Universidad de Talca
Facultad de Ingeniería
Departamento de Ciencias de la Computación

Diseño y análisis de algoritmos

Cálculo de mediana en tiempo lineal como mejor caso y caso promedio y estabilización del algoritmo de selección por pivotes IQS

Erik Regla
eregla09@alumnos.utalca.cl

20 de Junio del 2015

1. Algoritmos de selección basados en pivotes

Los algoritmos de ordenamiento y selección basados en la estrategia "dividir y conquistar"se fundamentan el el uso de pivotes para poder dividir el problema. Se ha demostrado la eficiencia de esta técnica en el caso promedio de un arreglo desordenado, pese a esto, cuando el conjunto de datos es la fuente del peor caso, el algoritmo de selección nada puede hacer. Este es el caso de algoritmos como *QuickSort* con pivote en la mediana del conjunto desordenado.

Se define entonces que un buen pivote representa a la mediana del conjunto ordenado, el cual permite dividir equitativamente el problema en dos para que así la subdivisión sea óptima y se cumpla el tiempo esperado.

2. Selección de mediana en tiempo lineal para el caso promedio

Existen métodos de encontrar la mediana geométrica de los datos en un tiempo lineal en promedio, utilizando el segmento de QuickSort que divide el arreglo en dos utilizando un pivote aleatorio, pero solo aplicando la recursión para el lado promisorio del arreglo (ver Algoritmo 2).

3. Selección de mediana en tiempo lineal para el peor caso

Está dicho que es imposible buscar el k-esimo elemento de un arreglo dado con un peor caso inferior a $nlog(n)$. Sin embargo, para efectos de balance, no nos interesa precisamente que el arbol esté perfectamente balanceado. Para nuestros efectos, sirve el que cumpla con un cierto umbral de confianza respecto a el resto de los valores.

Algoritmo 1 Selección de mediana utilizando QuickSelect

Precondición: A arreglo de datos desordenados
Postcondición: $A[p]$ es la mediana de A

```
1: i ← 0
2: j ← |A|
3: mientras p ≠ |A|/2 hacer
4:    p ← obtenerPivote(A, i, j)
5:    p ← particionar(A, i, j)
6:    si p > |A|/2 entonces
7:       i ← p
8:    si no
9:       j ← p
10:   fin si
11: fin mientras
12: devolver p
```

3.1. Hipótesis

Se puede obtener una mediana m perteneciente a $P_{30} \leq m \leq P_{70}$ en un arreglo A, si para $d = \frac{|A|}{k}$ elementos se le agrupan $\frac{k-1}{2}$ elementos menores y $\frac{k-1}{2}$ elementos mayores a cada uno de los d elementos dados.

3.2. Prueba

Sea n el tamaño de un arreglo K dado. Al ordenar K se tiene que $k_1 < k_2 < ... < k_n$. Entonces el elemento central $k_{\frac{n}{2}}$ posee $\frac{n-1}{2}$ elementos mayores ($k_1 < k_2 < ... < k_{\frac{n-1}{2}}$) y $\frac{n}{2}$ elementos mayores que ($k_{n+1} < k_{n+2} < ... < k_n$).

Si tomamos la mediana de K entonces sabemos que tiene exactamente $n-1$ elementos de los cuales sabemos si son mayores o menores que este. Ahora, tomemos un arreglo J de dimensión m. Sub-dividimos J en conjuntos K (que cumplen con la condición de K descrita anteriormente) de tamaño n. Si i es la mediana de K, ordenamos J en base al elemento i de cada K generado. Sea t la mediana del conjunto de J, sabemos que existen $\frac{m-1}{2n}$ grupos que poseen una mediana menor y $fracm - 12n$ conjuntos que poseen una mayor.

Entonces, de nuestros $\frac{n-1}{k}$ conjuntos, sabemos que cada uno de ellos tiene $\frac{n-1}{4}$ elementos que sabemos, son menores que t, y que otro grupo de $\frac{n-1}{k}$ conjuntos, sabemos que cada elemento también tiene $\frac{n-1}{4}$ elementos mayores que i (ver Figura 1). No es necesario recalcar que dado que los procesos de ordenamiento son ejecutados sobre K, el cual siempre es constante, este para tamaños relativamente pequeños de n puede ser considerado constante al ser aplicado sobre todo J.

Además, una variante de este algoritmo puede ser incluida para incrementar la precisión de la selección sin afectar significativamente el tiempo de ejecución al iterar el proceso sobre el conjunto de medianas, sin embargo, si lo que a uno le interesa es encontrar el pivote, conlleva a un gasto en

