CI-1221 Estructuras de Datos y Análisis de Algoritmos, II ciclo 2012, grupo 2: III Tarea Programada

Daniel Arguedas Calderón (B10542)
Daniel Obando Montero (A74639)

1. Búsqueda Exhaustiva

1. Encuentre una representación vectorial de la solución al problema, indicando claramente qué significa cada una de sus entradas.

   La solución esta representada por el vector sigma

   $$\sigma =< \sigma_1, \sigma_2, \ldots, \sigma_n >$$

   Donde σ_i representa el dedo (σ=1,2,3,4,5) con el que se va tocar la i-ésima nota ($i = 1, 2, \ldots, n$).

2. Determine el espacio E en el que habita el vector solución, definiendo claramente el (los) conjunto(s) a el (los) que pertenece(n) cada una de sus entradas.

   Definimos $\Delta|\{1, 2, 3, 4, 5\} \in \Delta$. Entonces, ya que tenemos n notas,

   $$\sigma \in E = \Delta \times \Delta \times \ldots \times \Delta = \Delta^n$$

3. Determine la cardinalidad del espacio.

   $$|E| = 5^n$$

4. Si es posible, acote el espacio utilizando una restricción del tipo $E' = \{\sigma \in E$:restricción en $\sigma\}$.

   Tenemos que la solución acotada pertenece al espacio:

   $$E' = \{\sigma \in E : \sigma_i \neq \sigma_{i+1}\}$$

   Así eliminamos todas las opciones que toquen dos teclas consecutivas con el mismo dedo. Sin embargo; debemos tomar en consideración el resto de restricciones. Por

Cuadro 1: Intervalos permitidos entre cada par de dedos.

Dedo	1	2	3	4	5
1	0	6	9	10	12
2	6	0	5	7	9
3	9	5	0	3	5
4	10	7	3	0	4
5	12	9	5	4	0

Cuadro 2: Intervalos permitos para pases de pulgar.

Dedo	2	3	4
Intervalo (teclas)	7	6	4

ejemplo, no se puede tocar una nota negra con el pulgar a menos de que se encuentre en una secuencias de teclas repetidas (la misma nota varias veces), en tal caso se permite siempre y cuando no inicie la secuencia. Otra restricción relacionada con las notas repetidas es que se debe usar una secuencia de de dedos descendente, con la excepción de que si llega al pulgar y es necesario volver a tocar la misma tecla se puede usar un dedo de orden superior. Para poder usar cualquier par de dedos hay que respetar los intervalos dados en el Cuadro 1, que indica el alcance máximo que hay entre dos dedos, si se observa con detenimiento la matriz es simétrica por lo que el intervalo funciona en ambas direcciones. Finalmente, está el criterio para pases de pulgar donde las restricciones son: no se pueden hacer pases de pulgar que involucren al dedo 5 (meñique); y que se respeten los intervalos que se muestran en el Cuadro 2. Estas dos, sin olvidar la restricción del pulgar tocando teclas negras.

5. Escriba un algoritmo en C++ que explore todas las soluciones candidatas en E o E' y devuelva una solución óptima, siguiendo el encabezado de función establecido en el archivo AXXXXX-BXXXXX-TP3.cpp.

6. Determine una cota asintótica para el tiempo de ejecución del algoritmo.

   Teniendo que el tiempo de ejecución es:

   $$5T(n-1) + c_0$$

   Resolvemos la recurrencia:

   $= 5[5T(n-2) + c_0] + c_0$

   $= 5^2T(n-2) + 5c_0 + c_0$

   $= 5^2[5T(n-3) + c_0] + 5c_0 + c_0$

   $= 5^3T(n-3) + 5^2c_0 + 5c_0 + c_0$

   $\vdots$

$= 5^k T(n-k) + 5^{k-1}c_0 + 5^{k-2}c_0 + \ldots + 5^0 c_0$

Entonces, cuando k=n:

$= 5^n T(0) + 5^{n-1}c_0 + 5^{n-2}c_0 + \ldots + c_0$

$= 5^n T(0) + \left[\frac{5^{n+1}-1}{4}\right] c_0$

$= 5^n$

Por lo tanto, la cota superior está dada por:

$$T(n) = O(5^n)$$

7. Corra su algoritmo con cada una de las melodías de prueba y registre para cada melodía el número de giros realizados y el tiempo de ejecución. Si el tiempo excede una hora para cualquiera de los tamaños, puede abortar la prueba y omitir las pruebas correspondientes a tamaños más grandes. Reporte en un cuadro el número de giros producidos para cada melodía y grafique el tiempo de ejecución vs. cantidad de notas para cada una de las melodías, representando los tiempos en una escala logarítmica . Diga si la progresión de tiempos obtenidos fue la esperada. Explique.

