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. [2011-11-24]

ATENCIÓN (1): Para aprobar deben obtener un **puntaje mínimo** de

- **50 %** en clases (Ej 1),
- **50 %** en programación (Ej 2),
- **50 %** en operativos (Ej 3) y
- **60 %** sobre las preguntas de teoría (Ej 4).

ATENCIÓN (2): Recordar que tanto en las clases (Ej. 1) como en los ejercicios de programación (Ej 2.) **deben usar la interfaz STL**.

[Ej. 1] [clases (20pt)] Insistimos: deben usar la interfaz STL.

a) **[oht-erase (7pt)]** Escribir una función `bool oht_erase(vector<list<T>> &V, bool (*equal)(T,T), unsigned int hash(T), T x);` que elimina el elemento `x` de la tabla de dispersión abierta `V`, utilizando la función de hash `hash()` y la relación de equivalencia dada por `equal()`.

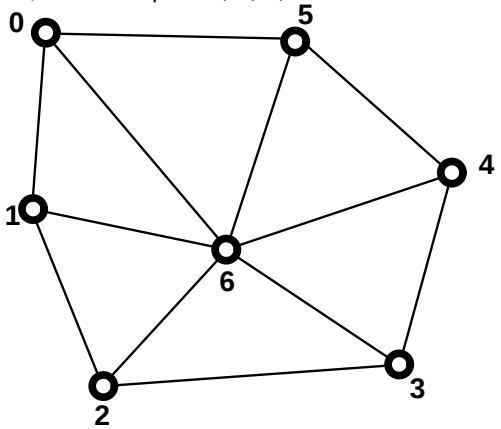
b) **[bst-find (7pt)]** Escribir la función `bool bst_find(btree<string> &T, bool (*less)(string,string), string x);` que devuelve `true` en caso que `x` esté en el ABB y `false` en caso contrario, utilizando `less()` como relación de orden.

c) **[bst-max (6pt)]** Escribir una función `string bst_max(btree<string> &T)` que devuelve el mínimo elemento de un árbol binario de búsqueda (ABB) `T`, siendo los valores contenidos en los nodos del árbol de tipo `string`. Si `T` es el árbol vacío, entonces debe retornar el string vacío (`""`).

[Ej. 2] [Programación (total = 40pt)] Insistimos: deben usar la interfaz STL.

Atención: Hay 3 ejercicios cuya suma es **50**, pero el total de la sección es **40**. (O sea, la nota final en esta sección es `min(40,n1+n2+n3)`).
Nota: En los ejercicios con grafos siguientes, los grafos son simples y se representan como un `map<int, set<int> >G`, donde `G[j]` es el conjunto de vértices adyacentes (esto es, conectados por una arista) con el vértice `j`.

a) **[is-indep (20pt)]** Dado un grafo `map<int, set<int> >G` y un conjunto `set<int> S` de vértices del grafo, escribir un predicado `bool is_indep(map<int, set<int> >&G,set<int>&S);` que determina si `S` es un conjunto **independiente** de `G`. Se dice que `S` es un conjunto independiente de `G`, si para cada par de vértices de `S`, NO existe una arista que los une en `G`. Por ejemplo, en la figura de abajo los conjuntos `{0,2,4}` y `{1,3,5}` son independientes, mientras que `{0,2,4,6}` no lo es.



b) **[is-hamilt (20pt)]** Dado un grafo `map<int, set<int> >G` y una lista de vértices `list<int> L` determinar si `L` es un **camino hamiltoniano** en `G`.

Ayuda: Mantener un `set<int> visited` con todos los vértices visitados. Para dos enteros `x`, `y` sucesivos en la lista `L`, verificar si son vértices de `G` y si son adyacentes. Verificar que el nuevo vértice `y` no fuera visitado previamente e insertarlo en `visited`. Al final, chequear que todos los vértices fueron visitados.
Nota: Recordamos que un camino Hamiltoniano en un grafo es un camino en el mismo que pasa por todos los nodos sin repetir ninguno. Por ejemplo en el grafo de arriba el camino `{0,1,2,3,4,5,6}` es Hamiltoniano.

c) **[verif-color (10pt)]** Dado un grafo `map<int, set<int> >G` y una coloración `map<int,string> C` escribir un predicado `bool verif_color(map<int, set<int> >&G,map<int,string> C);` que determina si `C` es una coloración válida, es decir si dados dos vértices adyacentes `x`,`y` de `G` sus colores son diferentes.

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

a) **[abb (5 pts)]** Dados los enteros $\{14, 8, 21, 3, 4, 11, 6, 5, 4, 13\}$ insertarlos, en ese orden, en un **árbol binario de búsqueda (ABB)**. Mostrar las operaciones necesarias para eliminar los elementos 14, 8 y 3 en ese orden.
b) **[hash-dict (5 pts)]** Insertar los números $\{4, 20, 30, 13, 12, 40, 20, 22, 9\}$ en una **tabla de dispersión cerrada** con $B = 8$ cubetas, con función de dispersión `h(x) = 3*x-2` 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 $\{7, 11, 6, 3, 7, 12, 10, 5, 5, 4, 13, 8\}$ 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 = 20pt, 4pt por pregunta)]

a) Cual es el costo de una **inserción exitosa** en una tabla de dispersión cerrada en función de su **tasa de ocupación**.
b) Discuta cuál es la influencia de las cubetas con valores `undef` y `deleted` en las **tablas de dispersión cerradas**.
c) Defina el concepto de **estabilidad** para algoritmos de ordenamiento. De un ejemplo.
d) Se quiere representar subconjuntos del conjunto universal de pares de enteros

$$U = \{(j, k) \ / \ 0 \leq j, k < 10\} \tag{1}$$

como un conjunto con **vectores de bits**.

1) ¿Cuál es el tamaño del conjunto universal?
2) Escribir las funciones `int indx(pair<int,int>)` y `pair<int,int> element(int)` correspondientes, que mapean el espacio de elementos del conjunto universal a índices y viceversa.

e) Discuta la cantidad de **intercambios** que realizan los 3 **algoritmos de ordenamiento lentos** que fueron estudiados.

mstorti@galileo/aed-2.0.4-53-g51a1e41/Thu Nov 24 13:08:17 2011 -0300