pila

Ejercicio 1 Determinar si una cadena `z` es de la forma `z = x y`, donde `y` es la cadena inversa (o espejo) de la cadena `x`, ignorando los espacios en blanco. Emplear una cola y una pila auxiliares. Resuelto en el archivo `cadena_pq.cpp`

correspondencia

Ejercicio 1 Escribir una función `void apply_map(list<int> &L, map<int,int> &M, list<int> &ML)` que, dada una lista `L` y una correspondencia `M` retorna por `ML` una lista con los resultados de aplicar `M` a los elementos de `L`. Si algún elemento de `L` no esta en el dominio de `M`, entonces el elemento correspondiente de `ML` no es incluido. Por ejemplo, si `L = (1,2,3,4,5,6,7,1,2,3)` y `M= (1,2),(2,3),(3,4),(4,5),(7,8)`, entonces, después de hacer `apply_map(L,M,ML)`, debe quedar `ML = (2,3,4,5,8,2,3,4)`. Restricciones: No usar estructuras auxiliares. El tiempo de ejecucion del algoritmo debe ser O(n), donde `n` es el numero de elementos en la lista, asumiendo que las operaciones usadaas de correspondencia son O(1). [Tomado en el 1er parcial del 20/4/2006]. Resuelto en el archivo `apply-map.cpp`

Ejercicio 2 Dos corresponencias `M1` y `M2` son inversas una de la otra si tienen el mismo numero de asignaciones y para cada par de asignacion `x->y` en `M1` existe el par `y->x` en `M2`. Escribir una funcion predicado `bool areinverse(map<int,int> &M1,map<int,int> &M2);` que determina si las correspondencias `M1`, `M2` son una la inversa de la otra o no. [Tomado en Primer Parcial 17-SET-2009]. Resuelto en el archivo `areinverse.cpp`

Ejercicio 3 Dada una correspondecia `M` y un elemento `x0`, podemos generar una secuencia `(x0,x1,x2,...)` de la forma `x_k+1=M(xk)`. La secuencia se detiene cuando uno de los valores `x_k` generados no pertenece a las claves de `M`. En ese caso la secuencia generada es finita. Por otra parte, puede ocurrir que un elemento de la secuencia se repita, es decir `x_k+m=x_k` con `m>0`. Es obvio que, a partir de alli la secuencia se va a repetir indefinidamente. _Consigna:_ escribir una Escribir una funcion `void cyclic(map<int,int> &M,list<int> &L);` que extrae en `L` todas aquellas claves de `M` que generan una secuencia ciclica infinita. Por ejemplo, si `M=(1,2),(2,5),(3,4),(4,6),(5,2)` entonces `cyclic(M,L)` debe retornar `L=(1,2,5)`. [Tomado en 1er parcial 25-SEP-2008]. Resuelto en el archivo `cyclic.cpp`

Ejercicio 4 Escribir una función `bool es_biyectiva (map <int,int> &A)` que retorna true si la misma representa una función biyectiva, esto es, si la correspondencia `A` describe una relación *uno a uno*. Por ejemplo, supongamos el conjunto `X = (0,1,2,3,4,5)` y consideremos las correspondencias `A1 = (0,2), (1,5), (2,0), (3,3), (4,4), (5,1)` y `A2 = (0,2), (1,1), (2,0), (3,3), (4,3), (5,1)` . En el primer caso, cada elemento (de 0 a 5) tiene preimagen, por lo que `es_biyectiva (A1)` debe retornar `true`. En cambio, en el segundo caso, los elementos 4 y 5 no tienen preimagen, por lo que `es_biyectiva (A2)` debe retornar `false`. Resuelto en el archivo `es_biyectiva.cpp`

Ejercicio 5 Una correspondencia es una _permutacion_ si el conjunto de los elementos del dominio (las claves) es igual al del contradominio (los valores). Consigna: escribir una funcion predicado `bool es_permutacion(map<int,int> &M)` que retorna `true` si `M` es una permutacion y `false` si no lo es. [Tomado en Primer Parcial 27-SET-2007]. Resuelto en el archivo `espermut.cpp`

Ejercicio 6 Implemente una función `void intersect_map(map< string,list<int> > &A, map< string, list<int> > &B,map< string, list<int> > &C)` que a partir de los diccionarios `A` y `B` construye un diccionario `C` de manera que las claves de `C` son la interseccion de las claves de `A` y `B` y para cada clave `k` en `C` la imagen `C[k]` es la interseccion de los valores en `A[k]` y `B[k]`. [Tomado en Primer Parcial 17-SET-2009]. Resuelto en el archivo `intersecmap.cpp`

