Apellido y Nombre: _____

Carrera: _____ DNI: _____

[Llenar con letra mayúscula de imprenta GRANDE]

Universidad Nacional del Litoral
Facultad de Ingeniería y Ciencias Hídricas
Departamento de Informática

Algoritmos y Estructuras de Datos

Algoritmos y Estructuras de Datos.
3er Parcial. [27 de Noviembre de 2009]

ATENCIÓN: Para aprobar deben obtener un **puntaje mínimo** del 50 % en clases (Ej 1), 25 % en operativos (Ej 3) y un 60 % sobre las preguntas de teoría (Ej 4).

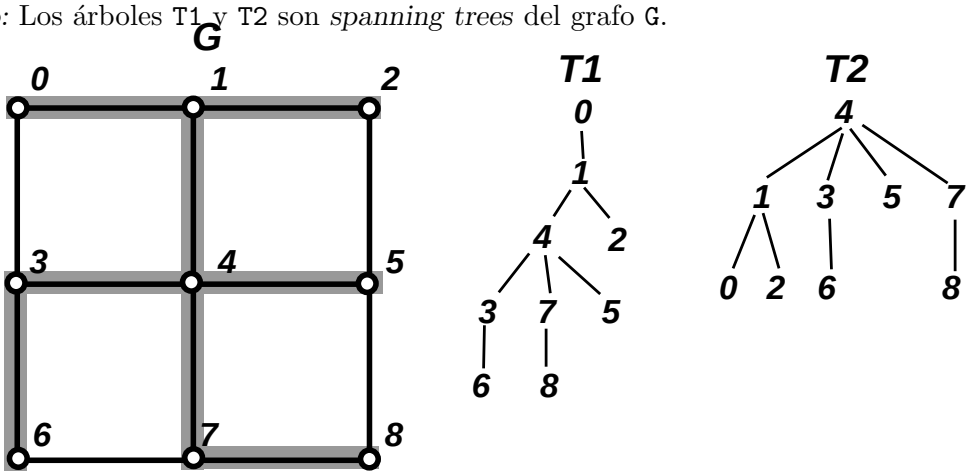
[Ej. 1] [clases (20pt)]

a) Implementar `set::iterator set::find(T x)` para conjuntos implementados por ABB. **No es necesario** escribir las declaraciones auxiliares de los miembros privados de la clase.
b) Implementar `int set::erase(T x)` para conjuntos implementados por listas ordenadas. Si se utiliza algún método auxiliar, también implementarlo. **No es necesario** escribir las declaraciones auxiliares de los miembros privados de la clase.
c) Implementar una función `bool openhashtable_insert(vector<list<T> >& table, unsigned int (*hashfunc)(T), T x)` que inserta el elemento `x` en la tabla de dispersión abierta `table` utilizando la función de dispersión `hashfunc` y retorna un booleano indicando si la inserción fue o no exitosa.
d) implementar una función `void vecbit_difference(vector<bool>&, A, vector<bool>& B, vector<bool>& C);` que devuelve `C=A-B` con `A,B,C` vectores de bits que representan conjuntos de un rango contiguo de enteros `[0, N)`, donde `N` es el tamaño de los vectores `A,B,C` (asumir el mismo tamaño para los tres, es decir, haciendo `int N=A.size();` es suficiente).

[Ej. 2] [programación (total 40pt)] Dado un grafo conexo `map<int, set<int> > G` y un árbol AOO `tree<int> T` decimos que `T` es un **spanning tree** de `G` si, se puede llegar desde

- El conjunto de nodos de `T` es igual al conjunto de vértices de `G`.
- Los nodos de `T` no están repetidos.
- Los caminos de `T` son caminos en el grafo y por lo tanto permiten llegar desde la raíz a cualquier otro nodo.

Podemos pensar a `T` como un subgrafo de `G`, pero que cumple las propiedades para ser un árbol.
Ejemplo: Los árboles `T1` y `T2` son *spanning trees* del grafo `G`.



Consigna: Escribir una función `bool is_spng_tree(map<int,set<int>> &G,tree<int> &T);` que retorna `true` sii `T` es *spanning tree* de `G`.
Ayuda:

a) [**tree-nodeset (10pt)**] Escribir una función `bool tree_nodeset(tree<int> &T,set<int> &nodeset);` que retorna `true` si todos los nodos de `T` son distintos y si ese es el caso retorna por `nodeset` el conjunto de nodos de `T`. *Ayuda:* Probablemente esto requiere hacerlo en forma recursiva sobre el árbol.

b) [**graph-nodeset (10pt)**] Escribir una función `void graph_vrtxset(map<int,set<int>> &G,set<int> &vrtxset);` que retorna por `vrtxset` el conjunto de vértices de `G`.

c) [**isspngtree (20pt)**] Con la ayuda de las dos funciones auxiliares previas:

- Comprobar que los nodos de `T` son únicos.
- Comprobar que el conjunto de nodos de `T` es igual al de vértices de `G`.
- Verificar que las *aristas* de `T` (es decir los pares padre-hijo) están contenidos en `G`. (**Ayuda:** escribir una función recursiva sobre el árbol).

[Ej. 3] [operativos (total 20pt)]

a) [**abb (5 pts)**] Dados los enteros $\{17, 11, 24, 6, 7, 14, 9, 8, 5, 16\}$ insertarlos, en ese orden, en un *"árbol binario de búsqueda"*. Mostrar las operaciones necesarias para eliminar los elementos 17, 9 y 6 en ese orden.

b) [**hash-dict (5 pts)**] Insertar los números $4, 20, 30, 13, 12, 40, 22, 9$ en una tabla de dispersión cerrada con $B = 8$ cubetas, con función de dispersión $h(x) = x$ `mod` 8 y estrategia de redispersión lineal.

c) [**heap-sort (5 pts)**] Dados los enteros $\{5, 9, 12, 6, 7, 17, 14\}$ ordenarlos por el método de *"montículos"* (*"heap-sort"*). Mostrar el montículo (minimal) antes y después de cada inserción/supresión.

d) [**quick-sort (5 pts)**] Dados los enteros $\{9, 13, 8, 5, 9, 14, 12, 7, 7, 6, 15, 10\}$ ordenarlos por el método de *"clasficación rápida"* (*"quick-sort"*). En cada iteración indicar el pivote y mostrar el resultado de la partición. Utilizar la estrategia de elección del pivote discutida en el curso, a saber el mayor de los dos primeros elementos distintos.

[Ej. 4] [Preguntas (total = 10pt, 4pt por pregunta)]

a) Defina la propiedad de **transitividad** para las relaciones de orden. Si $f(x)$ es una función sobre los reales y defino $a <_f b$ sii $f(a) < f(b)$, es $<_f$ transitiva?

b) Nombre diversas **relaciones de orden** que se pueden usar con los algoritmos de ordenamiento.

c) Discuta el valor de retorno de `insert(x)` para conjuntos.

d) Discuta el **número de inserciones** que requieren los algoritmos de ordenamiento lentos en el peor caso.

e) ¿Cuál es el costo de **inserción** en tablas de dispersión cerradas?