

Algunas cosas con árboles : Tree Rerooting + Euler Tour + Heavy-Light

Agustín Santiago Gutiérrez

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

Training Camp 2021

Contenidos

1

Truquitos con cuentitas

- Ejemplos
- Multiconjuntos co-unitarios
- Caso combinable

2

Cambio de raíz

- Planteo del problema
- Algoritmo para cambio de raíz
- Caso típico combinable
- Ejemplos no combinables

3

“Euler Tour” (o simplemente DFS en árbol)

- Euler Tour

4

Descomposición Heavy-Light

- Definición
- Uso para subir k veces
- Uso para LCA
- Segment Tree en caminos

A fool sees not the same tree that a wise man sees.

William Blake

When you start to treat the light weights like heavy weights, the heavy weights will go up a lot easier.

Edward "Ed" Ignatius Coan

Contenidos

1 Truquitos con cuentitas

Ejemplos

- Multiconjuntos co-unitarios
- Caso combinable

2 Cambio de raíz

- Planteo del problema
- Algoritmo para cambio de raíz
- Caso típico combinable
- Ejemplos no combinables

3 “Euler Tour” (o simplemente DFS en árbol)

- Euler Tour

4 Descomposición Heavy-Light

- Definición
- Uso para subir k veces
- Uso para LCA
- Segment Tree en caminos

Algunas cuentitas para hacer

Podríamos querer **calcular algo** sobre un multiconjunto de valores:

- Suma de pares: $ab + ac + ad + bc + bd + cd$ ($n = 4$)
- Suma de subconjuntos: $1 + a + b + c + ab + ac + ac + abc$ ($n = 3$)
- Cuántos positivos hay
- El máximo
- Los dos elementos más grandes
- El promedio

Algunas cuentitas para hacer (cont)

- Suma de pares: $\frac{(a+b+c+d)^2 - (a^2 + b^2 + c^2 + d^2)}{2}$ para $n = 4$
- Suma de subconjuntos: $(1 + a)(1 + b)(1 + c)$ para $n = 3$
- Las otras eran fáciles en $O(N)$

Contenidos

1

Truquitos con cuentitas

- Ejemplos
- **Multiconjuntos co-unitarios**
- Caso combinable

2

Cambio de raíz

- Planteo del problema
- Algoritmo para cambio de raíz
- Caso típico combinable
- Ejemplos no combinables

3

“Euler Tour” (o simplemente DFS en árbol)

- Euler Tour

4

Descomposición Heavy-Light

- Definición
- Uso para subir k veces
- Uso para LCA
- Segment Tree en caminos

Desafío: calcular para todos los co-unitarios

- Dado un multiconjunto, llamamos sus multiconjuntos *co-unitarios*, a los que se obtienen sacando exactamente un elemento
- Dado el multiconjunto $\{a, b, c, d\}$ original, los co-unitarios son $\{b, c, d\}$, $\{a, c, d\}$, $\{a, b, d\}$ y $\{a, b, c\}$
- Para cada “cuenta” anterior, queremos calcularla para **todos** los co-unitarios eficientemente. Idealmente en $O(N)$

Solución al desafío para suma de pares

- Hacemos la suma total $s = a + b + c + d$ y la suma de cuadrados total $q = a^2 + b^2 + c^2 + d^2$
- Al excluir al x , el co-unitario tendrá $s' = s - x$ y $q' = q - x^2$
- Como vimos antes, el valor es $\frac{s'^2 - q'}{2}$

Solución al desafío para suma de subconjuntos

- Hacemos el producto total $p = (1 + a)(1 + b)(1 + c)(1 + d)$
- Al excluir al x , el co-unitario tendrá $p' = \frac{p}{1+x}$
- Como vimos antes, p' es el valor buscado

Solución al desafío para cantidad de positivos

- Llamamos t a la cantidad total de positivos
- Al excluir a x , da $t - 1$ si $x > 0$ o bien t si no

Solución al desafío para el máximo

- Calculamos los dos elementos más grandes $m_1 \geq m_2$
- Al excluir a x , da m_2 si $x = m_1$ o bien m_1 si no

Solución al desafío para los dos elementos más grandes

- Calculamos los tres elementos más grandes $m_1 \geq m_2 \geq m_3$
- Al excluir a x , da
 - (m_2, m_3) si $x = m_1$
 - (m_1, m_3) si $x = m_2$
 - o bien (m_1, m_2) en otro caso

Solución al desafío para el promedio

- Calculamos la suma total s
- Al excluir a x , da $\frac{s-x}{n-1}$

Desventajas

- Cada caso fue un truquito diferente para pensar
- Algunos necesitan un inverso que puede no existir

Contenidos

1

Truquitos con cuentitas

- Ejemplos
- Multiconjuntos co-unitarios
- **Caso combinable**

2

Cambio de raíz

- Planteo del problema
- Algoritmo para cambio de raíz
- Caso típico combinable
- Ejemplos no combinables

3

“Euler Tour” (o simplemente DFS en árbol)

- Euler Tour

4

Descomposición Heavy-Light

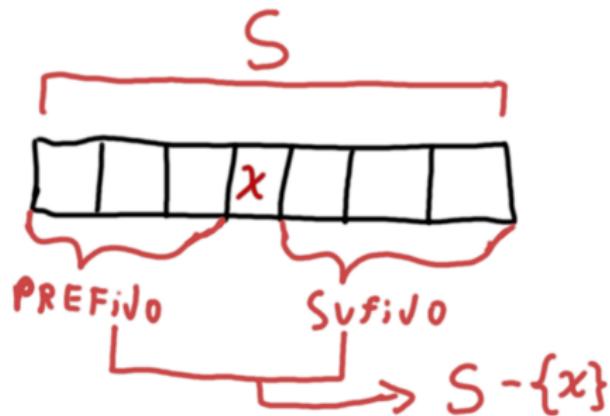
- Definición
- Uso para subir k veces
- Uso para LCA
- Segment Tree en caminos

Cálculo combinable

- Buscamos una forma más sistemática
- Un cálculo será *combinable*, si podemos usar los resultados sobre A y sobre B , para calcular sobre $A \cup B$
- Asumimos que sabemos calcular:
 - El conjunto vacío \emptyset
 - Los conjuntos unitarios $\{x\}$

Truco de prefijos y sufijos

- Si es combinable, podemos armar dos arreglos de $n + 1$ elementos $p[0..n]$ y $s[0..n]$
- p_i tendrá el cálculo sobre $\{x_0, x_1, \dots, x_{i-1}\}$, el rango $[0, i)$
- s_j tendrá el cálculo sobre $\{x_j, x_{j+1}, \dots, x_{n-1}\}$, el rango $[j, n)$
- Al excluir al i -ésimo, da $combinar(p_i, s_{i+1})$



Llenando p y s

- $p_0 = \text{valor}(\emptyset)$
- $p_i = \text{combinar}(p_{i-1}, \text{valor}(\{x_{i-1}\}))$
- $s_n = \text{valor}(\emptyset)$
- $s_j = \text{combinar}(s_{j+1}, \text{valor}(\{x_j\}))$
- Solo necesitamos *combinar* y los valores de *vacio* y de *unitarios*

¡Eran todos combinables!

- Suma de pares:
 - $valor(\emptyset) = (0, 0)$
 - $valor(\{x\}) = (x, 0)$
 - $combinar((s_1, p_1), (s_2, p_2)) = (s_1 + s_2, p_1 + p_2 + s_1 \cdot s_2)$
- Suma de subconjuntos:
 - $valor(\emptyset) = 1$
 - $valor(\{x\}) = 1 + x$
 - $combinar(a, b) = a \cdot b$
- Cuántos positivos hay:
 - $valor(\emptyset) = 0$
 - $valor(\{x\}) = 1 \text{ si } x > 0 \text{ y } 0 \text{ si no}$
 - $combinar(a, b) = a + b$

¡Eran todos combinables! (cont.)

