Conjuntos, Tablas y Grafos

Mauro Jaskelioff

12/06/2017

Más Estructuras

- Al diseñar algoritmos y estructuras, son comunes los siguientes conceptos:
  - Conjuntos
  - Tablas (arreglos asociativos)
  - Grafos
- Definiremos TADS para conjuntos y tablas
- Veremos diferentes representaciones de grafos.
- NO definiremos un TAD para grafos
  - las representaciones y los tipos de las operaciones varían levemente según la aplicación.

Conjuntos

- Los conjuntos son muy importantes en la matemática
- A diferencia de las secuencias, los conjuntos son colecciones no ordenadas de elementos.
- Es muy útil tener implementaciones de ellos ya que su uso es frecuente.
  - La mayoría de los lenguajes proveen alguna implementación de ellos.
- Presentaremos un TAD para conjuntos

TAD Conjuntos

$$
\begin{array}{lll}
\multicolumn{3}{l}{\textbf{tad}\ \mathit{Conjunto}\ [\mathbb{S}]\ (\mathit{Set})\ \textbf{where}} \\
\quad \mathit{empty} & : \mathbb{S}_A & = \emptyset \\
\quad \mathit{size} & : \mathbb{S}_A \to \mathbb{N} & = \lambda S \to |S| \\
\quad \mathit{singleton} & : A \to \mathbb{S}_A & = \lambda e \to \{\, e \,\} \\
\quad \mathit{map} & : (A \to B) \to \mathbb{S}_A \to \mathbb{S}_B & = \lambda f\ S \to \{\, f\ s \mid s\ \in\ S \,\} \\
\quad \mathit{filter} & : (A \to \mathit{Bool}) \to \mathbb{S}_A \to \mathbb{S}_A & = \lambda P\ S \to \{\, s\ \in\ S \mid P\ s \,\} \\
\quad \mathit{intersection} & : \mathbb{S}_A \to \mathbb{S}_A \to \mathbb{S}_A & = \lambda S\ S' \to S\ \cap\ S' \\
\quad \mathit{union} & : \mathbb{S}_A \to \mathbb{S}_A \to \mathbb{S}_A & = \lambda S\ S' \to S\ \cup\ S' \\
\quad \mathit{difference} & : \mathbb{S}_A \to \mathbb{S}_A \to \mathbb{S}_A & = \lambda S\ S' \to S - S'
\end{array}
$$

- Una implementación correcta de conjuntos necesita poder **comparar por igualdad** los elementos.
- Muchas implementaciones necesitan poder darle un **orden** a los elementos para poder ser eficientes.
- ¿Es posible generar un conjunto infinito con esta interfaz?

Especificación de costos

- Para una implementación con árboles balanceados.

Operación	**W**	**S**
empty *singleton* *size*	$O(1)$	$O(1)$
map f S *filter f S*	$O\left(\sum_{e \in S} W(f\ e)\right)$	$O\left(\lg \lvert S \rvert + \max_{e \in S} S(f\ e)\right)$
intersection S S' *union S S'* *difference S S'*	$O\left(C_W \cdot m \cdot \lg(1 + \frac{n}{m})\right)$	$O\left(C_S \cdot \lg(n+m)\right)$

- C_W y C_S son el trabajo y profundidad de comparar elementos.
- $n = \max(\lvert S \rvert, \lvert S' \rvert)$ y $m = \min(\lvert S \rvert, \lvert S' \rvert)$
- Si $n \sim m$ el trabajo de las últimas operaciones es $O(C_W \cdot n)$

Ejercicios

Otras operaciones del TAD se pueden definir a partir de las dadas:

