Mini Proyecto Final – Bingo_P
Documento de Pruebas

Burgos Panta Juliana Lisbeth
Díaz González Darwin Leonardo
Morales Vásquez Víctor Manuel
Saltos Preciado Andrés Alexander
Tumbaco Santana Gabriel Alejandro

1. Introducción

El presente documento describe el conjunto de pruebas realizadas sobre el sistema **Bingo_P**, con el objetivo de verificar el correcto funcionamiento de los algoritmos implementados, validar el cumplimiento de los requisitos establecidos, demostrar la aplicación de las estrategias algorítmicas (Dividir y Conquistar, Programación Dinámica) y garantizar la robustez de la solución implementada.

Las pruebas incluyen:

- **Pruebas algorítmicas** (verificación de Merge Sort, Búsqueda Binaria y Distancia de Edición).
- **Pruebas funcionales** (ingreso de datos, carga masiva, detección de ganadores).
- **Pruebas de validación** (manejo de errores, límites por idioma).
- **Pruebas de estrés** (rendimiento con grandes volúmenes de datos).

2. Metodología de Pruebas

Cada caso de prueba incluye seis elementos fundamentales:
(1) **Descripción:** Objetivo de la prueba
(2) **Entrada:** Datos utilizados
(3) **Proceso:** Ejecución paso a paso
(4) **Resultado Esperado:** Salida anticipada
(5) **Resultado Obtenido:** Salida real del sistema
(6) **Conclusión:** Validación del resultado obtenido.

3. Pruebas de Algoritmos Fundamentales

3.1. Caso 1: Merge Sort - Ordenamiento del Repositorio

Merge Sort - Español (Proceso Detallado)

Descripción: Verificar el ordenamiento correcto del repositorio de palabras en español utilizando el algoritmo Merge Sort implementado según CLRS pág. 31-34, demostrando la estrategia de Dividir y Conquistar.
Entrada: Lista desordenada de palabras: `perro, casa, gato, arbol, agua, mesa`
Proceso (Dividir y Conquistar):
Fase 1 - Dividir:

```
Nivel 0: [perro, casa, gato, arbol, agua, mesa]
         |
Nivel 1: [perro, casa, gato] | [arbol, agua, mesa]
         |                     |
Nivel 2: [perro] [casa,gato] | [arbol] [agua,mesa]
         |        |           |         |
Nivel 3: [perro] [casa][gato]| [arbol] [agua][mesa]
```

Fase 2 - Conquistar (Mezclar):

```
Nivel 3: [perro] [casa][gato] | [arbol] [agua][mesa]
         |        |            |         |
Nivel 2: [perro] [casa,gato]  | [arbol] [agua,mesa]
         |                     |
Nivel 1: [casa, gato, perro]  | [agua, arbol, mesa]
         |
Nivel 0: [agua, arbol, casa, gato, mesa, perro]
```

Resultado Esperado: Lista ordenada alfabéticamente en complejidad O(n log n): `[agua, arbol, casa, gato, mesa, perro]`
Resultado Obtenido: `[agua, arbol, casa, gato, mesa, perro]`
Conclusión: Merge Sort ordena correctamente el repositorio en español demostrando la estrategia de Dividir y Conquistar: divide el problema en subproblemas más pequeños (división recursiva), resuelve cada subproblema (ordenamiento de subarreglos), y combina las soluciones (mezcla ordenada).

Merge Sort - Otros Idiomas

Resultados obtenidos aplicando el mismo proceso recursivo:

- **Inglés:** `house, dog, cat, table, chair` → `[cat, chair, dog, house, table]`
- **Portugués:** `casa, cachorro, gato, mesa, cadeira` → `[cachorro, cadeira, casa, gato, mesa]`
- **Dutch:** `huis, hond, kat, tafel, stoel` → `[hond, huis, kat, stoel, tafel]`

Conclusión: Merge Sort funciona correctamente para los 4 idiomas según CLRS pág. 31-34.

3.2. Caso 2: Búsqueda Binaria - Validación Multiidioma

Búsqueda Binaria - 4 Idiomas

Descripción: Verificar que la búsqueda binaria funciona eficientemente en todos los idiomas para validar la existencia de palabras en los repositorios ordenados.
Entrada: Palabras a buscar en repositorios ordenados de diferentes tamaños (40-100 palabras por idioma).
Proceso: Aplicación del algoritmo de búsqueda binaria (CLRS Ejercicio 2.3-5) que divide repetidamente el espacio de búsqueda a la mitad hasta encontrar el elemento o determinar que no existe.

Idioma	Palabra	Iteraciones	Resultado
Español (100 pal.)	"casa"	4	Índice 17
Inglés (60 pal.)	"house"	3	Índice 23
Portugués (80 pal.)	"casa"	4	Índice 11
Dutch (40 pal.)	"huis"	3	Índice 18

Resultado Esperado: Búsqueda exitosa en complejidad O(log n) para cada idioma.
Resultado Obtenido: Todas las palabras fueron encontradas con el número esperado de iteraciones (máximo 4 iteraciones para 100 elementos).
Conclusión: La búsqueda binaria optimiza la validación en todos los repositorios con complejidad O(log n).

