications.

Enfin, souvenez-vous qu'une fois qu'une clé est rangée dans la table, cette clé *ne doit pas changer*.

Fichiers fournis

Les chapitres de *Madame Bovary*

Le texte de *Madame Bovary* est donné sous la forme de 35 fichiers (un par chapitre de `01.txt` à `35.txt`) dans l'archive `bovary.tar.gz` à désarchiver ainsi :

```
% tar zxf bovary.tar.gz
```

Vous disposez maintenant d'un sous-répertoire `bovary` qui contient les 35 chapitres. Le texte est soumis à cette licence.

Lecture des mots

Récupérez la classe des lecteurs des mots `WordReader` qui est conforme à la description donnée en cours. et compilez la.

Cette classe comprend entre autres, un constructeur **new** `WordReader (String filename)` qui prend un nom de fichier en argument ; ainsi qu'une méthode `main` d'essai :

```
% java WordReader bovary/01.txt
[Nous]
[étions]
[à]
[l'Etude,]
[quand]
[le]
[Proviseur]
[entra]
[suivi]
[d'un]
  ...
[<PAR>]
 ...
```

Outre les mots du premier chapitre écrit par Flaubert, vous voyez apparaître des mots `<PAR>` qui indiquent simplement les paragraphes. Ces mots `<PAR>` sont à traiter comme des mots ordinaires.

Interface de votre programme

Votre programme `Markov` doit obligatoirement se comporter ainsi.

- Il affiche son résultat sur la sortie standard (`System.out`) et seulement son résultat.
- Il lit la ligne de commande, à l'aide de la classe `Args` fournie.

  ```
  class Markov {
    ...
    public static void main (String [] arg) {
      Args a = new Args (arg) ;
      ...
    }
  }
  ```

  Dès, lors le comportement sera le suivant :

  – Si on invoque le programme avec un premier argument numérique, par exemple :

    ```
    % java Markov 4
    ```

    ou encore

    ```
    % java Markov 4 bovary/01.txt
    ```

    Alors `a.prefixLength` vaut la valeur de l'argument numérique. Sinon `a.prefixLength` a la valeur par défaut 2.

  – Les autres arguments se retrouvent dans le tableau `a.files`. Par exemple avec :

    ```
    % java Markov petit.txt
    ```

    le tableau `a.files` est `{ "petit.txt", }` (tableau à un élément). Cette disposition vous permet de tester votre programme sur de petits exemples, si `petit.txt` est le fichier suivant.

    ```
    a b c
    a b c
    a b c
    ```

    Alors le programme doit afficher « `a b c` » un certain nombre de fois.

    ```
    % java Markov petit.txt
    a b c a b c a b c a b c
    ```

Si il n'y pas d'arguments autres qu'un éventuel premier argument numérique, alors `a.files` a une valeur par défaut qui correspond aux 35 chapitres contenus dans le sous-répertoire `bovary`.