

Universidad de Costa Rica
Escuela de Ingeniería Eléctrica

Estructuras de Datos Abstractas
y Algoritmos para ingeniería (IE0217)

Proyecto de investigación
Algoritmo MOMSort

Estudiante:
Raquel Sofía Corrales Marín
B92378

Profesor:
Juan Carlos Coto Ulate

9 de diciembre de 2020

Índice

0.1. Introducción	2
0.2. Discusión	3
0.2.1. Estructura del Algoritmo.	3
0.2.2. Comportamiento del algoritmo en términos de com- plejidad	4
0.3. Implementación del algoritmo.	7
0.4. Conclusiones	10
Bibliografía	11

0.1. Introducción

El algoritmo MOMSort “*The Median of medians algorithm*”, fue publicado en 1972, tal como se detalla en [1], por los científicos de la computación Manuel Blum, Bob Floyd, Ron Rivest, Bob Tarjan y Vaughan Pratt, donde destaca que los primeros cuatro han sido galardonados con el premio Turing, donde este premio es dado por la *Association for Computing Machinery (ACM)* que tal como se indica en [2], esta consiste en la mayor sociedad de profesionales, educadores y científicos dedicados a la computación en todo el mundo. Cabe mencionar que [2] también hace referencia al premio Turing como el galardón “a grandes contribuciones de importancia duradera en computación” [2] y además, considerado por muchos como el Premio Nobel de la computación.

El algoritmo MOMSort, que tal como lo indica su nombre, constituye el algoritmo para obtener la mediana de las medianas, así mismo, consiste en una variación del “Quick-Select Algorithm”, donde [1] indica que ambos algoritmos se encuentran en la clasificación de algoritmos de “Divide and conquer”, que encuentra su traducción en el idioma español como “divide y vencerás”, donde en la referencia [1] se explica esta estrategia para resolver un problema, iniciando por descomponer en pequeños sub-problemas el problema general, donde haciendo uso de recursividad se procede a resolver estos pequeños sub-problemas y finalmente a construir o engranar las sub-soluciones obtenidas en una solución global para el problema. Siendo esta una efectiva y poderosa estrategia para abordar problemas en los cuales se pueda mantener los costos de descomponer y componer, así como el equilibrio en la complejidad de los sub-problemas.

Por tanto, bajo la estrategia de “divide y vencerás” nace el algoritmo MOMSort con el objetivo de encontrar la mediana de un arreglo de datos en desorden, contemplando una solución más optimizada que no incurra en emplear la solución trivial, la cual sería ordenar el arreglo y luego encontrar el dato que se encuentra en la posición central de los elementos, lo cual si pensamos en una lista de tamaño considerablemente extensa, no es tan eficiente considerando el tiempo que conllevaría comparar cada elemento para ordenarlo y recorrer la lista. La implementación de este algoritmo es aún más poderosa, pues logra encontrar la mediana de un arreglo en desorden haciendo uso de la mediana de las medianas de sub-grupos de elementos del mismo arreglo. De esta manera, a continuación, se profundizará a detalle en la estructura del algoritmo en análisis, así como en su complejidad e implementación.

0.2. Discusión

0.2.1. Estructura del Algoritmo.

En primer lugar, se explicará a profundidad la estructura y consideraciones del algoritmo en estudio, para esto se tomará como referencia la estructura para el algoritmo que describe [1].

Se parte por tanto del elemento en estudio, un arreglo en desorden de n elementos, donde para encontrar su mediana aplicando el algoritmo se debe seguir el siguiente listado de pasos:

- **Paso 1:** Dividir el arreglo en $\frac{n}{5}$ grupos, cada subgrupo conformado con 5 elementos, considerando que el último grupo puede quedar conformado por 5 o menos elementos.
- **Paso 2:** Proceder a encontrar la mediana de cada subgrupo de 5 elementos, para esto [1] indica que se puede utilizar cualquier tipo de resolución.
- **Paso 3:** Hacer un llamado recursivo al método *MOM-Select* para obtener el nuevo pivote “The median of medians”, la mediana de las medianas, a partir de las medianas de cada subgrupo encontradas en el *paso 2*, esto para obtener el pivote con valor x .
- **Paso 4:** Determinar el índice i del valor x e intercambiar $A[r]$ con $A[i]$, haciendo que el valor de x esté ahora en la posición del pivote.