- El máximo:

- $valor(\emptyset) = -\infty$
- $valor(\{x\}) = x$
- $combinar(a, b) = \max(a, b)$

- Los dos elementos más grandes:

- $valor(\emptyset) = (-\infty, -\infty)$
- $valor(\{x\}) = (x, -\infty)$
- $comb((a_1, b_1), (a_2, b_2)) = (\max(a_1, a_2), \max(\min(a_1, a_2), b_1, b_2))$

- El promedio:

- $valor(\emptyset) = (0, 0)$
- $valor(\{x\}) = (s, 1)$
- $combinar((s_1, n_1), (s_2, n_2)) = (s_1 + s_2, n_1 + n_2)$
- El promedio es simplemente $\frac{s}{n}$

Relación con la asociatividad [para los matemáticos]

- Si lo que buscamos calcular es el resultado de aplicar una operación asociativa sobre todos los elementos del multiconjunto, tenemos un valor combinable
- De $\text{valor}(A \cup B) = \text{combinar}(\text{valor}(A), \text{valor}(B))$ se desprende que la operación de combinar va a ser asociativa sobre los valores posibles
- No decimos directamente que valor es asociativo, porque no es operación binaria: transforma directamente un multiconjunto cualquiera, en un valor

Contenidos

1 Truquitos con cuentitas

- Ejemplos
- Multiconjuntos co-unitarios
- Caso combinable

2 Cambio de raíz

- **Planteo del problema**
- Algoritmo para cambio de raíz
- Caso típico combinable
- Ejemplos no combinables

3 “Euler Tour” (o simplemente DFS en árbol)

- Euler Tour

4 Descomposición Heavy-Light

- Definición
- Uso para subir k veces
- Uso para LCA
- Segment Tree en caminos

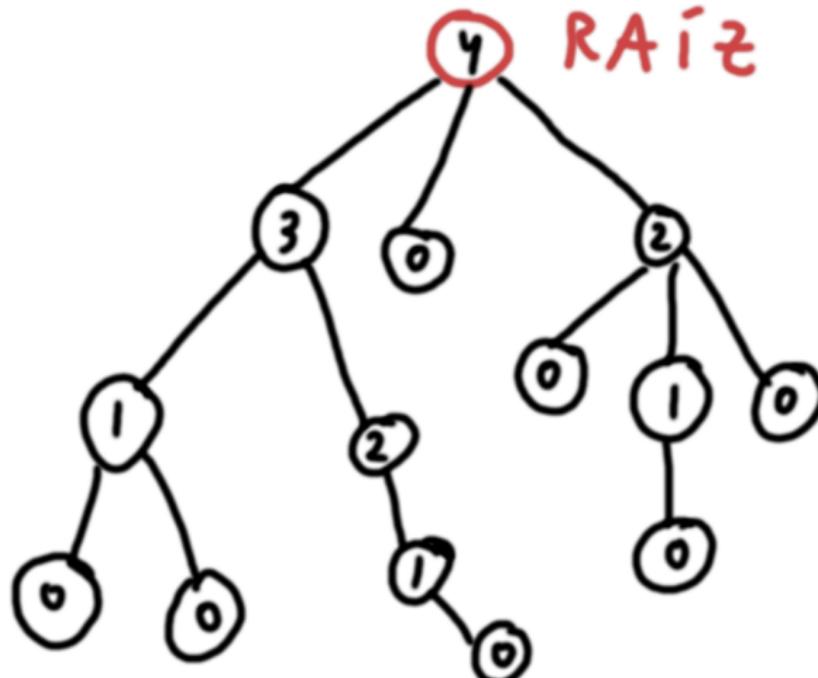
Recursión en árbol con raíz

Dado un árbol **con raíz**:

- Una *recursión* consiste en calcular en cada nodo **interno** (no hoja) un cierto *valor*(S), donde S es el multiconjunto de valores ya calculados recursivamente sobre sus **hijos**
- Las **hojas** serán los **casos base**
- No necesariamente valen todas las hojas lo mismo

