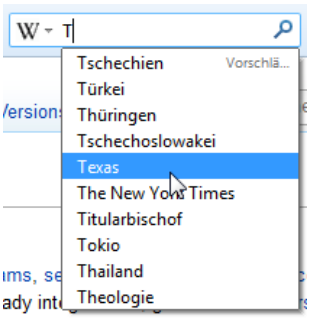
EDA (ETS d'Enginyeria Informàtica). Curs 2020-2021

Pràctica 4. Implementació de la funció *Autocompletar text* amb un ABB equilibrat (1 sessió)

Departament de Sistemes Informàtics i Computació. Universitat Politècnica de València

1. Descripció del problema

L'autocompletat de paraules és una característica que tenen moltes aplicacions que treballen amb text i que consisteix a predir la resta d'una paraula a partir del prefix que l'usuari escriu. El sistema suggereix una llista d'opcions i l'usuari pot triar la correcta; així es facilita la interacció de l'usuari amb el sistema.



Per a implementar aquesta funcionalitat es té un diccionari de paraules, que podria incloure informació sobre la freqüència d'ús d'aquestes; aquest diccionari es podria actualitzar per a adaptar-se a l'usuari afegint o eliminant termes. En aquesta pràctica aquest diccionari es representarà com un Arbre Binari de Cerca (*ABC*) de paraules i, donat el conjunt de caràcters escrits per l'usuari (prefix), la llista de suggeriments de completat serà la seqüència ordenada de paraules que comencen per aquest prefix. Per a això s'utilitzarà de forma repetida el mètode `sucesor(e)` d'un `ABB<E>` que permet obtindre l'element del `ABC` següent a `e` segons l'ordre establit en `E`. El cost d'aquesta operació, així com els de les operacions de cerca, inserció i esborrat en un ABC depenen de la topologia d'aquest. Només un ABC equilibrat permet realitzar aquestes operacions en temps logarítmics amb la talla del ABC.

Així, els objectius d'aquesta pràctica són els següents:

- Construir de manera eficient un ABC equilibrat a partir d'un conjunt de dades, així com re-equilibrar un ja existent quan s'ha desequilibrat després de diverses operacions d'insercions i/o esborrats.
- Implementar la funció autocompletar d'un *Editor Predictiu* utilitzant un ABC equilibrat.

2. Implementació eficient d'un `ABC`

Com s'ha comentat, la implementació que es faça de la classe `ABB` ha d'assegurar que els ABC que es construïsquen tinguen altures quasi logarítmiques. Atés que la topologia del ABC depén de l'ordre en el qual s'afigen les seues dades, la solució que es proposa per a aconseguir altures òptimes es basa a afegir les dades en l'ordre apropiat.

Activitat #1: El paquet `jerarquicos`

L'alumne ha de crear el subpaquete *librerias.estructurasDeDatos.jerarquicos* i afegir les classes `ABB`, `NodoABB` i `GUIAbb`, disponibles en PoliformaT.

Observe's que el mètode de recorregut per nivells de la classe `ABB` utilitza una cua per a emmagatzemar els elements durant la seua execució. Per això, és necessari incloure i compilar la interfície `Cola` en el paquet *librerias/estructurasDeDatos/modelos* i la seua implementació `ArrayCola` en *librerias/estructurasDeDatos/lineales*.

Activitat #2: Implementació dels mètodes per a construir el ABC equilibrat

Com s'ha comentat, la forma que té un ABC depén de l'ordre d'inserció dels elements en ell. Si l'ordre en el qual s'insereixen les dades és l'adequat, la relació altura nombre de nodes serà òptima. Una manera d'aconseguir això és la següent:

- Col·locar les dades que s'afegiran al ABC en un array `v` en ordre no decreixent.
- Recursivamente, fins que no queden dades per tractar:
  - Inserir `v[meitat]`, la mitjana, com a node arrel de l'arbre.
  - Construeix el seu fill esquerre de la mateixa manera a partir de les dades situades en l'array a l'esquerra de `meitat`.
  - Construeix el seu fill dret de la mateixa manera a partir de les dades situades en l'array a la dreta de `meitat`.

