Ideas con Raíz Cuadrada

Lazy update, SQRT Decomposition, Mo, etc.

Lautaro Lasorsa

Universidad de Buenos Aires - FCEN, Universidad Nacional de la Matanza - DIIT

Training Camp 2024

Gracias Sponsors!

Organizador

Diamond

Gold

Outline

Outline

Lazy Update

Definition (Lazy Update)

Es una técnica que aplicaremos cuando tenemos una estructura la cuál no soporta updates, pero si podemos aplicar un update a la respuesta de consultar a la estructura. Entonces, si tenemos u updates y q consultas:

- Cada $\sqrt{u}$ updates reconstruiremos la estructura en $O(T(n))$ con todos los updates hasta ahora.
- Al responder una query, respondemos con la estructura en su estado actual y luego aplicamos los $O(\sqrt{u})$ updates que faltan.
- La complejidad total es $O(\sqrt{u} * (T(n) + q))$.

Ejemplo de uso

Example

Suma en rango

- Tabla aditiva: Consultas $O(1)$ y construcción $O(n)$
- Actualizaciones: Sumar un valor x a todos los elementos de un rango.
- Consulta: Suma de un rango.
- Complejidad: $O(\sqrt{u} * (q + n))$

Outline

¿Dónde aplica?

Esta ídea tiene sentido sobre todo en gráfos porque vale lo siguiente

Theorem

Cota raíz cuadrada Si tenemos un grafo con m aristas, entonces hay a lo mucho $2 * \sqrt{m}$ nodos de grado $\sqrt{m}$ o más.

Utilizando esta idea, podemos resolver problemas con consultas/updates de la pinta:

- Update: Modificar el valor asociado a un nodo.
- Query: Consultar la agregación de los vecinos de un nodo.

Ejemplo de uso

Example

Dado un grafo con n nodos y m aristas, donde cada nodo inicialmente tiene un valor igual a 0. Resolver las siguientes consultas:

- 1. Dado un nodo u y un entero x, aumentar en x el valor de todos los vecinos de u
- 2. Dado un nodo u, devolver el valor de ese nodo.

Solución:

- Cuando aplico un update a un nodo, si este tiene grado $\leq \sqrt{m}$, actualizo efectivamente el valor de sus vecinos.
- Si tiene grado $> \sqrt{m}$, acumulo el update a un total asociado al nodo.
- Para responder una consulta, el valor del nodo será la suma del valor que guardo en el nodo y los updates que sus vecinos no propagaron. Cómo a lo mucho hay $2 * \sqrt{m}$ nodos de grado $\sqrt{m}$ o más (que no propagan sus updates), la complejidad es $O(\sqrt{m})$.

Otro ejemplo

En la Brazilian ICPC Summer School 2023 se tomó un problema similar al presentado anteriormente.

Power Link

Notar que este problema también puede ser resuelto con el enfoque de actualización lazy.

Subset Sum de elementos distintos

Example

Si tengo n elementos $x_i \geq 0$, y se que la suma $\sum_{i=0}^{n-1} x_i = S$, entonces hay hasta $O(\sqrt{S})$ elementos x_i distintos. Esto permite resolver el problema de subset sum de estos elementos realizando unicámente $O(\sqrt{S})$ pasos de actualizar que pueden ser $O(S)$.

Hecho así, la complejidad total es $O(S^{\frac{3}{2}})$.

Outline

Descomposición en Raíz Cuadrada

- Se divide al dominio en $O(\sqrt{n})$ trozos de tamaño $O(\sqrt{n})$.
- A cada trozo se le aplica un computo costoso en el tamaño del mismo.
- Se utilizán las respuestas a los mismos para generar una respuesta global o cómo precomputo para responder consultas.
- Por ejemplo, si el computo es $O(n^2)$, al aplicarlo a un bloque de tamaño $O(\sqrt{n})$, la complejidad será $O(n)$ y la complejidad total $O(n^{\frac{3}{2}})$.

Ejemplo de uso

Example (CSES: Increasing Array Queries)

- Tomo $n/base + 1$ puntos $k_0 = 0, k_1 = base, \ldots, k_i = i * base$. ($base \approx \sqrt{n}$)
- Para cada punto i calculo para el sufijo que inicia en ese punto:
  - El costo de hacer creciente cada prefijo del sufijo. $costo[i][j]$
  - El valor que tomaría cada elemento del sufijo si se hace creciente el prefijo que termina ahí (lo mismo que tomar el máximo de ese prefijo del sufijo). $max[i][j]$
  - La suma de los valores máximos de los elementos en ese prefijo. $sum\_max[i][j]$
- Para responder una query $[l, r]$:
  - Simulo el proceso desde l hasta el siguiente k_i (exclusive).
  - Agrego el costo de hacer creciente el rango $k_i \ldots r$.
  - Busco el último elemento j de ese rango que sería menor o igual al máximo c en $[l, k_i)$.
  - Agrego $c * (j - k_i) - sum\_max[i][j]$ al costo.

Outline

Árbol de un solo nivel

Aplicado al problema de consultas en rango y updates en punto 1D.

