

Enumeración explícita aplicada a la resolución de un problema de Bingo

NICOLÁS E. OLIVARES GONZÁLEZ^{1,*}

¹Ingeniería Civil en Informática, Departamento de Ingeniería Informática, Universidad de Santiago de Chile

*Correo electrónico del autor: nicolas.olivares.g@usach.cl

Compiled September 23, 2016

En el marco de la asignatura de Algoritmos Avanzados, impartida por el Departamento de Ingeniería Informática de la Universidad de Santiago, este artículo presenta el análisis de la solución para un problema presentado por el enunciado número 2 del laboratorio de la asignatura. Se aborda el uso de las técnicas de resolución de problemas en base a algoritmos.

OCIS codes:

<http://dx.doi.org/10.1364/optica.XX.XXXXXX>

1. INTRODUCCIÓN

Este artículo aborda los conceptos estudiados en cursos anteriores de algoritmos, se basa en el enfoque de resolución de problemas con el uso de algoritmos de enumeración explícita o como son mejor conocido por la comunidad en general algoritmos de *Fuerza Bruta*. Para el desarrollo de este laboratorio se utiliza el lenguaje de programación C y para la compilación del programa se trabaja en el ambiente de Linux.

2. DESCRIPCIÓN DEL PROBLEMA

La marca de juegos de azar Bingo! tiene como característica en sus juegos la elección de números entre el 1 al 20. El jugador ganador solo debe acertar a 8 de estos números. Esta gran cantidad de combinaciones representa una dificultad a la hora de realizar los juegos.

3. MARCO TEÓRICO

Al momento de solucionar un problema, se plantean diversas maneras o métodos de resolución en base a las características del problema. Los algoritmos de enumeración descriptiva actúan sobre diversos tipos de problemas. Estos algoritmos son la primera opción de resolución que se viene a la mente al pensar en un problema donde involucra combinaciones de algún conjunto de números o elementos. Combinaciones sin repetición o combinaciones ordinarias de m elementos tomados de n en n (de orden n) son los distintos grupos de n elementos distintos que se pueden hacer con los m elementos que tenemos, de forma que dos grupos se diferencian en algún elemento y no en el orden de colocación. Se representa por $C_{m,n}$. ($n \geq m$).[1]

4. DESCRIPCIÓN DE LA SOLUCIÓN

Teniendo en consideración que el único dato de entrada proporcionado es un número entero $n > 8$, se describe la idea de solución.

A. IDEA 1

- Partiendo desde un conjunto de 8 números del 1 al 8 ordenados.
- Por cada número desde la derecha hacia la izquierda.
- El límite para cada posición está dado por $n-i$, siendo $i = 0 : 7$
- Mientras el número en la posición no sobrepase el límite, sumarle y ponerlo en la lista de combinaciones.

Algorithm 1. Combinaciones de Bingo 1

```

1: procedure BINGO1( $n$ )
2:    $arreglo = 1 : 8$ 
3:   for  $limite = n; limite \geq n - 7; limite --$  do
4:      $i = 7$ 
5:     while  $i \geq 0$  do
6:       while  $arreglo[i] < limite$  do
7:          $arreglo[i] ++$ 
8:         imprimir  $arreglo$ 
9:         if then
10:            $limite --$ 
11:            $i --$ 

```

B. IDEA 2

Esta idea se basa en que el primer grupo ordenado de 8 números menores será

$$1, 2, 3, 4, 5, 6, 7, 8$$

y el último grupo ordenado de 8 números de mayor valor dependiendo de n será

$$(a)$$

$$n - 7, n - 6, n - 5, n - 4, n - 3, n - 2, n - 1, n - 0$$

- Partiendo con un conjunto del 1 al 8, generado a partir de la anidación de cada número dependiente de su compañero a la izquierda.
- Teniendo el número límite de $n-7$ hasta n
- Sumar a las posiciones independientemente hasta que se alcance el límite y pasar a la suma del siguiente a la izquierda

Algorithm 2. Combinaciones de Bingo 2

```

1: procedure BINGO2(n)
2:   for  $a = 1; a \leq n - 7; a++$  do
3:     for  $b = a + 1; b \leq n - 6; b++$  do
4:       for  $c = b + 1; c \leq n - 5; c++$  do
5:         for  $d = c + 1; d \leq n - 4; d++$  do
6:           for  $e = d + 1; e \leq n - 3; e++$  do
7:             for  $f = e + 1; f \leq n - 2; f++$  do
8:               for  $g = f + 1; g \leq n - 1; g++$  do
9:                 for  $h = g + 1; h \leq n; h++$  do
10:                  imprimir  $abcdefgh$ 

```

5. ANÁLISIS DE LOS RESULTADOS

En esta sección se realizará un análisis del algoritmo implementado en base a la serie de preguntas habituales que se realizan para comprobar la validez de un algoritmo.

A. Análisis: Preguntas para el Algoritmo 1

1. ¿El algoritmo para en algún momento?
Si, el algoritmo para, cuando se haya pasado por todas las posiciones.
2. Y si para, ¿lo hace con la solución?
Este algoritmo no entrega todas las soluciones posibles, solo entrega aquellas que tienen a n como límite máximo B , por lo tanto no entregan aquellas restantes donde el límite máximo va desde n a 20
3. ¿Es eficaz? ¿Soluciona el problema?
Al no obtener la solución no se define como eficaz.
4. ¿Es eficiente?
Su orden de complejidad queda dado por la entrada n de manera $O(8n^3)$. Por ser de manera polinómica se puede decir eficiente pero aún así demora más
5. ¿Se puede mejorar?
Se puede mejorar aplicando otra manera de manejar el comportamiento del conjunto

6. ¿Existe otro método para solucionar el problema?

