Algorithmique TP 20 : Justification d'un texte en lignes

Le problème du formatage d'un texte en lignes[1], sous sa forme la plus simple, peut se formuler de la manière suivante : soit une suite de mots m_0,...,m_{n-1} que l'on désire découper en lignes de longueur L. On note $l(m)$ la longueur d'un mot m. Les mots sont séparés par des espaces dont la taille idéale est e, mais qui peuvent être étirés ou comprimés au besoin (sans toutefois provoquer de chevauchement de mots), de manière que la ligne $m_i m_{i+1} \ldots m_j$ ait exactement la longueur L. Cependant, on attribue une pénalité d'étirement ou de compression égale à la valeur absolue totale des étirements ou des compressions subies par les blancs de la ligne. Autrement dit, le coût de la mise en forme de la ligne $m_i m_{i+1} \ldots m_j$ est $|e' - e|$ où e', la longueur réelle des espaces, est $\frac{(L-l(m_i)-\ldots-l(m_j))}{(j-i)}$. Cependant, si $j = n-1$ (on a atteint la dernière ligne), le coût est nul à moins que $e' < e$, puisqu'on n'a pas à jouer sur l'elasticité des espaces de cette dernière ligne. On veut formater un texte de façon à avoir un coût minimal.

Un algorithme dynamique de résolution de ce problème de formatage peut s'exprimer comme suit.

$c[i] := \infty$ pour $i = 0, \ldots, n$
$c[n] := 0$
Pour i entre $n-1$ et 0
  $c[i] := \min_{j=i \ldots n-1}(cout(i,j) + c[j+1])$
  $d[i] :=$ le j correspondant au min précédent
i := 0
TantQue $i < n$
  Ecrire $m_i \ldots m_{d[i]}$
  $i := d[i] + 1$

où $c[i]$ est le coût minimum pour écrire les mots $m_i \ldots m_{n-1}$ (avec m_i en début de ligne), $d[i]$ est l'indice du dernier mot de la première ligne pour ce coût minimum et $cout(i,j)$ est le coût de l'écriture des mots $m_i \ldots m_j$ sur une même ligne (éventuellement ∞).

1. En utilisant le module `Lib20` (`lib20.mli`, `lib20.cmi` et `lib20.cmo` sous `/usr/local/serveur/PROG/algo`) implanter l'algorithme de formatage et afficher dans la fenêtre graphique. L'interface du module `lib20` est la suivante :

`lib20.mli`

```
val infini : int
val espace : int
val longueur_mots : string array -> int -> int -> int
val cout_ligne : string array -> int -> int -> int -> int
val exemple : string array
exception Too_much
```

où
- `espace` est la valeur de l'espace idéal (e) entre deux mots ;
- `longueur_mots mots i j` calcule la somme des longueurs des mots d'indice `i`,`i+1`,...`j` du tableau `mots` ;
- `cout_ligne mots i j largeur` est le coût de formatage des mots d'indice `i`,`i+1`,...`j` du tableau `mots` dans une ligne de largeur `largeur` ; l'exception `Too_much` est levée si les mots ne tiennent pas sur la ligne ;
- `exemple` est un exemple de phrase à formater.

La fonction de formatage prend en argument un vecteur de mots et la largeur de la ligne :

```
justifie : string array -> int -> unit
```

Pour utiliser la fenêtre graphique (voir `The graphics library` dans la documentation), on devra utiliser les fonctions suivantes :

[1] D'après Structure de données et algorithmes, A. Aho, J. Hopcroft & J. Ullman, page 352

- `open_graph: string -> unit` (à appeler avec l'argument `""`) pour ouvrir la fenêtre graphique ;
- `moveto: int -> int -> unit` pour se déplacer à une position voulue :
- `draw_string string -> unit` pour afficher une chaîne de caractère à la position courante.

Pour compiler, utiliser la commande suivante :

```
ocamlc -o tp20 graphics.cma -I /usr/local/serveur/PROG/algo lib20.cmo tp20.ml
```