Ejercicio 7 Dada una correspondencia `M` y asumiendo que es invertible o biunivoca (esto es, todos los valores del contradominio son distintos), la correspondencia 'inversa' `N` es aquella tal que, si `y=M[x]`, entonces `x=N[y]`. Por ejemplo, si `M=(0,1),(1,2),(2,0)`, entonces la inversa es `N=(1,0),(2,1,(0,2))`. Consigna: Escribir una función `bool inverse(map<int,int> &M,map<int,int> &N)` tal que, si `M` es invertible, entonces retorna true y `N` es su inversa. En caso contrario retorna falso y `N` es la correspondencia 'vacia' (sin asignaciones) [Tomado en el 1er parcial del 20/4/2006]. Resuelto en el archivo `inverse.cpp`

Ejercicio 8 Dado un grafo `G`, y un arbol `T`, decimos que `T` "expande" a `G` si la raiz `n` de `T` es un vertice de `G`, y los caminos de `T` permiten llegar desde `n` hasta cualquier otro nodo de `G`. Escribir una funcion `bool is_spng_tree(G,T)` (por "is-spanning-tree") que determina si `T` expande a `G`. [Tomado en el 3er parcial del 2009-11-27]. Resuelto en el archivo `isspngtree.cpp`

Ejercicio 9 Dada una lista `L` de enteros escribir una función `bool es_constante (list <int> &L)` que retorna true solo si todos sus elementos son iguales. Hacerlo con (i) sólo operaciones del TAD lista y (ii) mediante una correspondencia. Escriba un procedimiento `void imprime (map <int,int> &M);` para imprimir una correspondencia. Resuelto en el archivo `lista_cte.cpp`

Ejercicio 10 Escribir las funciones `map2list()` y `list2map()` de acuerdo a las siguientes especificaciones. `void map2list(map<int,int> &M,list<int> &keys,list<int> &vals);` dado un map `M` retorna las listas de claves y valores. `void list2map(list<int> &keys,list<int> &vals,map<int,int> &M);` dadas las listas de claves `(k1,k2,k3...)` y valores `(v1,v2,v3...)` retorna el map `M` con las asignaciones correspondientes `(k1,v1),(k2,v2),(k3,v3),....` (Nota: Si hay *claves repetidas*, solo debe quedar la asignacion correspondiente a la *ultima* clave en la lista. Si hay menos valores que claves, utilizar cero como valor. Si hay mas valores que claves, ignorarlos). [Tomado en Primer Parcial 17-SET-2009]. Resuelto en el archivo `map2list.cpp`

Ejercicio 11 Considere una red de n computadoras enumeradas $i = 0, 1, ...(n-1)$ las cuales pueden estar conectadas entre si de diversas maneras. Dicha información de la interconexión se la puede almacenar mediante un vector

de n listas cuyos elementos son los enteros $[0, n)$ que designan a cada computadora. Cada sublista V_i enumera las primeras "vecinas" de la computadora i, para $0 \leq i < n$. Por ejemplo, supongamos que el vector de listas $V = [v_0, v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9]$, de las 10 computadoras enumeradas entre 0 y 9 es: $V = [(6), (8), (7), (6), (8), (7), (0, 3), (2, 5, 9), (1, 4, 9), (7)]$. Por ejemplo, las primeras vecinas de las computadoras 7 y 8 son son (2,5,9) y (1,4,9), respectivamente. Si hacemos un dibujo de la red resultante (ver programa demo) observaremos que hay dos subredes, a saber, la subred 1 conformada por las computadoras G1 = (0,3,6) y la subred 2 conformada por las computadoras G2 = (1,2,4,5,7,8,9). Ver otro ejemplo en el programa demo. Entonces, dado un vector de listas V que representa una red posiblemente desconexa interesa identificar todas las subredes que pueda contener. Esta tarea se puede resolver construyendo la correspondencia $M = M(X, Y)$ que mapea del dominio $X = [0, n)$ de las computadoras presentes en la red, al rango $Y = [1, g_{max}]$ de las subredes posibles. Un algoritmo que ejecuta esta tarea se resume en el programa demo. Consigna: usando las operaciones de los TAD lista, mapeo y cola, escribir un procedimiento `void map_explora (vector < list <int> > & V, map <int,int> & M)` en el cual, dada una red de n computadoras representada por un vector de listas de enteros `V`, construya dicho mapeo M. Resuelto en el archivo `map_explora.cpp`

Ejercicio 12 Dadas dos correspondencias `A` y `B`, que asocian enteros con listas ordenada de enteros, escribir una funcion `void merge_map(map<int,list<int>> &A, map<int,list<int>> &B, map<int,list<int>> &C)` que devuelve en `C` una correspondencia que asigna al elemento `x` la fusion ordenada de las dos listas `A[x]` y `B[x]`. Si `x` no es clave de `A`, entonces `C[x]` debe ser `B[x]` y viceversa. Por ejemplo: si `M=(1,2),(2,5),(3,4),(4,6),(5,2)` entonces `cyclic(M,L)` debe dejar `L=(1,2,5)`. [Tomado en 1er parcial 25-SEP-2008]. Resuelto en el archivo `mergemap.cpp`