1. Definir y dar los costos de las siguientes operaciones
   - $find : \mathbb{S}_A \rightarrow A \rightarrow Bool$, tal que $find\ S\ e = e\ \in\ S$
   - $insert : \mathbb{S}_A \rightarrow A \rightarrow \mathbb{S}_A$, tal que $insert\ S\ e = S\ \cup\ \{e\}$
   - $delete : \mathbb{S}_A \rightarrow A \rightarrow \mathbb{S}_A$, tal que $delete\ S\ e = S - \{e\}$
2. Dada la siguiente función

$$\begin{array}{ll} fromSeq & : \ Seq\ A \rightarrow \mathbb{S}_A \\ fromSeq\ s = reduce\ union\ empty\ (map\ singleton\ s) & \end{array}$$

   2.1 ¿A qué TAD pertenece cada operación en la definición?
   2.2 Suponiendo la comparación $O(1)$, mostrar que $fromSeq$ es trabajo $O(n \lg n)$ y profundidad $O(\lg^2 n)$.

Costos de las operaciones

- Los costos de las operaciones son:

Operación	**W**	**S**
fromSeq S	$O(\lvert S\rvert \lg \lvert S\rvert)$	$O\left(\lg^2 \lvert S\rvert\right)$
find S e *insert S e* *extract S e*	$O(C_W \lg \lvert S\rvert)$	$O(C_S \lg \lvert S\rvert)$

Tablas

- Las tablas se conocen con varios nombres:
  - Diccionarios, arreglos asociativos, mapas, funciones parciales ...
- Una tabla es un TAD que almacena un dato para cada clave.
- Son, esencialmente, el grafo de una función parcial.
  - Conjunto de pares clave-valor
  - Cada clave tiene asociado un solo valor
  - No necesitan estar todas las claves en el grafo.
- Notación:

$$\{(k_1, v_1), (k_2, v_2), \ldots, (k_n, v_n)\}$$

  o también

$$\{(k_1 \mapsto v_1), (k_2 \mapsto v_2), \ldots, (k_n \mapsto v_n)\}$$

- ¿Por qué no usar directamente el TAD Conjunto?

TAD Tabla

$$
\begin{array}{llll}
\multicolumn{4}{l}{\textbf{tad}\ \ \mathit{Tabla}\ (\mathbb{K} : \mathit{Set})\ [\mathbb{T}]\ (\mathit{Set})\ \textbf{where}} \\
\quad \mathit{empty} & : \mathbb{T}_{\mathbb{V}} & & = \emptyset \\
\quad \mathit{size} & : \mathbb{T}_{\mathbb{V}} \to \mathbb{N} & & = \lambda T \to |T| \\
\quad \mathit{singleton} & : (\mathbb{K} \times \mathbb{V}) \to \mathbb{T}_{\mathbb{V}} & & = \lambda(k, v) \to \{(k, v)\} \\
\quad \mathit{filter} & : (\mathbb{V} \to \mathit{Bool}) \to \mathbb{T}_{\mathbb{V}} \to \mathbb{T}_{\mathbb{V}} & & \\
 & & & = \lambda f\ T \to \{(k, v) \in T \mid f\ v\} \\
\quad \mathit{map} & : (\mathbb{K} \to \mathbb{V} \to \mathbb{V}') \to \mathbb{T}_{\mathbb{V}} \to \mathbb{T}_{\mathbb{V}'} & & \\
 & & & = \lambda f\ T \to \{(k, f\ k\ v) \mid (k, v) \in T\} \\
\quad \mathit{extract} & : \mathbb{T}_{\mathbb{V}} \to \mathbb{S}_{\mathbb{K}} \to \mathbb{T}_{\mathbb{V}} & & = \lambda T\ S \to \{(k, v) \in T \mid k \in S\} \\
\quad \mathit{erase} & : \mathbb{T}_{\mathbb{V}} \to \mathbb{S}_{\mathbb{K}} \to \mathbb{T}_{\mathbb{V}} & & = \lambda T\ S \to \{(k, v) \in T \mid k \notin S\} \\
\quad \mathit{merge} & : (\mathbb{V} \to \mathbb{V} \to \mathbb{V}) \to \mathbb{T}_{\mathbb{V}} \to \mathbb{T}_{\mathbb{V}} \to \mathbb{T}_{\mathbb{V}} & & \\
 & & & = \lambda f\ T\ T' \to \forall k \in \mathbb{K}
\end{array}
$$

$$
\begin{array}{llll}
 & (k, f\ v\ v') & \mathit{si} & (k, v) \in T \wedge (k, v') \in T' \\
\mathit{sino} & (k, v) & \mathit{si} & (k, v) \in T \vee (k, v) \in T'
\end{array}
$$