Es importante mencionar que [5] expone algunas consideraciones entre los pasos descritos anteriormente, las cuales son: En primer lugar, para el *paso 2*, especifica que se debe ordenar cada uno de los grupos de 5 elementos para encontrar la mediana, haciendo este paso más específico en el modo de calcular cada una de las medianas.

Luego del *paso 3*, cuando ya se ha encontrado la mediana de las medianas de cada grupo, indica dividir el grupo alrededor de dicho valor, x . Tomando en cuenta que siendo k uno más que el número de elementos en el lado bajo de la partición, de modo que x es el k -ésimo elemento más pequeño y hay $n-k$ elementos en el lado alto de la división, se debe tomar las siguientes consideraciones:

Recordando que i corresponde al índice del valor x (*The median of medians*), si $i=k$, devolver el valor x . De lo contrario, repetir todo el método de forma recursiva para encontrar el i -ésimo elemento más pequeño en el lado de elementos menores a x si $i < k$, o el $i-k$ elemento más pequeño en el lado de elementos mayores a x si $i > k$.

0.2.2. Comportamiento del algoritmo en términos de complejidad

Tal como indican los autores en [1], [3], [4], el *algoritmo MOMSort* “*The Median of medians algorithm*”, presenta una complejidad lineal, de la forma:

$$O(n) \quad (1)$$

Esto se explicará haciendo uso del siguiente diagrama, que tal como lo explica [1], cada círculo representa un valor del conjunto de n elementos, donde las columnas verticales constituyen los grupos de 5 elementos y por su parte, los círculos blancos constituyen las medianas de cada grupo, además, el círculo destacado por la letra x es la mediana de las medianas.

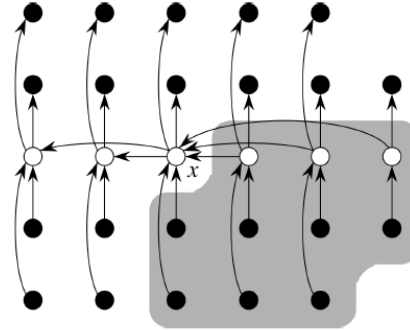


Figura 1: Diagrama de la complejidad del algoritmo. [1]

Continuando con la explicación de la complejidad del algoritmo, [1] indica que las flechas están colocadas partiendo de valores mayores y dirigidas a valores menores, por tanto la parte sombreada delimita valores mayores que el valor x , donde además, este subconjunto representa

una fracción del total de elementos, donde tal como se observa en la *figura 1*, cada columna vertical aporta al menos 3 elementos a este subconjunto, representada en su forma fraccional como $\frac{3n}{10}$, por tanto, al realizar un llamado recursivo al método, este deberá hacerse sobre el $\frac{7n}{10}$ restante.

Sin embargo, para concluir la linealidad de la complejidad del algoritmo en estudio, se deben tomar mayores consideraciones, donde tal como se detalla y se concluye en [1], los pasos 1 y 4 descritos anteriormente que constituyen los procedimientos del algoritmo, se realizan en un tiempo lineal de la forma $O(n)$. Así mismo, el paso 2, porque cada uno de los grupos creados tienen como máximo 5 elementos, por tanto, se puede encontrar su mediana en tiempo constante independientemente del método empleado para este fin. Considerando el paso 3 y la recursividad que se le debe aplicar a este en alrededor de un total de $\frac{n}{5}$, [1] concluye que la recurrencia y complejidad del algoritmo está dada por:

$$T(n) \leq an + T\left(\frac{n}{5}\right) + T\left(\frac{7n}{10}\right) \quad (2)$$

