Proyecto 1

Análisis y diseño de Algoritmos II

Juan Camilo Diaz Valencia
Carlos Stiven Ruiz Rojas
Juan David Rojas Narvaez
Jhony Fernando Duque Villada

Ingeniería de Sistemas

Diciembre 2024

Universidad del Valle

Sede Tuluá
2024

1. Problema de la terminal inteligente

1.1 Contextualización

El problema de la terminal inteligente transforma una cadena A[1...n] a una cadena B[1...n] por medio de cinco operaciones buscando la solución con menor costo.

Las operaciones son las siguientes:

- advance
- replace
- insert
- delete
- kill

El costo está dado por la ecuación:

$$Costo = a + r + i + d + k \tag{1}$$

Se plantean tres soluciones al problema con el objetivo de compararlas, analizar su comportamiento y concluir sobre cada una.

Las soluciones propuestas son las siguientes:

- Programación a fuerza bruta.
- Programación dinámica.
- Programación voraz.

1.2 Programación a fuerza bruta

1.2.1 ¿En que consiste?

La programación a fuerza bruta explora todas las posibles transformaciones entre a[1...n] y b[1...n] mediante las operaciones, esta misma lo hace mediante la función recursiva *rec_transform*, la cual se encarga de evaluar las diferentes opciones y calcular el costo mínimo de transformación.

1.2.2 Formalización

$$C(x_i, y_i) = \begin{cases} 0 & \text{si } x_i = \text{len}(x) \text{ y } y_i = \text{len}(y) \\ (\text{len}(y) - y_i) \cdot i_k & \text{si } x_i = \text{len}(x) \text{ y } y_i < \text{len}(y) \\ k & \text{si } y_i = \text{len}(y) \text{y} x_i < \text{len}(x) \end{cases}$$

¿Cómo se implementa?

Se define la función recursiva *rec_transform* que es la que lleva a cabo la comparación de las cadenas y determina el costo mínimo de las transformaciones. Esta función recibe los índices x_i, y_i para las posiciones actuales de las cadenas x y y.

Casos base

1. Si ambos índices (x_i, y_i) alcanzan el final de las cadenas ($\text{x}_i == len(x)$) y ($\text{y}_i == len(y)$), significa que ya hemos transformado completamente las dos cadenas, por lo que no se requieren más operaciones. Su costo es 0.
2. Si hemos llegado al final de x pero quedan caracteres por insertar en y se calcula el costo de insertar los caracteres restantes de y (El número de caracteres multiplicado por el costo de inserción i)
3. Si hemos llegado al final de y pero quedan caracteres por eliminar de x, se realiza la operación de *kill*, eliminando todos los caracteres restantes de x con el costo correspondiente.

Operaciones recursivas

Después de los casos base, la función evalúa los operadores y calcula el costo mínimo para cada uno.

- Advance → Si x y y son iguales y llama recursivamente aumentando los índices.
- Replace → Si x y y son diferentes, llama recursivamente aumentando los índices.
- Delete → Avanza solo en la cadena x (sin avanzar en y) y se suma el costo de la operación.
- Insert → Avanza solo en la cadena y (sin avanzar en x) y se suma el costo de la operación.

Llamada inicial

Finalmente, la función *brute_force_transform* llama a *rec_transform* con los índices iniciales $\text{x}_i = 0$ y $\text{y}_i = 0$, y retorna el costo mínimo y la secuencia de operaciones para la transformación.

Complejidad temporal

La función realiza llamados recursivos sobre todas las combinaciones de índices, evaluando cuatro posibles operaciones. Puesto que la operación *kill* se realiza en el peor de los casos, $n - x_i$ veces, donde x_i es el índice actual de la cadena.

Aunque la operación se realiza cuando se han agotado los caracteres de y, cada vez que esta operación se activa, se hace una llamada recursiva para cada uno de los caracteres restantes en x. El coste de liminar los caracteres restantes de x se calcula como un costo k de esta operación multiplicado por la cantidad de caracteres que deben eliminarse.

Por lo tanto, la operación *kill* contribuye a la recursión, pero no cambia la complejidad exponiencial general.

La complejidad temporal de la función depende del número total de combinaciones de las posiciones de las dos cadenas. En el peor de los casos, esto es un arbol de recursión de profundidad *min(len(x), len(y))*, y en cada nivel del árbol se evalúan 4 operaciones.

Por lo tanto, el número de llamados en el peor de los casos es exponencial con respecto al tamaño de las cadenas, el costo temporal es:

$$O\left(4^{\max(\text{len}(x),\text{len}(y))}\right) = O(4^n) \tag{2}$$

1.3 Programación Dinámica

1.3.1 ¿En qué consiste?

La función implementa una transformación óptima entre las dos cadenas, por medio de una estrategia dinámica con memorización.

Estructura y funcionamiento

Utiliza una matriz de $((n + 1) * (m + 1) * 2)$ que almacena los costos mínimos para transformar una subsecuencia. Los índices son las posiciónes en las cadenas (x:source, y:target).

cursor indica si el cursor está en estado normal o en un estado especial para operaciones como *kill*.

el valor $dp[x][y][cursor]$ representa el costo mínimo para transformar *source[x:]* en *target[y:]*.

Casos base

- $dp[n][m][0] = dp[n][m][1] = 0$ Si ya hemos recorrido ambas cadenas, el costo es cero.
- *operation_trace[n][m] = []*: No se requiere ninguna operación.

Llamado recursivo

$min\_cost(x, y, cursor)$ calcula el costo mínimo desde la posición actual. Considera todas las operaciones posibles.

- Si $x < n$, se pueden aplicar los operadores *advance,delete,replace o kill.*
- Si $y < m$, se puede aplicar insert.
- Memorización: Cada operación se evalúa recursivamente, acumulando su costo y seleccionando el menor. Si un costo ya fue calculado, se reutiliza.
- Operaciones óptimas: Se rastrean las operaciones óptimas con *operation_trace.*
- Una vez calculado el costo mínimo desde la posición inicial $x = 0, y = 0, cursor = 0$ se obtiene la secuencia de operaciones desde *operation_trace.*

Costo temporal

1. Tamaño de la matriz: $(n+1) * (m+1) * 2$ donde $n = len(source)$ y m=len(target).
2. Cada posición se evalúa una vez gracias a la memorización, en cada operación se realizan $O(5)$ operaciones como máximo (correspondientes a los operadores)
3. La complejidad total es de $O(n * m)$

Formalización

1. Estado del DP: Definición del estado:

   $dp[x][y]$[cursor] = Costo mínimo para transformar $source[x:]$ en $target[y:]$ dado cursor.

2. Transición del estado:

$$dp[x][y][\text{cursor}] = \min \begin{cases} dp[x+1][y+1][0] + a & (\text{si } source[x] = target[y] \text{ y 'cursor' } = 0) \\ dp[x+1][y][0] + d & (\text{delete}) \\ dp[x+1][y+1][0] + r & (\text{replace}) \\ k + (m-y) \cdot i & (\text{kill}) \\ dp[x][y+1][\text{cursor}] + i & (\text{insert}) \end{cases}$$

Caso base

$dp[n][m][0] = dp[n][m][1] = 0.$

Solución

La solución óptima es:

$total\_cost = dp[0][0][0]$, $sequence = operation\_trace[0][0]$

Esta se enceunra en el estado inicial de la matriz de programación dinámica, esto debido a que esta representa el costo mínimo para transformar toda lacadena $source[0:]$ en $target[0:]$.

La matriz dp se construye de manera que se llenan los estados finales primero. $dp[n][m]$ y luego se retrocede hacia el estado inicial $dp[0][0]$ mientras se evalúan todas las posibles transiciones. $operation\_trace[0][0]$ contiene la secuencia completa de operaciones para la transformación con el menor costo posible.

Complejidad

Dado que:

- Hay $O(n * m)$ estados.
- Cada estado toma $O(1)$ tiempo para evaluar todas las operaciones gracias a la memorización.

La complejidad temporal es: $O(n * m)$

Problema 2. El problema de las subastas

2.1. Contextualización

El mecanismo utilizado por el gobierno para realizar las subastas de acciones es el siguiente:

El gobierno pone en subasta un total de A acciones a un precio mínimo de B

Cada oferente presenta una oferta en forma de tripleta pi, mi, Mi donde pi es el precio que está dispuesto a pagar por cada acción, mi es el número mínimo de acciones que desea comprar, y Mi es el número máximo de acciones que podría comprar.

El gobierno se encargará de comprar las acciones sobrantes a su precio mínimo B

De entre los diferentes oferentes, el gobierno debe decidir cuántas acciones asignar a cada uno, lo cual se representa por una lista $X =< x1, x2, ...xk >$, donde xi es el npumero de acciones asignadas al oferente i. Se debe cumplir que $mi \leq xi \leq Mi$ y que la suma total de las acciones asignadas debe ser igual a A, es decir, $\sum xi = A$. Esto garantiza que todas las acciones se vendan.

El valor recibido por una asignación de acciones X se define como el valor

$$vr(X) = \sum_{i=1}^{k} x_i p_i,$$

donde x_i es el número de acciones asignadas al oferente i y p_i es el precio por acción ofrecido por dicho oferente.

El objetivo es encontrar la asignación de las A acciones que maximice el valor $vr(X)$, respetando las restricciones de cada oferente y el precio mínimo del gobierno.

2.2. Ejemplo

Supongamos un caso en donde se tiene lo siguiente:

$A = 1000$,$B = 100$,$n = 2$

Donde A es el número total de acciones disponibles, B es el precio mínimo al cual el gobierno compra las acciones sobrantes, y n es el número de oferentes.

Se tienen las siguientes ofertas:

- $A \rightarrow (500, 100, 600)$
- $B \rightarrow (450, 400, 800)$
- $Gobierno \rightarrow (100, 0, 1000)$

debemos intentar asignar la mayor cantidad de acciones posible a los oferentes que pagan más, en este caso el primer oferente, que ofrece un precio de 500 por acción.

Asignaciones:

Asignación 1:

- Asignamos 600 acciones al primer oferente (su límite máximo)..
- Asignamos 400 acciones al segundo oferente.
- Las 0 acciones restantes serán compradas por el gobierno a un precio de 100 por acción.

Por lo tanto, la asignación sería:

- $x_1 = 600$
- $x_2 = 400$
- $x_3 = 0$

Cálculo de $vr(X)$:

$$vr(X) = 600 * 500 + 400 * 450 + 0 * 100 = 300000 + 180000 \tag{3}$$

Resultado:

- Asignación: (600,400,0)
- Valor de vr: 480,000

Asignación 2:

- Asignamos 500 acciones al primer oferente.
- Asignamos 500 acciones al segundo oferente.
- El gobierno comprará las 0 acciones sobrantes a 100 por acción.

