IIC2283 — Diseño y Análisis de Algoritmos
Ayudantía 11

Grupos conmutativos

Un grupo G es conmutativo cuando para todos $x, y \in G$, se cumple que $xy = yx$. Como notación, se definirá que $[a, b] = a^{-1}b^{-1}ab$.

Parte 1 Demuestre que $ab = ba$ si y solo si $[a, b] = 1$.

Solución

$$\begin{aligned} ab &= ba \\ b^{-1}ab &= b^{-1}ba \\ b^{-1}ab &= a \\ a^{-1}b^{-1}ab &= a^{-1}a \\ [a, b] &= 1 \end{aligned}$$

Parte 2 Decimos que un grupo G es generado por $S \subseteq G$ si todo elemento $g \in G$ se puede expresar como el producto de elementos de S, o inversos de elementos de S. Es decir, si $g \in G$, entonces $g = x_1 \cdots x_n$, donde x_i o x_i^{-1} pertenece a S.

Desarrolle y analice un algoritmo que dados un conjunto finito $(g_1, \ldots, g_n)$, y una operación binaria como caja negra, determine si $\langle g_1, \ldots, g_n \rangle$ es conmutativo.

Al decir que la operación binaria funciona como caja negra, nos referimos a que es posible obtener el resultado de $a \circ b$ en tiempo $O(1)$, pero no se conoce el funcionamiento interno de la función.

Solución Evaluar todos los generadores entre sí. Por cada par de generadores g_i, g_j, se verifica usando el operador que $[g_i, g_j]$ sea igual a 1. En caso de que algún par de generadores no conmute, entonces directamente se obtiene que el grupo no es conmutativo, y si todos los generadores conmutan, entonces es conmutativo.

Para la última parte, se demuestra que si $[g_i, g_j] = 1$, entonces $[g_i^{-1}, g_j] = 1$:

$$\begin{aligned} [g_i, g_j] &= 1 \\ g_i^{-1}g_j^{-1}g_ig_j &= 1 \\ g_ig_j &= g_jg_i \\ g_j &= g_i^{-1}g_jg_i \\ g_jg_i^{-1} &= g_i^{-1}g_j \\ g_i^{-1} &= g_j^{-1}g_i^{-1}g_j \\ 1 &= g_ig_j^{-1}g_i^{-1}g_j \\ [g_i^{-1}, g_j] &= 1 \end{aligned}$$

y luego, que si $ab \in G$, entonces $ba \in G$. Por definición de grupo generado se tiene que si $a, b \in G$, entonces:

$$a = x_1 \cdots x_n$$
$$b = y_1 \cdots y_m$$

con $x_i, y_j \in \{g_1, \ldots, g_n, g_1^{-1}, \ldots, g_n^{-1}\}$. Luego, como los generadores y sus inversos conmutan entre sí, se cumple que:

$$\begin{aligned} ab &= x_1 \cdots x_n \cdot y_1 \cdots y_m \\ &= y_1 \cdots y_m \cdot x_1 \cdots x_n \\ &= ba \end{aligned}$$

Parte 3 Dado un grupo G, se define el centro de G como:

$$Z(G) = \{x \in G | \text{ para todo } y \in G \text{ se cumple que } [x, y] = 1\}$$

y además, para cada elemento $g \in G$, el centralizador de g como:

$$C(g) = \{x \in G | [x, g] = 1\}$$

Demuestre que $Z(G)$ es un subgrupo de G, y que para todo $g \in G$, $C(g)$ también es un subgrupo.

Idea de solución Si $g \in Z(G)$, entonces $g^{-1} \in Z(G)$ por el argumento similar al de la parte 2. Además, si $a, b \in Z(G)$, y $t \in G$ es arbitrario, entonces:

$$\begin{aligned} abt &= atb \\ &= tab \end{aligned}$$

y por lo tanto $ab \in Z(G)$.

El caso con los centralizadores es similar.

Parte 4 Dado un grupo $G = \langle g_1, \ldots, g_n \rangle$, definimos un subproducto aleatorio como

$$g_1^{\epsilon_1} \cdots g_n^{\epsilon_n} \in G$$

donde los valores de $\{\epsilon_i\}_i$ son elegidos de forma uniforme e independiente desde $\{0, 1\}$.

Demuestre que si H es un subgrupo propio de G, entonces:

$$Pr[g_1^{\epsilon_1} \cdots g_n^{\epsilon_n} \notin H] \geq \frac{1}{2}$$

Solución Como H es distinto de G, entonces no contiene a todos los generadores. Si definimos ℓ como

$$\ell = \min_i \{i | g_i \notin H\},$$

entonces podemos expresar el subproducto aleatorio como $u g_\ell^{\epsilon_\ell} v$, donde $u = g_1^{\epsilon_1} \cdots g_{\ell-1}^{\epsilon_{\ell-1}}$, y $v = g_{\ell+1}^{\epsilon_{\ell+1}} \cdots g_n^{\epsilon_n}$.