Especificación de costos

Operación	**W**	**S**
empty *singleton* *size*	$O(1)$	$O(1)$
filter f T	$O\left(\sum_{(k,v)\in T} W(f\ v)\right)$	$O\left(\lg \lvert T\rvert + \max_{(k,v)\in T} S(f\ v)\right)$
map f T	$O\left(\sum_{(k,v)\in T} W(f\ k\ v)\right)$	$O\left(\lg \lvert T\rvert + \max_{(k,v)\in T} S(f\ k\ v)\right)$
extract T T′ *merge f T T′* *erase T T′*	$O\left(C_W \cdot m \cdot \lg(1 + \frac{n}{m})\right)$	$O\left(C_S \cdot \lg(n+m)\right)$

- C_W y C_S son el trabajo y profundidad de comparar elementos.
- $n = \max(\lvert T\rvert, \lvert T'\rvert)$ y $m = \min(\lvert T\rvert, \lvert T'\rvert)$
- Si $n \sim m$ el trabajo de las últimas operaciones es $O(C_W \cdot n)$

Otras operaciones

$$
\begin{aligned}
\mathit{find} &: \mathbb{T}_{\mathbb{V}} \to \mathbb{K} \to \mathit{Maybe}\ \mathbb{V} \\
&= \lambda T\ k \to \begin{cases} \mathit{Just}\ v & \mathit{si}\ (k, v) \in T \\ \mathit{Nothing} & \mathit{en\ otro\ caso} \end{cases} \\
\mathit{insert} &: (\mathbb{V} \to \mathbb{V} \to \mathbb{V}) \to \mathbb{T}_{\mathbb{V}} \to (\mathbb{K} \times \mathbb{V}) \to \mathbb{T}_{\mathbb{V}} \\
&= \lambda f\ T\ (k, v) \to \\
&\quad \begin{cases} \{(k, v)\} \cup T & \mathit{si}\ \neg \exists v'.\ (k, v') \in T \\ \{(k, f\ v'\ v)\} \cup \mathit{delete}\ T\ k & \mathit{si}\ \exists v'.\ (k, v') \in T \end{cases} \\
\mathit{delete} &: \mathbb{T}_{\mathbb{V}} \to \mathbb{K} \to \mathbb{T}_{\mathbb{V}} \\
&= \lambda T\ k \to \{(k', v) \in T \vee k' \neq k\}
\end{aligned}
$$

Ejercicio:

1. ¿Cuáles de estas tres operaciones pueden definirse en base a las vistas anteriormente?
2. Argumentar en qué sentido el TAD Conjunto es un caso especial del TAD Tabla.

Costos de las operaciones

- Los costos de las operaciones son:

Operación	W	S
find T k *insert* T k *delete* T k	$O(C_W \lg \lvert T \rvert)$	$O(C_S \lg \lvert T \rvert)$

- Más operaciones de Tablas:

$domain : \mathbb{T}_{\mathbb{V}} \rightarrow \mathbb{S}_{\mathbb{K}}$	$range : \mathbb{T}_{\mathbb{V}} \rightarrow Seq\ \mathbb{V}$	
domain T *range* T	$O(\lvert T \rvert)$	$O(\lg \lvert T \rvert)$

$tabulateT : (\mathbb{K} \rightarrow \mathbb{V}) \rightarrow \mathbb{S}_{\mathbb{K}} \rightarrow \mathbb{T}_{\mathbb{V}}$		
tabulateT f S	$O\left(\sum_{k \in S} W(f\ k)\right)$	$O\left(\max_{k \in S} S(f\ k)\right)$