   Los resultados de las pruebas para cada melodía se muestran en el Cuadro 3. Cuando la melodía no tuvo solución se indicó con NA (No Aplica).

   Los tiempos promedio se muestran en el Cuadro 4. Se excluyeron las melodías que no aplicaron en el cálculo de los promedios.

   La gráfica representando el tiempo de ejecución promedio para un grupo de melodías según la cantidad de notas vs. cantidad de notas se muestra en la Figura 1. Claramente coincide con la progresión esperada porque entre más notas hay más soluciones posibles por revisar y el algoritmo tiene que probarlas todas para determinar la mejor. Se ven ciertas excepciones como la melodía de 15 notas y las de 16, dada la poca cantidad de melodías probadas con esa cantidad de notas. Además, existe la posibilidad de que no todos los caminos sean válidos, por lo que al encontrar un *callejón sin salida*, deja de probar por ese camino y se devuelve a uno que si sea válido, evitando así un tiempo de búsqueda mayor en caminos que no llevan a una solución válida por la infracción de algunas de las restricciones especificadas para el algoritmo.

Cuadro 3: Comparación de la cantidad de pases de pulgar y tiempos de ejecución de las distintas melodías utilizando búsqueda exhaustiva.

No.	Melodía	Cantidad de Notas	Pases de Pulgar	Tiempo (ms)
1	El día que me quieras	14	0	201, 82
2	Estrellita	14	0	1 459, 76
3	La Cucaracha	17	0	9 228, 36
4	The Legend Of Zelda	14	0	53, 82
5	Tetris	19	0	75 573, 52
6	Futurama	16	0	1 726, 74
7	Auld lang Syne	14	0	1 790, 65
8	Minuet	16	0	167, 76
9	You belong with me	20	2	424, 11
10	Mario Bros	13	0	72, 39
11	Happy Birthday To You	12	0	449, 61
12	Martinillo	14	0	2 082, 07
13	Inspector Gadget	17	1	1 226, 05
14	Oda a la alegría	20	1	528 387, 40
15	Hakuna Matata	11	0	62, 86
16	Schindler's List	16	2	11, 08
17	Never Say Never	14	NA	NA
18	Campanero	14	0	2 032, 48
19	Joy to the world	14	1	727, 25
20	La paloma	13	0	1 054, 14
21	Fantaisie Impromptu	15	0	60, 40
22	Moonlight Sonata	12	0	0, 82
23	Clocks	16	NA	NA
24	Pumped Up Kicks	12	0	145, 94
25	Jingle Bells	14	0	2 136, 34

Cuadro 4: Tiempos promedio de acuerdo a la cantidad de notas utilizando búsqueda exhaustiva.

Cantidad de Notas	Número de Melodías	Promedio
11	1	62, 86
12	3	198, 79
13	2	563, 27
14	8	1 310, 52
15	1	60, 40
16	3	635, 19
17	2	5 227, 21
19	1	75 573, 52
20	2	264 405, 76