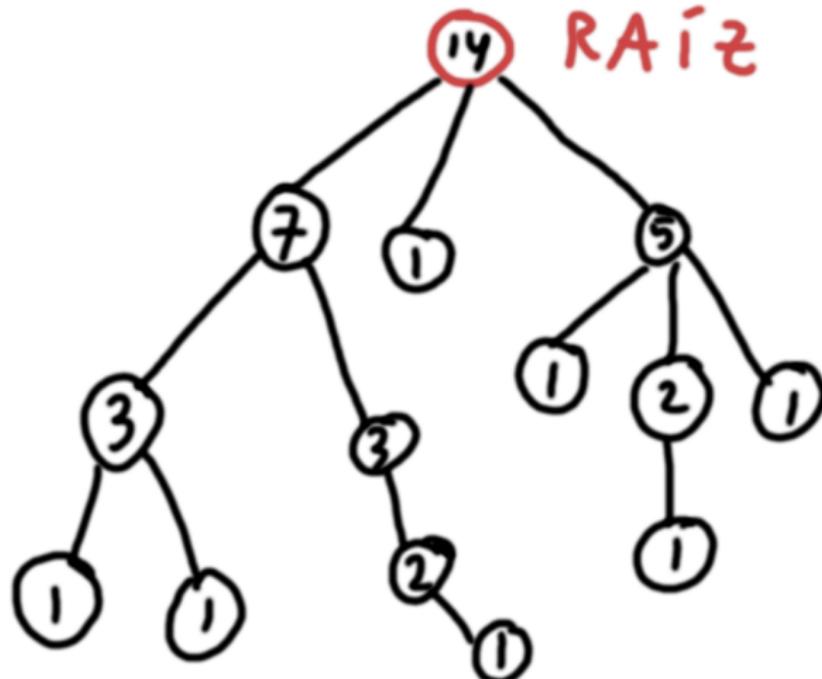
Ejemplo: altura de subárbol

- $\text{valor}(S) = 1 + \max_{x \in S} x$
- $\text{valor}(h) = 0$ en cada hoja



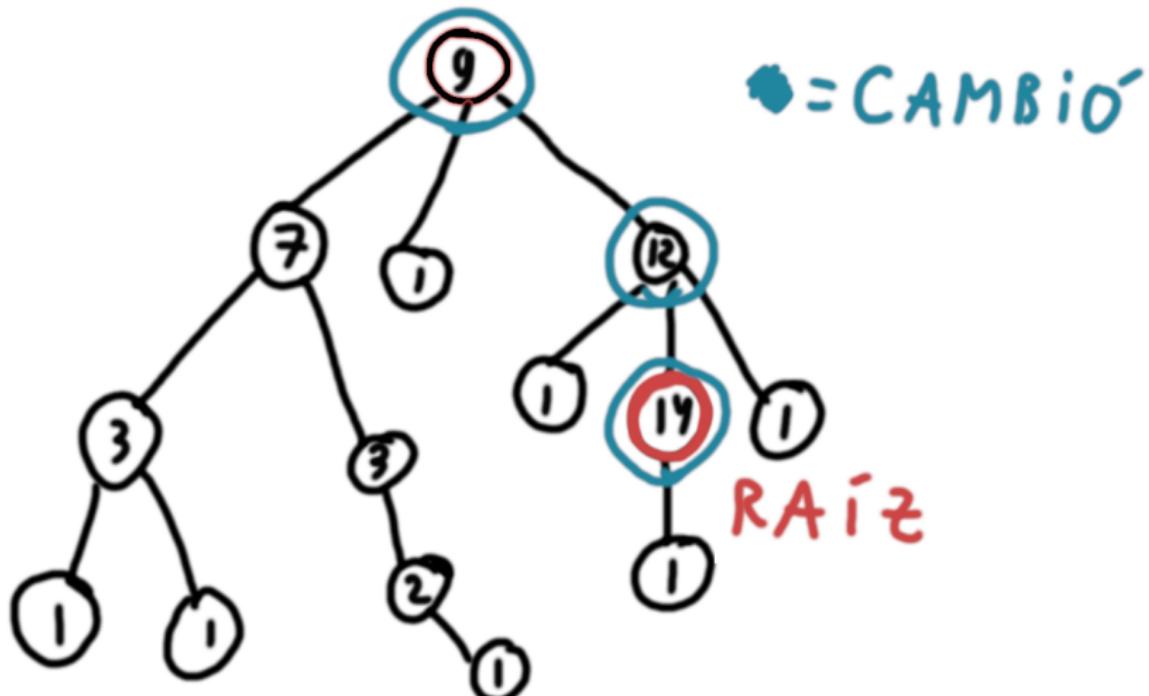
Ejemplo: tamaño de subárbol

- $\text{valor}(S) = 1 + \sum_{x \in S} x$
- $\text{valor}(h) = 1$ en cada hoja



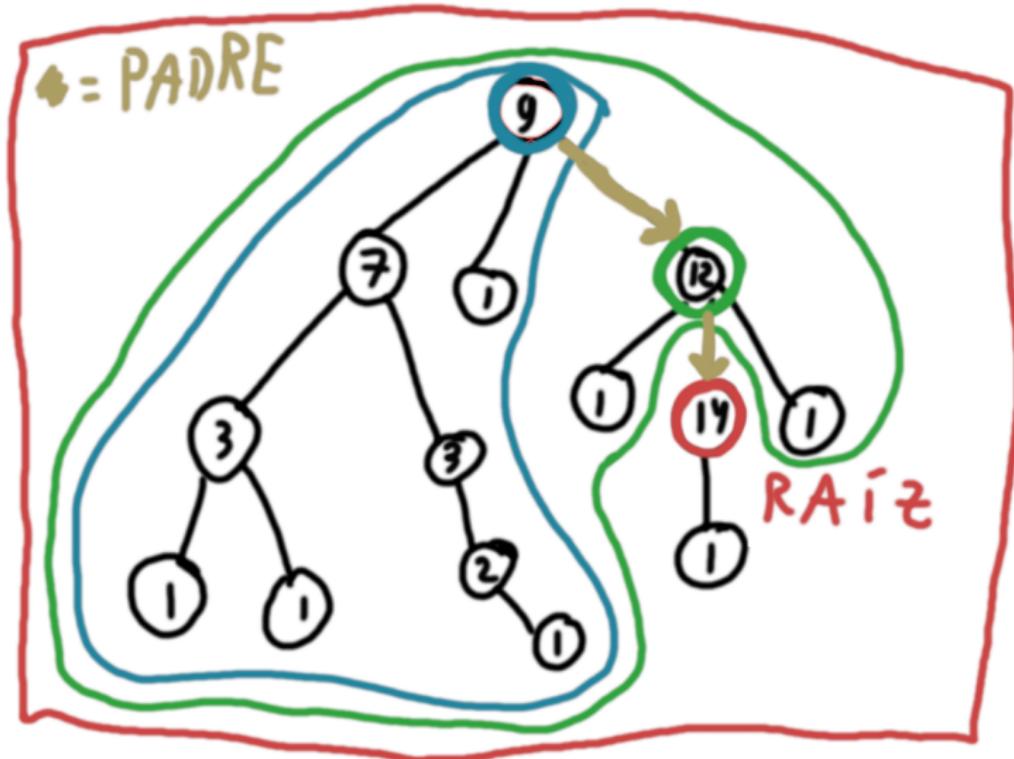
Cambio de raíz

En ambos casos anteriores, los valores cambian si se **cambia la raíz**
Ejemplo con tamaño de subárbol:



Cambio de raíz (cont.)

Cambian los padres, ergo, cambian los subárboles de cada nodo



Nuestro sueño

- Calcular el valor en la raíz, para **todas** las raíces
- Más aún: Con cada raíz calcular **todos los subárboles**
- Que sea eficiente: idealmente todo en $O(N)$

Contenidos

1 Truquitos con cuentitas

- Ejemplos
- Multiconjuntos co-unitarios
- Caso combinable

2 Cambio de raíz

- Planteo del problema
- **Algoritmo para cambio de raíz**
- Caso típico combinable
- Ejemplos no combinables

3 "Euler Tour" (o simplemente DFS en árbol)