Ejemplos

- Podemos definir *reduce* sobre tablas haciendo reduce sobre la secuencia rango

$$
\begin{aligned}
&\mathit{reduceT} : (\mathbb{V} \to \mathbb{V} \to \mathbb{V}) \to \mathbb{V} \to \mathbb{T}_{\mathbb{V}} \to \mathbb{V} \\
&\mathit{reduceT} \oplus b\ T = \mathit{reduce} \oplus b\ (\mathit{range}\ T)
\end{aligned}
$$

- Podemos definir *collect* sobre tablas

$$
\begin{aligned}
&\mathit{collectT} \quad : \mathit{Seq}\ (\mathbb{K}\ \times\ \mathbb{V}) \to \mathbb{T}\ (\mathit{Seq}\ \mathbb{V}) \\
&\mathit{collectT}\ s = \mathit{fromSeq}\ .\ \mathit{collect}
\end{aligned}
$$

- En una implementación real, estas operaciones se podrían hacer directamente sobre la representación interna

Grafos

- Los grafos son utilizados para modelar infinidad de problemas
- Existen diferentes formas de representar un grafo.
- Cuál elegir depende de las operaciones que se quieran realizar sobre el grafo
- Las principales representaciones de un grafo (V, E) $(n = |V|, m = |E|)$ son:
  - Matriz de adyacencia:
    - matriz de $n \times n$, $M(i, j) = ((i, j) \in E)$.
  - Lista de adyacencia:
    - arreglo de tamaño n, $A(n)$ es una lista con los ́vertices vecinos al vértice n
  - Lista de lados
    - Una lista de pares $(i, j) \in E$.

Operaciones sobre grafos

- Algunas de las operaciones que comúnmente se usan son:
  1. *deg* G v, encontrar el grado de un vértice;
  2. *eslado* G (i, j), encontrar si un lado pertenece al grafo;
  3. realizar un *map* de una función sobre los lados;
  4. iterar sobre todos los vecinos de un vértice.
- Notar que hay diferentes variantes de grafos (dirigidos vs no dirigidos), con pesos en los lados, con pesos en los vértices, etc.
- Las representaciones y operaciones cambiarán levemente según el tipo de grafo.

Conjunto de lados

- Nos interesan representaciones con operaciones paralelizables
- La definición matemática dice que tenemos un *conjunto de lados*.
- Podemos usar esto como representación si lo implementamos con el TAD de conjuntos.
- Nos independizamos de la estructura subyacente al TAD conjuntos (comparar con "lista de lados").
- Si tomamos los costos de la implementación con árboles balanceados:
  - Buscar un lado es barato ($O(\lg m)$)
  - Buscar vecinos, no ($\Theta(m)$)

Tabla de adyacencia

- Para obtener rápido acceso a los vecinos podemos usar una *tabla de adyacencia*
- Cada vértice (claves) es mapeado a un conjunto de vecinos.
- Acceder los vecinos es barato (buscar un elemento en la tabla es $O(\lg n)$).
- Buscar un lado sigue siendo $O(\lg n)$.
- Es una abstracción de la lista de adyacencia.

Resumen de costos

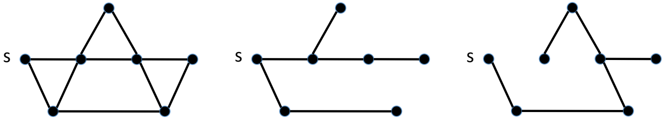
- Los siguientes costos suponen una implementación de conjuntos y tablas con árboles balanceados

	Conj de lados		**Tabla de adyacencia**	
Operación	**W**	**S**	**W**	**S**
esLado $g\ (u, v)$	$O(\lg n)$	$O(\lg n)$	$O(\lg n)$	$O(\lg n)$
map sobre lados	$O(m)$	$O(\lg n)$	$O(m)$	$O(\lg n)$
map sobre vecinos	$O(m)$	$O(\lg n)$	$O(\lg n + d_G(v))$	$O(\lg n)$
$d_G(v)$	$O(m)$	$O(\lg n)$	$O(\lg n)$	$O(\lg n)$