Luego, por definición de ℓ se tiene que $u \in H$, y se separan las probabilidades según v. Como notación, se define $r = g_1^{\epsilon_1} \cdots g_n^{\epsilon_n}$:

$$\begin{aligned} Pr[r \notin H] &= Pr[r \notin H | v \in H] Pr[v \in H] + Pr[r \notin H | v \notin H] Pr[v \notin H] \\ &= 1/2 Pr[v \in H] + Pr[r \notin H | v \notin H] Pr[v \notin H] \\ &\geq 1/2 Pr[v \in H] + 1/2 Pr[v \notin H] \\ &= 1/2 \end{aligned}$$

En la primera parte, se usa que si $v \in H$ entonces el producto no pertenece a H si y solo si ϵ_ℓ es 1, lo que sucede con probabilidad $1/2$.

En la segunda, se considera que si ϵ_ℓ es 0, entonces el producto no pertenece a H, y por lo tanto que: $Pr[r \notin H | v \notin H] \geq Pr[\epsilon_\ell = 0] = 1/2$.

Parte 5 Describa y analice un algoritmo aleatorizado que dados generadores $(g_1, \ldots, g_n)$, y una operación binaria como caja negra, determine si $\langle g_1, \ldots, g_n \rangle$ es conmutativo.

Idea de solución El algoritmo genera dos subproductos aleatorios independientes h, h', y luego verifica si $[h, h'] = 1$. En este caso se realiza una cantidad $O(n)$ de multiplicaciones.

Si el grupo es conmutativo, entonces el algoritmo entrega la respuesta correcta con probabilidad 1, por que $[h, h']$ debe ser igual a 1.

Si el grupo no es conmutativo, entonces se debe evaluar la probabilidad de que $[h, h'] = 1$.

$$\begin{aligned} Pr[[h, h'] \neq 1] &= Pr[h' \notin C(h)] \\ &= Pr[h' \notin C(h) | h \in Z(G)] Pr[h \in Z(G)] + Pr[h' \notin C(h) | h \notin Z(G)] Pr[h \notin Z(G)] \\ &= 0 Pr[h \in Z(G)] + Pr[h' \notin C(h) | h \notin Z(G)] Pr[h \notin Z(G)] \\ &\geq 1/2 \cdot 1/2 \\ &= 1/4 \end{aligned}$$

Luego, la probabilidad de error cuando G no es conmutativo es $Pr[[h, h'] = 1] \leq 3/4$.

Evaluación de árboles booleanos

Se tiene como input un árbol binario completo con profundidad $2n$. Las hojas del árbol tienen valores de $\{0, 1\}$, y los nodos internos tienen conectivos $\wedge, \vee$, que se alternan de acuerdo a la profundidad, es decir, los hijos de un nodo $\wedge$ son nodos $\vee$, y los hijos de un nodo $\vee$ son nodos $\wedge$. El costo de evaluar un árbol de este tipo se define como cantidad de hojas revisadas.

Desarrolle y analice un algoritmo aleatorizado que permita evaluar un árbol de este tipo con costo esperado menor a la cantidad de hojas. El algoritmo debe retornar una evaluación correcta con probabilidad 1.

Idea de solución El algoritmo se basa en que si un árbol tiene raíz $\wedge$, y se determinó que un subárbol tiene valor 0, entonces no es necesario evaluar el otro, y de forma similar cuando la raíz es $\vee$, y se determina que un subárbol tiene valor 1.

Más específicamente, el algoritmo evalúa recursivamente el árbol desde la raíz, y por cada nodo elige aleatoriamente cual subárbol evaluar primero. El segundo subárbol se evalúa solo si es necesario.

El costo promedio de evaluar un árbol de altura $2t$ es 3^t, lo que se demuestra por inducción en la altura.

Si el árbol tiene altura 2 hay 4 casos:

1. Raíz $\wedge$ que evalúa a 1
2. Raíz $\wedge$ que evalúa a 0
3. Raíz $\vee$ que evalúa a 1
4. Raíz $\vee$ que evalúa a 0

En el caso 1, los subárboles tienen valor 1, y por lo tanto se deben evaluar ambos. Por otra parte, los subárboles tienen raíz $\vee$ con salida 1, y por lo tanto hay dos posibilidades:

- Ambas hojas son 1, en cuyo caso solo se evalúa una hoja, o
- Una hoja es 1, y la otra 0. En este caso, la cantidad esperada de hojas evaluadas es:

  $$1/2 \cdot 1 + 1/2 \cdot 2 = 3/2$$

  donde el valor depende de cual subárbol se evalúa primero: Si se elige primero la hoja con valor 1, se evalúa una hoja, y si se elige primero la con valor 0, se evalúan 2.