En la referencia [1] se utiliza el siguiente árbol de recursividad para confirmar y demostrar de una forma más gráfica la complejidad del algoritmo de las medianas, donde se explica que este árbol tiene un nodo correspondiente a cada llamada recursiva, además, cada nodo está etiquetado con el costo local (no recursivo) de la llamada. Además, [1] indica que el costo total del llamado se obtiene sumando el costo en cada nivel y posteriormente sumando los costos de cada uno de los niveles. A continuación, en la *figura 2* se muestra el árbol recursivo utilizado en el análisis de la complejidad del algoritmo en estudio.

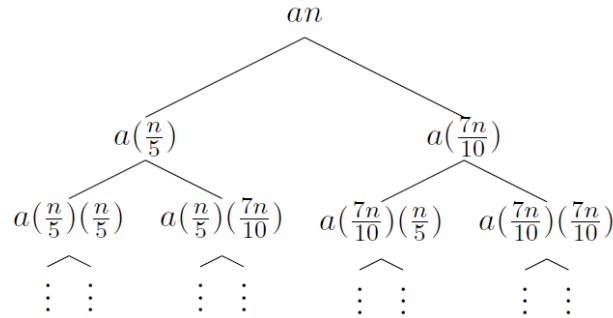


Figura 2: Árbol de recursividad. [1]

Por tanto, siguiendo la recomendación de [1] mencionada anteriormente para hallar el costo total, a partir de la *figura 2*, tenemos que la suma de costos por niveles arrojan el siguiente resultado:

$$Nivel\ 1 = an$$

$$Nivel\ 2 = a\left(\frac{n}{5}\right) + a\left(\frac{n}{5}\right) = a\left(\frac{9}{10}\right)n$$

$$Nivel\ 3 = a\left(\frac{n}{5}\right)\left(\frac{n}{5}\right) + a\left(\frac{n}{5}\right)\left(\frac{7n}{10}\right) + a\left(\frac{7n}{10}\right)\left(\frac{n}{5}\right) + a\left(\frac{7n}{10}\right)\left(\frac{7n}{10}\right) = a\left(\frac{81}{100}\right)n$$

$$\vdots$$

Por tanto sumando los costos de cada nivel se obtiene:

$$an + a\left(\frac{9}{10}\right)n + a\left(\frac{81}{100}\right)n, \dots, a\left(\frac{9}{10}\right)^i n \quad (3)$$

Lo cual a su vez, con la *ecuación (2)* y *(3)*, se puede representar de la siguiente manera:

$$T(n) \leq an \left(\sum_{i=0}^{\infty} \left(\frac{9}{10}\right)^i \right) = an \left(\frac{1}{1-\left(\frac{9}{10}\right)} \right) = 10an = O(n)$$

$$T(n) \leq O(n) \quad (4)$$

Por tanto, tal como se detalló y demostró con anterioridad, se concluye que el algoritmo en estudio, *MOMSort algoritmo*, se ejecuta en tiempo lineal de la forma $O(n)$. Resulta además, oportuna la conclusión de [4], la cual indica que la llamada recursiva que calcula la mediana a partir de este algoritmo, no excede el comportamiento lineal para el peor de los casos, esto porque la lista de medianas es el 20 % del tamaño de la conjunto total, mientras que la otra llamada recursiva se repite en como máximo el 70 % del otro subconjunto restante, tal como se detalla en la *ecuación 2* y de la mano con lo abordado en [1].

