Algoritmos y Estructuras de Datos.
TPL3. Trábajo Práctico de Laboratorio 3. [2013-11-09]

PASSWD PARA EL ZIP: `FW4R92XA5BK2`

Instrucciones

- El examen consiste en que escriban las funciones descriptas más abajo; impleméntandolas en C++ de tal forma que el código que escriban **compile y corra correctamente**, es decir, no se aceptará un código que de algún error de compilación o que tire alguna excepción/señal de interrupción en runtime. Básicamente se hace una evaluación de caja negra, aunque le daremos un rápido vistazo al código.
- Pueden utilizar todas las funciones y utilidades del estándar de C++ que por supuesto contiene a la librería STL.
- Se incluye un template llamado **`program.cpp`**. En principio sólo tienen que escribir el cuerpo de las tres funciones pedidas. El paquete ya incluye el header **`tree.h`**.
- También se incluyen con el paquete la clase de arbol binario **`btree.h`**.
- Algunas funciones utilitarias que pueden servir

```
void print(set<int> &S);
void print(list<set<int> > &LS);
void print(vector<set<int> > &VS);
T.lisp_print();
```

Ejercicios

[Ej. 1] **[gatherset]** Dado una serie de conjuntos de enteros S_j, con $j \in [0, N_S)$ juntarlos entre sí aquellos que tienen al menos un elemento en común. Es decir debemos encontrar otro grupo de conjuntos W_j, con $j \in [0, N_W)$ tal que

- Los W_j son disjuntos entre sí, esto es $W_j \cap W_k = \emptyset$ si $j \neq k$
- Dos enteros están en el mismo W_k (sólo en uno porque son disjuntos) si y sólo si comparten alguno de los S_j.

Por ejemplo, si **`S0={5,7,11}`**, **`S1={0,1,3}`**, **`S2={1,2,10}`**, y **`S3={7,13,22}`**. Entonces tenemos **`W0={5,7,11,13,22}`**, **`W1={0,1,2,3,10}`**.

Consigna: Escribir una función
`void gatherset(list< set<int<> > &S, list< set<int<> > &W);` que realiza la tarea descripta.

Ayuda:

- Inicializar **`W`** como una lista vacía.
- Mientras que **`S`** no este vacío):
  - Hacer **`tmp=`** algún elemento de **`S`** y eliminarlo de **`S`**
  - Para cada **`w`** en **`W`**:
    - Si la intersección de **`w`** y **`tmp`** no es nula
      - Hacer **`tmp=tmp`** $\cup$ **`w`**.
      - Eliminar **`w`** de **`W`**.

- Insertar **`tmp`** en **`W`**

[Ej. 2] **[maxshare]** Dado una serie de conjuntos de enteros S_j, con $j \in [0, N_S)$ y otro conjunto W encontrar aquel S_k cuya intersección con W tiene el máximo tamaño, es decir el conjunto tal que

$$|S_k \cap W| \leq |S_j \cap W|, \ \forall j \in [0, N_s)$$

donde $|\ .\ |$ denota el tamaño del conjunto. Por ejemplo, si **`W={1,3,5}`** y **`S0={0,2,4,8}`**, **`S1={0,1,4,9}`**, **`S2={5,10}`**, y **`S3={1,3,6}`**. Entonces tenemos que el elegido debe ser **`S3={1,3,6}`** ya que el tamaño de la intersección con **`W`** es 2 y para los otros es menor.

Consigna: Escribir una función **`set<int> maxshare(vector< set<int> > &S, set<int> &W);`** que realiza la tarea descripta. Si hay varios conjuntos que tienen el misma tamaño de intersección debe retornar el primero de ellos.

Ayuda: Ir recorriendo **`S`** y mantener unas variables auxiliares **`set<int> Smax; int nsmax;`** que son el conjunto de **`S`** que tiene la máxima intersección hasta el momento, y el tamaño de su intersección con **`W`**. Para cada conjunto de **`S`** calcular la intersección con **`W`**, y si es estrictamente mayor que **`nsmax`** reemplazar los mínimos actuales.

[Ej. 3] **[sccount]** Dado una árbol binario (AB) **`T`**, determinar cuantos nodos de **`T`** tienen exactamente un solo hijo (*single child count*).

Ejemplos:
Para **`T=(8 (7 9 2) (3 . (9 . 1)));`** debe retornar **`2`**
Para **`T=(2 8 (1 (2 . 6) .));`** debe retornar **`2`**
Para **`T=(4 (9 (2 6 .) .) 8);`** debe retornar **`2`**
Para **`T=(8 (6 (5 . (4 3 (4 . (9 . 6)))) .) 3);`** debe retornar **`4`**

Definición recursiva:

$$\texttt{sccount}(n) = \begin{cases} 0; & \text{si } n = \Lambda, \\ s + \texttt{sccount}(l) + \texttt{sccount}(r); & \text{caso contrario,} \end{cases} \quad (1)$$

donde $s = 1$, si n tiene exactamente un sólo hijo y 0 en caso contrario. l, r son los hijos izquierdo y derecho de n.

Consigna: Escribir una función **`int sccount(btree<int> &T);`** que realiza la tarea descripta.

Curiosidad: Coincide con la cantidad de puntos que aparecen en notación Lisp, pero no recomendamos encarar esa estrategia.