A partir d'aquesta idea, s'han de completar els següents mètodes de la classe `ABB`:

1. `construirEquilibrado`: donat un subarray `v[ini,fi]` ordenat de forma no decreixent, retorna l'arrel del ABC equilibrat amb els elements de dita subarray:

```
protected NodoABB<E> construirEquilibrado(E[] v, int ini, int fin) { ... }
```

2. `ABB`: constructor que crea un ABB equilibrat amb els elements de l'array que es passa com a paràmetre usant el mètode `construirEquilibrado`:

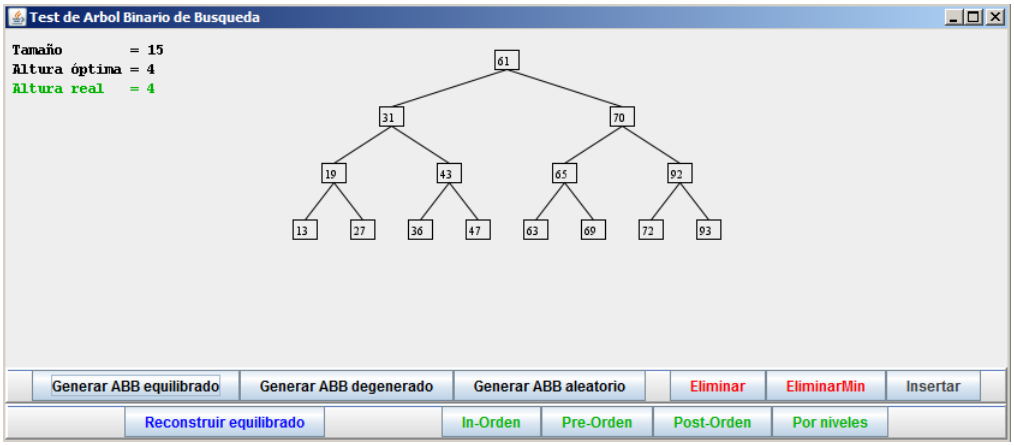
```
public ABB(E[] v) { ... }
```

3. `reconstruirEquilibrado`: que reequilibra `this` ABB utilitzant els mètodes `inOrden` i `construirEquilibrado`:

```
public void reconstruirEquilibrado() { ... }
```

Activitat #3: Validació de la classe `ABB`

Executa el programa `GUIAbb` per a comprovar el funcionament dels mètodes dissenyats. En concret, usa els botons `Generar ABC equlibrat` i `Generar ABB aleatori` i després `Reconstruir equilibrat`. En la figura següent es mostra l'execució d'aquest programa una vegada seleccionat el botó `Generar ABC equlibrat`. També pots observar el funcionament dels mètodes d'inserció, esborrat d'elements i esborrat del mínim en un ABC, així com comprovar el resultat dels recorreguts que poden realitzar-se sobre ell (per nivells, in-ordre, pre-ordre i post-ordre).