0.3. Implementación del algoritmo.

```
1 def median_of_medians(arreglo):
2     """
3     Metodo: "Median of medians" implementacion del algoritmo
4     que obtiene la mediana de
5     un arreglo a partir de la mediana de sub-arreglos
6     Parametro: arreglo en desorden
7     Retornar: llamado al metodo "seleccionar pivote" si el
8     arreglo es no nulo
9     """
10
11     if arreglo is None or len(arreglo) == 0:
12         return None
13
14     return seleccionar_pivote(arreglo, len(arreglo) // 2)
15
16 def seleccionar_pivote(arreglo, k):
17     """
18     Seleccionar el pivote que corresponde al k-esimo elemento
19     del arreglo
20     Parametro: (arreglo) del cual se necesita obtener la
21     mediana
22     (k) k-esimo elemento del arreglo
23     Retornar: El valor del pivote
24     """
25
26     # Dividir el arreglo en grupos de 5
27     grupos = [arreglo[i : i+5] for i in range(0, len(arreglo)
28     , 5)]
29
30     #Ordenar cada n\5 grupo
31     ordenar_grupos = [sorted(grupo) for grupo in grupos]
32
33     #Obtener la mediana de cada n\5 grupo
34     medianas = [grupo[len(grupo) // 2] for grupo in
35     ordenar_grupos]
36
37     #Encontrar la mediana de las medianas
38     if len(medianas) <= 5:
39         pivote = sorted(medianas)[len(medianas) // 2]
40     else:
41         #Llamado recursivo al metodo
42         pivote = seleccionar_pivote(len(medianas) // 2)
```



```

1 #Dividir el arreglo alrededor del pivote
2     p = dividir_arreglo(arreglo, pivote)
3
4     #Si el pivote esta en la posicion k
5     if k == p:
6         #Seleccionar ese pivote como "Median of medians"
7         return pivote
8     #Si el elemento k es menor al pivote
9     if k < p:
10        #Seleccionar un nuevo pivote al lado izquierdo de la
11        division (p).
12        return seleccionar_pivote(arreglo[0:p], k)
13    #Si el elemento k es mayor al pivote
14    else:
15        #Seleccionar un nuevo pivote al lado derecho de la
16        division (p).
17        return seleccionar_pivote(arreglo[p+1:len(arreglo)],
18        k - p - 1)
19
20 def dividir_arreglo(arreglo, pivote):
21     """
22     Dividir el arreglo alrededor del pivote encontrado
23     Parametros: (arreglo) arreglo para dividir, (pivote) el
24     pivote para dividir el arreglo
25     Retorna: posicion final del pivote utilizado para la
26     division dentro del arreglo
27     """
28     menor = 0
29     mayor = len(arreglo) - 1
30     i = 0
31
32     while i <= mayor:
33         if arreglo[i] == pivote:
34             i += 1
35
36         elif arreglo[i] < pivote:
37             arreglo[menor], arreglo[i] = arreglo[i], arreglo[
38             menor]
39             menor += 1
40             i += 1
41
42         else:
43             arreglo[mayor], arreglo[i] = arreglo[i], arreglo[
44             mayor]
45             mayor -= 1
46
47     return menor

```


Tal como se observa con anterioridad, se procedió a implementar los métodos (median_of_medians), (seleccionar_pivote) y (dividir_arreglo), tomando las consideraciones descritas en [1] para la estructura del algoritmo en estudio, ahora como parte del código se procede a hacer el llamado de la función “median_of_medians” pasándole por parámetro los arreglos definidos y finalmente imprimiendo el valor que retorna el cual corresponde a la mediana del arreglo, tal como se muestra a continuación:

```

1 arreglo_1 =
  [25,24,33,39,3,18,19,31,23,49,45,16,1,29,40,22,15,20,
2 24,4,13,34]
3 arreglo_2 =
  [24,14,7,5,21,76,88,5,2,11,3,32,7,16,9,35,28,1,18]
4 mediana_1 = median_of_medians(arreglo_1)
5 mediana_2 = median_of_medians(arreglo_2)
6 print ("Median of medians del arreglo 1:",mediana_1)
7 print ("Median of medians del arreglo 2:",mediana_2)