Ejercicio 13 Dada una correspondencia `M` tal que el conjunto de sus valores es igual al conjunto de sus claves, encontrar el índice nilpotente, de la misma, es decir el número de veces `n` que hay que componerla consigo misma hasta llegar a la identidad, es decir $M^n = I$. _Consigna:_ Escribir una función `int nilpot(map<int,int> &M);` que dada una correspondencia `M` retorna el mínimo entero `n` tal que $M^n = I$. [Tomado en el 1er parcial 21/4/2005]. Resuelto en el archivo `nilpot.cpp`

conjunto

Ejercicio 1 gado un grafo como `map<int,set<int>> G` encontrar los subconjuntos del mismo `list<set<int>> D` que estan desconectados. Por ejemplo, si `G=1->2,2->1,3->4,4->3`, entonces debe retornar `D=(1,2,3,4)`. La signatura de la funcion a implementar es `void connected(map<int,set<int>> &G, list<set<int>> &D);` [Tomado en el 3er parcial 2008-11-20]. Resuelto en el archivo `connected.cpp`

Ejercicio 2 Dada una lista de conjuntos de enteros `list< set<int> > l` escribir una función `void diffsym(list< set<int> > &L, set<int> &ad)` que

retorna en `ad` el conjunto de los elementos que pertenecen a uno y sólo uno de los conjuntos de `L`. Por ejemplo, si `L = (1,2,3,2,4,5,4,6)` entonces `ad` debe ser `1,3,5,6`. Notar que si el número de conjuntos en `l` es 2 y los llamamos `A` y `B`, entonces debe retornar `ad = (A-B) union (B-A)`. [Tomado en el 3er parcial 23/6/2005]. Resuelto en el archivo `diffsym.cpp`

Ejercicio 3 Dado un conjunto `S` y una relacion de orden `bool comp(int x,int y)` separar `S`, segun sus clases de equivalencia en una lista `list<set<int>> L`. Signatura: `void eqclass(set<int> &S, bool (*)(int x,int y),list<set<int>> &L)` [Tomado en el 3er parcial 2008-11-20]. Resuelto en el archivo `eqclass.cpp`

Ejercicio 4 Dado un grafo `vector<set<int>> G` y un vertice de partida `x` se desea determinar la estructuras de capas de vecinos de `G` alrededor de `x` definida de la siguiente forma: la capa 0 es `x`, la capa 1 son los vecinos de `x`. A partir de alli la capa `n>=2` esta formada por los vecinos de los nodos de la capa `n-1` (es decir la adyacencia de la capa) pero que no estan en las capas anteriores (en realidad basta con verificar que no esten en las capas `n-1` ni `n-2`). [Tomado en tercer parcial 22-NOV-2007]. Resuelto en el archivo `graphlayers.cpp`

Ejercicio 5 Dados `n` conjuntos A_0, A_1, ... A_{n-1} determinar si alguno de ellos (digamos A_j) incluye a todos los otros. Es decir $A_j \subset A_k$ para todo `k`. En ese caso, retornar el indice j, si no retornar -1. `int includes_all(vector< set<int> > &setv);` [Tomado en tercer parcial 22-NOV-2007]. Resuelto en el archivo `incall.cpp`

Ejercicio 6 Escribir un predicado `bool incluido(set<int> &A, set<int> &B);` que retorna verdadero si y solo si `A` esta incluido en `B`. [Tomado en el 3er parcial 23/6/2005]. Resuelto en el archivo `incluido.cpp`

Ejercicio 7 Escribir una función predicado `bool is_mapped_set(set<int> &A,set<int> &B,int (*mapfun)(int));` que retorna verdadero si el conjunto `B` contiene los elementos de `A`, mapeados via la funcion `mapfun`. [Tomado en recuperatorio 29-NOV-2007]. Resuelto en el archivo `ismapset.cpp`

Ejercicio 8 Dado un grafo `G`, y un arbol `T`, decimos que `T` "expande" a `G` si la raiz `n` de `T` es un vertice de `G`, y los caminos de `T` permiten llegar desde `n` hasta cualquier otro nodo de `G`. Escribir una funcion `bool is_spng_tree(G,T)` (por "is-spanning-tree") que determina si `T` expande a `G`. [Tomado en el 3er parcial del 2009-11-27]. Resuelto en el archivo `isspngtree.cpp`