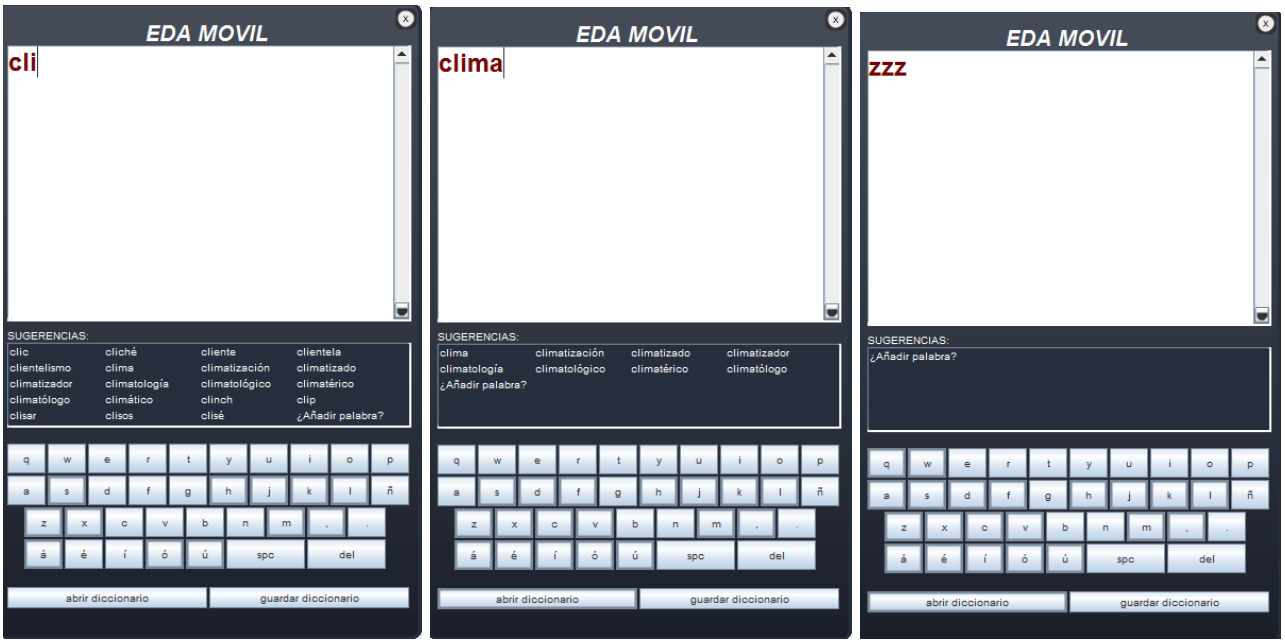


3. Implementació de la funció *Autocompletar palabras*

La funció d'autocompletat de paraules ha d'oferir una sèrie de suggeriments per a completar la paraula conforme l'usuari la va escrivint, sense que haja necessàriament d'acabar d'escriure-la. Per a implementar aquesta funcionalitat s'ha dissenyat la classe `EditorPredictivo` com un `ABB<String>`; en concret, tindrà la següent estructura:

```
public class EditorPredictivo extends ABB<String> {
  public EditorPredictivo() { ... }
  public EditorPredictivo(String nomFitxer) { ... }
  public void incluir(String nova) { ... }
  public void guardar(String nomFitxer) { ... }
  public ListaConPI<String> recuperarSucesores(String prefixe, int n) { ... }
}
```

Aquesta classe serà la utilitzada per la interfície gràfica `GUIEditorPredictivo`; en la figura següent es mostra l'execució aquesta interfície per a tres entrades: `cli`, `clima` i `zzz`. En l'últim cas, la llista està buida perquè no hi ha ningúna paraula que tinga com a prefix zzz. Aquesta interfície permet també, obrir un diccionari diferent o afegir noves paraules al diccionari i guardar-lo.



Activitat #4: El paquet `editorPredictivo`

Crear el paquet *aplicaciones/editorPredictivo* i afegir les classes `EditorPredictivo` i `GUIEditorPredictivo` en aquest paquet. Així mateix, l'alumne haurà de copiar el fitxer `castellano.txt`, disponible també en PoliformaT, en el directori associat al paquet *aplicaciones/editorPredictivo*; aquest fitxer conté més de 460000 paraules en castellà ordenades alfabèticament (les dades amb els quals, per defecte, es construirà un Editor Predictiu).

Note's que el segon mètode constructor de la classe `EditorPredictivo`, donat un fitxer de paraules ordenat ascendentment, crea un `ABC` equilibrat de `String` fent ús del mètode *construirEquilibrado* per a inserir les paraules de l'array, obtenint així un ABC equilibrat.

Activitat #5: El mètode `recuperarSucesores` de la classe `EditorPredictivo`

Completar el mètode *recuperarSucesores* de la classe `EditorPredictivo`, amb el perfil següent:

```
public ListaConPI<String> recuperarSucesores(String prefijo, int n)
```

Aquest mètode retorna una Llista Amb Punt d'Interés amb els `n` primers successors d'un `prefix` donat; recorde's que aquesta llista ha d'incloure a `prefix` com a primer element de la llista sempre que aquest ja figure en el ABC (és a dir, que el prefix siga ja una paraula completa). Per a implementar aquest mètode es pot fer ús del mètode `sucesor` de la classe `ABC` com segueix:

1. Buscar el prefix en l'`ABC`, per si aquest fóra ja una paraula completa.
2. Recuperar del `ABC` els següents successors del prefix fins a trobar un que ja no comence per aquest prefix o s'hagen afegit n successors.

Activitat #6: Prova de l'Editor Predictiu

Comprova que el codi que has escrit utilitzant la interfície gràfica `GUIEditorPredictivo`. A més de provar amb els exemples de la figura anterior, a continuació es mostren els suggeriments que han d'aparéixer per a una sèrie de prefixos de prova:

Prefijo	**Sugerencias**			
catar	catar catarroso	catarata catarsis	catarral	catarro
mar	mar maraco maraquero maratón maravilloso	mara maracucho marar maravilla maraña	marabunta maracuyá marasmo maravillar marañero	maraca maraquear maratoniano maravillosamente
tene	tenebrosidad tenencia tenería	tenebroso tener	tenedor teneraje	teneduría tenerife
criti	criticable criticidad	criticador criticón	criticar critiquizar	criticastro