```

Cabe mencionar que se utilizaran los arreglos 1 y 2, con valores seleccionados al azar. A continuación se muestra el resultado obtenido luego de correr el código:



```

Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/pscore6

PS C:\Users\juanc> conda activate base
PS C:\Users\juanc> & C:/Users/juanc/Anaconda3/python.exe "c:/Users/juanc/Desktop/Estructuras (Proyecto)/MOMSort.py"
Median of medians del arreglo 1: 24
Median of medians del arreglo 2: 5
PS C:\Users\juanc>

```

Figura 3: Resultado de compilación en terminal.

Para comprobar que el resultado es correcto, procedemos a ordenar el arreglo de forma manual y a seleccionar el dato que se encuentra en la posición central:

Cuadro 1: Comprobación de mediana arreglo 1.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
1	3	4	13	15	16	18	19	20	22	*23	*24	24	25	29	31	33	34	39	40	45	49

Cuadro 2: Comprobación de mediana arreglo 2.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	2	3	5	5	7	7	9	11	14	16	18	21	24	28	32	35	76	88

Cabe mencionar que para el arreglo 1, al ser una cantidad de datos par, se procedió a encontrar la media entre los datos centrales (23 y 24) $\rightarrow \frac{23+24}{2} = 23.5 \approx 24$, tal como lo detalla en [6] y para el caso del arreglo 2, al ser una cantidad de datos impar, se procedió a encontrar el valor central (14). Comprobando de esta manera que el resultado del código es correcto.

0.4. Conclusiones

El *algoritmo MOMSort “The Median of medians algorithm”*, resulta ser muy eficiente, tomando como referencia el comportamiento del algoritmo en términos de complejidad, pues tal como se detalló con anterioridad, este algoritmo se desarrolla en tiempo lineal de la forma $O(n)$. Además, en [1] y [5] se concluye que el enfoque de la mediana de las medianas es muy representativo en los algoritmos de partición, ya que la estructura que utiliza este para encontrar el pivote más acertado, es muy utilizado en algoritmos de selección que se desarrollen de manera más exacta, por ejemplo los algoritmos de selección rápida. Finalmente es importante mencionar que la estructura del algoritmo es muy clara, por tanto la implementación del algoritmo no resulta ser tan compleja, esto gracias a que el algoritmo se encuentra bajo la clasificación de “divide y vencerás” lo cual hace que la implementación sea dividida en pequeñas tareas que retornan resultados claves, que finalmente engranados en la distintas etapas, convergen en una solución global, “The medians of medians”.

Bibliografía

- [1] A. TOLMACH, *Lecture 1: Introduction; Divide-And-Conquer; Selection*. 2017. [Online]. Disponible: https://web.cecs.pdx.edu/~apt/cs584_2017/
- [2] S. UCHITEL, *El premio Turing*. Buenos Aires, Argentina:Departamento de computación, Facultad de Cs. Exactas y Naturales, Universidad de Buenos Aires, 2012. [Online]. Disponible: <https://www.dc.uba.ar/wp-content/uploads/2018/12/Julio2012.pdf>
- [3] A. ALEXANDRESCU, *Fast Deterministic Selection*. The D Language Foundation, 2016. [Online]. Disponible: <https://dblp.org>
- [4] H. SARAOGI, *Median of medians algorithm*. 2013. [Online]. Disponible: <http://himangi774.blogspot.com/2013/09/medianof-medians.html>.
- [5] T. H. CORMEN, C. STEIN, R. L. RIVEST, AND C. E. LEISERSON, *Introduction to Algorithms*. McGraw-Hill Higher Education, 2nd edition, 2001.
- [6] F. J. MARCO, *Mediana*. Economipedia, 2020. [Online]. Disponible: <https://economipedia.com/definiciones/mediana.html>