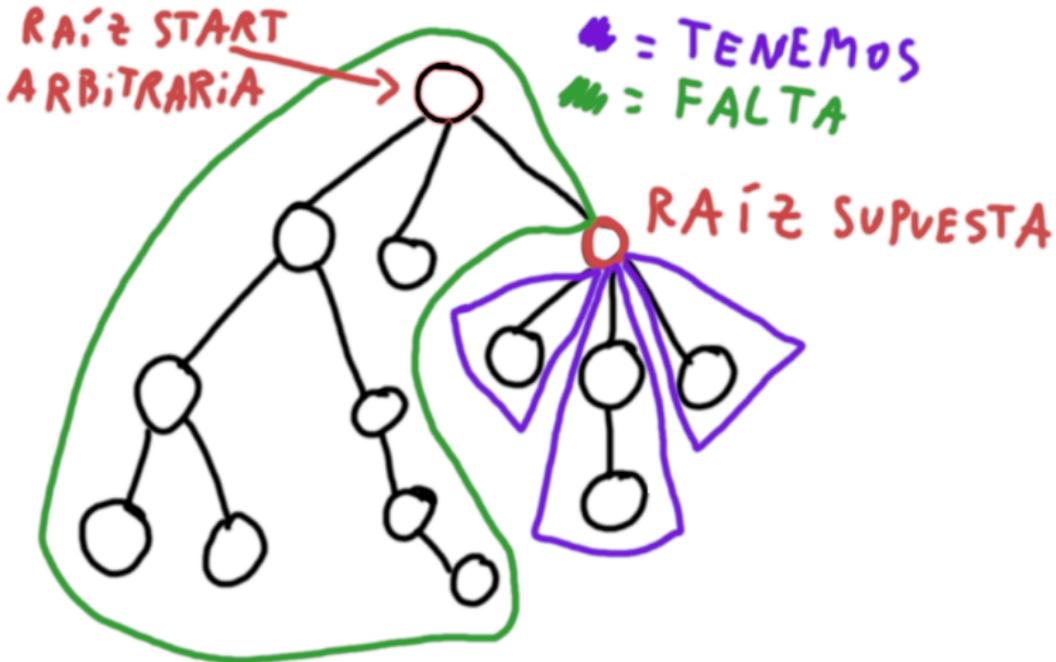
- Euler Tour

4 Descomposición Heavy-Light

- Definición
- Uso para subir k veces
- Uso para LCA
- Segment Tree en caminos

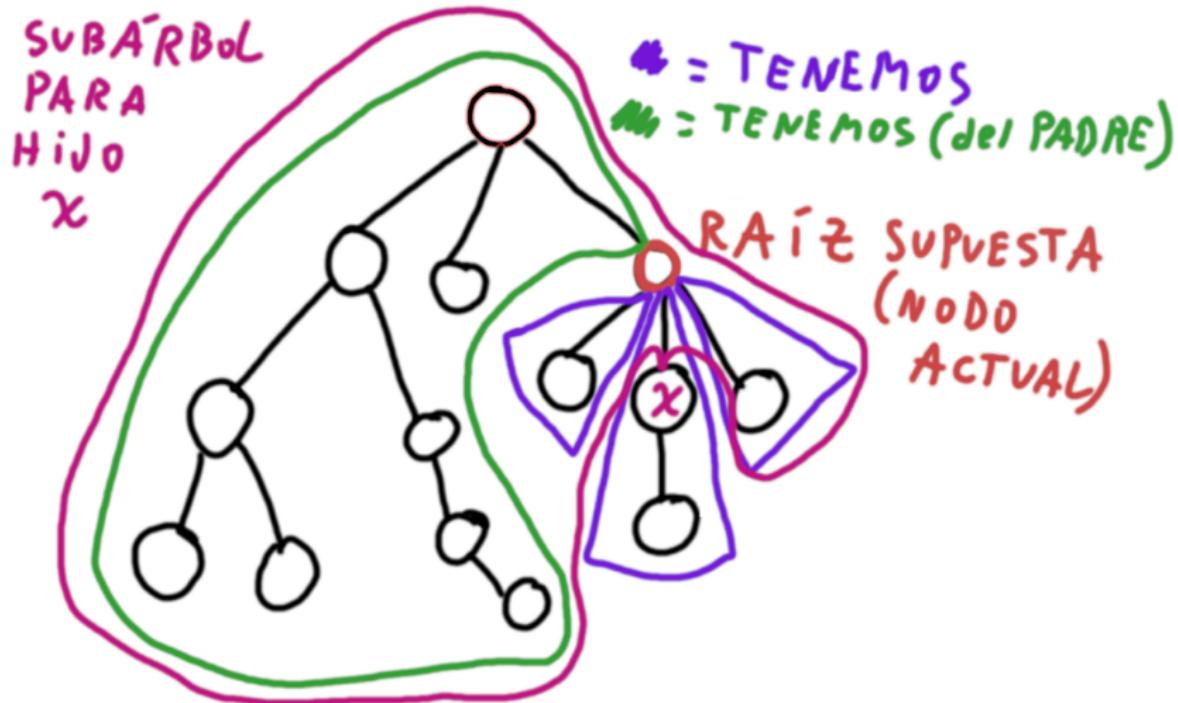
Primera pasada

- Primera pasada de recursión normal con subárboles “hacia abajo”
- Para poner un nodo de raíz nos faltaría el subárbol “hacia arriba”
- Conseguiremos estos valores en una segunda pasada



Segunda pasada

- Cada nodo recibe del padre el subárbol “hacia arriba” ya listo
- Cada nodo calcula los subárboles “hacia arriba” para sus hijos



Segunda pasada (cont.)

- El valor en este nodo puesto como raíz, se calcula con todos los subárboles hijos, y con el subárbol del padre
- Cada hijo necesita el valor usando un conjunto co-unitario
- Si podemos calcular el valor para los co-unitarios en $O(N)$, todo el algoritmo corre en $O(N)$
- Hemos calculado todos los subárboles posibles (en primera pasada si es “hacia abajo”, o en segunda pasada si es “hacia arriba”)

Valores en las aristas

- A veces se usan valores de los nodos y de las aristas
- Al calcular el valor se tiene un multiconjunto de pares (v, e):
 - v es el valor en el hijo (recursivamente)
 - e es el valor en la arista que conduce al hijo
- Si podemos calcular el valor en los co-unitarios funciona igual

Contenidos

1 Truquitos con cuentitas

- Ejemplos
- Multiconjuntos co-unitarios
- Caso combinable

2 Cambio de raíz

- Planteo del problema
- Algoritmo para cambio de raíz
- **Caso típico combinable**
- Ejemplos no combinables

3 "Euler Tour" (o simplemente DFS en árbol)

- Euler Tour

4 Descomposición Heavy-Light

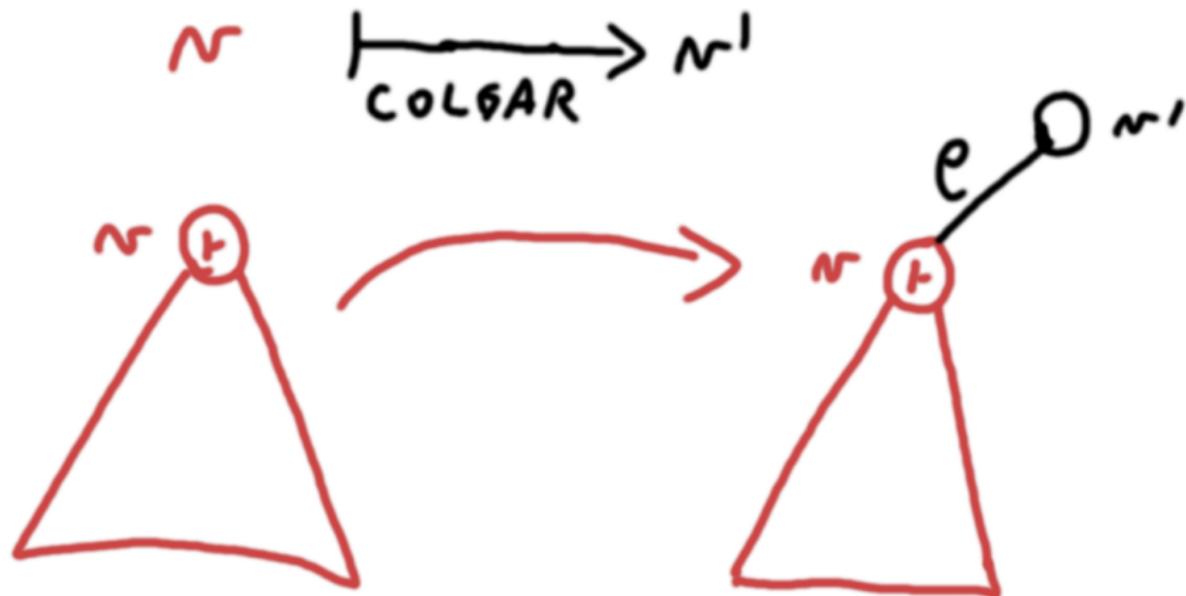
- Definición
- Uso para subir k veces
- Uso para LCA
- Segment Tree en caminos

Calcular cierta información sobre un subárbol

- Si queremos calcular algo sobre subárboles tal que:
 - Lo tenemos para cada hoja
 - Se pueden combinar dos cálculos al “unir” dos subárboles
 - Se puede recalculiar al “colgar” de una arista
- Entonces podemos hacer cambio de raíz
- El valor buscado será combinable (vale usar prefijo y sufijo)

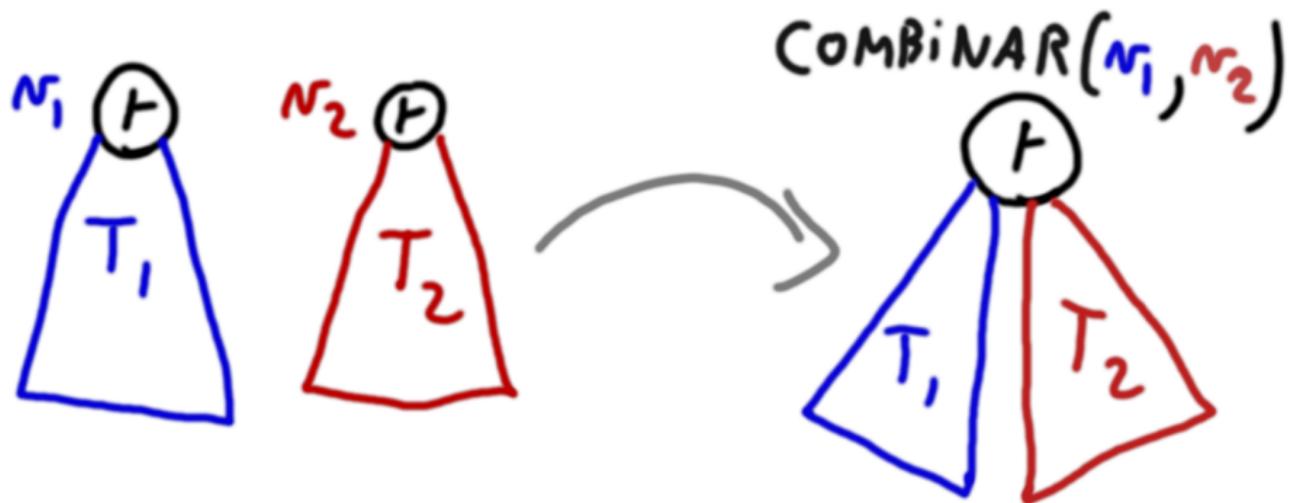
Operación de colgar

Esto es colgar de una arista e (y se recalcula el valor v al nuevo v'):