Ejercicio 9 Dada una lista de conjuntos `list< set<int> > L`, escribir una función predicado `bool proper_subset(list< set<int> > &L)`, que determina si los conjuntos de la lista `L` son subconjuntos propios en forma consecutiva. Es decir, si $L = (A_0, A_1, ..., A_{n-1})$, determinar si A_j es subconjunto propio de A_{j+1}, para $j = 0, ..., n-2$. [Tomado en el examen final 7/7/2005]. Resuelto en el archivo `propsubset.cpp`

lista

Ejercicio 1 Escribir una función `void apply_map(list<int> &L, map<int,int> &M, list<int> &ML)` que, dada una lista `L` y una correspondencia `M` retorna por `ML` una lista con los resultados de aplicar `M` a los elementos de `L`. Si algún elemento de `L` no esta en el dominio de `M`, entonces el elemento correspondiente de `ML` no es incluido. Por ejemplo, si `L = (1,2,3,4,5,6,7,1,2,3)` y `M= (1,2),(2,3),(3,4),(4,5),(7,8)`, entonces, después de hacer `apply_map(L,M,ML)`, debe quedar `ML = (2,3,4,5,8,2,3,4)`. Restricciones: No usar estructuras auxiliares. El tiempo de ejecucion del algoritmo debe ser O(n), donde `n` es el numero de elementos en la lista, asumiendo que las operaciones usadaas de correspondencia son O(1). [Tomado en el 1er parcial del 20/4/2006]. Resuelto en el archivo `apply-map.cpp`

Ejercicio 2 Dos corresponencias `M1` y `M2` son inversas una de la otra si tienen el mismo numero de asignaciones y para cada par de asignacion `x->y` en `M1` existe el par `y->x` en `M2`. Escribir una funcion predicado `bool areinverse(map<int,int> &M1,map<int,int> &M2);` que determina si las correspondencias `M1`, `M2` son una la inversa de la otra o no. [Tomado en Primer Parcial 17-SET-2009]. Resuelto en el archivo `areinverse.cpp`

Ejercicio 3 En ciertas aplicaciones interesa separar las corridas ascendentes en una lista de números $L = (\ a_1,\ a_2,\ ...,\ a_n\ )$, donde cada corrida ascendente es una sublista de números consecutivos a_i, a_{i+1}, ..., a_{i+k}, la cual termina cuando $a_{i+k} > a_{i+k+1}$, y es ascendente en el sentido de que $a_i \leq a_{i+1} \leq ... \leq a_{i+k}$. Por ejemplo, si la lista es L = (0,5,6,9,4,3,9,6,5,5,2,3,7), entonces hay 6 corridas ascendentes, a saber: (0,5,6,9), (4), (3,9), (6), (5,5) y (2,3,7). *Consigna:* usando las operaciones de la clase lista, escribir una función `int ascendente (list <int> &L, list < list<int> > &LL)` en la cual, dada una lista de enteros `L`, almacena cada corrida ascendente como una sublista en la lista de listas `LL`, devolviendo además el número z de corridas ascendentes halladas. *Restricciones:* a) El tiempo de ejecución del algoritmo debe ser $O(n)$, b) La lista de listas `LL` inicialmente está vacía, c) No usar otras estructuras auxiliares. [Tomado en Examen Final 29-JUL-2004]. Resuelto en el archivo `ascendente.cpp`

Ejercicio 4 Determinar si una cadena `z` es de la forma `z = x y`, donde `y` es la cadena inversa (o espejo) de la cadena `x`, ignorando los espacios en blanco. Emplear una cola y una pila auxiliares. Resuelto en el archivo `cadena_pq.cpp`

Ejercicio 5 Escribir una función `void chunk_revert(list<int> &L,int n);` que dada una lista `L` y un entero `n`, invierte los elementos de la lista tomados de a `n`. Si la longitud de la lista no es múltiplo de `n` entonces se invierte el resto también. Por ejemplo, si `L=1,3,2,5,4,6,2,7` entonces después de hacer `chunk_revert(L,3)` debe quedar `L=2,3,1,6,4,5,7,2`. Restricciones: Usar a lo sumo una estructura auxiliar. (En tal caso debe ser lista, pila o cola). [Tomado en el 1er parcial 21/4/2005]. Resuelto en el archivo `chunk-revert.cpp`

