# Proyecto 4: Índice de Poder de Banzhaf

Ricardo Castro Jiménez
Carmen Hidalgo Paz
Jorge Guevara Chavarría

*Escuela de Ingeniería en Computación, Tecnológico de Costa Rica, San José, Costa Rica*

riccastro@estudiantec.cr
carmenhidalgopaz@estudiantec.cr
joguevara@estudiantec.cr

En este proyecto trabajamos con el problema de la Suma de Subconjuntos usando backtracking para encontrar coaliciones ganadoras (tienen igual o mayor que la cantidad de votos pedida) en sistemas de votación ponderada. La idea principal fue calcular el Índice de Poder de Banzhaf (IPB) para cada votante, detectando cuáles son los votos críticos de cada coalición y mostrar todo esto en una interfaz gráfica hecha en C con GTK y Glade.

Básicamente, el programa recibe un conjunto de votantes (enteros positivos ordenados entre 3 y 12) cada uno con una cantidad de votos y un mínimo de votos necesarios para ganar. El algoritmo busca todos los subconjuntos que cumplen ciertas condiciones y calcula qué tan influyente es cada votante.

**Variantes del problema**

Nos centramos en la variante "Mayor o Igual" del problema de suma de subconjuntos:

**Variante Mayor o Igual**

Se buscan todos los subconjuntos cuya suma sea mayor o igual que la cantidad necesaria para ganar. La búsqueda no se detiene cuando se encuentra una solución, sino que explora todas para conseguir todas las coaliciones ganadoras. Esto nos permite ver todas las coaliciones mínimas y calcular con precisión quién tiene votos críticos para el IPB.

## Implementación

### Backtracking y votos críticos

El corazón del proyecto es la función recursiva sumaSubconjuntosV3_collect. Esta función hace backtracking para recorrer todas las combinaciones posibles que sumen al menos la cantidad pedida por el usuario. Además, durante esta exploración, cuando encuentra una coalición ganadora, calcula directamente qué votantes son críticos: es crítico aquel votante que si se quita, la coalición deja de ser ganadora.

Esto lo hicimos para no tener que hacer un recorrido extra después, sino que mientras se generan las soluciones ya contamos los votos críticos, lo que optimiza bastante el proceso y simplifica el flujo.

### Interfaz gráfica y visualización

Creamos una interfaz con GTK y Glade para que el usuario pueda:

- Elegir entre 3 y 12 votantes, poner sus votos (enteros positivos, ordenados) y cambiar sus nombres y colores.
- Definir la cantidad de votos mínimos para ganar.
- Ver la lista de coaliciones ganadoras con checkboxes que indican quién está en cada coalición y la suma total del conjunto de votos en esa coalición.
- Ver un gráfico tipo parlamento que muestra la distribución de votos, con colores personalizados.
- Ver estrellas doradas que indican qué votantes son críticos en una coalición ganadora.
- Ver el IPB de cada votante en decimal y en fracción, y el total de votos críticos en un label visible.

Así, no solo se muestra la información numérica sino que se entiende visualmente la importancia de cada votante.

**Resultados y validaciones**

Probamos con distintos números de votantes y valores de votos mínimos para ganar, verificando que:

- El backtracking encuentra todas las coaliciones ganadoras sin saltarse ninguna.
- Los votos críticos calculados son correctos y coinciden con el IPB teórico.
- La interfaz responde rápido y actualiza todo cuando cambian los datos.
- Validamos que no haya errores por entradas inválidas, números negativos, o datos mal ordenados.

Con eso garantizamos que el programa es confiable y útil para análisis reales.

**Conclusión**

Este proyecto nos ayudó a entender bien cómo aplicar backtracking en problemas combinatorios y cómo integrar cálculos complejos, como el IPB, dentro del mismo proceso para ganar eficiencia.

Además, desarrollar la interfaz con GTK y Glade fue muy útil para aprender a hacer aplicaciones gráficas que muestren datos complicados de forma sencilla y clara. En resumen, fue un buen ejercicio de combinar lógica algorítmica con diseño de interfaces interactivas, todo en C.

**Fuentes consultadas**


[1] GeeksforGeeks, “Subset Sum Problem”, [En línea]. Disponible en: https://www.geeksforgeeks.org/subset-sum-problem-dp-25/ [Accesado: 10 de mayo del 2025].

[2] GeeksforGeeks, “Backtracking with Constraints”, [En línea]. Disponible en: https://www.geeksforgeeks.org/backtracking-algorithms/ [Accesado: 10 de mayo del 2025].

[3] Enunciado del Proyecto 3, Análisis de Algoritmos, Tecnológico de Costa Rica, 2025.

[4] K. O’Kane, “Linux Gtk Glade Programming Part 1”, YouTube, [video en línea]. Disponible en: https://www.youtube.com/watch?v=g-KDOH_uqPk [Accesado: 10 de mayo del 2025].

[5] K. O’Kane, “Linux Gtk Glade Programming Part 25 Dynamic Scrollable Grid of Buttons”, YouTube, [video en línea]. Disponible en: https://www.youtube.com/watch?v=9SHHjzc9wqA [Accesado: 11 de mayo del 2025].