Operación de unir

Esto es unir dos subárboles (se combinan sus valores):



Ejemplo: tamaño de subárbol

- En las hojas el tamaño es 1
- Colgar es sumar 1 porque se agrega un nodo
- Combinar es $\text{combinar}(a, b) = a + b - 1$
- La operación asocia, pero no hace falta pensarla ni demostrarla especialmente: **si está calculando bien la información sobre la unión de subárboles, será asociativa automáticamente**

Ejemplo: altura de subárbol

- En las hojas la altura es 0
- Colgar es sumar 1 porque crece la altura
- Combinar es $\text{combinar}(a, b) = \max(a, b)$

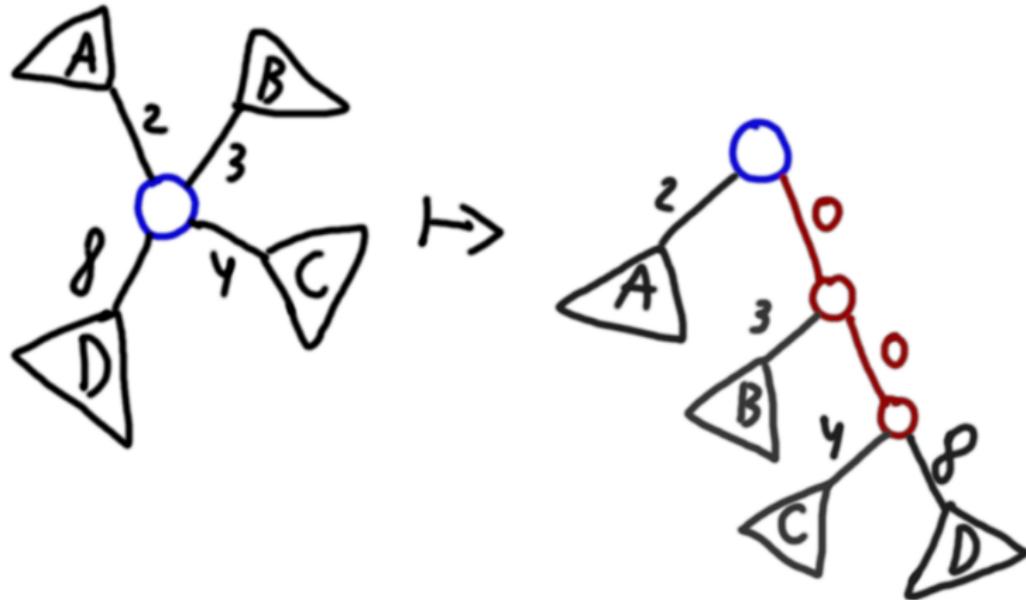
Ejemplo: máximo camino simple

Asumimos aristas con pesos (pueden ser incluso negativas)

- Calculamos tres valores:
 - El camino máximo simple del subárbol
 - Los dos caminos máximos desde raíz (usando hijos distintos)
- En las hojas es $(0, 0, 0)$
- Colgar con una arista de peso p es $(d, h_1, h_2) \rightarrow (\max(d, x), x, 0)$ donde $x = \max(0, h_1 + p)$
- Combinar es
$$\text{combinar}((d, h_1, h_2), (d', h'_1, h'_2)) = (\max(d, d', x_1 + x_2), x_1, x_2)$$
 donde $x_1 = \max(h_1, h'_1)$ y $x_2 = \max(\min(h_1, h'_1), h_2, h'_2)$

Truquito: Binarizar el árbol

- Lo anterior funciona bien con el truco de prefijo y sufijo
- Si el árbol fuera binario tendría grado máximo 3
- Se podrían calcular los co-unitarios por fuerza bruta bc, ac, ab
- Truco opcional: binarizar poniendo aristas “neutra”



Contenidos

1 Truquitos con cuentitas

- Ejemplos
- Multiconjuntos co-unitarios
- Caso combinable

2 Cambio de raíz

- Planteo del problema
- Algoritmo para cambio de raíz
- Caso típico combinable
- **Ejemplos no combinables**

3 "Euler Tour" (o simplemente DFS en árbol)

- Euler Tour

4 Descomposición Heavy-Light

- Definición
- Uso para subir k veces
- Uso para LCA
- Segment Tree en caminos

No son habituales en problemas

- Lo más común en problemas es tener el caso típico combinable
- El algoritmo de cambio de raíz igualmente no lo necesita
- Alcanza con calcular el valor de la recursión, sobre todos los subconjuntos co-unitarios

Ejemplo no combinable 1: Hash de árbol

- Para resolver isomorfismo de árbol con raíz, se computa un hash con los hashes de los hijos **ordenados**
- Tener que ordenarlos destruye toda esperanza de combinar
- Tampoco va a funcionar binarizar el árbol
- Se puede igualmente computar el hash de todos los co-unitarios
- Conseguimos así los N hashes de un árbol con raíz en cada nodo

Ejemplo no combinable 2: Recursión con la mediana

- La recursión toma la mediana de los valores de los hijos
- No hay forma de combinar sin guardar **todos** los valores
- Pese a esto, es fácil calcular las medianas de los co-unitarios
- Podemos usar Tree Rerooting sobre esta... cosa

Contenidos

1

Truquitos con cuentitas

- Ejemplos
- Multiconjuntos co-unitarios
- Caso combinable

2

Cambio de raíz

- Planteo del problema
- Algoritmo para cambio de raíz
- Caso típico combinable
- Ejemplos no combinables

3

"Euler Tour" (o simplemente DFS en árbol)

- Euler Tour

4

Descomposición Heavy-Light

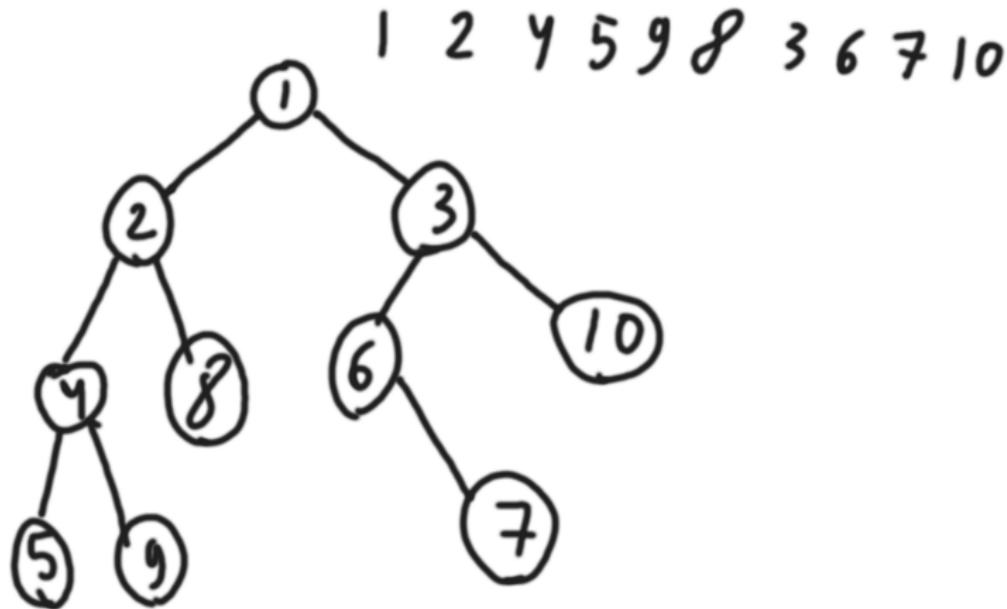
- Definición
- Uso para subir k veces
- Uso para LCA
- Segment Tree en caminos

