𐤌𐤀𐤉𐤏𐤎

Indivil i Mandoni volen deixar constància de les seves possessions en una sèrie de taules de plom. El problema és que aquestes poden patir petites deformacions i alterar el text. Per a evitar-ho, ens demanen si podem idear un sistema per a verificar que el text no s'ha modificat, i a ser possible poder-lo corregir. Aquest sistema ha de ser ràpid, que no costi gaire de computar.

Exemple

Per exemple, donat el text original "4E3IF7U VL9LW PQKWT MREPJWMTEUB FDNFL", i el text rebut "4E3IF7U VL9LW PQKWT MREPJWMTEUB FDNFT", ha de detectar que hi ha hagut una modificació, i si és possible, corregir-lo.

Requeriments

Es demana realitzar:

- Un programa que llegeixi les dades d'un fitxer de text amb majúscules i dígits i afegeixi uns caràcters (majúscules o dígits) per verificar el congingut, i un altre programa que retorni si aquest text ha estat modificat. El mètode de detecció ha de tenir un cost de $O(n)$, i el de correcció de $O(n^2)$, on n és la llargada del text. La codificació haurà de superar el test del `makefile`.
- Un informe que contingui, sobre els algorismes principals: pseudo-codi en disseny recursiu i iteratiu, cost teòric i experimental.

Arguments i parametres

L'execuci'o de l'aplicació haurà de seguir la següent sintaxi:

```
$./senderscribe <fitxer text enviat> <fitxer text rebut>
$./receiverscribe <fitxer text rebut> <fitxer resultat>
```

Els arguments opcionals del programa són els següents:

fitxer text enviat : Fitxer sense codificar.

fitxer text rebut : Fitxer codificat segons el vostre mètode.

fitxer resultat : Fitxer on direm si hi ha hagut error en l'enviament (`KO`) o no n'hi ha hagut (`OK`). Si es fa una correcció, s'indicarà la posició i el caràcter que hi pertoca.

Si no hi ha arguments opcionals, l'aplicació llegeix de l'entrada estàndard i escriu a la sortida estàndard.

Exemple

Si en el fitxer `text.txt` posem el text de l'exemple anterior:

```
$./senderscribe text.txt enviat.txt
$./receiverscribe enviat.txt resultat.txt
$ cat resultat.txt
KO
33 L
```

Avaluació

En l'informe de la pràctica, els algorismes principals, les taules i les gràfiques han d'estar comentats.

Els programes han de compilar sense errors ni warnings i passar els tests amb `make test`, altrament no s'avaluarà. Per passar el test de detecció de canvis, haurà de detectar canvis com a mínim del 25% del text, i el corrector haurà de corregir canvis de com a mínim el 15% del text.

La baremació de la pràctica (sobre 10) serà la següent:

Costos 3 punts, anàlisi de costos teòrics de l'algorisme principal:

- 1 punt, recursiu
- 1 punt, iteratiu
- 1 punt, empírics dels algorismes;

Disseny 3 punts, recursiu i iteratiu. Es tindrà en compte la dificultat, la capacitat de detecció i correcció, i el cost de l'algorisme;

Implementació 4 punts

- 1 punt, iteratiu en python (opcionalment en C)
- 1 punt, recursiu en python. La recursivitat només s'ha d'implementar en l'algorisme principal.
- 1 punt, bones pràctiques de programació (`pylint`)
- 1 punt, utilització dels recursos del llenguatge de programació.

Enviament

L'assignació és per parelles, i representa un 20% de la nota final. Presenteu la pràctica al Campus Virtual de la UdL amb dos fitxers: un pdf per a l'informe, i un arxiu comprimit, `tgz` o `zip` (no s'admeten altres formats de compressió), per al codi font.

Els programes que s'hagin de compilar ho han de fer amb:

```
$ make
```

Els programes es testejen fent:

```
$ make test
```

La pràctica podria ser verificada individualment el primer dia de laboratori després de l'entrega.