Ejercicio 6 Coloquemos n números enteros positivos alrededor de una circunferencia inicial. Construyamos ahora sucesivas circunferencias concéntricas *ha-*

cia el exterior, de igual cantidad de elementos, los cuales son obtenidos restando (en valor absoluto) pares consecutivos de la última circunferencia exterior. Entonces, puede verificarse que si $n = 2^k$ en alguna iteración p aparecerán n números iguales. En ese momento se detiene la iteración. Por ejemplo, supongamos $k = 2$, $(n = 4)$ y que la circunferencia inicial sea $C_0 = (8, 2, 5, 7)$, entonces iteramos y obtendremos sucesivamente, $C_1 = (6, 3, 2, 1)$, $C_2 = (3, 1, 1, 5)$, $C_3 = (2, 0, 4, 2)$, $C_4 = (2, 4, 2, 0)$ y $C_5 = (2, 2, 2, 2)$, por lo que el número de circunferencias iteradas es $p = 5$. Entonces, dada una lista $L = [x_0, x_1, ..., x_{n-1}]$ de n números enteros que representan los valores iniciales alrededor de la circunferencia inicial, escribir una función `int circulo(list<int> & L);` que ejecuta esta tarea y devuelva además el número de circunferencias iteradas p. *Restricción:* el algoritmo debe ser *in place*. *Ayuda:* Pensar a la lista en un *"sentido circular"*. Tener cuidado al generar la diferencia correspondiente al extremo. [Tomado en el 1er parcial del 21/4/2005]. Resuelto en el archivo `circulo.cpp`

Ejercicio 7 Escriba procedimientos para concatenanar: a) dos listas `L1` y `L2` usando `insert`; b) un vector `VL` de `n` listas usando `insert`; c) una lista `LL` de `n` sublistas usando `insert` "básico"; d) una lista `LL` de `n` sublistas usando una opción de `insert`; e) una lista `LL` de `n` sublistas usando `splice`. Resuelto en el archivo `concatena.cpp`

Ejercicio 8 Escribir una función `void creciente(queue<int> &Q)` que elimina elementos de `Q` de tal manera de que los elementos que quedan estén ordenados en forma creciente. [Tomado en el 1er parcial 27-APR-2004] Resuelto en el archivo `creciente.cpp`

Ejercicio 9 Dada una correspondecia `M` y un elemento `x0`, podemos generar una secuencia `(x0,x1,x2,...)` de la forma `x_k+1=M(xk)`. La secuencia se detiene cuando uno de los valores `x_k` generados no pertenece a las claves de `M`. En ese caso la secuencia generada es finita. Por otra parte, puede ocurrir que un elemento de la secuencia se repita, es decir `x_k+m=x_k` con `m>0`. Es obvio que, a partir de alli la secuencia se va a repetir indefinidamente. _Consigna:_ escribir una Escribir una funcion `void cyclic(map<int,int> &M,list<int> &L);` que extrae en `L` todas aquellas claves de `M` que generan una secuencia ciclica infinita. Por ejemplo, si `M=(1,2),(2,5),(3,4),(4,6),(5,2)` entonces `cyclic(M,L)` debe retornar `L=(1,2,5)`. [Tomado en 1er parcial 25-SEP-2008]. Resuelto en el archivo `cyclic.cpp`

Ejercicio 10 Implemente una función `encuentra(list<int> &L1, list<int> &L2, list<int> &indx)` que verifica si los elementos de 'L2' estan en 'L1' (en el mismo orden, pero no necesariamente en forma consecutiva). Si es asi, retorna true y en 'indx' retorna los indices de los elementos de 'L1' que corresponden a los elementos de 'L2'. Resuelto en el archivo `encuentra.cpp`

Ejercicio 11 Escriba procedimientos para intercalar (*merge*): (i) dos listas ordenadas `L1` y `L2` en una nueva lista `L`; (ii) un vector `VL` de `n` listas ordenadas como nueva lista `L`. Notar que *intercalar* (*merge*) implica en ambos casos

que la nueva lista L debe resultar también *ordenada*. Resuelto en el archivo intercala.cpp

Ejercicio 12 Implemente una función void intersect_map(map< string,list<int> > &A, map< string, list<int> > &B,map< string, list<int> > &C) que a partir de los diccionarios A y B construye un diccionario C de manera que las claves de C son la interseccion de las claves de A y B y para cada clave k en C la imagen C[k] es la interseccion de los valores en A[k] y B[k]. [Tomado en Primer Parcial 17-SET-2009]. Resuelto en el archivo intersecmap.cpp

Ejercicio 13 Dada una correspondencia M y asumiendo que es invertible o biunivoca (esto es, todos los valores del contradominio son distintos), la correspondencia 'inversa' N es aquella tal que, si y=M[x], entonces x=N[y]. Por ejemplo, si M=(0,1),(1,2),(2,0), entonces la inversa es N=(1,0),(2,1,(0,2)). Consigna: Escribir una función bool inverse(map<int,int> &M,map<int,int> &N) tal que, si M es invertible, entonces retorna true y N es su inversa. En caso contrario retorna falso y N es la correspondencia 'vacia' (sin asignaciones) [Tomado en el 1er parcial del 20/4/2006]. Resuelto en el archivo inverse.cpp