Idea

- Recorrer un árbol con DFS
- Ir anotando los nodos o aristas por los que se va pasando
- “Se vuelca” así el árbol sobre un arreglo
- El orden de las cosas en el arreglo es interesante

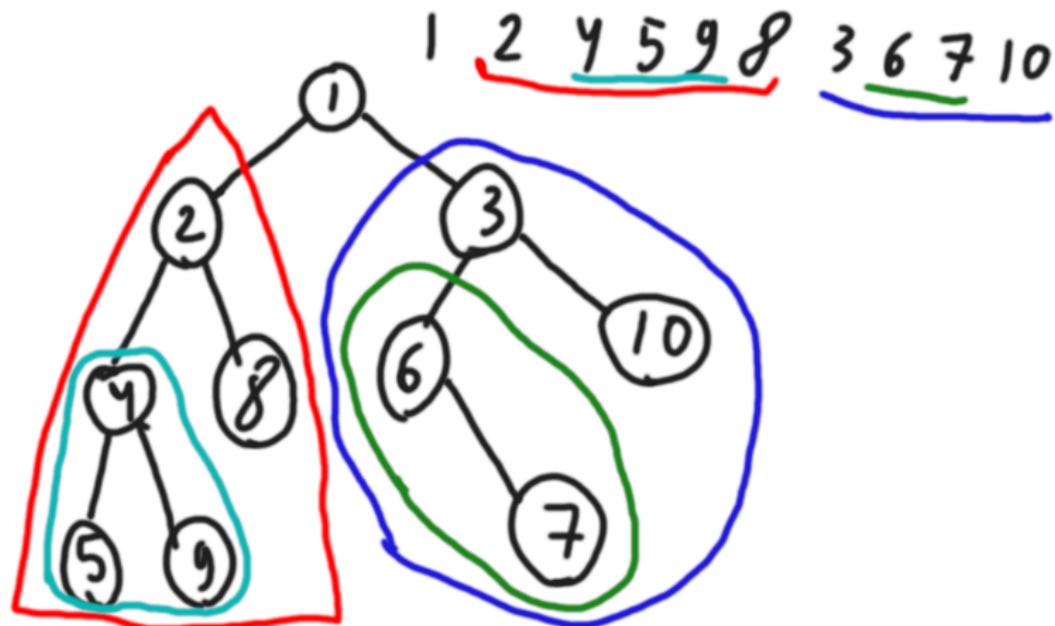
Versión 1: Anotar nodos al descubrir

- La más habitual
- Clave: cada subárbol se va a un intervalo
- Tiempos de descubrimiento y fin en el DFS marcan el intervalo



Versión 1: Anotar nodos al descubrir (cont.)

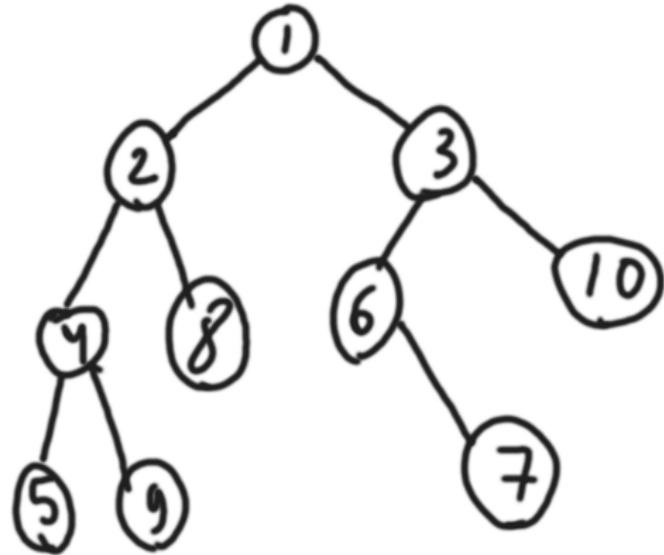
- La más habitual
- Clave: cada subárbol se va a un intervalo
- Tiempos de descubrimiento y fin en el DFS marcan el intervalo



Versión 2: Anotar nodos en cada visita

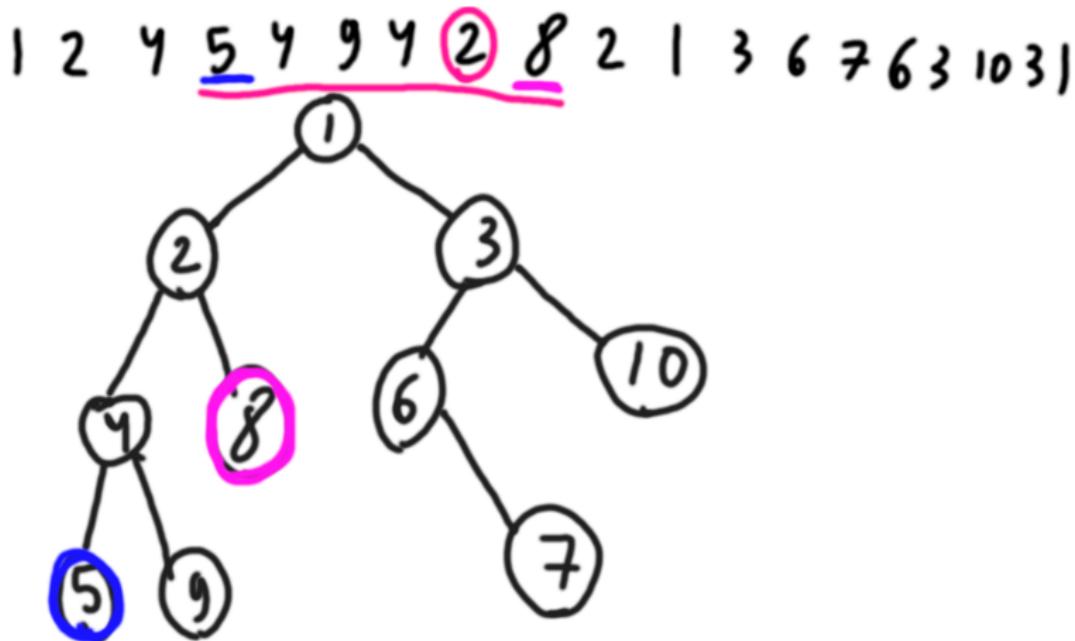
- La clásica para LCA
- El LCA es el de **mínima profundidad** en el rango entre a y b
- Lo anterior y sparse table permiten LCA en $O(1)$

1 2 4 5 4 9 4 2 8 2 1 3 6 7 6 3 10 3 1



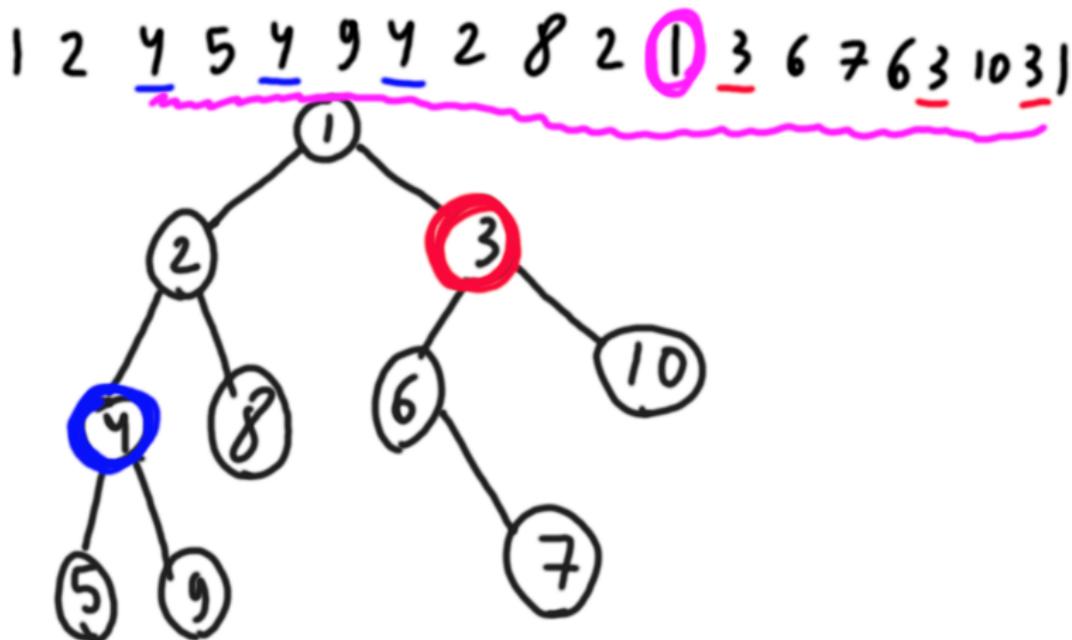
Versión 2: Anotar nodos en cada visita (cont.)