Cálculo de $vr(X)$:

$$vr = 500 * 500 + 500 * 450 + 0 * 100 = 250000 + 225000 \tag{4}$$

Resultado:

- Asignación: (500,500,0)
- Valor de $vr(X)$: 475,000

2.3. Primeras aproximaciones

Dado el siguiente algoritmo:

$$\langle y_1, y_2, \ldots, y_{n-1} \rangle \quad \sum_{i=1}^{n-1} y_i = A \tag{5}$$

Se debe aplicar a la entrada:

$A = 1000, B = 100, n = 4, < 500, 400, 600 >, < 450, 100, 400 >,$
$< 200, 50, 200 >, < 400, 100, 400 >,$ la oferta del gobierno $< 100, 0, 1000 >$

Implementación:

Se exploran todas las combinaciones posibles de asignaciones y_1, y_2, y_3, y_4 donde $y_i \in \{0, Mi\}$. Y la suma total $y_1, y_2, y_3, y_4 = A = 1000$.

Se obtienen todas las combinaciones posibles:

- $y_1 = 0, 600$
- $y_2 = 0, 400$
- $y_3 = 0, 400$
- $y_4 = 0, 200$

Se filtran aquellas que cumplan la restricción $A = 1000$

Combinaciones:

- $(600, 400, 0, 0) : 600 + 400 + 0 + 0 = 1000$
- $(600, 0, 400, 0) : 600 + 0 + 400 + 0 = 1000$
- $(600, 0, 0, 200) : 600 + 0 + 0 + 200 = 1000$
- $(0, 400, 400, 200) : 0 + 400 + 400 + 200 = 1000$

Cálculo de $vr(X)$:
Dado que el algoritmo selecciona el mejor valor de vr, el algoritmo siempre encuentra la solución óptima.

Asignación	**Cálculo de** $vr(X)$	**Valor** $vr(X)$
(600, 400, 0, 0)	$300000 + 180000 + 0 + 0$	480000
(600, 0, 400, 0)	$300000 + 0 + 160000 + 0$	460000
(600, 0, 0, 200)	$300000 + 0 + 0 + 40000$	340000
(0, 400, 400, 200)	$0 + 180000 + 160000 + 40000$	380000

Table 1: Tabla con los valores de $vr(X)$.

2.4. Estructura de la solución

- $dp[i][j]$: Valor máximo de vr posible con los primeros i oferentes asignando exactamente j acciones.
- Cada subproblema depende de cuantas acciones se asignan al i oferente. Esto teniendo en cuenta el mínimo m_i y el max M_i de acciones que puede tomar dicho oferente.

Decisiones

Para cada oferente i y cada cantidad j de acciones restantes:

- No asignar acciones al oferente i
- Asignar x acciones al oferente i, donde $m_i \leq x \leq min(M_i, j)$

Relación de recurrencia

La relación de recurrencia para llenar la tabla de programación dinámica es:

$$dp[i][j] = \max\left(dp[i-1][j], \max_{x \in [m_i, \min(M_i, j)]} (dp[i-1][j-x] + x \cdot p_i)\right) \tag{6}$$

- El valor óptimo es $dp[n][A]$, que es el valor máximo de vr consideraendo todos los n oferentes y asignando exactamente A acciones.

Recontrucción de la solución óptima

- Partimos desde $dp[n][A]$.
- Rastreando hacia atrás, verificamos si la asignación x acciones al oferente i maximiza el valor. Si sucede, añadimos x a la solución y restamos x a las acciones restantes$j$.
- Repetimos el proceso para todos los oferentes, reconstruyendo la lista de asignación. $[y_1, y_2, ..., y_n]$.

Relación de recurrencia
La relación de recurrencia que determina el costo es:

$$T(n, A) = T(n-1, A) + \sum_{j=0}^{A} (\min(M_i, j) - m_i + 1) \tag{7}$$

Donde:

- $T(n-1, A)$: Costo de resolver el problema para $n-1$ oferentes.
- $\sum_{j=0}^{A} (\min(M_i, j) - m_i + 1)$: Costo de explorar todas las combinaciones de acciones x para el i oferente, considerando las restricciones min y max.

Costo total
El costo total para llenar la tabla dp y resolver el problema es:

$$O(n * a * M) \tag{8}$$

Donde M es el máximo de Mi sobre todos los oferentes.

Ejemplo

Entrada:

- Número total de acciones: $A = 10$
- Precio mínimo: $B = 50$
- Oferentes:
  1. $< 100, 2, 5 >$ (precio: 100, mínimo: 2, máximo: 5)
  2. $< 80, 1, 4 >$ (precio: 80, mínimo: 1, máximo: 4)
  3. $< 70, 0, 6 >$ (precio: 70, mínimo: 0, máximo: 6)

Estructura de la Tabla

- $dp[i][j]$ almacena el valor máximo para asignar exactamente j acciones usando los primeros i oferentes.
- Las asignaciones válidas para cada oferente deben respetar $m_i \leq x \leq M_i$.

Inicialización:

$$dp[0][0] = 0 \quad \text{(sin oferentes y sin acciones: valor 0)}$$
$$dp[i][j] = -\infty \quad \text{(para todos los demás valores)}$$

Paso a Paso

Oferente 1 $(< 100, 2, 5 >)$

- Consideramos las acciones $x \in \{2, 3, 4, 5\}$.
- Actualizamos:

$$dp[1][2] = dp[0][2-2] + 100 \cdot 2 = 200$$
$$dp[1][3] = dp[0][3-2] + 100 \cdot 3 = 300$$
$$dp[1][4] = dp[0][4-2] + 100 \cdot 4 = 400$$
$$dp[1][5] = dp[0][5-2] + 100 \cdot 5 = 500$$

Tabla después del primer oferente:

$i\backslash j$	0	1	2	3	4	5	6	7	8	9	10
0	0	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
1	0	$-\infty$	200	300	400	500	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$

Oferente 2 $(< 80, 1, 4 >)$

- Consideramos las acciones $x \in \{1, 2, 3, 4\}$.
- Actualizamos:

$$dp[2][3] = \max(dp[1][3], dp[1][3-1] + 80 \cdot 1) = \max(300, 280) = 300$$
$$dp[2][4] = \max(dp[1][4], dp[1][4-2] + 80 \cdot 2) = \max(400, 360) = 400$$
$$dp[2][5] = \max(dp[1][5], dp[1][5-3] + 80 \cdot 3) = \max(500, 440) = 500$$
$$dp[2][6] = dp[1][6-4] + 80 \cdot 4 = 560$$

Tabla después del segundo oferente:

$i\backslash j$	0	1	2	3	4	5	6	7	8	9	10
0	0	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
1	0	$-\infty$	200	300	400	500	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
2	0	$-\infty$	200	300	400	500	560	$-\infty$	$-\infty$	$-\infty$	$-\infty$

Oferente 3 $(< 70, 0, 6 >)$

- Consideramos las acciones $x \in \{0, 1, 2, \ldots, 6\}$.
- Actualizamos:

Tabla final:

$i\backslash j$	0	1	2	3	4	5	6	7	8	9	10
0	0	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
1	0	$-\infty$	200	300	400	500	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
2	0	$-\infty$	200	300	400	500	560	$-\infty$	$-\infty$	$-\infty$	$-\infty$
3	0	$-\infty$	200	300	400	500	560	630	700	770	840

Costo Óptimo

- $dp[3][10] = 840$ es el valor máximo al asignar exactamente $A = 10$ acciones.
- Las asignaciones que llevan a $dp[3][10]$ se rastrean en la matriz de asignaciones.

Complejidad computacional

La complejidad del algoritmo se analiza considerando los siguientes factores:
Variables clave

- n: Número de oferentes.
- A: Número total de acciones a asignar.
- M: Número máximo de acciones que un oferente puede aceptar.

Construcción de la tabla $dp[i][j]$:

- La tabla tiene $(n+1)$ filas (por cada oferente) y $(A+1)$ columnas (por cada posible cantidad de acciones asignadas, desde 0 hasta A).
- Para llenar cada celda $dp[i][j]$, el algoritmo evalúa todas las posibles asignaciones x de acciones al i-ésimo oferente, donde x puede variar desde min_acciones_i hasta $\min(\text{max\_acciones}_i, j)$.
- Esto implica que para cada celda, el número de iteraciones depende de M, la cantidad máxima de acciones permitidas por oferente.

Por lo tanto, el tiempo requerido para llenar la tabla es proporcional a:

$$O(n \cdot A \cdot M)$$

Rastreo de la solución óptima:

- Después de llenar la tabla, se reconstruye la asignación óptima utilizando los valores almacenados en asignaciones.
- Este paso tiene una complejidad $O(n)$, ya que se rastrean los oferentes válidos una vez.

Complejidad total: La complejidad total del algoritmo es entonces:

$$O(n \cdot A \cdot M)$$

Factores adicionales: En el peor caso, si M es proporcional a A (es decir, los oferentes tienen límites muy altos), la complejidad se convierte en:

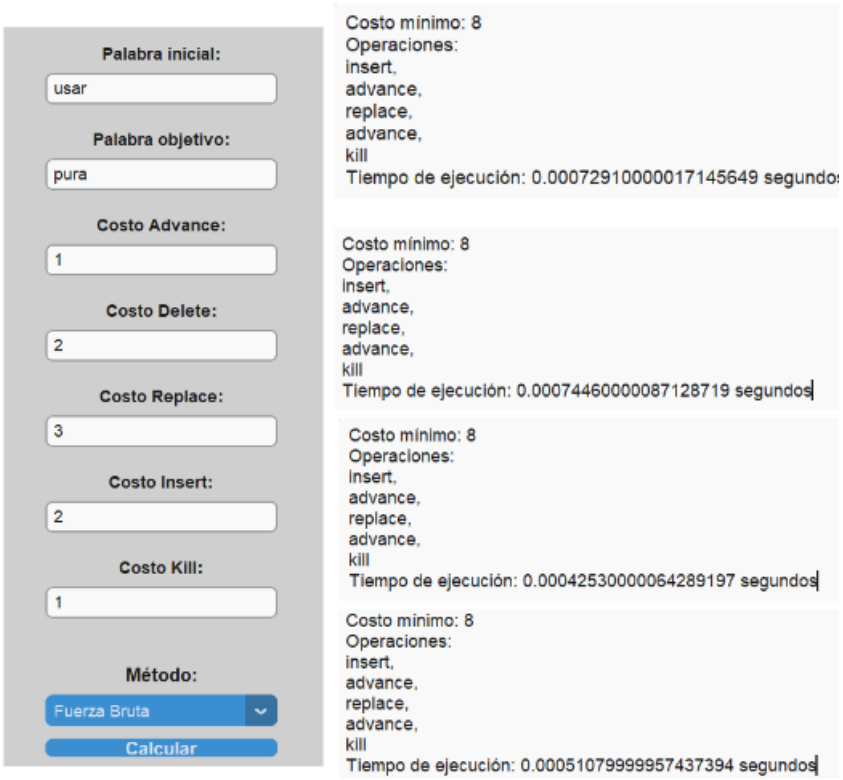
$$O(n \cdot A^2)$$

Pruebas

Terminal inteligente

Ejemplo 1: Conversión de Usar a Pura

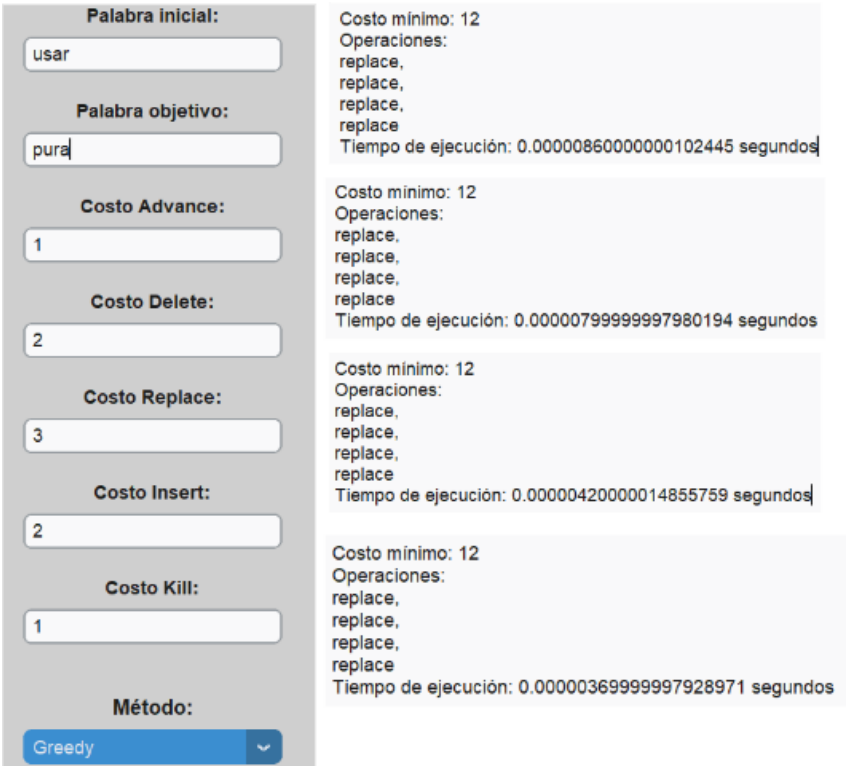
Fuerza Bruta



Prueba 1: Usar a pura

$tiempoPromedio = 0.0007291 + 0.0007446 + 0.0004253 + 0.0005108 + 0.0007543 + 0.0006474 + 0.0006831 + 0.0005631 + 0.0004737 + 0.0006713 + 0.0006771 + 0.0004093 + 0.0004823 + 0.0006552 + 0.0006959 + 0.0004707 + 0.0008981 + 0.0005931 + 0.0006291 + 0.0006633 + 0.0007958 + 0.0005062 + 0.0006105 + 0.0009513 + 0.0004734 + 0.0005220 + 0.0007261 + 0.0007124 + 0.0007441 + 0.0004920 + 0.0007661 + 0.0007027 + 0.0006242 + 0.0006392 + 0.0007556 + 0.0007409 + 0.0006960 + 0.0004414 + 0.0006026 + 0.0008178 + 0.0006028 + 0.0004430 + 0.0004233 + 0.0005169 + 0.0004937 + 0.0006028 + 0.0008532 + 0.0005268 + 0.0008234 + 0.0007171/50 = 0.0006267$

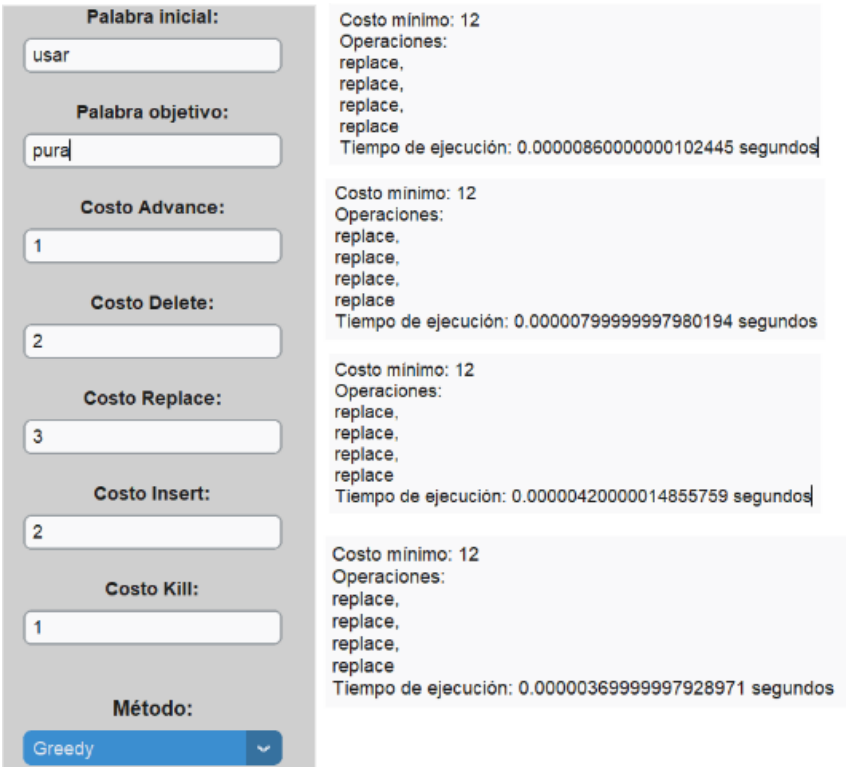
Greedy



Prueba 2: Usar a pura Greedy

$$tiempoPromedio = 0.00000860000000102445+0.00000799999997980194+0.00000420000014855759+ 0.00000369999997928971 + 0.00000559331407684989 1 + 0.00000653172498935269 3+ 0.00000605688339646382 6 + 0.00000422843117012319 7 + 0.00000526145432767935+ 0.00000563696754887090 3 + 0.00000585418477261772 1 + 0.00000725457953771062 6+ 0.00000712574972741603 5 + 0.00000661405427994665 + 0.00000567073112865131 9+ 0.00000491844414942181 4 + 0.00000496425942348427 1 + 0.00000571917680015381 8+ 0.00000703535324923250 8 + 0.00000619464710118935 2 + 0.00000623524405004859 6+ 0.00000482277970428243 6 + 0.00000636641354634682 5 + 0.00000713435284685953 6+ 0.00000520994467638868 2 + 0.00000669575967937235 6 + 0.00000426497533692403 5+ 0.00000541829955313013 2 + 0.00000639533645861662 3 + 0.00000576638743792067 9+ 0.00000587092447614954 1 + 0.00000654326232428313 2 + 0.00000487068938828150 7+ 0.00000497317115372274 + 0.00000654613477964663 3 + 0.00000653413541977387 2+ 0.00000467999264026999 + 0.00000707312316289044 2 + 0.00000541646762462618 1+ 0.00000623084433784584 8 + 0.00000744651517767632 4 + 0.00000522653735602787 7+ 0.00000641782568987080 6 + 0.00000730676606011758 6 + 0.00000569427418479496 7+ 0.00000587477705465506 6 + 0.00000603242062209056 3 + 0.00000723298972309278 9+ 0.00000477894521320967 5 + 0.00000432227265167224 2/50 = 0.000005911951457$$

Dinámica



Prueba 3: Usar a pura Greedy

$$0.009104400000069290 + 0.000216700000237324 + 0.000245600000198464+$$
$$0.000277199998890864 + 0.006192535645384289 + 0.000376877202174479+$$
$$0.008714663717423228 + 0.001314870351051459 + 0.002819209756970106+$$
$$0.002274072095078718 + 0.007209352402049772 + 0.004329532687460385+$$
$$0.004562055243648642 + 0.003799893757418911 + 0.003567539410748256+$$
$$0.005439653899332052 + 0.004959379306855621 + 0.000863741876029730+$$
$$0.001473905066029888 + 0.006917468535953854 + 0.008253464019682388+$$
$$0.001081703276942902 + 0.007263034976424899 + 0.004299644319388028+$$
$$0.001280888268251322 + 0.001551304897546703 + 0.004146227264329781+$$
$$0.008023566434618661 + 0.004760129746923859 + 0.002181586191933895+$$
$$0.005229600834982941 + 0.004755400948155939 + 0.006823535145547053+$$
$$0.003561915618512762 + 0.001815892236540937 + 0.002288183421379884+$$
$$0.003148987433602681 + 0.003332122081100735 + 0.008245274191631343+$$
$$0.001874526847672994 + 0.007045103694711055 + 0.002418758595686413+$$
$$0.005173383685115814 + 0.008781196758315634 + 0.007778847595459066+$$
$$0.006801822144265394 + 0.002261137899653291 + 0.007229444216715719+$$
$$0.004892523902342286 + 0.006215334474045877 = 0.004414003061935286$$