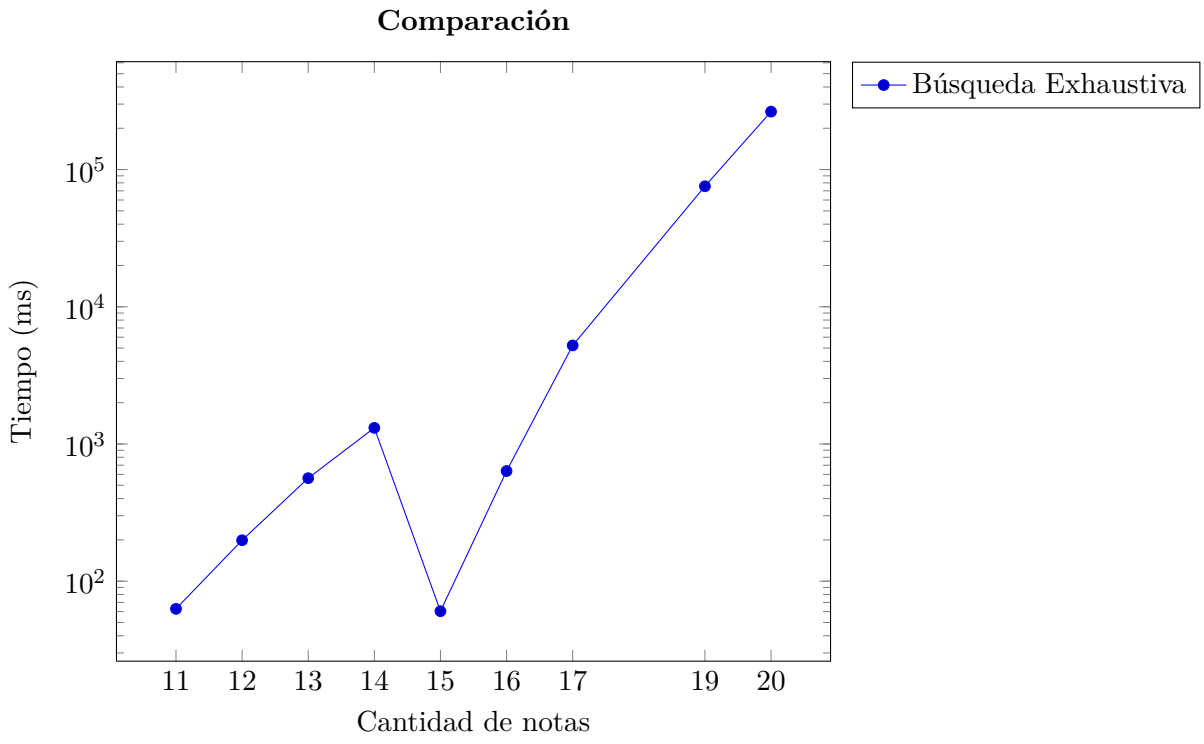

Figura 1: Gráfico comparativo de los tiempos promedio de ejecución de búsqueda exhaustiva según la cantidad de notas de las melodías.

2. Programación Dinámica

1. Determine si el problema se puede resolver a partir de soluciones a subproblemas, y si es así, describa cómo construir tal solución.

   Para lograr resolver el problema mediante subproblemas, debemos notar que cada vez que tocamos una nota, queda una nota menos por tocar. Luego, si encontramos la menor cantidad de pases, desde la primera nota hasta la i-ésima nota ($i = 1 \dots n$), podemos encontrar cuál es el mejor dedo del que debemos *venir* para poder tocar esa nota con el j-ésimo dedo ($j = 1 \dots 5$). Entonces, si encontramos la mejor secuencia de dedos y de pases de pulgar viniendo de la i-ésima nota hacia la $i+1$-ésima nota, podemos determinar la mejor secuencia para la $i+2$-ésima nota apartir de lo calculado para la $i+1$-ésima nota y así sucesivamente hasta llegar a la n-ésima nota.

2. Determine un oráculo (describiendo claramente su significado y el de sus argumentos) , el valor objetivo, y los pasos base y recursivo que permiten construirlo.

   Oráculo:

   $\rho_i(j)$ =cantidad mínima de pases obtenible desde la nota $1 \dots i$ tocando la i-ésima nota ($i = 1 \dots n$) con el j-ésimo dedo($j = 1 \dots 5$).

   Base:

   $\rho_0(j) = 0$ para $j = 1 \dots 5$

   Meta:

   $$\text{mín} \begin{cases} \rho_n(1) \\ \rho_n(2) \\ \rho_n(3) \\ \rho_n(4) \\ \rho_n(5) \end{cases}$$

   Recursivo:

   mín $\rho_{i-1}(k) + x$

   Tal que:

   $x = 0$; si no hay pase de pulgar entre el k-ésimo dedo, tocando la $(i-1)$-ésima nota, y el j-ésimo dedo tocando la i-ésima nota.

   $x = 1$; si hay pase de pulgar entre el k-ésimo dedo, tocando la $(i-1)$-ésima nota, y el j-ésimo dedo tocando la i-ésima nota.

   Con $i = 1 \dots n$ y $k = 1 \dots 5$.