Ejercicio 14 Escribir una función void junta (list <int> &L, int n) que, dada una lista L, agrupa de a n elementos dejando su suma IN PLACE. Por ejemplo, si la lista L contiene L=(1,3,2,4,5,2,2,3,5,7,4,3,2,2), entonces depués de junta (L,3) debe quedar L=(6,11,10,14,4). Prestar atención a no usar posiciones inválidas después de una supresión. El algoritmo debe tener un tiempo de ejecución O (m), donde m es el número de elementos en la lista original. [Tomado en el examen final del 1/8/2002] Resuelto en el archivo junta.cpp

Ejercicio 15 Dada una lista L de enteros escribir una función bool es_constante (list <int> &L) que retorna true solo si todos sus elementos son iguales. Hacerlo con (i) sólo operaciones del TAD lista y (ii) mediante una correspondencia. Escriba un procedimiento void imprime (map <int,int> &M); para imprimir una correspondencia. Resuelto en el archivo lista_cte.cpp

Ejercicio 16 Escribir las funciones map2list() y list2map() de acuerdo a las siguientes especificaciones. void map2list(map<int,int> &M,list<int> &keys,list<int> &vals); dado un map M retorna las listas de claves y valores. void list2map(list<int> &keys,list<int> &vals,map<int,int> &M); dadas las listas de claves (k1,k2,k3...) y valores (v1,v2,v3...) retorna el map M con las asignaciones correspondientes (k1,v1),(k2,v2),(k3,v3),.... (Nota: Si hay *claves repetidas*, solo debe quedar la asignacion correspondiente a la *ultima* clave en la lista. Si hay menos valores que claves, utilizar cero como valor. Si hay mas valores que claves, ignorarlos). [Tomado en Primer Parcial 17-SET-2009]. Resuelto en el archivo map2list.cpp

Ejercicio 17 Considere una red de n computadoras enumeradas $i = 0, 1, ...(n-1)$ las cuales pueden estar conectadas entre si de diversas maneras. Dicha información de la interconexión se la puede almacenar mediante un vector

de n listas cuyos elementos son los enteros $[0, n)$ que designan a cada computadora. Cada sublista V_i enumera las primeras "vecinas" de la computadora i, para $0 \leq i < n$. Por ejemplo, supongamos que el vector de listas $V = [v_0, v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9]$, de las 10 computadoras enumeradas entre 0 y 9 es: $V = [(6), (8), (7), (6), (8), (7), (0, 3), (2, 5, 9), (1, 4, 9), (7)]$. Por ejemplo, las primeras vecinas de las computadoras 7 y 8 son son (2,5,9) y (1,4,9), respectivamente. Si hacemos un dibujo de la red resultante (ver programa demo) observaremos que hay dos subredes, a saber, la subred 1 conformada por las computadoras G1 = (0,3,6) y la subred 2 conformada por las computadoras G2 = (1,2,4,5,7,8,9). Ver otro ejemplo en el programa demo. Entonces, dado un vector de listas V que representa una red posiblemente desconexa interesa identificar todas las subredes que pueda contener. Esta tarea se puede resolver construyendo la correspondencia $M = M(X, Y)$ que mapea del dominio $X = [0, n)$ de las computadoras presentes en la red, al rango $Y = [1, g_{max}]$ de las subredes posibles. Un algoritmo que ejecuta esta tarea se resume en el programa demo. Consigna: usando las operaciones de los TAD lista, mapeo y cola, escribir un procedimiento `void map_explora (vector < list <int> > & V, map <int,int> & M)` en el cual, dada una red de n computadoras representada por un vector de listas de enteros `V`, construya dicho mapeo M. Resuelto en el archivo `map_explora.cpp`

Ejercicio 18 Dadas dos correspondencias `A` y `B`, que asocian enteros con listas ordenada de enteros, escribir una funcion `void merge_map(map<int,list<int>> &A, map<int,list<int>> &B, map<int,list<int>> &C)` que devuelve en `C` una correspondencia que asigna al elemento `x` la fusion ordenada de las dos listas `A[x]` y `B[x]`. Si `x` no es clave de `A`, entonces `C[x]` debe ser `B[x]` y viceversa. Por ejemplo: si `M=(1,2),(2,5),(3,4),(4,6),(5,2)` entonces `cyclic(M,L)` debe dejar `L=(1,2,5)`. [Tomado en 1er parcial 25-SEP-2008]. Resuelto en el archivo `mergemap.cpp`