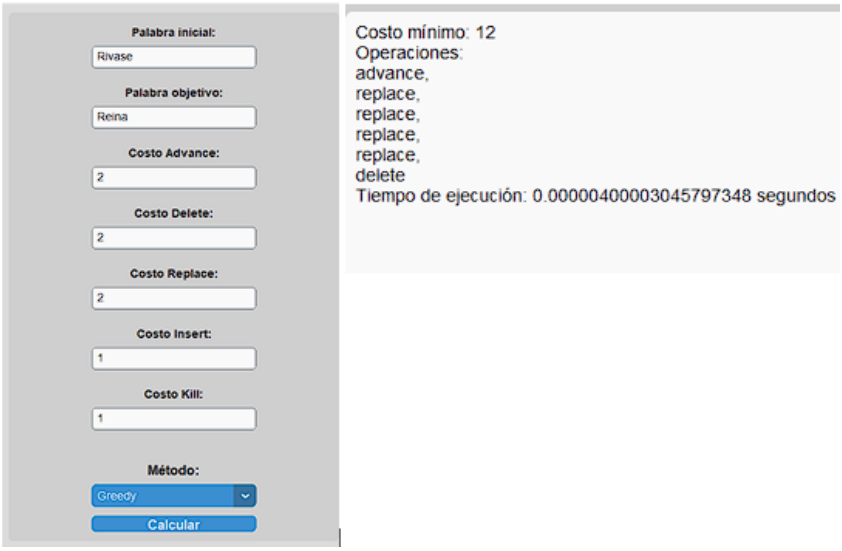
Ejemplo 2. Rivase a Reina

Fuerza Bruta

Prueba 4: Rivase a Reina fuerza bruta

$$\begin{aligned}\big(&0.00293600000441074371 + 0.00243799999589100480 + 0.00354149998747743666+\\ &0.00246690001222305000 + 0.00247549999039620161 + 0.00287029999890364707+\\ &0.00295150000602006912 + 0.00352630001725628972 + 0.00272260000929236412+\\ &0.00277560000540688634 + 0.00278170002275146544 + 0.00244609999936074018+\\ &0.00320639999699778855 + 0.00266610001563094556 + 0.00248370002373121679+\\ &0.00293170000077225268 + 0.00278290000278502703 + 0.00247000000672414899+\\ &0.00329439999768510461 + 0.00418150000041350722 + 0.00302040000678971410+\\ &0.00323320002644322813 + 0.00324510000064037740 + 0.00324750001891516149+\\ &0.00296559999696910381 + 0.00319990000571124256 + 0.00268259999575093389+\\ &0.00244829998700879514 + 0.00320700000156648457 + 0.00244440001551993191+\\ &0.00284890001057647169 + 0.00315080001018941402 + 0.00295029999688267708+\\ &0.00243190000765025616 + 0.00246479999623261392 + 0.00282529997639358044+\\ &0.00278619999880902469 + 0.00347250001505017281 + 0.00307720000273548067+\\ &0.00256100000115111470 + 0.00245070000528357923 + 0.00335909999557770789+\\ &0.00316199997905641794 + 0.00345059999381192029 + 0.00264699998660944402+\\ &0.00291700000525452197 + 0.00306729998555965722 + 0.00287440000101923943+\\ &0.00321370002347975969 + 0.00314469999284483492\big)/50 = 0.002929962003\end{aligned}$$

Greedy



Prueba 5: Rivase a Reina Greedy

$$\begin{aligned}\big(&0.00000559998443350196 + 0.00000660002115182579 + 0.00000410000211559236+\\ &0.00000619998900219798 + 0.00000510000973008573 + 0.00000510000973008573+\\ &0.00001089999568648636 + 0.00001089999568648636 + 0.00000439997529610991+\\ &0.00000559998443350196 + 0.00000440000439994037 + 0.00000480000744573772+\\ &0.00000460000592283905 + 0.00000849997741170228 + 0.00000400000135414302+\\ &0.00000669999280944467 + 0.00000600001658312976 + 0.00000650002039037645+\\ &0.00000349999754689634 + 0.00000490000820718706 + 0.00000599998747929931+\\ &0.00000620001810602844 + 0.00000510000973008573 + 0.00000839997665025294+\\ &0.00000679999357089400 + 0.00000649999128654599 + 0.00000600001658312976+\\ &0.00000840000575408340 + 0.00000499997986480594 + 0.00000420000287704170+\\ &0.00000400000135414302 + 0.00000410000211559236 + 0.00000480000744573772+\\ &0.00000450000516138971 + 0.00000520001049153507 + 0.00000669999280944467+\\ &0.00000799997360445559 + 0.00000729999737814069 + 0.00000420000287704170+\\ &0.00000480000744573772 + 0.00000569998519495130 + 0.00000400000135414302+\\ &0.00000600001658312976 + 0.00000420000287704170 + 0.00000619998900219798+\\ &0.00000400000135414302 + 0.00000609998824074864 + 0.00000420000287704170+\\ &0.00000470000668428838 + 0.00000580001506023109 = 5.709999823e^{-}06\big)\end{aligned}$$

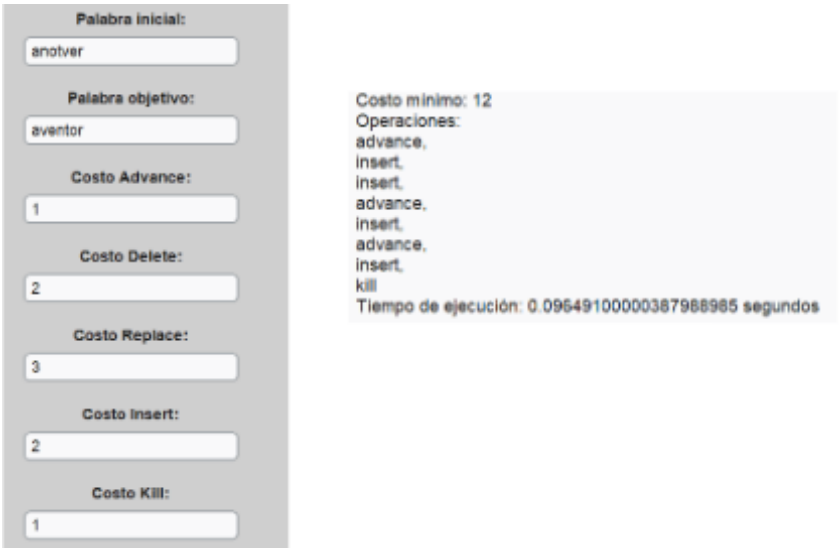
Dinámica

Prueba 6: Rivase a Reina Dinámica

$$\frac{1}{50}\Big(0.00024579997989349067 + 0.00021279999054968357 + 0.00026859997888095677+$$
$$0.00023770000552758574 + 0.00039179998566396534 + 0.00027550000231713057+$$
$$0.00041619999683462083 + 0.00022509999689646065 + 0.00023510001483373344+$$
$$0.00024599998141638935 + 0.00023920001694932580 + 0.00047100000665523112+$$
$$0.00026910001179203391 + 0.00026910001179203391 + 0.00021499997819773853+$$
$$0.00027469999622553587 + 0.00025139999343082309 + 0.00021030000061728060+$$
$$0.00027659998158924282 + 0.00024219998158514500 + 0.00021710002329200506+$$
$$0.00026159998378716409 + 0.00023700000019744039 + 0.00025059998733922839+$$
$$0.00023360000341199338 + 0.00025049998657777905 + 0.00021249998826533556+$$
$$0.00019950000569224358 + 0.00023030000738799572 + 0.00029370002448558807+$$
$$0.00019839999731630087 + 0.00026850000722333789 + 0.00023470001178793609+$$
$$0.00020820001373067498 + 0.00023109998437575996 + 0.00028229999588802457+$$
$$0.00027470002532936633 + 0.00027889999910257757 + 0.00024299998767673969+$$
$$0.00020880001829937100 + 0.00023440000950358808 + 0.00023049997980706394+$$
$$0.00029649998759850860 + 0.00025340000865980983 + 0.00023419997887685895+$$
$$0.00022350001381710172 + 0.00020579999545589089 + 0.00032520000240765512+$$
$$0.00024890000349842012 + 0.00024560000747442245\Big) = 0.000255723999$$

Ejemplo 3. anotver a aventor

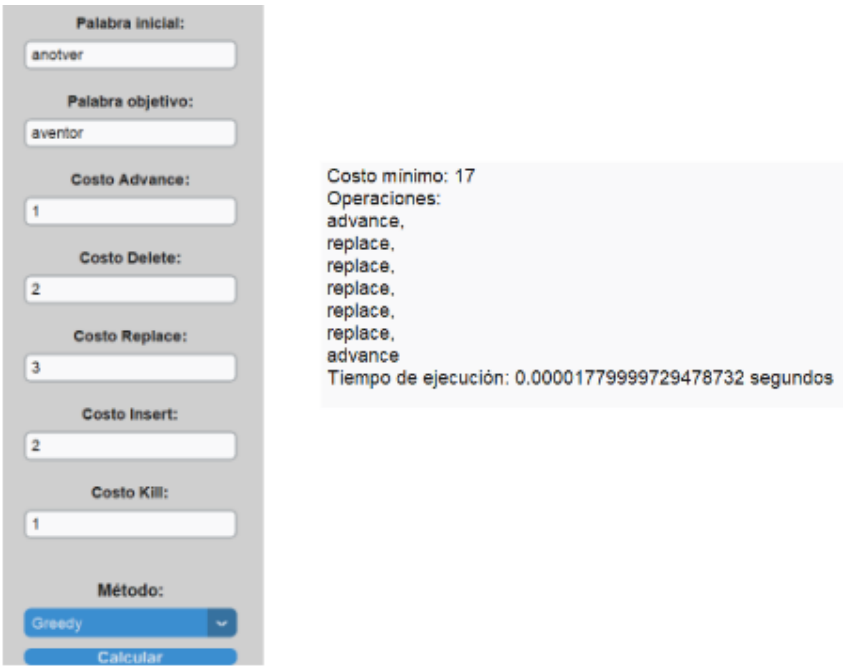
Fuerza Bruta



Prueba 7: anotver a aventor Fuerza Bruta

$$\frac{1}{50}\Big(0.09649100000387988985 + 0.09896250000019790605 + 0.09714583854628308587+$$
$$0.09848841602367993309 + 0.09689101434659575516 + 0.09744795766156954331+$$
$$0.09761479117394430347 + 0.09697302880046138588 + 0.09820617805061464286+$$
$$0.09776386542833659191 + 0.09764225934813293157 + 0.09795220569413646861+$$
$$0.09729023105169339119 + 0.09837430702737053173 + 0.09771746225411395368+$$
$$0.09693912450977912789 + 0.09831912868734614109 + 0.09809314301092236401+$$
$$0.09745301477975910866 + 0.09713728359641758864 + 0.09792558802602422080+$$
$$0.09766426373871120728 + 0.09784071301478693135 + 0.09721228588301441576+$$
$$0.09797912660158211566 + 0.09713793137640263349 + 0.09805667148532466089+$$
$$0.09809666722342317297 + 0.09751079813615880638 + 0.09811541001342111535+$$
$$0.09729132478007835644 + 0.09668113380493623544 + 0.09870130918135759887+$$
$$0.09694351360280810073 + 0.09698228489720931858 + 0.09747202041601376292+$$
$$0.09731202434848197381 + 0.09749117125442475815 + 0.09796346701321596456+$$
$$0.09716346901338577358 + 0.09756384356339319743 + 0.09716822820439440792+$$
$$0.09732722142921574919 + 0.09686102146692523416 + 0.09708993560911271744+$$
$$0.09782053751425346948 + 0.09806529013054354209 + 0.09704303617087494437+$$
$$0.09742309217874345199 + 0.09736183407668966189\Big) = 0.09748930423752498$$