3.3. Caso 3: Distancia de Edición - Sugerencias Multiidioma

Distancia de Edición - Español

Descripción: Verificar que el algoritmo de Programación Dinámica genera sugerencias correctas para palabras con errores tipográficos, implementando el algoritmo de distancia de edición según CLRS Problema 15-5.
Entrada: Palabra con error tipográfico: "agau" (debería ser "agua")
Proceso: Construcción de tabla de Programación Dinámica que calcula el mínimo número de operaciones (insertar, eliminar, reemplazar) necesarias para transformar la palabra errónea en cada candidata del repositorio.
Tabla DP para "agau" vs "agua":

	ϵ	a	g	u	a
ϵ	0	1	2	3	4
a	1	0	1	2	3
g	2	1	0	1	2
a	3	2	1	1	2
u	4	3	2	1	$\boxed{2}$

La tabla muestra cómo se calcula la distancia mínima. La celda [4,4] contiene el resultado final: distancia = 2. Las operaciones necesarias son: (1) eliminar 'u' de "agau", (2) insertar 'u' en la posición correcta.
Resultado Esperado: Sugerencia de la palabra "agua" con distancia de edición = 2
Resultado Obtenido: Sistema genera el mensaje "¿Quisiste decir 'agua'?" con distancia calculada de 2
Conclusión: El algoritmo de Programación Dinámica identifica correctamente la palabra más cercana en el repositorio español usando memoización para evitar recálculos.

Distancia de Edición - Otros Idiomas

Descripción: Verificar el funcionamiento del algoritmo en inglés, portugués y holandés.
Entrada: Palabras con diversos errores tipográficos en los tres idiomas.
Proceso: Cálculo de distancia de edición usando Programación Dinámica para cada par (palabra errónea, candidata del repositorio). El algoritmo construye una tabla similar para cada comparación.

Idioma	Error	Sugerencia	Distancia
Inglés	"hose"	"house"	1
Inglés	"dg"	"dog"	1
Portugués	"csa"	"casa"	1
Portugués	"cachoro"	"cachorro"	1
Dutch	"hus"	"huis"	1
Dutch	"hond"	"hond"	0 (correcta)

Resultado Esperado: Sugerencias precisas con distancias mínimas calculadas correctamente.
Resultado Obtenido: Todas las sugerencias coinciden con las palabras correctas esperadas.
Conclusión: El algoritmo de distancia de edición (CLRS Problema 15-5) funciona correctamente en los 4 idiomas, demostrando la efectividad de la Programación Dinámica para resolver el problema de corrección ortográfica.

4. Pruebas de Ingreso y Validación

4.1. Caso 4: Ingreso Manual Multiidioma

Validación por Idioma

Descripción: Verificar que el sistema acepta y valida correctamente cartones ingresados manualmente en los cuatro idiomas soportados.
Entrada: Cartones con identificación, jugador y palabras para cada idioma.
Proceso: Para cada cartón se realiza: (1) Validación del formato del ID (2 letras de idioma + 6 dígitos), (2) Verificación de que las palabras existen en el repositorio usando búsqueda binaria, (3) Validación de que no exceda el máximo de palabras permitido para el idioma, (4) Agregación al sistema si todas las validaciones son exitosas.
Resultados por idioma:

- **Español (SP123456, J001):** Palabras: `casa perro gato sol mesa` → Cartón agregado (5/24 palabras)
- **Inglés (EN123456, J002):** Palabras: `house dog cat table chair` → Cartón agregado (5/14 palabras)
- **Portugués (PT123456, J003):** Palabras: `casa cachorro gato mesa cadeira` → Cartón agregado (5/20 palabras)
- **Dutch (DT123456, J004):** Palabras: `huis hond kat tafel stoel` → Cartón agregado (5/10 palabras)

Resultado Esperado: Todos los cartones deben ser aceptados y agregados al sistema.
Resultado Obtenido: Los 4 cartones fueron agregados correctamente, validando las palabras mediante búsqueda binaria en sus respectivos repositorios ordenados.
Conclusión: El sistema valida correctamente cartones en los 4 idiomas usando búsqueda binaria.

4.2. Caso 5: Detección de Errores con Sugerencias

Errores y Sugerencias

Descripción: Verificar detección de errores y sugerencias con distancia de edición. **Entrada:** Cartones con palabras erróneas. **Proceso:** (1) Validar con búsqueda binaria, (2) Calcular distancia de edición, (3) Sugerir palabra con menor distancia.
Resultados: SP: 'agau'→'agua', 'arie'→'aire', 'amro'→'amor'; EN: 'hose'→'house', 'dg'→'dog', 'tble'→'table'; PT: 'csa'→'casa', 'cachoro'→'cachorro', 'gto'→'gato'; DT: 'hus'→'huis', 'kt'→'kat'.
Conclusión: Sistema genera sugerencias inteligentes para todos los idiomas.