Ejercicio 19 Escribir una función `void ordenag (list <int> &l, int m)` que, dada una lista `l`, va ordenando sus elementos de a grupos de `m` elementos. Por ejemplo si m=5, entonces `ordenag` ordena los primeros 5 elementos entre si, despues los siguientes 5 elementos, y asi siguiendo. Si la longitud `n` de la lista no es un múltiplo de `m`, entonces los últimos `n mod m` elementos también deben ser ordenados entre si. Por ejemplo, si l = (10 1 15 7 2 19 15 16 11 15 9 13 3 7 6 12 1), entonces después de `ordenag (5)` debemos tener l = (1 2 7 10 15 11 15 15 16 19 3 6 7 9 13 1 12). [Tomado en el examen final del 5-Dic-2002]. Resuelto en el archivo `ordenag.cpp`

Ejercicio 20 Usando las operaciones del TAD lista, escribir una función `void particiona (list<int> &L, int a)` la cual, dada una lista de enteros `L`, reemplace aquellos que son mayores que `a` por una sucesión de elementos menores o iguales que `a` pero manteniendo la suma total constante. [Ejercicio tomado en el Exámen Final del 05/07/01] Resuelto en el archivo `particiona.cpp`

Ejercicio 21 Escriba una función `void print_back (list<int> & L, list <int>::iterator p)` que, en forma *recursiva*, imprima una lista en sentido inverso, es decir, desde el final al principio de la lista. Se le da como dato

el procedimiento a la primera posición de la lista. [Ejercicio 3 del final del 14/02/2002] Resuelto en el archivo `print_back.cpp`

Ejercicio 22 Usando las operaciones del TAD lista, escribir una función `void random_shuffle (list <int> &L)` que, dada una lista de enteros `L`, reordena sus elementos en forma aleatoria. Se sugiere el siguiente algoritmo: usando una lista auxiliar `Q` se van generando números enteros desde 0 a `length (L) - 1`. Se extrae el elemento j-ésimo de `l` y se inserta en `Q`. Finalmente, se vuelven a pasar todos los elementos de la cola `Q` a la lista `L`. [Ejercicio tomado en el Exámen Final del 05/07/01] Resuelto en el archivo `random_shuffle.cpp`

Ejercicio 23 Dada una lista de enteros `L` y dos listas `SEQ` y `REEMP` escribir una función `void reemplaza (list<int> &L, list<int> &SEQ, list<int> &REEMP)` que busca todas las secuencias de `SEQ` en `L` y las reemplaza por `REEMP`. Por ejemplo, si L=(1 2 3 4 5 1 2 3 4 5 1 2 3 4 5), SEQ=(4 5 1) y REEMP=(9 7 3), entonces despues de llamar a `reemplaza` debe quedar L=(1 2 3 9 7 3 2 3 9 7 3 2 3 4 5). Este procedimiento tiene un efecto equivalente a la función `reemplazar` de los editores de texto. [Tomado el 1er parcial, 16 abril 2002] Resuelto en el archivo `reemplaza.cpp`

Ejercicio 24 (a) Escriba una función `void refina (list<double> &L, double delta)` tal que dada una lista inicial de reales clasificados de menor a mayor `L`, refina inserta elementos entre los de `L`, de tal modo que la diferencia máxima entre elementos de la lista final sea menor o igual que delta; (b) Escriba una función `void desrefina (list<double> &L, double delta)` tal que dada una lista inicial de reales clasificados de menor a mayor `L`, desrefina suprime elementos de `L`, de tal modo que la diferencia minima entre elementos de la lista final sea mayor o igual que delta. Resuelto en el archivo `refina.cpp`

Ejercicio 25 Usando las operaciones del TAD lista, escribir una función `void rejunta (list<int> &L, int A)` que, dada una lista de enteros `L`, agrupe elementos de tal manera que en la lista queden solo elementos mayores o iguales que `A`. El algoritmo recorre la lista y, cuando encuentra un elemento menor, empieza a agrupar el elemento con los siguientes hasta llegar a `A` o hasta que se acabe la lista. Por ejemplo, si L=[3,4,2,4,1,4,4,3,2,2,4,1,4,1,4,4,1,4,4,2], entonces rejunta (L,10) da L=[13,12,13,10,10]. En la lista final NO deben quedar elementos menores que `A` salvo, eventualmente, el último. [Ejercicio tomado en el Exámen Final del 05/07/01] Resuelto en el archivo `rejunta.cpp`

Ejercicio 26 Escriba procedimientos para insertar, suprimir y buscar un elemento en una lista ordenada `L`. Versión **sin** `funciones genéricas` (comparar con `sorted_list2.cpp` y `sorted_list3.cpp`). Resuelto en el archivo `sorted_list1.cpp`

Ejercicio 27 Escriba procedimientos para insertar, suprimir y buscar un elemento en una lista ordenada `L`. Versión **únicamente** con `funciones genéricas` (comparar con `sorted_list1.cpp` y `sorted_list3.cpp`). Resuelto en el archivo `sorted_list2.cpp`