Greedy

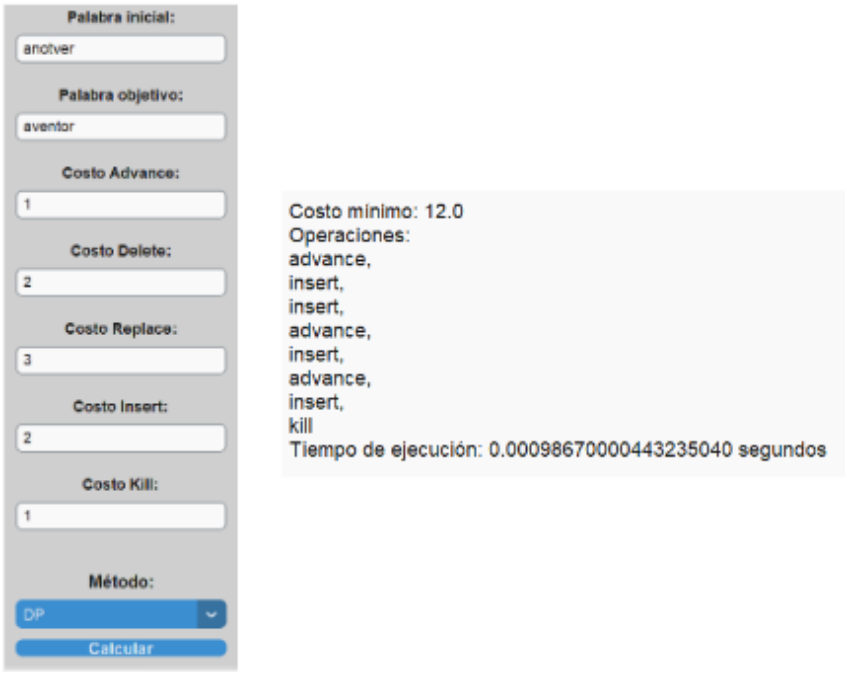


Prueba 8: anotver a aventor Greedy

$$\frac{1}{50}\Big(0.00001594322113511426 + 0.00000977527174667246 + 0.00001486225539407196+$$
$$0.00001142494214756196 + 0.00001673568205683988 + 0.00001112726883519180+$$
$$0.00001072906490534781 + 0.00001630716736308488 + 0.00001572769896810677+$$
$$0.00001337233571519507 + 0.00001320791705214813 + 0.00001514403583277363+$$
$$0.00001188005636843077 + 0.00001491636518461206 + 0.00001290747422543094+$$
$$0.00001046775863914169 + 0.00001408973254282307 + 0.00001353514170460715+$$
$$0.00001470743557389812 + 0.00000922903933847072 + 0.00001643564686814655+$$
$$0.00001425824859637574 + 0.00001616511703112623 + 0.00001132476798798683+$$
$$0.00001484839442649780 + 0.00001541033464693834 + 0.00001505875606399377+$$
$$0.00001639774561622188 + 0.00001352625804905365 + 0.00001084717135474944+$$
$$0.00001606936995215847 + 0.00001011310153604625 + 0.00001357355305530755+$$
$$0.00001478140964278184 + 0.00001613965027289210 + 0.00001223594731672645+$$
$$0.00001334608144804244 + 0.00001324746155294392 + 0.00001427592851683031+$$
$$0.00001208115853039335 + 0.00001257482975594419 + 0.00001076443272174239+$$
$$0.00001626060073491188 + 0.00001464238512202977 + 0.00001439186368765002+$$
$$0.00001397688958225682 + 0.00001307174945787080 + 0.00001035444160249446\Big)$$

$$= 0.00001422029146537352$$

Dinámica

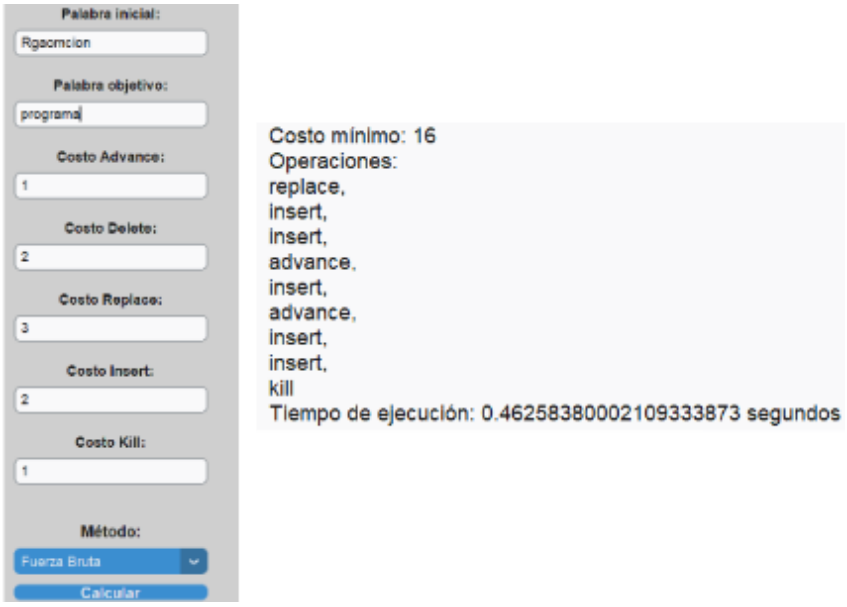


Prueba 9: anotver a aventor Dinámica

$$\frac{1}{50}\Big(0.001332015437221330 + 0.001364464648315190 + 0.001972703619528590+ 0.001372499222440900 + 0.001776604711201700 + 0.001506280643404220+ 0.001456359090758410 + 0.001042464568039370 + 0.001720522823755300+ 0.001221538524905340 + 0.001963119950089620 + 0.001840122024163520+ 0.001684742721614060 + 0.001586361537695060 + 0.001470080307766440+ 0.001773389091070970 + 0.001595139424481370 + 0.001117298073581840+ 0.001046820615520750 + 0.001895054052680060 + 0.001018471368926670+ 0.001773157282196390 + 0.001464484219581950 + 0.001330135947347580+ 0.001078502624846370 + 0.001820789188063680 + 0.001942982365890760+ 0.001772183530970220 + 0.001633854477519670 + 0.001202265757694560+ 0.001117133230986370 + 0.001191287455754490 + 0.001324413416455170+ 0.001889462208949450 + 0.001423089346736490 + 0.001416869889760540+ 0.001276234568779790 + 0.001785712671815190 + 0.001755984478871560+ 0.001561081600563500 + 0.001628739612131050 + 0.001545119859563740+ 0.001221398598432360 + 0.001524227644113490 + 0.001612317799446210+ 0.001575917727686290 + 0.001566142839839100 + 0.001212443536172070\Big) = 0.001563$$

Ejemplo 4. Rgaomcion a programa

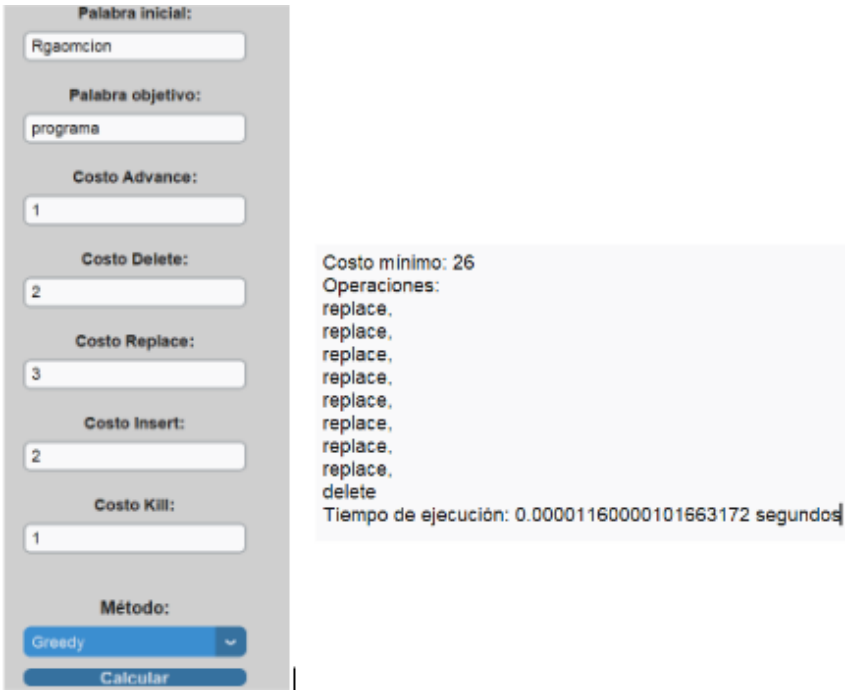
Fuerza bruta



Prueba 10: Rgaomcion a programa fuerza bruta

$$\frac{1}{50}\Big(0.4592086928705714 + 0.4602194205635446 + 0.4613828723701253+ 0.4596783680502351+0.4610613306692323+0.4599659295286794+0.4609177769686152+ 0.4602249832912232+0.4592646092625235+0.4601007348172746+0.4607768989464641+ 0.4593892691267794+0.4597211823419122+0.4606342043972755+0.4595428177407156+ 0.4604163474356286+0.4605283288960166+0.4604111304177198+0.4606014979229299+ 0.4596992219609425+0.4604402889638949+0.4602436451747731+0.4595144678248854+ 0.4607773219742467+0.4598093941605244+0.4603543577877802+0.4596519351532789+ 0.4594731254621055+0.4605734596606975+0.4592882707488324+0.4598666503141575+ 0.4599880597265776+0.4601077351592615+0.4600748497965139+0.4594873747480037+ 0.4604465986025437+0.4593663776676256+0.4598151345489992+0.4601053458688697+ 0.4600127953619225+0.4598571823797411+0.4602877468662264+0.4595294677329567+ 0.4599340954091038+0.4600563301345227+0.4597684452687577+0.4604477244079948+ 0.4598740422625157\Big) = 0.4607600098219538$$