- $n = \|V\|$
- $m = \|E\|$

Buscando en Grafos

- Dado un vértice s, se visitan nuevos vértices tachando lados.
  - Ningún vértice se visita dos veces.
  - Hay que recordar que vértices se visitaron.
- Los dos métodos más estándar son DFS y BFS.
  - Visitan todos los vértices que se pueden alcanzar desde s,
  - pero en distinto orden:
  - BFS (Breadth First Search): Visitar todos los vecinos en la frontera, expandir la frontera y repetir.
  - DFS (Depth First Search): Explorar tan lejos como sea posible antes de retomar el último camino sin explorar.
- Generan un árbol de búsqueda (implícito o explicíto).



Usos de Búsqueda en Grafos

- Pueden ser usados, por ejemplo, para:
  - Generar un árbol de expansión
  - Verificar si un grafo es conexo
- DFS:
  - Ordenamiento topológico.
  - Encontrar componentes fuertemente conectadas.
  - Test de planaridad.
- BFS:
  - Encontrar camino con menos lados entre dos vértices.
  - Determinar si un grafo es bipartito.
  - Encontrar el flujo máximo de una red.

Buscando en paralelo

- DFS es inherentemente secuencial
  - No podemos empezar otra búsqueda hasta que la actual no termine.
- BFS tiene buen paralelismo cuando el grafo es *playo* (los caminos más cortos entre el origen y los otros vértices son razonablemente chicos).
- Sin embargo, en la práctica, muchos grafos son playos.

Algoritmo BFS

- Sea v el vértice origen.
- Sea X_i los vértices visitados en el paso i ($X_0 = \{v\}$).
- La *frontera* F_i representa los nodos agregados en el paso i

$$F_0 = \{v\} \qquad F_{i+1} = N_G(F_i) - X_i$$

(donde $N_G(S) = \cup_{v \in S}(getNbrs\ G\ v)$)

- En pseudocódigo:

$$\begin{aligned}
bfs\ G\ s = \textbf{let}\ & bfs'\ X\ \emptyset\ \ i = (X, i) \\
& bfs'\ X\ F\ i = \textbf{let}\ F' = N_G\ F \\
& \qquad\qquad\qquad \textbf{in}\ bfs'\ (X\ \cup\ F')\ (F' - X)\ (i+1) \\
\textbf{in}\ & bfs'\ \{s\}\ \{s\}\ 0
\end{aligned}$$

Costo de BFS

- Suponemos $Graph = Table\ V\ \mathbb{S}_V$
- El costo de BFS es determinado por el costo de N_G que depende de la estructura del grafo.

$$
\begin{array}{l}
getNbrs : Graph \to V \to \mathbb{S}_V \\
getNbrs\ G\ v = \mathbf{case}\ (find\ G\ v)\ \mathbf{of} \\
\qquad\qquad\qquad\qquad Nothing \to empty \\
\qquad\qquad\qquad\qquad Just\ a \to a
\end{array}
$$

$$
\begin{array}{l}
N \quad\ : Graph \to \mathbb{S}_V \to \mathbb{S}_V \\
N_G\ F = \mathbf{let}\ nghs = extract\ G\ F \\
\qquad\quad \mathbf{in}\ reduce\ (merge\ (\lambda x\ y \to x))\ empty\ nghs
\end{array}
$$

- Se puede demostrar que $W_{bfs} = O(m \lg n)$ y $S_{bfs} = O(d \lg^2 n)$
- donde $n = V$, $m = |E|$, y d es la cantidad de pasos de bfs (long del mas largo de los caminos mas cortos).

Resumen

- TAD de Conjuntos
- TAD de Tablas
- Representaciones de Grafos
- Búsqueda paralelizable $\rightarrow$ BFS