- Al igual que antes, dividimos el vector en $O(\sqrt{n})$ bloques de tamaño $O(\sqrt{n})$.
- Mantengo el valor para cada punto y el valor agregado de cada bloque.
- Para responder una query $[l, r]$:
  - Si l y r pertenecen al mismo bloque, calculo la respuesta en $O(\sqrt{n})$. Sino:
  - Calculo el valor del sufijo de bloque que empieza en l ($[l, k_i)$).
  - Agrego los bloques intermedios cubiertos completamente por la consulta.
  - Agrego los elementos del prefijo ($[k_j, r]$).
- Permite (si la operación es conmutativa e inversible) hacer updates en $O(1)$ y consultas en $O(\sqrt{n})$.

Ejemplo de uso

Example (MEX de un multiconjunto con updates)

Enunciado:

- Dado un multiconjunto de n enteros x_i $(1 \leq x_i, n \leq 2 * 10^5)$.
- Se realizan q consultas y u actualizaciones intercalados de los tipos:
- Consulta: Calcular el MEX (mínimo entero no presente) del multiconjunto.
- Update:
  - 1. Agregar un elemento x al multiconjunto.
  - 2. Elimina una aparición de un elemento x del multiconjunto.
- $1 \leq q \leq 2 * 10^5$, $1 \leq u \leq 10^7$.

Solución a MEX con updates

- Mantengo un histograma de los elementos.
- Para cada bloque del histograma calculo la cantidad de elementos distintos.
- Estas dos cosas se actualizán en $O(1)$.
- Al hacer una consulta, busco el primer bloque que no este completo usando la cantidad de distintos en cada bloque.
- Busco el primer elemento del bloque con 0 apariciones linealmente.
- Esto permite consultas en $O(\sqrt{n})$

Outline

2 punteros

- Tengo un problema de realizar consultas en un vector de n elementos.
- Tengo una estructura de datos que puede representar un rango $[l, r)$ y permite responder la consulta sobre ese rango en $O(Q)$.
- Puedo modificar la estructura para pasar a representar $[l + 1, r)$, $[l - 1, r)$,$[l, r + 1)$ y $[l, r - 1)$ en $O(U)$.
- Permite responder todas las consultas en $O((q + n) * \sqrt{n} * U + q * Q)$ (q consultas).

Ordenar las consultas

- Divido las consultas en $O(\sqrt{n})$ bloques de tamaño $O(\sqrt{n})$ en función de su posición de inicio.
- Dentro de cada bloque, ordeno las consultas por su posición de fin.
- Es decir, ordeno las consultas (l, r) por $(l//\sqrt{n}, r)$.
- Voy a responder las consultas en este orden actualizando la estructura al pasar de una consulta a otra (modificando los extremos como sea necesario).
- Al pasar de una consulta a otra en el mismo bloque:
  - El extremo l puede moverse a lo mucho $\sqrt{n}$ veces.
  - El extremo r solo puede avanzar. En cada bloque puede avanzar en total hasta $O(n)$ veces.
- Cada bloque realiza hasta n updates y cada consulta genera hasta $O(\sqrt{n})$ updates del extremo izquierdo.
- Al cambiar de un bloque a otro:
  - El extremo l avanza hasta n veces.
  - El extremo r retrocede hasta $n - l$ veces.
- Entonces, al cambiar de bloque ($O(\sqrt{n})$ veces), se realizan $O(n)$ updates.

Ejemplo

Notar que con la estructura anterior permite resolver consultas de MEX en rango en $O(n * \sqrt{n})$.

Caso con updates: Las consultas tienen 3 índices

Ahora una consulta está definida por 3 índices (l, r, t):

- l y r son los extremos de la consulta.
- t es la cantidad de actualizaciones antes de esta consulta.

Sea $base \approx n^{\frac{2}{3}}$, ordeno las consultas según $(l//base, r//base, t)$. Notar que esto parte a los ejes l y r en $O(n^{\frac{1}{3}})$ partes. La grilla queda dividida en $O(n^{\frac{2}{3}})$ bloques.

La complejidad

Son n elementos, q consultas y u updates.

- Con un razonamiento similar al anterior, podemos notar que cada consulta que no cambie los extremos de bloque realiza $O(n^{\frac{2}{3}})$ updates al mover los extremos l y r.
- Dentro de cada bloque hay hasta $O(u)$ updates al avanzar el índice t.
- Al cambia de bloque, se realizán $O(n)$ updates al cambiar los extremos de la consulta y $O(u)$ updates de retroceder el t. Esto se realiza $O(n^{\frac{2}{3}})$ veces.
- La complejidad total es $O((q+n+u) * n^{\frac{2}{3}} * U + q * Q)$.

Ejemplo de uso

Cómo en el caso sin updates, con la estructura de SQRT Decomposition permite resolver MEX en rangos con updates de forma eficiente.

Ejemplos generales

- Suma en rango: Mo y SQRT Decomposition.
- Máximo en rango: SQRT Decomposition.
- DQUERY (elementos distintos): Mo.

Gracias por ver

Medios de contacto:

- laulasorsa@unlam.edu.ar
- lautarolasorsa@gmail.com
- @lautaro-lasorsa en LinkedIn