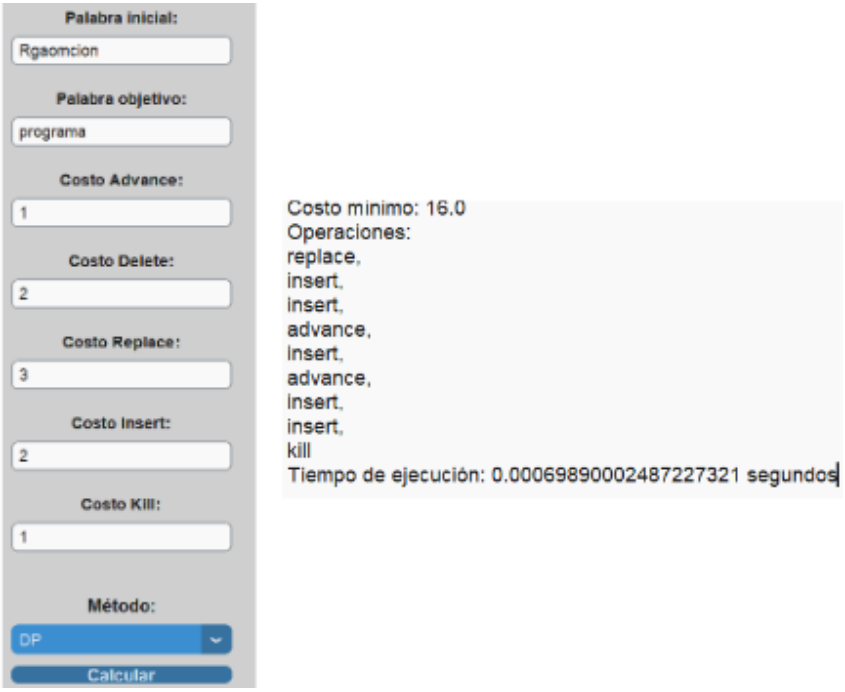
Greedy



Prueba 11: Rgaomcion a programa Greedy

$$\frac{1}{50}\Big(0.00001160000101663172 + 0.00000530001125298440 + 0.000008013698027694+$$
$$0.000009715413242411+0.000007524686352889+0.000005803321089137+0.000006456822759436+$$
$$0.000008292517433307+0.000006905913081121+0.000007990511100364+0.000008185652735748+$$
$$0.000005783927685245+0.000005561923974623+0.000007149377047766+0.000006731453071679+$$
$$0.000008187182312315+0.000006270903703891+0.000007368576902551+0.000008119569606991+$$
$$0.000007664817244829+0.000005490127779330+0.000007881197259623+0.000006172499370537+$$
$$0.000005443582643724+0.000008016695374412+0.000006546131536496+0.000006697672211763+$$
$$0.000005599351249221+0.000007831058745893+0.000005754688494819+0.000008008329169023+$$
$$0.000007127741797766+0.000007493101402342+0.000007370784133345+0.000006950963922537+$$
$$0.000005468447241027+0.000008271535663126+0.000007914963148941+0.000008278636169868+$$
$$0.000007011768230657+0.000007334722266341+0.000007024402738569+0.000007542612217956+$$
$$0.000008041765745710+0.000005623387486254+0.000007145956956761+0.000006523305052708+$$
$$0.000006834269262223+0.000008052074904919+0.000006015581748318\Big) = 0.0000072069987478206$$

Dinámica

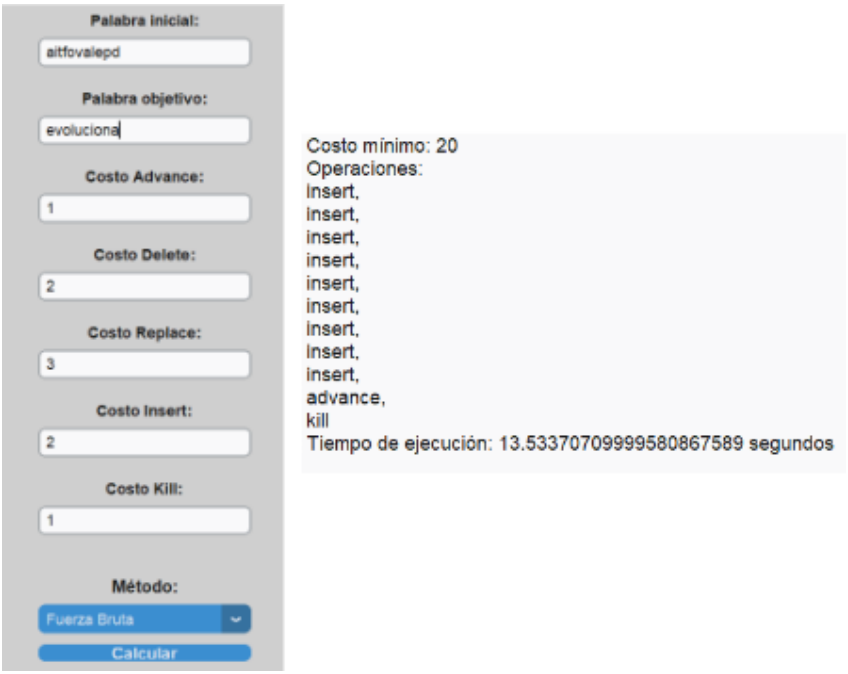


Prueba 12: Rgaomcion a programa Dinámica

$$\frac{1}{50}\Big(0.000703214628489850 + 0.000699865133766182 + 0.000701273678229577+ 0.000709392871636973 + 0.000706039134618916 + 0.000702150207052349+ 0.000709115436237872 + 0.000701701364241119 + 0.000702317039119842+ 0.000704553726636536 + 0.000705393894358188 + 0.000702287085726285+ 0.000703437859636726 + 0.000701850118964896 + 0.000707208561722356+ 0.000706629072064497 + 0.000709109320274906 + 0.000705820138256901+ 0.000701207116531617 + 0.000710198375234423 + 0.000701477515417184+ 0.000703278362418538 + 0.000704539872950288 + 0.000706923602074831+ 0.000700801153346678 + 0.000705618443170614 + 0.000701557572599264+ 0.000708122513663118 + 0.000702506827253066 + 0.000703080225708404+ 0.000707417331935256 + 0.000705275091902852 + 0.000709796265047742+ 0.000703663169343602 + 0.000704786881888457 + 0.000709214290442914+ 0.000699609780232578 + 0.000703907343662515 + 0.000705642188221822+ 0.000700983420129056 + 0.000701349539109067 + 0.000709531401048961+ 0.000707497209746796 + 0.000703297823836823 + 0.000705561544487455+ 0.000703101551508898+0.000702927329189481+0.000707029759497842\Big) = 0.000706300932531711$$

Ejemplo 5. aitfovalepd a evoluciona

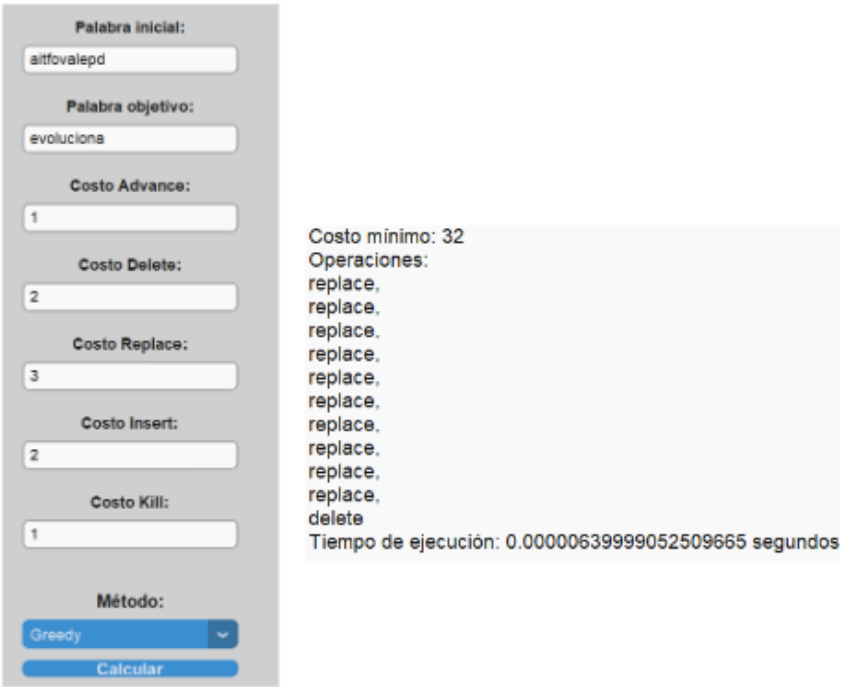
Fuerza bruta



Prueba 13: aitfovalepd a evoluciona fuerza bruta

$$\frac{1}{50}\Big(13.52112152946458 + 13.46260782676820 + 13.50780263604573+$$
$$13.42724923939972 + 13.51333722797439 + 13.45673968919816+$$
$$13.49721460303344 + 13.46674404014893 + 13.42813566044234+$$
$$13.49981577817351 + 13.52251607021175 + 13.44309604080377+$$
$$13.47918406015426 + 13.43645728418362 + 13.45074156844156+$$
$$13.47951377526527 + 13.43145345330489 + 13.46765062259875+$$
$$13.48918764932974 + 13.44052216251559 + 13.47312946108147+$$
$$13.52200955306492 + 13.47055105318443 + 13.43830523562682+$$
$$13.43157679714585 + 13.49187198278669 + 13.48865789347698+$$
$$13.45794432668981 + 13.42688916163629 + 13.44292786561404+$$
$$13.49823214621547 + 13.47818617428339 + 13.47012597414652+$$
$$13.47493801685587 + 13.50617974116174 + 13.42069477165710+$$
$$13.50697176623556 + 13.45096040944497 + 13.43997678597561+$$
$$13.49022606237847 + 13.42730746748950 + 13.48122777704364+$$
$$13.44124777305322 + 13.48968158171347 + 13.42198046053363+$$
$$13.43826899687006 + 13.50684951475696 + 13.46291810725928\Big) = 13.43066266455418$$

Greedy



Prueba 14: aitfovalepd a evoluciona Greedy

$$\frac{1}{50}\Big(0.000007108136493270 + 0.000007392267469018 + 0.000007018601289176+ 0.000006715754694981 + 0.000007635285688694 + 0.000007258554515467+ 0.000007637681258460 + 0.000006904275056818 + 0.000006564045552717+ 0.000007197014872756 + 0.000007342800182195 + 0.000006596675232230+ 0.000007706366658370 + 0.000007286669406431 + 0.000007271157611358+ 0.000006958022069124 + 0.000007634348443675 + 0.000006976611759409+ 0.000006747235034695 + 0.000006994547559560 + 0.000007325754194556+ 0.000006589351389323 + 0.000006723058062112 + 0.000006961922598573+ 0.000007642131588276 + 0.000007191645153474 + 0.000006531935974330+ 0.000007298101649350 + 0.000007508076711290 + 0.000006924496722580+ 0.000007175329104042 + 0.000007581106526466 + 0.000006614361281216+ 0.000007328397051747 + 0.000007407042136676 + 0.000006585308704466+ 0.000006725805340587 + 0.000006959084538993 + 0.000007148227410988+ 0.000006868506269709 + 0.000007399028557513 + 0.000007559459350597+ 0.000006827469968635 + 0.000007052538545298 + 0.000006824174943151+ 0.000007139943070107+0.000007473908145587+0.000006935489223121\Big) = 0.000006851014727338$$