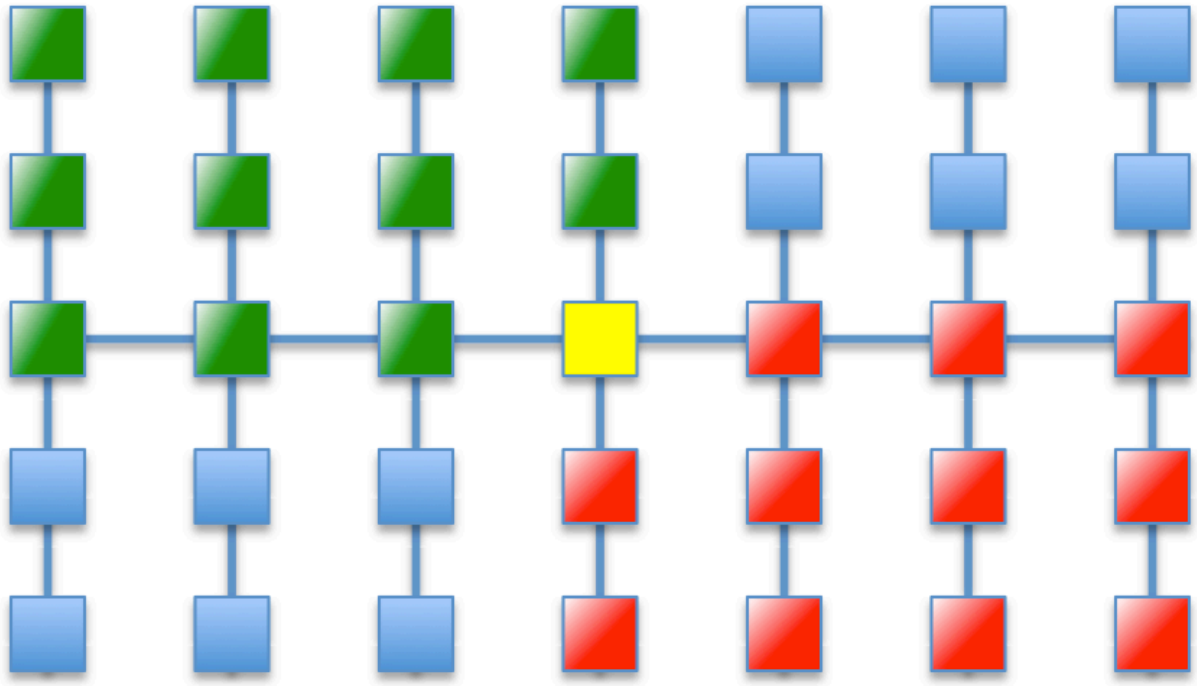

Figura 1: Ejemplo de ejecución del algoritmo de mediana de medianas para un conjunto de 35 elementos. En amarillo es la mediana de de lis i generados, en verde los elementos que son garantizados menores que i y en rojo los elementos garantizados de ser mayores que i y en celeste cada dato del cual no se tiene información relativa a su posición. Cada línea vertical de tamaño k (en este caso igual a 5) representa un conjunto K, generado en base a J. La línea horizontal representa el arreglo de medianas de cada grupo de K.

espacio extra al almacenar los índices o bien a un incremento en el tiempo de ejecución debido a que es necesario mover cada elemento dentro el mismo arreglo.

Actualmente no hay documentación respecto a la existencia de mecanismos que puedan aprovechar esta situación de ordenamiento parcial sobre cierto grupo de elementos en el arreglo. Como trabajo futuro, se estudiarán estos fenómenos.

4. Metodología

Los pasos a seguir se enumeran a continuación:

1. Investigar respecto a algoritmos para cálculo de medianas.
2. Demostrar matemáticamente que los algoritmos cumplen con los requerimientos.
3. Describir las estructuras de datos involucradas en cada uno de ellos.

Algoritmo 2 Selección de mediana aproximada utilizando mediana de medianas

Precondición: J arreglo de datos desordenados, k el largo de la lista temporal K
Postcondición: $A_{\frac{|A|}{2}}$ es la mediana aproximada de J

1: $A \leftarrow []$
2: $i \leftarrow 0$
3: **mientras** $i + k < |A|$ **hacer**
4: $\quad K \leftarrow A_{i...min(i+k,|A|)}$
5: $\quad K \leftarrow ordenar(K)$
6: $\quad insertar K_{\frac{|K|}{2}} en A$
7: $\quad i \leftarrow i + k$
8: **fin mientras**
9: $A \leftarrow ordenar(A)$
10: **devolver** $A_{\frac{|A|}{2}}$

4. Implementar los algoritmos correspondientes para realizar una comparativa entre cada uno de ellos.

5. Llevar a cabo una investigación experimental, analizar e interpretar los resultados.

Lo que está hecho:

- Investigar respecto a algoritmos para cálculo de medianas.
- Describir las estructuras y algoritmos involucrados.
- Implementar los algoritmos correspondientes para realizar una comparativa entre cada uno de ellos.

Lo que falta:

- Mejorar documentación del algoritmo de mediana promedio (aunque no creo que sea necesario, dado que no es más que usar QuickSelect con pivote aleatorio).

4.1. Resultados de la investigación hasta ahora

En la literatura se encuentran demostraciones de cuando es atingente implementar estos algoritmos descritos para acelerar o estabilizar otros. Un ejemplo de esto, es el algoritmo *QuickSelect*.

4.2. Implementación del caso de estudio

Para realizar las comparativas se tomó como víctima a *IQS*[1], el cual es un algoritmo de selección basado en pivotes diseñado para entregar los primeros k-ésimos elementos en un tiempo

[1]También conocido como *Incremental Quick Select*

$m + k\ log(k)$ de forma online en k. Actualmente como parte del mecanismo, la implementación original utiliza como pivote el elemento del medio entre el rango que actualmente está buscando. El objetivo de esta investigación es cambiar la implementación del método *getPivot()*, el cual elige como pivote un elemento al azar y estudiar como cambia el comportamiento al ser implementado este cambio de forma permanente de selección y de forma introspectiva. buscando mejorar la cota superior del peor caso de IQS sin afectar notorialmente el rendimiento en el caso promedio.