Definitivamente dado que este Algoritmo no soluciona por completo el problema, se busca otra manera de resolver el problema

Análisis: Preguntas para el Algoritmo 2

1. ¿El algoritmo para en algún momento?
Si para al completar todos los 8 ciclos anidados asociados.
2. Y si para, ¿lo hace con la solución?
Si entrega la solución para cada conjunto de elementos, esta solución presenta las combinaciones posibles como un conjunto de números
3. ¿Es eficaz? ¿Soluciona el problema?
Al entregar la solución completa, soluciona el problema por lo tanto es eficaz
4. ¿Es eficiente?
A pesar de entregar la solución, no es eficiente dado que su orden de complejidad $O(n^8)$ polinomial, demuestra que su falta de eficiencia.
5. ¿Se puede mejorar?
Con un buen uso de estructuras de datos y un proceso meticuloso de
6. ¿Existe otro método para solucionar el problema?
Definitivamente existen otros métodos que ayudan a resolver el problema y encontrar una solución

6. TRAZA DE LA SOLUCIÓN

La Figura 1 muestra la traza de cómo esta pensada la solución. Partiendo siempre desde un conjunto de 1 a 8, se tiene un límite $n-i$ para cada posición del conjunto con $i = 0 : 7$. Cuando el número en una posición vaya sumando 1 unidad a sí mismo, comprueba si se alcanza el límite $n-i$, si esto ocurre, es el número en la posición de la izquierda el que repite el proceso de suma y comprobación.

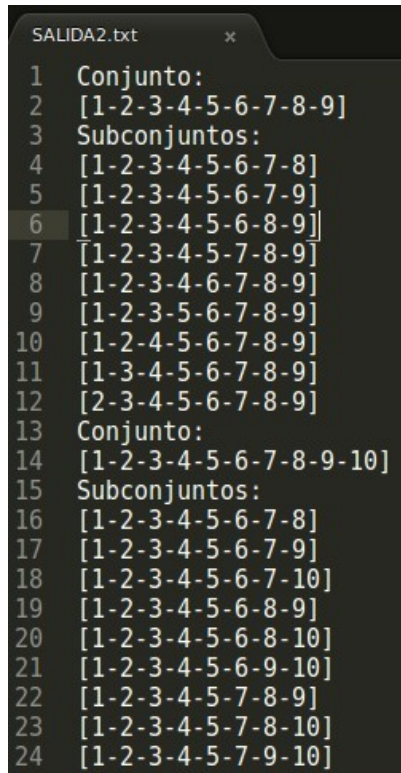
Fig. 1. Explicación de la traza para una entrada de $n = 9$

1	2	3	4	5	6	7	8
1	2	3	4	5	6	7	8
1	2	3	4	5	6	8	9
1	2	3	4	5	7	8	9
1	2	3	4	6	7	8	9
1	2	3	5	6	7	8	9
1	2	4	5	6	7	8	9
1	3	4	5	6	7	8	9
2	3	4	5	6	7	8	9

Los números destacados con colores representan los cambios al número en las posiciones, es decir la suma de una unidad.

En el programa generado, se puede obtener una idea del funcionamiento de ambos algoritmos. Para el Algoritmo 1 se pueden revisar los resultados en el archivo *SALIDA.TXT*, mientras que para el Algoritmo 2 se puede ver el resultado en el archivo *SALIDA2.TXT*.

Fig. 2. Demostración de algoritmo 2



```
SALIDA2.txt
1 Conjunto:
2 [1-2-3-4-5-6-7-8-9]
3 Subconjuntos:
4 [1-2-3-4-5-6-7-8]
5 [1-2-3-4-5-6-7-9]
6 [1-2-3-4-5-6-8-9]
7 [1-2-3-4-5-7-8-9]
8 [1-2-3-4-6-7-8-9]
9 [1-2-3-5-6-7-8-9]
10 [1-2-4-5-6-7-8-9]
11 [1-3-4-5-6-7-8-9]
12 [2-3-4-5-6-7-8-9]
13 Conjunto:
14 [1-2-3-4-5-6-7-8-9-10]
15 Subconjuntos:
16 [1-2-3-4-5-6-7-8]
17 [1-2-3-4-5-6-7-9]
18 [1-2-3-4-5-6-7-10]
19 [1-2-3-4-5-6-8-9]
20 [1-2-3-4-5-6-8-10]
21 [1-2-3-4-5-6-9-10]
22 [1-2-3-4-5-7-8-9]
23 [1-2-3-4-5-7-8-10]
24 [1-2-3-4-5-7-9-10]
```

La Figura 2 muestra el resultado del Algoritmo 2

7. CONCLUSIONES

Tras la realización del laboratorio y análisis de resultados, se concluye lo siguiente:

- Solo el Algoritmo 2 resulta acabar con la solución al problema.
- Si se itera el Algoritmo 1 puede acabar con las combinaciones restantes para los conjuntos desde n hasta 20
- Ninguno de los dos algoritmos en sí son de enumeración explícita como tal, o como se describe *Fuerza Bruta*, ya que se encuentran las combinaciones posibles inmediatamente y no se pasa por un proceso de revisión de criterio a *TODAS* las combinaciones.
- La aplicación de técnicas algorítmicas para la resolución de problemas implica el conocimiento de los diferentes tipos de recursos disponibles, incluso aquellos que no son eficientes y no son usados por la gran mayoría de los entendidos en el tema.

REFERENCES

1. L. B. Calmaestra, "Unidad didáctica: Combinatoria. combinaciones sin repetición." (2016).