Ejercicio 28 Escriba procedimientos para insertar, suprimir y buscar un elemento en una lista ordenada `L`. Versión mediante una `clase genérica` (comparar con `sorted_list1.cpp` y `sorted_list2.cpp`). Resuelto en el archivo `sorted_list3.cpp`

arbol orientado

Ejercicio 1 El `cumsum(v)` de un vector `v` es la suma acumulada, es decir en la posicion `v[j]` debe quedar la suma de los elementos de `v[0..j]`. Para un arbol lo podemos extender diciendo que en cada nodo del arbol queda la suma de los valores de los nodos de su subarbol ANTES de la operacion. Por ejemplo si `T=(1 (2 (3 4 5 6)))` entonces despues de `cumsum(T)` debe quedar `T=(21 (2 (18 4 5 6)))`. La version hacia abajo corresponde a que en cada camino `n0,n1,...,nk` queden los valores `cumsum[*n0,*n1,...,*nk]`. Resuelto en el archivo `cumsum.cpp`

Ejercicio 2 Listado de árboles orientados en diferentes ordenes. Orden previo, posterior y simétrico. Resuelto en el archivo `listarbo.cpp`

Ejercicio 3 El listado en orden de nivel de los nodos de un árbol lista primero la raiz, luego todos los nodos de profundidad 1, después todos los de profundidad 2, y asi sucesivamente. Los nodos que estén en la misma profundidad se listan en orden de izquierda a derecha. Escribir una función `void orden_de_nivel (tree <int> &t)` para listar los nodos de un árbol en orden de nivel. Resuelto en el archivo `orden_nivel.cpp`

cola

Ejercicio 1 Determinar si una cadena `z` es de la forma `z = x y`, donde `y` es la cadena inversa (o espejo) de la cadena `x`, ignorando los espacios en blanco. Emplear una cola y una pila auxiliares. Resuelto en el archivo `cadena_pq.cpp`

Ejercicio 2 Escribir una función `void cum_sum_cola (queue<int> &Q)` que modifica a la cola `Q` dejando la suma acumulada de sus elementos, es decir, si los elementos de `Q` antes de llamar a `cum_sum_cola(Q)` son `Q = (a_0,a_1,...,a_n-1)`, entonces después de llamar a `cum_sum_cola(Q)` debe quedar `Q = (a_0, a_0 + a_1, ..., a_0 + a_1 + ... + a_n)`. Por ejemplo, si `Q = (1,3,2,4,2)` entonces después de hacer `cum_sum_cola (Q)` debe quedar `Q = (1,4,6,10,12)`. Restricciones: (i) usar una cola auxiliar; (ii) usar la interfase STL para colas (`clear (), front (), pop (), push (T x)+, size (), empty ()`); (iii) NO usar más estructuras auxiliares que la indicada ni otros algoritmos de STL; y (iv) el algoritmo debe ser $O(n)$. [Tomado en el Primer Parcial 27-ABR-2004] Resuelto en el archivo `cum_sum_cola.cpp`

Ejercicio 3 Considere una red de n computadoras enumeradas $i = 0, 1, ...(n-1)$ las cuales pueden estar conectadas entre si de diversas maneras. Dicha información de la interconexión se la puede almacenar mediante un vector

de n listas cuyos elementos son los enteros $[0, n)$ que designan a cada computadora. Cada sublista V_i enumera las primeras "vecinas" de la computadora i, para $0 \leq i < n$. Por ejemplo, supongamos que el vector de listas $V = [v_0, v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9]$, de las 10 computadoras enumeradas entre 0 y 9 es: $V = [(6), (8), (7), (6), (8), (7), (0, 3), (2, 5, 9), (1, 4, 9), (7)]$. Por ejemplo, las primeras vecinas de las computadoras 7 y 8 son son (2,5,9) y (1,4,9), respectivamente. Si hacemos un dibujo de la red resultante (ver programa demo) observaremos que hay dos subredes, a saber, la subred 1 conformada por las computadoras G1 = (0,3,6) y la subred 2 conformada por las computadoras G2 = (1,2,4,5,7,8,9). Ver otro ejemplo en el programa demo. Entonces, dado un vector de listas V que representa una red posiblemente desconexa interesa identificar todas las subredes que pueda contener. Esta tarea se puede resolver construyendo la correspondencia $M = M(X, Y)$ que mapea del dominio $X = [0, n)$ de las computadoras presentes en la red, al rango $Y = [1, g_{max}]$ de las subredes posibles. Un algoritmo que ejecuta esta tarea se resume en el programa demo. Consigna: usando las operaciones de los TAD lista, mapeo y cola, escribir un procedimiento `void map_explora (vector < list <int> > & V, map <int,int> & M)` en el cual, dada una red de n computadoras representada por un vector de listas de enteros `V`, construya dicho mapeo M. Resuelto en el archivo `map_explora.cpp`