Lenguajes de programación

Aplicación de listas en SCHEME

Estructuras de datos en Scheme

Tipos de estructuras de datos

- **Lineales**:
  - Arreglos
  - Registros
  - Tablas
  - Pilas
  - Filas
  - …
- **Jerárquicas** :
  - Árboles binarios
  - Árboles de expresiones
  - Montículos
  - …
- **Red**:
  - Grafos dirigidos
  - Grafos no dirigidos
  - Grafos ponderados
  - …

Representación en Scheme de ED

- La lista tiene la capacidad de representar a cualquier estructura de datos, sin importar si es lineal o no.
- Las ED en Scheme implícitamente utilizan memoria dinámica, pero no son estructuras cuyo estado se modifique en memoria…
- Por lo tanto, se implementarán procedimientos que transforman ED de entrada en nuevas ED modificadas.

Lista como arreglo unidimensional

- Cada elemento de la lista, corresponde a un elemento del arreglo.
- **DESVENTAJA**:

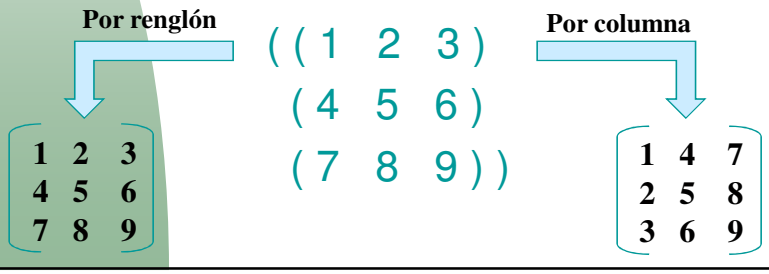
  El acceso de elementos es menos eficiente ya que debe realizarse secuencialmente

Ejemplo: ordenar arreglo

- Ordenar los elementos de un arreglo.
- ¿Qué algoritmo de sort se puede utilizar?
- **CASO BASE**: Arreglo vacío
  - Ya está ordenado…
- **CASO GENERAL**: Arreglo con elementos
  - Suponer que: ya se tiene ordenado el resto del arreglo…
  - insertar el primer elemento en el resto del arreglo ordenado…. SORT POR INSERCIÓN.

Listas que almacenan matrices

- POSIBILIDAD: Lista de renglones (o columnas) de la matriz, donde cada reglón o columna, es a su vez, una lista con los datos correspondientes.



Ejemplo: sumar matrices

- Asumir que las matrices son de las mismas dimensiones.
- La suma es elemento por elemento.
- **Estrategia**: sumar renglones!!
- **CASO BASE**: Matrices sin renglones (listas vacías)
  - No hay nada que sumar => matriz vacía
- **CASO GENERAL**: Matrices con renglones
  - Suponer que: ya se sumaron el resto de los renglones…
  - Insertar al inicio el resultado de sumar los primeros renglones de las matrices.

Lista como registro

- Dado que una lista en Scheme acepta elementos de diferentes tipos, una lista de campos es la relación directa para almacenar un registro.
- POSIBILIDADES:
  - *(176432 Juan Pérez 78 95 87)*
  - *(176432 (Juan José Pérez Pérez) (78 95 87))*
  - *(176432 ((Juan José) (Pérez) (de Pérez)) (78 95 87) ...)*
  - *((matricula 176432) (nombre (Juan José) (Pérez) (de Pérez)) (notas 78 95 87) ...)*

Tablas = listas de registros

- ((registro$_1$) (registro$_2$) … (registro$_n$))
- **EJEMPLO**: Dada una lista en donde se almacenan datos de empleados en el siguiente formato de registro:

  (nómina (nombre) #depto sueldo)

  Obtener la lista de empleados del depto. X

Ejemplo: Obtener nombres de empleados de depto X

- ¿Cómo se accesa el campo sueldo del registro (nómina (nombre) #depto sueldo) ?

  → (cadddr registro)

- **CASO BASE**: Tabla sin empleados (lista vacía)
  - No hay nombres que obtener => lista vacía
- **CASO GENERAL**: Tabla con empleados
  - Suponer que: ya se obtuvieron los nombres de los empleados en el resto de la tabla
  - Si el primer empleado pertenece al departamento X, insertar su nombre al inicio de la lista de nombres que ya se obtuvo.
  - Si no, regresar los nombres de los empleados que ya se obtuvieron.

Listas como Pilas y Filas

Pilas

Agregar un dato

*(define (**push** pila dato)*
(cons dato pila))

Eliminar un dato

*(define (**pop** pila)*
(cdr pila))

Tope de la pila

*(define (**top** pila)*
(car pila))

Filas

Agregar un dato

*(define (**insert** fila dato)*
(append fila (list dato)))

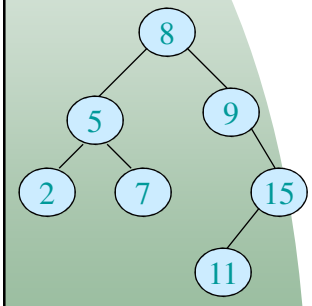
Eliminar un dato

*(define (**remove** fila)*
(cdr fila))

Primero de la fila

*(define (**first** fila)*
(car fila))

Listas que almacenan árboles



POSIBILIDAD: Lista donde el primer elemento representa la raíz del árbol, seguida por dos sublistas que describen recursivamente los subárboles izquierdo y derecho.

(raíz (subárbol-izquierdo)
(subárbol-derecho))

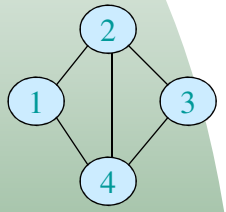
Ejemplo:

(8
(5 (2 () ()) (7 () ()))
(9 () (15 (11 () ()) ())))

Ejemplo: Determinar existencia en árbol

- Determinar si un valor se encuentra en un **árbol binario de búsqueda**; árbol binario donde el valor en cada raíz de subárbol es mayor que los valores de su subárbol izquierdo, pero menor que los valores de su subárbol derecho.
- **CASO BASE**: Árbol vacío (lista vacía)
  - Valor no encontrado => #f
- **CASO GENERAL**: Árbol con elementos
  - Si la raíz es el valor buscado => #t
  - Si no, buscar en el subárbol adecuado dependiendo si el valor buscado es menor o mayor que la raíz.

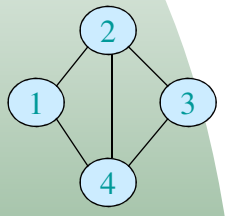
Listas que almacenan grafos



- ¿Cómo se puede representar un grafo en memoria?

  POSIBILIDADES:
  - Lista de adyacencias
    - ((1 2 4) (2 1 3 4) (3 2 4) (4 1 2 3))
  - Matriz de adyacencias
    - ((0 1 0 1)(1 0 1 1)(0 1 0 1)(1 1 1 0))
  - Nodos y aristas
    - ((1 2 3 4)((1 2)(1 4)(2 3)(2 4)(3 4)))
  - etc.

Ejemplo: Determinar el máximo de adyacencias



- Determinar el número máximo de aristas en las que participan los nodos de un grafo representado mediante una lista de adyacencias.
- **CASO BASE**: Grafo vacío (lista vacía)
  - => 0 adyacencias
- **CASO GENERAL**: Grafo con nodos
  - Asumir que ya se sabe el # máximo de adyacencias del resto de los nodos.
  - Si el # de adyacencias del primer nodo es mayor que el resto, regresarlo.
  - Si no, regresar el del resto.