3. Basado en su respuesta al punto anterior, escriba un algoritmo que resuelva el problema.

4. Determine una cota asintótica para el tiempo de ejecución del algoritmo.

   $T(n) = 10n + (n)[(5 + c_1)(5)] + n + c_0$

$= 10n + 25n + 5nc_1 + n + c_0$

$= n(36 + 5c_1) + c_0$

Por lo tanto, la cota superior está dada por:

$$T(n) = O(n)$$

5. Corra su algoritmo con cada una de las melodías de prueba e indique si las soluciones obtenidas contienen la misma cantidad de pases del pulgar que las obtenidas mediante el algoritmo de búsqueda exhaustiva. Grafique los tiempos de ejecución vs. el tamaño del arreglo, representando los tiempos en escala logarítmica, y agregue al gráfico la curva correspondiente a los tiempos producidos por el algoritmo de búsqueda exhaustiva. Compare las curvas y diga si la relación entre los tiempos fue la esperada.

   Los resultados de las pruebas para cada melodía se muestran en el Cuadro 5. Cuando la melodía no tuvo solución se indicó con NA (No Aplica).

   Los tiempos promedio se muestran en el Cuadro 6. Se excluyeron las melodías que no aplicaron en el cálculo de los promedios. Se observa que la mayoría de los tiempos se encuentran en 1 ms, esto es de esperarse dado que su cota asintótica es $O(n)$, además las cantidades de notas no varían sinificativamente por eso los tiempos son tan similares.

   La gráfica representando el tiempo de ejecución promedio para un grupo de melodías según la cantidad de notas vs. cantidad de notas se muestra en la Figura 2. Claramente se ve que el algoritmo de programación dinámica tiene un tiempo de ejecución mucho mejor que el de búsqueda exhaustiva, y se podría decir que tiene un leve crecimiento de acuerdo a la cantidad de notas. En algunos casos se observa que el tiempo promedio de las melodías con mayor cantidad de notas es menor que el de las que se probaron con menor cantidad de notas, esto se debe a la cantidad de melodias probadas con ciertas cantidades de notas son menores. Tómese el caso de las melodías de de 17 notas (2 melodías) con un promedio de $1,39$ ms a la de 19 notas (1 melodía) con un promedio de $1,18$.

6. Finalmente, en el Cuadro 7 se muestran los resultados de correr las melodías completas con el algoritmo de programación dinámica, lo tiempos son menores que los del Cuadro 5 porque se omitió la parte del código que imprimía el vector sigma con la solución. Nuevamente, se utilizó NA para denotar que la melodía no tiene solución cumpliendo con las restricciones dadas. Es importante destacar que dados los tiempos de ejecución de programación dinámica se pudieron hacer pruebas con las melodías completas, 25 notas en adelante que también se pueden observar en la Figura 2.

Cuadro 5: Comparación de la cantidad de pases de pulgar y tiempos de ejecución de las distintas melodías utilizando programación dinámica.

No.	Melodía	Cantidad de Notas	Pases de Pulgar	Tiempo (ms)
1	El día que me quieras	14	0	1,24
2	Estrellita	14	0	1,00
3	La Cucaracha	17	0	1,20
4	The Legend Of Zelda	14	0	1,00
5	Tetris	19	0	1,18
6	Futurama	16	0	1,19
7	Auld lang Syne	14	0	1,00
8	Minuet	16	0	1,02
9	You belong with me	20	2	1,30
10	Mario Bros	13	0	0,94
11	Happy Birthday To You	12	0	0,87
12	Martinillo	14	0	1,16
13	Inspector Gadget	17	1	1,58
14	Oda a la alegría	20	1	1,24
15	Hakuna Matata	11	0	0,85
16	Schindler's List	16	2	1,11
17	Never Say Never	14	NA	NA
18	Campanero	14	0	1,02
19	Joy to the world	14	1	0,96
20	La paloma	13	0	1,16
21	Fantaisie Impromptu	15	0	1,20
22	Moonlight Sonata	12	0	0,96
23	Clocks	16	NA	NA
24	Pumped Up Kicks	12	0	0,86
25	Jingle Bells	14	0	1,08

Cuadro 6: Tiempos promedio de acuerdo a la cantidad de notas utilizando programación dinámica.