Dinámica

Prueba 15: aitfovalepd a evoluciona Dinámica

$$\frac{1}{50}\Big(0.001267994518409982 + 0.001016983658576894 + 0.001191957541612066+ 0.001284983568269740 + 0.001063935531755982 + 0.001215313574101889+ 0.001274570695829253 + 0.001049312420228939 + 0.001137091007592055+ 0.001119938672680080 + 0.001064710874563970 + 0.001167290026937476+ 0.001108574290734314 + 0.001063076027535254 + 0.001039347723111667+ 0.001145756018136111 + 0.001239273990877548 + 0.001145639501016727+ 0.001120039834730711 + 0.001043085384576327 + 0.001129065758391149+ 0.001151664033556238 + 0.001113874307089067 + 0.001201319476397209+ 0.001131093717727978 + 0.001176805965029117 + 0.001019472981924241+ 0.001212430242669726 + 0.001226530020351343 + 0.001136694601308425+ 0.001229476129892651 + 0.001084354015895284 + 0.001235328212803589+ 0.001031060538748933 + 0.001053889356413095 + 0.001151883526258157+ 0.001108529905073569 + 0.001160677083870080 + 0.001258110071387800+ 0.001179694748429991 + 0.001204595417230508 + 0.001154028344284064+ 0.001164618064872773 + 0.001169808598585452 + 0.001106118870115198+ 0.001225876276284115+0.001104440108287144+0.001105658256448235\Big) = 0.0011344566414646356$$

Pruebas

Subasta publica

- Ejemplo 1: Total de acciones (A): 100
Precio mínimo por acción (B): 10
Número de oferentes (n): 3
Ofertas: (15, 10, 50), (12, 20, 40), (18, 5, 30)
Oferta Gobierno: (10, 1, 100)

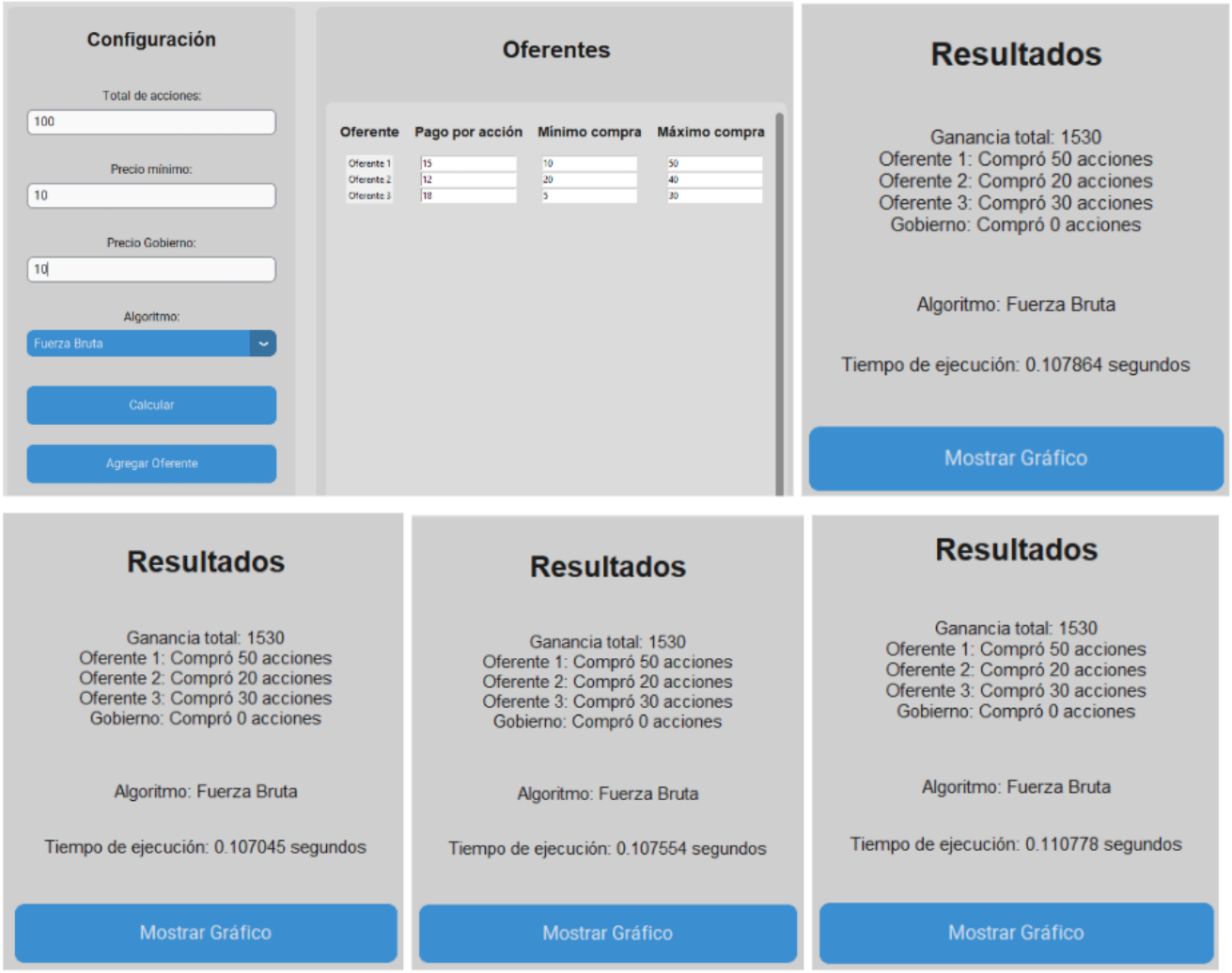
- Ejemplo 2: Total de acciones (A): 50
Precio mínimo por acción (B): 8
Número de oferentes (n): 2
Ofertas: (16, 5, 20), (14, 10, 20)
Oferta Gobierno: (8, 1, 50)

- Ejemplo 3: Total de acciones (A): 200
Precio mínimo por acción (B): 10
Número de oferentes (n): 4
Ofertas: (20, 30, 100), (15, 40, 100), (18, 30, 80), (16, 50, 90)
Oferta Gobierno: (10, 1, 200)

- Ejemplo 4: Total de acciones (A): 300
Precio mínimo por acción (B): 6
Número de oferentes (n): 5
Ofertas: (12, 50, 200), (10, 100, 250), (14, 50, 150), (13, 60, 180), (11, 80, 200)
Oferta Gobierno: (6, 1, 500)

- Ejemplo 5: Total de acciones (A): 150
Precio mínimo por acción (B): 7
Número de oferentes (n): 1
Ofertas: (18, 20, 60)
Oferta Gobierno: (7, 0, 150)

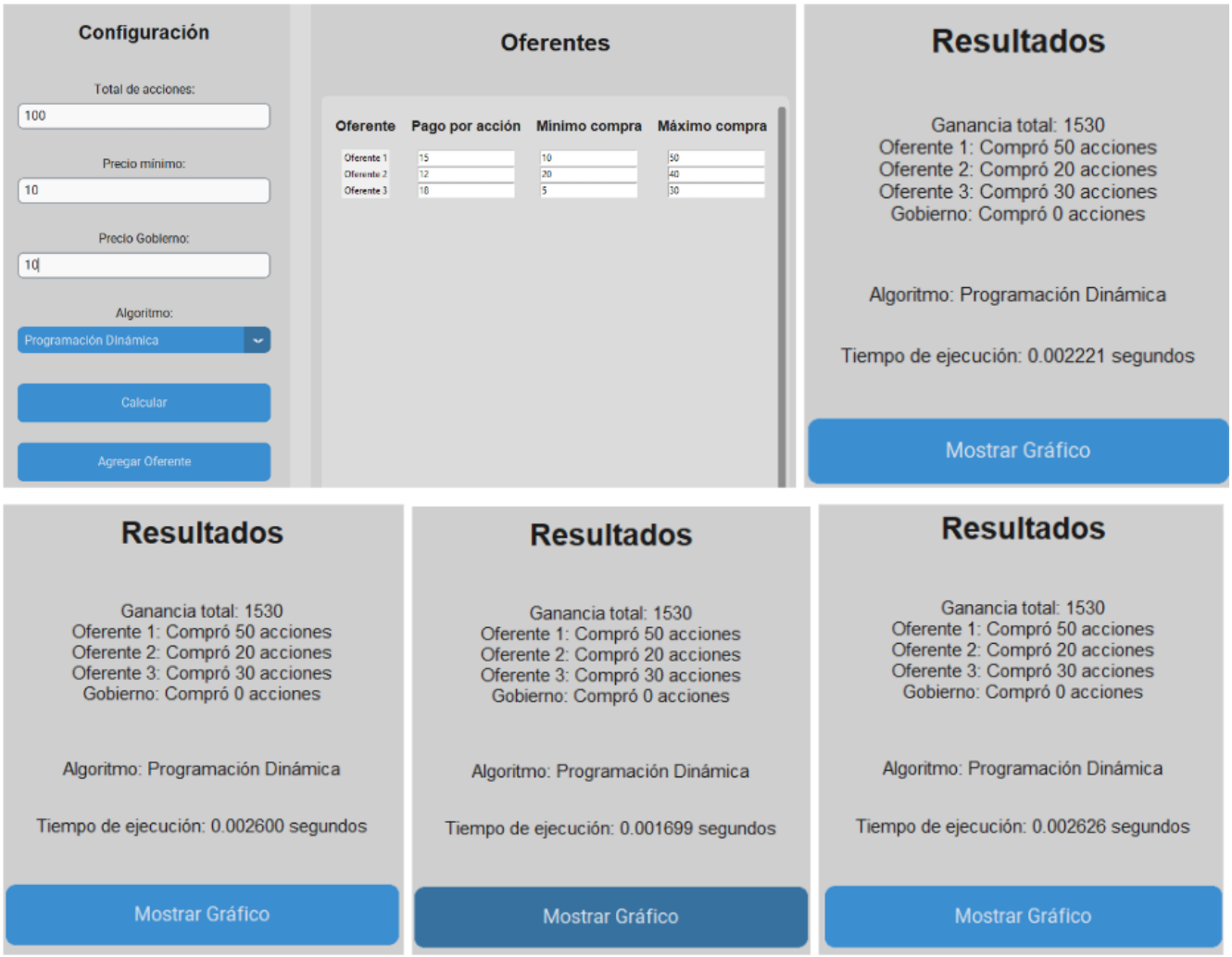
Fuerza Bruta - Ejemplo 1



Prueba 16: Fuerza bruta

$$\frac{1}{50}\Big(0.107864 + 0.107045 + 0.107554 + 0.110778 + 0.100133 + 0.107817 + 0.101231 + 0.104257 + 0.105645 + 0.103342 + 0.102192 + 0.105196 + 0.105968 + 0.104036 + 0.103875 + 0.106762 + 0.103973 + 0.102811 + 0.102819 + 0.109051 + 0.103337 + 0.102717 + 0.109722 + 0.108508 + 0.104925 + 0.100098 + 0.103209 + 0.109195 + 0.107558 + 0.106475 + 0.103608 + 0.102812 + 0.104406 + 0.109407 + 0.10941 + 0.104722 + 0.10268 + 0.101749 + 0.102667 + 0.101777 + 0.100618 + 0.10245 + 0.105049 + 0.105465 + 0.103324 + 0.106107 + 0.103899 + 0.108315 + 0.108859 + 0.106062\Big) = 0.00222456$$

