TP Algorithmique et Optimisation Discrète

En informatique, un *patch* est une suite d'instructions de transformation appliquées à un flot en entrée (par exemple un fichier) et dont le résultat est écrit sur le flot de sortie. Le flot est ici traité ligne par ligne, de la première à la dernière[1]. Une ligne se termine par le caractère NEWLINE (`\n`). Chaque ligne a un numéro à partir de 1 pour la première ligne ; le début du fichier (avant la première ligne) correspond par convention à une ligne virtuelle de numéro 0.

Un patch est une suite d'instructions, chacune écrite sur une ou deux lignes. La première ligne d'une instruction de patch est composée d'un caractère définissant l'instruction (`+`, `=`, `d` ou `D`), suivi d'un caractère espace puis d'un entier $k \geq 0$ indiquant le numéro de la ligne du flot d'entrée affectée par l'instruction. Cette première ligne de l'instruction peut être :

"`+` k`\n`" *Ajout* : la ligne suivante dans le patch est ajoutée sur le flot de sortie après la ligne k du flot d'entrée. Si $k = 0$, la ligne est insérée avant la première ligne (numérotée 1) du flot d'entrée.

"`=` k`\n`" *Substitution* : la ligne k du flot d'entrée est remplacée sur le flot de sortie par la ligne suivante dans le patch.

"`d` k`\n`" *Destruction* : la ligne k du flot d'entrée n'est pas recopiée sur le flot de sortie.

"`D` k m`\n`" *Multi-destruction* : les m lignes de la ligne k à la ligne $k+m-1$ du flot d'entrée ne sont pas recopiées sur le flot de sortie.

Le fonctionnement est le suivant :

- Toutes les lignes du flot d'entrée qui ne sont pas détruites et dont le numéro n'apparait pas dans une instruction de substitution sont recopiées sur le flot de sortie.
- Les numéros de lignes k dans les instructions du patch apparaissent par ordre croissant.
- Les instructions sont exécutées dans l'ordre du début à la fin ; ainsi deux commandes d'ajout consécutives avec un même numéro de ligne entraînent l'ajout consécutif des deux lignes opérandes dans l'ordre sur la sortie.
- Une instruction de substitution a un numéro de ligne strictement supérieur à l'instruction qui la précède.
- Toutes les instructions qui suivent une instruction de destruction "`d` k" (resp. "`D` k m") ont un numéro de ligne supérieur ou égal à $k+1$ (resp. $k+m$). De plus, une instruction de destruction (`d` ou `D`) ne peut pas porter sur la même ligne qu'une instruction de substitution.

La commande fournie `applyPatch` écrit sur la sortie standard le résultat de l'application du patch *patch_file* au fichier en lecture *original_file* :

`applyPatch` *patch_file original_file*

Côut d'un patch. Le coût d'une instruction de patch dont la première ligne commence par "`+`" ou "`=`" est 10+s, où s est le nombre de caractères de la ligne qui suit dans le patch (en incluant `\n`). Le coût d'une instruction de destruction commençant par "`d`" (resp. "`D`") est 10 (resp. 15). Le coût d'un patch est la somme des coûts de ses instructions.

Exemple. Pour le patch P (listing 1) et le fichier F (listing 2), la commande : `applyPatch P F` écrit sur la sortie standard les lignes du listing 3. Le patch P est de coût 67.

```
1 + 0
2 @#?!
3 + 1
4 u
5 = 3
6 v
7 D 4 2
8 + 6
9 ww
```

Listing 1 – Fichier `P` de patch.

```
1 a b
2 ccc
3 d
4 ee
5 ff
6 g
```

Listing 2 – Fichier `F` d'entrée.

```
1 @#?!
2 a b
3 u
4 ccc
5 v
6 g
7 ww
```

Listing 3 – Sortie de `applyPatch P F`

1. La spécification simplifiée dans ce TP diffère des commandes `diff` et `patch` de Unix, mais le principe est similaire.