5. Pruebas Funcionales del Juego

5.1. Caso 6: Detección de Ganador - Todos los Idiomas

Partida con 4 Idiomas

Descripción: Verificar que el sistema detecta correctamente a los ganadores en cada ronda utilizando el índice invertido para búsqueda eficiente.
Entrada: Cuatro cartones de prueba con 3 palabras cada uno:

- SP000001 (J001): casa, perro, gato
- EN000001 (J002): house, dog, cat
- PT000001 (J003): casa, cachorro, gato
- DT000001 (J004): huis, hond, kat

Proceso: (1) Iniciar partida generando orden aleatorio de idiomas, (2) Para cada ronda, extraer palabras del repositorio correspondiente, (3) Usar índice invertido para encontrar cartones afectados en O(1), (4) Marcar palabras y verificar si algún cartón completa todas sus palabras.
Orden aleatorio generado: Dutch → Inglés → Español → Portugués
Resultado Esperado: El sistema debe detectar ganadores cuando sus cartones se completen.
Resultado Obtenido:

- **Ronda 1 (Dutch):** Palabra "huis": DT000001 (1/3), Palabra "hond": DT000001 (2/3), Palabra "kat": DT000001 (3/3) → GANADOR DETECTADO
- **Ronda 2 (Inglés):** Palabra "house": EN000001 (1/3), Palabra "dog": EN000001 (2/3), Palabra "cat": EN000001 (3/3) → GANADOR DETECTADO

Conclusión: El índice invertido detecta ganadores eficientemente en todos los idiomas.

5.2. Caso 7: Particionamiento por Idioma

Distribución de Cartones

Descripción: Verificar que D&C optimiza procesamiento evaluando solo cartones relevantes. **Entrada:** 160 cartones (40 por idioma). **Proceso:** Particionar cartones por idioma, acceder solo a partición correspondiente.

Idioma	Cartones	Max Pal.	Lím. Extrac.
Español	40	24	80
Inglés	40	14	48
Portugués	40	20	64
Dutch	40	10	32

Resultado Esperado: Procesar solo cartones del idioma actual.
Resultado Obtenido: Cada ronda evalúa 40 cartones en lugar de 160.
Conclusión: D&C reduce complejidad procesando solo datos relevantes.

6. Pruebas de Límites y Validación

6.1. Caso 8: Validación de Límites por Idioma

Límites Máximos

Descripción: Verificar que el sistema rechaza cartones que exceden el máximo de palabras permitido para cada idioma.
Entrada: Cartones en el límite exacto y cartones que exceden el límite para cada uno de los 4 idiomas.
Proceso: Intentar agregar cartones con diferentes cantidades de palabras y verificar que el sistema realice la validación correctamente según los límites establecidos.

Cartón	Palabras	Límite	¿Válido?	Resultado
SP000001	24	24	Sí	Aceptado
SP000002	25	24	No	Rechazado
EN000001	14	14	Sí	Aceptado
EN000002	15	14	No	Rechazado
PT000001	20	20	Sí	Aceptado
PT000002	21	20	No	Rechazado
DT000001	10	10	Sí	Aceptado
DT000002	11	10	No	Rechazado

Resultado Esperado: Aceptar cartones que están en el límite, rechazar los que lo exceden.
Resultado Obtenido: Todos los cartones en el límite fueron aceptados correctamente, y todos los que exceden el límite fueron rechazados con mensaje de error apropiado.
Conclusión: El sistema respeta y valida correctamente los límites específicos de cada idioma.

6.2. Caso 9: Límite de Extracciones

Cálculo de Límites

Descripción: Verificar cálculo correcto de límite de extracciones.
Entrada: Repositorios con tamaños diferentes.
Proceso: Aplicar fórmula: límite = máx(max_pal $\times$ 3, total_repo $\times$ 0,8).
Cálculos: SP: máx$(72, 80) = 80$; EN: máx$(42, 48) = 48$; PT: máx$(60, 64) = 64$; DT: máx$(30, 32) = 32$.
Resultado Esperado: Límites correctos, rondas finalizan al alcanzarlos.
Resultado Obtenido: Coincidencia con valores esperados.
Conclusión: Sistema finaliza rondas garantizando equidad.

7. Pruebas de Estrés

7.1. Caso 10: Gran Volumen - 160 Cartones

Distribución de Carga

Descripción: Evaluar rendimiento con carga significativa.
Entrada: 160 cartones (40 por idioma), $\sim$3,200 palabras totales.
Proceso: (1) Cargar 160 cartones, (2) Ordenar repositorios con Merge Sort, (3) Validar con búsqueda binaria, (4) Construir índices invertidos, (5) Ejecutar partida completa.
Rendimiento: SP: 0.8s, EN: 0.6s, PT: 0.7s, DT: 0.5s. Total: 3.2 segundos.
Resultado Esperado: Manejo sin degradación.
Resultado Obtenido: Carga en 3.2s, juego fluido, detección instantánea.
Conclusión: Solución escalable y eficiente para todos los idiomas.