- La clásica para LCA
- El LCA es el de **mínima profundidad** en el rango entre a y b
- Lo anterior y sparse table permiten LCA en $O(1)$



Versión 2: Anotar nodos en cada visita (cont.)

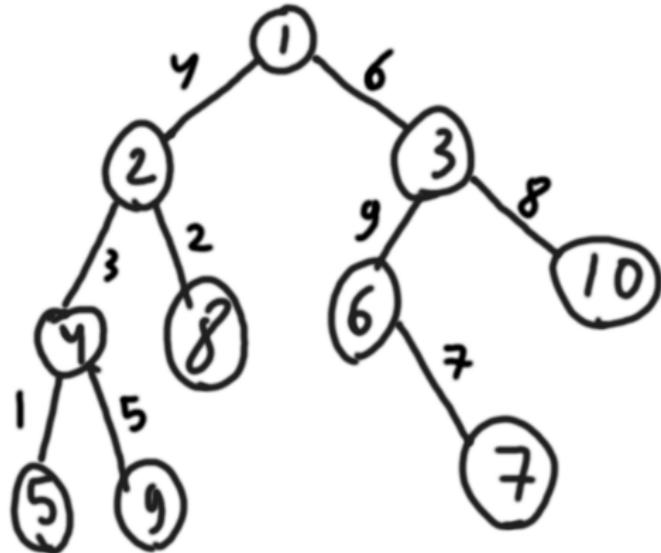
- La clásica para LCA
- El LCA es el de **mínima profundidad** en el rango entre a y b
- Lo anterior y sparse table permiten LCA en $O(1)$



Versión 3: Anotar aristas en cada visita

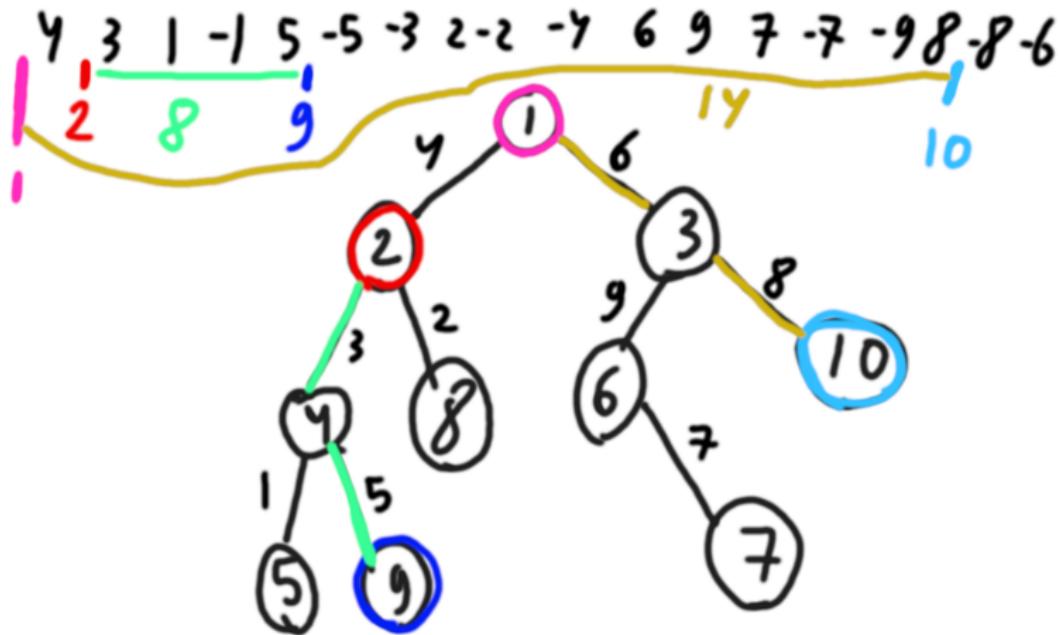
- Versión light de la descomposición heavy-light
- Se anota el valor sumando al bajar y restando al subir
- Sumar en el arreglo de x a un descendiente suma en el árbol

4 3 1 -1 5 -5 -3 2 -2 -7 6 9 7 -7 -9 8 -8 -6



Versión 3: Anotar aristas en cada visita (cont.)

- Con dos ramas desde el $LCA(a, b)$ se forma el camino (a, b)
- Con segment/fenwick tree se pueden actualizar aristas del árbol
- Caso especial: con el xor no hace falta LCA



Contenidos

1

Truquitos con cuentitas

- Ejemplos
- Multiconjuntos co-unitarios
- Caso combinable

2

Cambio de raíz

- Planteo del problema
- Algoritmo para cambio de raíz
- Caso típico combinable
- Ejemplos no combinables

3

“Euler Tour” (o simplemente DFS en árbol)