Cantidad de Notas	Número de Melodías	Promedio
11	1	0,85
12	3	0,90
13	2	1,05
14	8	1,06
15	1	1,20
16	3	1,11
17	2	1,39
19	1	1,18
20	2	1,27

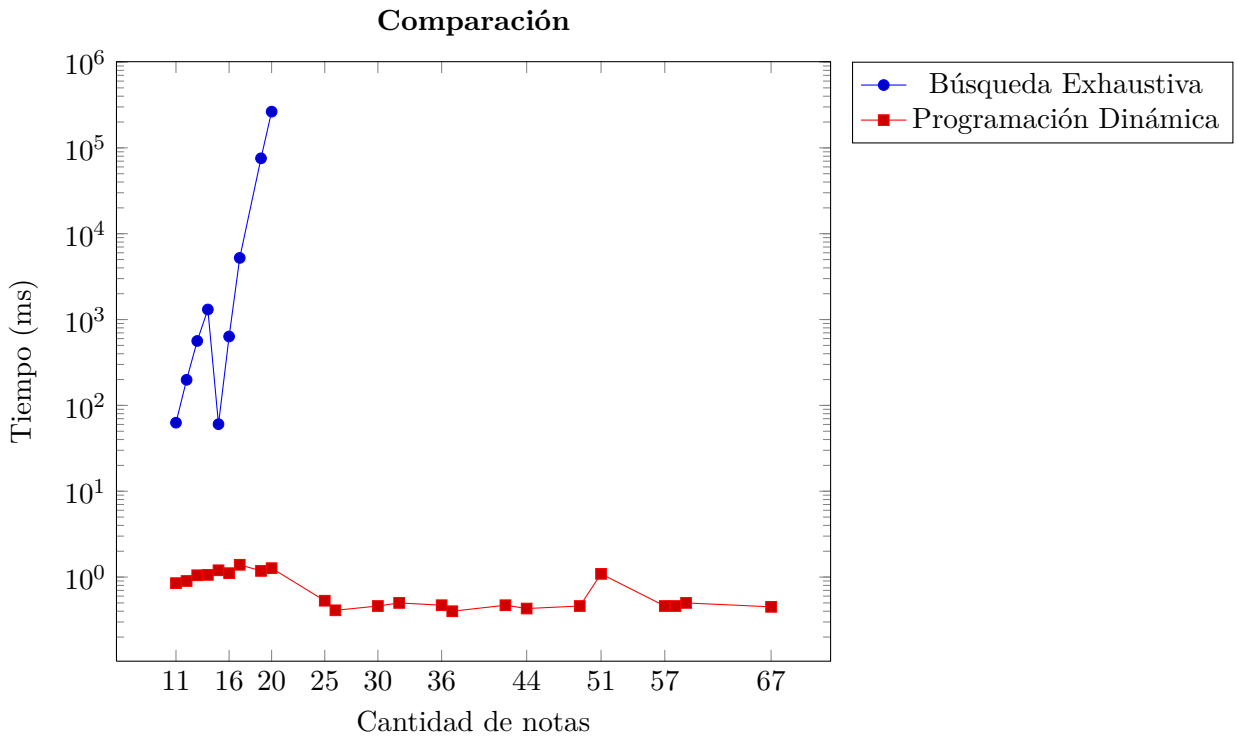

Figura 2: Gráfico comparativo de los tiempos promedio de ejecución de búsqueda exhaustiva y programación dinámica según la cantidad de notas de las melodías.

Cuadro 7: Comparación de la cantidad de pases de pulgar y tiempos de ejecución de las distintas melodías utilizando programación dinámica con melodías completas.

No.	Melodía	Cantidad de Notas	Pases de Pulgar	Tiempo (ms)
1	El día que me quieras	57	2	0,46
2	Estrellita	42	1	0,49
3	La Cucaracha	36	1	0,47
4	The Legend Of Zelda	44	0	0,43
5	Tetris	37	0	0,40
6	Futurama	42	0	0,45
7	Auld lang Syne	59	0	0,50
8	Minuet	67	1	0,45
9	You belong with me	58	4	0,46
10	Mario Bros	49	0	0, 46
11	Happy Birthday To You	25	0	0, 53
12	Martinillo	32	0	0,43
13	Inspector Gadget	30	NA	NA
14	Oda a la alegría	30	2	0,41
15	Hakuna Matata	26	0	0, 41
16	Schindler's List	45	NA	NA
17	Never Say Never	25	NA	NA
18	Campanero	32	0	0, 56
19	Joy to the world	30	1	0, 51
20	La paloma	44	1	0, 43
21	Fantaisie Impromptu	59	NA	NA
22	Moonlight Sonata	45	NA	NA
23	Clocks	96	NA	NA
24	Pumped Up Kicks	144	NA	NA
25	Jingle Bells	51	0	1,09

