Informe Final — Análisis y Diseño de Algoritmos

Camino mínimo en una matriz (Comparación de paradigmas)

Curso: Análisis y Diseño de Algoritmos – Universidad EAFIT
Estudiante: Samuel Moncada Mejía
Cédula: 1022003803
Fecha: Noviembre de 2025

1. Descripción del problema

El problema del camino mínimo en una matriz consiste en encontrar la ruta de menor costo desde la esquina superior izquierda de una matriz n×n hasta la esquina inferior derecha. Cada celda tiene un valor que representa su costo de entrada, y el objetivo es determinar el camino cuya suma de costos sea la mínima posible, moviéndose solo hacia abajo o hacia la derecha.

2. Paradigmas implementados

Paradigma	Descripción	Complejidad temporal
Backtracking (DFS con poda)	Explora recursivamente todas las rutas posibles y poda aquellas cuyo costo parcial supera el mejor costo encontrado.	Exponencial O(2^(2n))
Greedy (Dijkstra)	Usa una cola de prioridad para expandir siempre la celda con menor costo acumulado, garantizando el resultado óptimo.	$O(n^2 \log n)$
Programación Dinámica (Bottom-Up)	Construye una tabla dp[i][j] con el costo mínimo para llegar a cada celda, reutilizando subsoluciones previas.	$O(n^2)$

3. Pseudocódigo de cada enfoque

- Backtracking con poda

```
best = +∞
dfs(i, j, costo):
  si costo ≥ best → retornar
  si (i,j) es destino → best = min(best, costo + a[i][j])
  si puede bajar → dfs(i+1, j, costo + a[i][j])
  si puede derecha → dfs(i, j+1, costo + a[i][j])
```

- Greedy (Dijkstra)

```
dist[0][0] = a[0][0]
Cola prioridad Q = {(a[0][0], (0,0))}
mientras Q no vacía:
  (d, (i,j)) = extraer mínimo
  si (i,j) = destino → retornar d
  para cada vecino válido (abajo, derecha):
    nd = d + a[vecino]
    si nd < dist[vecino]:
      dist[vecino] = nd
      insertar (nd, vecino) en Q
```

- Programación Dinámica (Bottom-Up)

```
dp[0][0] = a[0][0]
para i en [0..n-1]:
  para j en [0..n-1]:
    si (i,j) ≠ (0,0):
      dp[i][j] = a[i][j] + min(dp[i-1][j], dp[i][j-1])
retornar dp[n-1][n-1]
```

4. Análisis teórico

Recurrencia de la PD: $T(i, j) = C[i][j] + \min(T(i-1, j), T(i, j-1))$

Paradigma	Tiempo	Espacio	Características
Backtracking	Exponencial	O(n)	Busca todas las combinaciones posibles; solo viable para n pequeños.
Dijkstra (Greedy)	$O(n^2 \log n)$	$O(n^2)$	Usa estructura de datos eficiente; encuentra el óptimo en todos los casos.
Programación Dinámica	$O(n^2)$	$O(n^2)$ o O(n)	Reutiliza subsoluciones; simple y eficiente para este tipo de problema.

5. Resultados experimentales

Archivo: results/resultados.csv

Caso	n	Algoritmo	Costo	Tiempo (ms)
0	3	Backtracking	7	0
		Dijkstra	7	0
		Programación Dinámica	7	0
1	4	Backtracking	10	0
		Dijkstra	10	0
		Programación Dinámica	10	0
2	6	Backtracking	11	0
		Dijkstra	11	0
		Programación Dinámica	11	0

Los tres algoritmos encuentran el mismo costo mínimo, eso lo que demuestra es que esta bien hecho. El tiempo de ejecución aparece como 0 ms por el tamaño reducido de las matrices, con valores de n mayores, el Backtracking crece exponencialmente mientras Dijkstra y la Programación Dinámica se mantienen eficientes.

6. Discusión comparativa

Caracteristica	Backtracking	Dijkstra	Programación Dinámica
Rendimiento	Muy lento en n grandes	Rápido y escalable	Muy rápido
Exactitud	Óptimo (con poda)	Óptimo	Óptimo
Facilidad de implementación	Media	Alta (cola de prioridad)	Muy alta
Reutilización de subproblemas	No	Parcial	Total
Aplicabilidad	n pequeños o educativos	Grafos con pesos positivos	Cuadrículas o estructuras regulares

7. Conclusiones personales

Backtracking garantiza la exactitud, pero es lento para n grandes. Dijkstra representa el paradigma de Greedy, y logra el mismo resultado óptimo con mucho mejor rendimiento, y por último la Programación Dinámica es la más eficiente y directa, porque aprovecha las subestructuras óptimas y evita recomputaciones.

8. Declaración ética de uso de IA

Utilice ChatGPT como herramienta de apoyo para darle una mejor estructura y también para redactar este trabajo. Además, también la utilicé en dudas o problemas que tuve al momento de realizar los códigos y para realizar los pseudocódigos, pero todo fue comprendido.