- Euler Tour

4

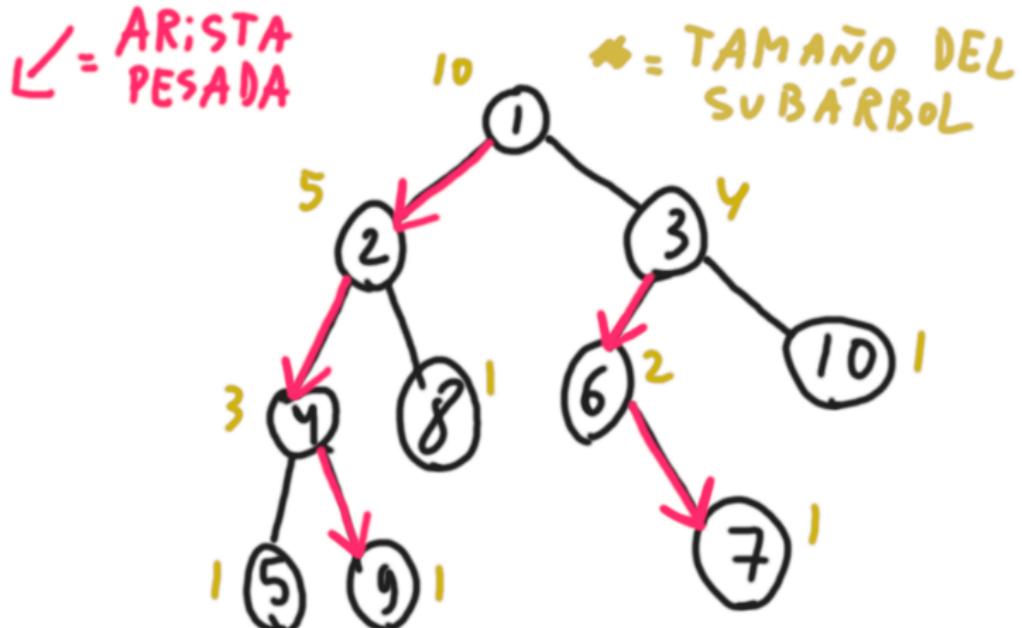
Descomposición Heavy-Light

● Definición

- Uso para subir k veces
- Uso para LCA
- Segment Tree en caminos

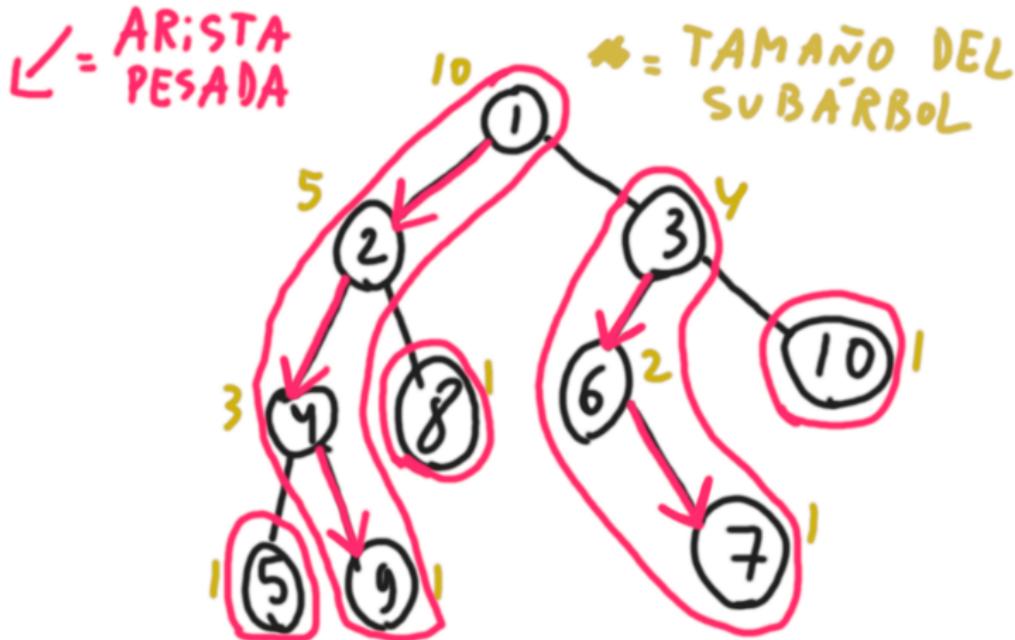
Definición

- A cada nodo interno se le asigna una **arista pesada** a un hijo
- La arista pesada va al **subárbol más grande**
- En caso de empate, se toma cualquiera
- Las aristas que no son pesadas las llamamos livianas



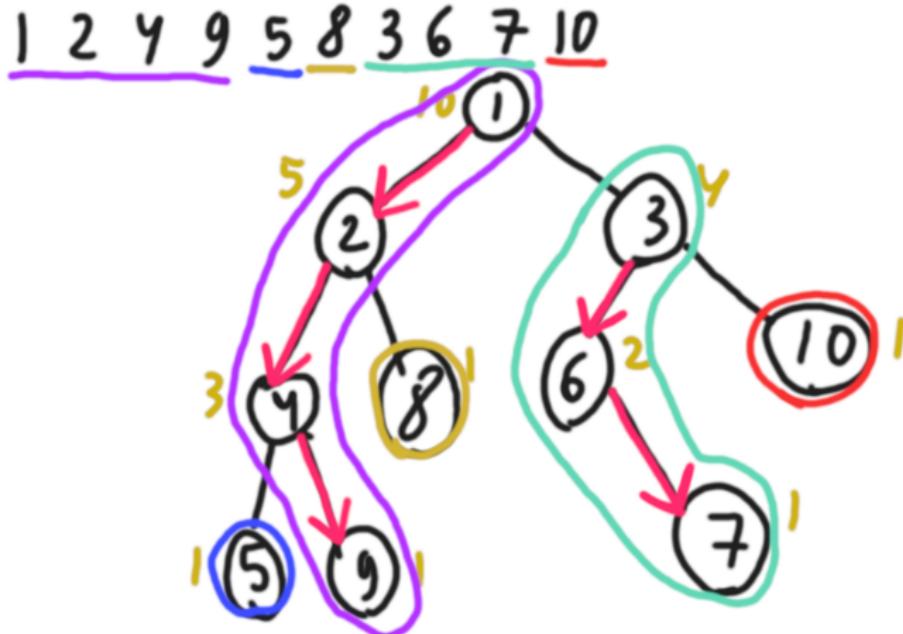
Definición (cont.)

- Las aristas pesadas partitionan los nodos en caminos pesados
- Al bajar una arista liviana, **se reduce el tamaño a la mitad o menos**



Cómputo

- Podemos usar Euler Tour en su versión 1 (nodos al descubrir)
- Un detalle: bajamos **primero** por la arista pesada
- Al hacerlo los caminos pesados quedan **consecutivos**



Contenidos

1 Truquitos con cuentitas

- Ejemplos
- Multiconjuntos co-unitarios
- Caso combinable

2 Cambio de raíz

- Planteo del problema
- Algoritmo para cambio de raíz
- Caso típico combinable
- Ejemplos no combinables

3 “Euler Tour” (o simplemente DFS en árbol)

- Euler Tour

4 Descomposición Heavy-Light

- Definición
- **Uso para subir k veces**
- Uso para LCA
- Segment Tree en caminos

Subir k veces en $O(\lg N)$

- Por cada nodo, sabemos comienzo y fin de su camino pesado
- La clave es que subiendo a la raíz hay $O(\lg N)$ aristas livianas
- Dentro de un camino pesado, subir k veces es restar k al índice
- Si hay lugar en el camino pesado para subir k , dar la respuesta
- Si no, saltar al comienzo del camino y subir por la arista liviana
- Repetir hasta terminar
- Notar que mientras subimos recorremos:
 - Intervalos del arreglo (parte de un camino pesado)
 - Al cambiar de camino, se recorre una arista liviana
 - En total son $O(\lg N)$ de ambas cosas

Contenidos

1 Truquitos con cuentitas

- Ejemplos
- Multiconjuntos co-unitarios
- Caso combinable

2 Cambio de raíz

- Planteo del problema
- Algoritmo para cambio de raíz
- Caso típico combinable
- Ejemplos no combinables

3 “Euler Tour” (o simplemente DFS en árbol)

- Euler Tour

4 Descomposición Heavy-Light

- Definición
- Uso para subir k veces
- **Uso para LCA**
- Segment Tree en caminos

Cómputo de LCA en $O(\lg N)$

- Conviene tener anotada la profundidad p también
- Si están en distinto camino pesado, subir aquel cuyo camino pesado comience más abajo
- Si están en el mismo camino pesado, el LCA es el de arriba
- Como antes, recorremos $O(\lg N)$ partes de camino pesado y aristas livianas

Contenidos

1 Truquitos con cuentitas

- Ejemplos
- Multiconjuntos co-unitarios
- Caso combinable

2 Cambio de raíz

- Planteo del problema
- Algoritmo para cambio de raíz
- Caso típico combinable
- Ejemplos no combinables

3 “Euler Tour” (o simplemente DFS en árbol)

- Euler Tour

4 Descomposición Heavy-Light

- Definición
- Uso para subir k veces
- Uso para LCA
- Segment Tree en caminos

Segment Tree en caminos y subárboles

- Ahora además ponemos un Segment Tree sobre el arreglo
- Cualquier camino en el árbol se descompone pasando por el LCA
- Se descompone en $O(\lg N)$ aristas livianas e intervalos del arreglo
- Podemos entonces soportar modificaciones y consultas de puntos y rangos (eventualmente con segment tree con Lazy Propagation), sobre cualquier camino del árbol, en tiempo $O(\lg^2 N)$
- Además utilizando el mismo segment tree, como antes, podemos operar con subárboles en $O(\lg N)$

Post con la implementación

- <https://codeforces.com/blog/entry/53170>