3. Algoritmos ávidos

1. Determine si es posible resolver el problema mediante un algoritmo ávido cuyo tiempo de ejecución sea asintóticamente mejor que el de programación dinámica. Explique su respuesta. Se llegó a la conclusión de que el problema no puede ser resuelto por algoritmos ávidos, no por el hecho de que el tiempo se ejecución sea mejor que el programación dinámica, sino por que no daría una solución óptima en todos los casos. A continuación se muestran unos contraejemplos con tres tipos de soluciones ávidas propuestas:

   a) Primera solución ávida: Se sabe que lo que se quieren minimizar son los pases de pulgar, una solución extrema sería evitarlos por completo y tratar de tocar la pieza usando únicamente los intervalos permitidos que se muestran en el Cuadro 1. Basta con tomar una melodía en la cual se haga un pase de pulgar. Tómese la melodía *Joy to the World* que hace un pase de pulgar con 14 notas. En el Cuadro 8 vemos que las primeras 5 notas son consecutivas y descendentes. Al ser notas descendentes el algoritmo ávido va a tratar de usar el dedo de mayor orden posible para tocar las notas respentando todas las restricciones. Siguiendo con esta idea, nos damos cuenta que al llegar a la sexta nota se quedaría sin opciones de avanzar por lo que ni siquiera daría una solución.

   b) Segunda solución ávida: esta se deriva de la anterior, lo que cambia es que si se van a permitir pases de pulgar y como es ascendente tomaría el dedo de menor orden posible para tocar las teclas. Tómese una escala ascendente de C (Do mayor). Observe el Cuadro 9 donde se muestra la solución con programación dinámica y la solución ávida propuesta. Como siempre va a tomar el dedo de menor posible inciaría con el dedo 1, luego tocaría la nota siguiente con el dedo 2 y en lugar de utilizar el dedo 3 para la próxima volvería a utilizar el dedo 1, de esta forma si produciría una solución, pero no sería óptima. La razón por la que la solución no es la mejor es que el algoritmo ávido daría 3 pases de pulgar (en las notas 2 - 3, 4 - 5 y 6 - 7) y el algortimo de progamación dinámica sólo tuvo un pase de pulgar (notas 3 - 4).

   c) Tercera solución ávida. podría ser que el algoritmo siempre utilice el dedo más cercano al dedo con el que tocó la nota anterior; siempre y cuando, se respeten las restricciones. Además, el algoritmo siempre intentará tocar la nota con el dedo *más alto* posible si la parte de la pieza que va tocando es descendente; o el dedo más bajo si va ascendente. Sin embargo, es muy probable que el algoritmo ávido llegue a un *callejón sin salida* y por lo tanto, no daría una solución óptima o por lo menos no peor a la de programación dinámica o búsqueda exhaustiva.

   A continuación uno de estos casos.Supongamos que el vector de melodía contiene lo siguiente:

   $$< 61, 63, 65, 66, 68, 70, 72, 73, 75 >$$

Cuadro 8: Comparación de la solución dada por búsqueda exhaustiva y la solución ávida propuesta con notas descendentes.

		Algoritmo	
No. Nota	Joy To The World	Exhaustivo	Ávido
1	72	5	5
2	71	4	4
3	69	3	3
4	67	2	2
5	65	1	1
6	64	4	x
7	62	3	x
8	60	2	x
6	67	4	x
10	69	5	x
11	69	4	x
12	71	5	x
13	71	4	x
14	72	5	x

En este caso, la única manera de que puede tocar toda la melodía sería si tocará la nota 65 con el dedo 1, ya que si no lo hace de esta manera, después llegaría a un punto donde debe tocar una negra con el pulgar (lo cuál es inválido) o hacer un *pase de pulgar* desde el dedo 4 (que también es inválido). Con el algortimo ávido propuesto, la solución dada sería la siguiente:

$$< 5, 4, 3, 2, ?, ?, ?, ?, ? >$$

Como se puede ver, el algoritmo llegó a un *callejón sin salida* por considerar una sola opción; y asegura que no hay solución para esa pieza. Sin embargo; como se detalló anteriormente, la solución se encuentra si toca la tercera nota con el dedo 1, opción que el algoritmo ávido ni siquiera sabía que existía. Entonces, la solución correcta sería de la siguiente manera:

$$< 5, 4, 1, 4, 3, 2, 1, 4, 3 >$$

Cuadro 9: Comparación de la solución dada por búsqueda exhaustiva y la solución ávida propuesta con notas descendentes.

		Algoritmo	
No. Nota	Escala Do mayor	Dinámico	Ávido
1	60	2	1
2	62	3	2
3	64	4	1
4	65	1	2
5	67	2	1
6	69	3	2
7	71	4	1
8	72	5	2