PRACTICA V: ABSTRACCIóN DE DATOS

Vector Disperso)

26 de noviembre de 2014

1. Objetivo

El objetivo de esta práctica es que el alumno/a comprenda la importancia de la abstracción de datos y se familiarice con las herramientas que C++ proporciona para llevarlos a cabo.

2. Introducción: Tipo de Dato Vector Disperso

Un vector disperso es un vector en el que la mayoría de los elementos tienen el mismo valor (conocido como valor por defecto). La presencia de este tipo de valores acarrea inconvenientes desde el punto de vista del almacenamiento (un mismo elemento se repite un número considerable de veces) como desde el punto de vista de la eficiciencia (pues distintos algoritmos pueden aprovechar este hecho).

Un vector se dice que es disperso cuando en él sólo se almacena los elemenos que son distintos del valor nulo.

Ejemplo: Supongamos que tenemos el conjunto de lemas del castellano (fichero lema.txt) almacenados en un vector, eso si, ordenado alfabéticamente, `Lemas`. Sea V un vector de enteros donde V[i] representa el numero de ocurrencias del i-ésimo lema (`Lemas[i]`) en un párrafo de un texto (por ejemplo, el Quijote.txt). Este vector tendrá una dimensión de 85918 (número total de lemas distintos) y se podría declarar como

```
vector<int> V(85918,0):
```

Obviamente el número de palabras en un párrafo es mucho menor que el numero de posibles lemas. Por ejemplo, el segundo parrafo de esta sección tiene un total de 19 palabras distintas, y por tanto, sólo 19 posiciones del

vector V tomarán un valor distinto de 0. Almacenar toda la información en el vector puede ser muy costoso en

- espacio: pues almacenaríamos 84999 ceros.
- tiempo: para contar el numero total de palabras del párrafo, tendríamos que iterar por las 85918 posiciones y acumular aquellas que son distintas de cero.

Una solución consiste en utilizar un tipo de dato abstracto que nos permita trabajar con este tipo de información de forma eficiente. En cualquier caso, debemos de dotar al vector disperso de un interfaz similar a la que ofrece la clase vector. En concreto en esta práctica proponemos que el vector disperso esté dotado de los métodos que se indican en la sección 2.1.

Así, por ejemplo el vector de nuestro ejemplo se podría declarar como

```
vectorD V(85918,0):
```

y podríamos indicar que el lema 1546 ha ocurrido 12 veces en el párrafo o sumar todas las ocurrencias mediante

```
V.assign(1546,12);
...
for (int i=0; i<V.size(); i++)
  suma+=V[i];
```

Sin embargo, este vector NO almacenaría todos los posibles valores, sólo los valores no nulos. Para ello, es suficiente con considerar para cada valor no nulo la posición en la que se encuentra y el valor asociado. En la sección 3 propondremos una posible representación para el vector disperso, basada en la clase list de la STL.

2.1. Especificación

Ver la especificación en el fichero vectorD.h

3. Representación

Para la representación se propone, usando la clase list de la STL,

- `list< pair<int,TIPO> > vd; // vector que aloja los valores no nulos`
- `int n_ele; // numero de elementos totales`
- `TIPO v_nulo; //valor nulo del vector`

La función de abstracción y el invariante de la representación asociadas a esta representación son:

3.1. Función de Abstracción

```
/* FA: rep -- > vector
  FA(V) dado un vectorD V (vd,n_ele,v_nulo) con
  vd=[ (a,v1), (b,v2), ..., (n,vn) ]
  n_ele = M
  v_nulo = t
  ---------->  vector M, tal que M[pos]
     pos: 0 1 2 ... a-1 .a ... x .... b....  ... n-1  n n+1 ..... M-1
  M[pos]: t t t ....t   v1 ... t .... v2 ..  ... t   vn  t  .....  t
  */
```

3.2. Invariante de la Representación

```
/* IR : rep ----> bool
 IR(V) dado un vectorD V (vd,n_ele,v_nulo) se satisface que:
  0 <= vd.size() < n_ele;
  vd[i].second != v_nulo, para todo i = 0, ..., vd.size()-1;
  vd[i].first >=0, para todo i = 0, ..., vd.size()-1;
  vd[i].first < vd[j].first si i<j
 */
```

Para chequear el invariante de la representación podemos utilizar la siguiente función

4. Se entrega.

- vectorD.h: Especificación + Representación
- ejemplo.cpp: Fichero de prueba del vectorD

4.1. Se pide:

- Implementar la clase vector disperso, junto con la documentación asociada con Doxygen.
- Discutir e implementar los métodos de la clase vectorD cuando utilizamos un `map<int,T>` como su representación interna.
- Implementar un tda polinomio, donde como representación interna consideremos el tipo vector disperso. Dicho tipo polinomio debe tener entre otros los siguientes métodos:

1. Constructores
2. Consultores
   - grado del polinomio
   - coeficiente
3. Modificadores
   - asignar coeficiente y grado a un determinado monomio
4. Operadores
   - operator=
   - operator+
   - operator-
5. Iteradores