En ambos casos la cantidad promedio de hojas es menor o igual a $3/2$, por lo que en promedio se evalúan a lo más $2 \cdot 3/2 = 3$ hojas en total. En los otros casos también se puede demostrar que el promedio de hojas es menor o igual a 3.

Finalmente, si la propiedad se cumple para árboles de altura $2n$, entonces en un árbol de altura $2(n+1)$ se consideran los dos primeros niveles. Como el árbol esta fijo, los nodos a profundidad 2 tienen valores determinados, y por el mismo argumento del caso base se necesitan evaluar 3 de ellos en promedio. Luego, como los árboles desde esos nodos tienen altura $2n$, entonces por hipótesis de inducción se revisan 3^n hojas en cada uno, y por lo tanto en todo el árbol el costo promedio es menor o igual a 3^{n+1}.

Corte mínimo en grafos no dirigidos

Dado un grafo no dirigido $G = (V, E)$, se consideran particiones de nodos en dos conjuntos S_1, S_2. Luego, se define el costo de la partición como la cantidad de arcos del grafo original, que tengan un extremo en cada parte.

Un algoritmo aleatorizado que resuelve este problema está dado en el algoritmo 1.

Parte 1 Analice el tiempo de ejecución del algoritmo.

Idea de solución El algoritmo realiza $|V|$ contracciones de arcos, cuyo costo depende de la representación del grafo. Usando directamente una matriz de adyacencia, cada contracción toma tiempo $O(|V|)$ por que se deben revisar 2 filas y 2 columnas, y por lo tanto el tiempo de ejecución del algoritmo es $O(|V|^2)$.

Parte 2 Dado un corte mínimo fijo, se define el conjunto de arcos con un extremo en cada parte como F, con $|F| = k$, y también se define que $n = |V|$.

Demuestre que el grafo tiene al menos $\frac{kn}{2}$ arcos, y acote la probabilidad de elegir aleatoriamente un arco de F.

Solución Se utiliza que si un nodo tiene grado g, entonces el corte mínimo debe involucrar al menos g arcos. Lo anterior se cumple por que para todo nodo, siempre existe el corte formado por ese nodo, y todos los otros, cuyo costo es igual al grado del nodo.

Con el argumento anterior, si el corte mínimo tiene costo k, entonces todos los nodos tienen grado mayor o igual a k, y por lo tanto existen al menos $\frac{kn}{2}$ arcos.

Luego, la probabilidad de que un arco elegido aleatoriamente pertenezca a F es a lo más

$$\frac{k}{(kn/2)} = \frac{2}{n}.$$

Parte 3 Acote la probabilidad de que en la iteración $j+1$ se contraiga un arco de F, dado que hasta la iteración j no se ha contraído un arco de ese conjunto.

Solución Tras j iteraciones hay $n-j$ nodos, ya que cada iteración contrae dos nodos en uno. Luego, como no se han contraído arcos de F, entonces la probabilidad buscada es a lo más $2/(n-j)$.

Parte 4 Determine la probabilidad de que no se contraiga un arco de F tras $n-2$ iteraciones, y acote la probabilidad de error del algoritmo.

Idea de solución Si denominamos E_i a la variable aleatoria que indica que en la iteración i no se eligió un arco de F, entonces la probabilidad es:

$$P = Pr[E_1] \cdot Pr[E_2|E_1] \cdot Pr[E_3|E_2, E_1] \cdots Pr[E_{n-2}|E_{n-3}, \ldots, E_1]$$

Con la parte 3, esto se puede acotar por

$$\begin{aligned} P &\geq \left(1 - \frac{2}{n}\right) \cdot \left(1 - \frac{2}{n-1}\right) \cdots \left(1 - \frac{2}{3}\right) \\ &= \left(\frac{n-2}{n}\right) \cdot \left(\frac{n-3}{n-1}\right) \cdots \left(\frac{1}{3}\right) \\ &= \frac{2 \cdot (n-2)!}{n!} \\ &= \binom{n}{2}^{-1} \end{aligned}$$

Luego la probabilidad de error es a lo más $1 - 1/\binom{n}{2}$. Para reducir la probabilidad de error bajo alguna constante se puede repetir el algoritmo $\binom{n}{2}$ veces, con lo cual el tiempo de ejecución aumenta a $O(n^4)$. En ese caso la probabilidad de error es menor o igual a $1/e$.

Algorithm 1 Algoritmo de Karger

```
function KARGER(G = (V, E))
    for v ∈ V do
        S[v] ← {v}
    end for
    while |V| > 2 do
        Elegir e = (u, v) desde E de forma uniforme
        Contraer e, es decir reemplazar los extremos del arco por un único nodo z_uv, tal que E(z_uv, x) ssi E(u, x) o E(v, x)
        S[z_uv] ← S[u] ∪ S[v]
    end while
    (v_1, v_2) ← V
    return (S[v_1], S[v_2])
end function
```