Estructuras - Parte 2

Problemas, Algoritmos y Programación

Octubre de 2016

Contenidos

Repaso de la clase pasada

La clase pasada vieron algunas estructuras para resolver consultas en rangos aplicando operadores asociativos:

- Prefijos:
  - Consultas estáticas: Tablas aditivas.
  - Consultas dinámicas: Binary Index Trees (Fenwick).
- Intervalos generales:
  - Consultas estáticas: Sparse Tables.
  - Consultas dinámicas: Segment Trees.

Hoy nos enfocaremos sólo en los Segment Trees, y los dotaremos de más poder del que tienen en sus versiones como las vimos la clase pasada.

Repaso de Segment Trees

Los Segment Trees se basan en guardar la respuesta para el intervalo entero ("rellenando" con neutros hasta llegar a tamaño potencia de dos, a lo sumo duplicando su tamaño, para hacer más sencilla su implementación en caso de ser necesario o conveniente), y luego, para cada intervalo para el cuál tenemos la respuesta, guardar la respuesta para la primer mitad, y para la segunda mitad. (Dibujo en el pizarrón, para más referencias consultar las diapositivas de la clase pasada).

Repaso de Segment Trees

Poder hacer consultas dinámicas significa que podemos modificar la estructura, y seguir haciendo consultas rápidamente, sin tener que procesar toda la estructura nuevamente.

Los costos que tiene el Segment Tree son los siguientes:

- Inicializar: $\mathcal{O}(N)$
- Consultar un rango cualquiera: $\mathcal{O}(\log N)$
- Actualizar un valor: $\mathcal{O}(\log N)$

El problema es que sólo podemos actualizar UN valor con este costo. Sería deseable, ya que manejamos intervalos, poder actualizar todo un intervalo.

Actualizando intervalos

Si quisieramos actualizar un intervalo, de la manera en la que sabemos hacerlo, deberíamos actualizar cada elemento del intervalo, llegando a l hojas, siendo l la cantidad de elementos del intervalo.

Actualizando intervalos

Si quisieramos actualizar un intervalo, de la manera en la que sabemos hacerlo, deberíamos actualizar cada elemento del intervalo, llegando a l hojas, siendo l la cantidad de elementos del intervalo.
Pero cuando consultamos por un intervalo no miramos todas las hojas, ¿porqué hacerlo cuando actualizamos?. Si sabemos que un nodo del árbol va a propagar a sus hijos la actualización, para que todas sus hojas descendientes hagan esa actualización, podemos guardar esa información y actualizar más adelante.

Lazy Propagation

Esta idea se conoce como Lazy Propagation (propagación perezosa), y consiste en que cada vértice del árbol se guarde toda la información necesaria para propagarla a sus hijos en el momento en el que sea necesario y ya no pueda seguir postergandose.

La idea consiste en los pasos que se explican en la próxima diapositiva.

Lazy Propagation

- La actualización comienza en la raíz, indicando el cambio que se le hace a todos los elementos del intervalo (a todos el mismo), la cual se encarga de propagarle el cambio a sus hijos.
- Cada nodo que recibe una actualización toma una decisión basada en el caso en el que corresponda:
  - El intervalo no debe ser actualizado, en ninguno de sus elementos: El cambio se ignora.
  - Algunos elementos del intervalo deben ser actualizados: Decisión difícil, la pateo a los hijos.
  - Todo el intervalo debe ser actualizado: Me guardo el cambio para propagarlo a mis hijos cuando sea necesario.
- Cuando es necesario, se propaga el cambio a los hijos (de la misma) manera que en el segundo caso del paso anterior.

Lazy Propagation

- La actualización comienza en la raíz, indicando el cambio que se le hace a todos los elementos del intervalo (a todos el mismo), la cual se encarga de propagarle el cambio a sus hijos.
- Cada nodo que recibe una actualización toma una decisión basada en el caso en el que corresponda:
  - El intervalo no debe ser actualizado, en ninguno de sus elementos: El cambio se ignora.
  - Algunos elementos del intervalo deben ser actualizados: Decisión difícil, la pateo a los hijos.
  - Todo el intervalo debe ser actualizado: Me guardo el cambio para propagarlo a mis hijos cuando sea necesario.
- Cuando es necesario, se propaga el cambio a los hijos (de la misma) manera que en el segundo caso del paso anterior.

Veamos ahora cuándo es necesario propagar el cambio a los hijos.

Lazy Propagation

Cuando un intervalo debe ser actualizado por completo, podemos guardar esta información en el vértice que contiene a ese intervalo, y no transmitirsela a los hijos. Podría llegar a pasar, que nunca sea necesario consultar los hijos, y por lo tanto, no sea necesario propagar esta información.

Veamos como funciona esto con un ejemplo en el pizarrón.

Lazy Propagation

Si una consulta llega a un vértice que no propagó sus actualizaciones a sus hijos, y la consulta debe ser transmitida a sus hijos, el nodo que guarda la información que le va a mandar a sus hijos, antes de derivar la consulta, se da cuenta que sus hijos tienen que tener la información actualizada, y propaga recién en ese momento.

Lazy Propagation

Si una consulta llega a un vértice que no propagó sus actualizaciones a sus hijos, y la consulta debe ser transmitida a sus hijos, el nodo que guarda la información que le va a mandar a sus hijos, antes de derivar la consulta, se da cuenta que sus hijos tienen que tener la información actualizada, y propaga recién en ese momento.
Otra forma de ver esto es que, si a un vértice le llega una consulta que no sabe responder, le dice a quien trae la consulta: "¿Vas a ver a mis hijos? Tomá, llevales esta actualización y deciles que la apliquen antes de responderte".

Complejidad del update por rangos

Ya sabemos que la consulta se puede responder en orden logarítmico. Pero usando esta técnica, las actualizaciones por rango, también son logarítmicas, ya que toman las mismas decisiones (deciden cuándo patearle el update a los hijos y cuándo no con el mismo criterio que usan las consultas, basándose únicamente en los extremos del intervalo a consultar o actualizar).

Además, la propagación que se hace sólo cuando es necesaria, es necesaria, sólo es necesaria cuando la consulta visita los nodos del árbol, por lo que para cada operación de avisarle a los hijos que deben ser actualizados, existe una operación asociada de consultar a los hijos, luego esta actualización se puede ver como parte de la consulta, a la cual no le cambia su complejidad.

De esta manera podemos ver que el update por rangos es también logarítmico.