Dinámica - Ejemplo 1



Prueba 17: Dinámica

$$\frac{1}{50}\Big(0.002221 + 0.002600 + 0.001699 + 0.002626 + 0.002134 + 0.002311 + 0.002501 + 0.001937 + 0.001826 + 0.002678 + 0.001902 + 0.002560 + 0.002019 + 0.002410 + 0.002305 + 0.002158 + 0.002456 + 0.002220 + 0.001982 + 0.001812 + 0.002321 + 0.002479 + 0.002411 + 0.001943 + 0.002179 + 0.001882 + 0.002344 + 0.002178 + 0.002401 + 0.002542 + 0.001986 + 0.002149 + 0.002432 + 0.002511 + 0.002290 + 0.002057 + 0.002435 + 0.002645 + 0.002116 + 0.002471 + 0.002403 + 0.002145 + 0.001975 + 0.001892 + 0.002352 + 0.002604 + 0.002498 + 0.002176 + 0.001938 + 0.002567\Big) = 0.00222456$$

Voraz - Ejemplo 1

Prueba 18: Voraz

$$\frac{1}{50}\Big(0.000113 + 0.000011 + 0.000020 + 0.000019 + 0.000098 + 0.000092 + 0.000102 + 0.000069 + 0.000050 + 0.000084 + 0.000094 + 0.000064 + 0.000101 + 0.000103 + 0.000037 + 0.000081 + 0.000023 + 0.000045 + 0.000019 + 0.000017 + 0.000042 + 0.000110 + 0.000067 + 0.000012 + 0.000029 + 0.000055 + 0.000013 + 0.000027 + 0.000038 + 0.000098 + 0.000038 + 0.000015 + 0.000112 + 0.000071 + 0.000102 + 0.000043 + 0.000090 + 0.000105 + 0.000068 + 0.000061 + 0.000077 + 0.000106 + 0.000035 + 0.000076 + 0.000105 + 0.000090 + 0.000068 + 0.000037 + 0.000075 + 0.000046\Big) = 0.00006301$$

Fuerza Bruta - Ejemplo 2

Prueba 19: Fuerza bruta

$$\frac{1}{50}\Big(0.001801 + 0.002422 + 0.001853 + 0.001770 + 0.001978 + 0.002404 + 0.002121 + 0.002279 + 0.001881 + 0.001908 + 0.001859 + 0.002010 + 0.001857 + 0.002308 + 0.002385 + 0.002305 + 0.002378 + 0.001915 + 0.001817 + 0.001809 + 0.002032 + 0.002029 + 0.002093 + 0.002374 + 0.002219 + 0.002395 + 0.002135 + 0.001848 + 0.002194 + 0.001881 + 0.002038 + 0.002303 + 0.001832 + 0.001989 + 0.002145 + 0.002137 + 0.001789 + 0.001811 + 0.001875 + 0.002164 + 0.002112 + 0.001798 + 0.002043 + 0.001873 + 0.002388 + 0.001845 + 0.001899 + 0.002278 + 0.002387 + 0.002346\Big) = 0.00203061$$

Dinámica - Ejemplo 2

Prueba 20: Dinámica

$$\frac{1}{50}\Big(0.007428 + 0.000601 + 0.000329 + 0.000579 + 0.006165 + 0.005965 + 0.003109 + 0.005101 + 0.004971 + 0.001524 + 0.006306 + 0.005563 + 0.001037 + 0.002778 + 0.002454 + 0.004593 + 0.001661 + 0.004133 + 0.004218 + 0.000654 + 0.000549 + 0.002623 + 0.000901 + 0.003967 + 0.006660 + 0.004246 + 0.002706 + 0.003733 + 0.007251 + 0.006143 + 0.003969 + 0.006554 + 0.002671 + 0.004825 + 0.002568 + 0.006762 + 0.001920 + 0.000767 + 0.005480 + 0.005207 + 0.006040 + 0.004633 + 0.001566 + 0.000884 + 0.000938 + 0.003838 + 0.001217 + 0.006928 + 0.000899 + 0.004912\Big) = 0.00361046$$

Voraz - Ejemplo 2

Prueba 21: Voraz

$$\frac{1}{50}\Big(0.000102 + 0.000013 + 0.000012 + 0.000016 + 0.000467 + 0.009573 + 0.000624 + 0.001422 + 0.000281 + 0.002656 + 0.006874 + 0.008288 + 0.005459 + 0.005791 + 0.009032 + 0.000023 + 0.003546 + 0.003690 + 0.008839 + 0.007101 + 0.002399 + 0.005556 + 0.003761 + 0.007921 + 0.009622 + 0.000551 + 0.001241 + 0.007830 + 0.005723 + 0.004482 + 0.007689 + 0.000799 + 0.005366 + 0.006481 + 0.009909 + 0.002801 + 0.005917 + 0.000162 + 0.003251 + 0.005330 + 0.004796 + 0.004246 + 0.003354 + 0.003075 + 0.005787 + 0.000273 + 0.003708 + 0.003314 + 0.009543 + 0.006192\Big) = 0.004298$$

Fuerza Bruta - Ejemplo 3

Prueba 22: Fuerza bruta

$$\frac{1}{50}\Big(2.746607 + 2.848173 + 2.853504 + 2.926486 + 2.860449 + 2.818011 + 2.862820 + 2.896826 + 2.808390 + 2.792628 + 2.809273 + 2.797805 + 2.853475 + 2.832594 + 2.752757 + 2.749288 + 2.830371 + 2.841684 + 2.916348 + 2.882204 + 2.775226 + 2.752107 + 2.800158 + 2.823750 + 2.846143 + 2.818153 + 2.854298 + 2.787204 + 2.883960 + 2.830201 + 2.850263 + 2.883035 + 2.762389 + 2.756933 + 2.866527 + 2.926315 + 2.855802 + 2.807210 + 2.872519 + 2.845769 + 2.778468 + 2.765813 + 2.798433 + 2.798105 + 2.847080 + 2.783921 + 2.769688 + 2.792650 + 2.925426 + 2.747094\Big) = 2.825687$$

Dinámica - Ejemplo 3

Prueba 23: Dinámica

$$\frac{1}{50}\Big(0.009779 + 0.004038 + 0.005790 + 0.006618 + 0.009672 + 0.004936 + 0.009651 + 0.009555 + 0.005716 + 0.009422 + 0.005311 + 0.007748 + 0.004721 + 0.006271 + 0.005131 + 0.006260 + 0.007432 + 0.006591 + 0.008373 + 0.005986 + 0.004553 + 0.005910 + 0.008847 + 0.004107 + 0.009126 + 0.007485 + 0.009209 + 0.005609 + 0.008507 + 0.006446 + 0.009704 + 0.005434 + 0.006728 + 0.008782 + 0.005678 + 0.005764 + 0.005644 + 0.008300 + 0.006968 + 0.004368 + 0.005962 + 0.006640 + 0.005333 + 0.006024 + 0.006147 + 0.007050 + 0.009676 + 0.007741 + 0.004591 + 0.006027\Big) = 0.006827$$

Voraz - Ejemplo 3

Prueba 24: Voraz

$$\frac{1}{50}\Big(0.000018 + 0.000030 + 0.000018 + 0.000011 + 0.000020 + 0.000017 + 0.000026 + 0.000018 + 0.000013 + 0.000025 + 0.000022 + 0.000029 + 0.000014 + 0.000024 + 0.000028 + 0.000017 + 0.000016 + 0.000018 + 0.000014 + 0.000027 + 0.000017 + 0.000015 + 0.000018 + 0.000030 + 0.000028 + 0.000020 + 0.000014 + 0.000019 + 0.000012 + 0.000027 + 0.000030 + 0.000025 + 0.000014 + 0.000026 + 0.000016 + 0.000025 + 0.000028 + 0.000014 + 0.000025 + 0.000029 + 0.000016 + 0.000028 + 0.000027 + 0.000028 + 0.000013 + 0.000027 + 0.000028 + 0.000021 + 0.000024 + 0.000022\Big) = 0.000021$$

Fuerza Bruta - Ejemplo 4

Prueba 25: Fuerza bruta

$$\frac{1}{50}\Big(13.142388 + 13.592095 + 14.634143 + 17.075916 + 16.672294 + 16.643829 + 13.911156 + 15.217320 + 14.093830 + 16.968732 + 15.529913 + 16.568932 + 13.973110 + 14.977807 + 14.121907 + 15.200912 + 15.454214 + 16.946603 + 14.146683 + 13.283542 + 13.744159 + 16.195193 + 16.763096 + 14.610352 + 14.256513 + 14.695052 + 14.155859 + 14.927457 + 16.306474 + 14.657673 + 16.888096 + 17.048387 + 14.788184 + 15.882408 + 13.446409 + 14.163529 + 16.323868 + 14.237896 + 15.747878 + 17.013788 + 13.481856 + 13.361661 + 15.673610 + 15.564013 + 13.264108 + 16.238137 + 14.713028 + 15.053480 + 13.249163 + 16.580817\Big) = 15.103749$$

Dinámica - Ejemplo 4

Prueba 26: Dinámica

$$\frac{1}{50}\Big(0.017270 + 0.018276 + 0.018873 + 0.014013 + 0.017378 + 0.018225 + 0.017937 + 0.017140 + 0.015770 + 0.018693 + 0.017612 + 0.015595 + 0.015697 + 0.016824 + 0.016797 + 0.017510 + 0.016416 + 0.014599 + 0.014066 + 0.018684 + 0.017980 + 0.014105 + 0.018763 + 0.014039 + 0.014328 + 0.018050 + 0.016947 + 0.018725 + 0.017927 + 0.015593 + 0.018602 + 0.017177 + 0.016687 + 0.014241 + 0.016061 + 0.018662 + 0.016676 + 0.014493 + 0.018608 + 0.016634 + 0.014865 + 0.018268 + 0.015140 + 0.016110 + 0.017453 + 0.014803 + 0.015687 + 0.017478 + 0.018850 + 0.016178\Big) = 0.016730$$

Voraz - Ejemplo 4

Prueba 27: Voraz

$$\frac{1}{50}\Big(0.000034 + 0.000013 + 0.000020 + 0.000017 + 0.000031 + 0.000014 + 0.000018 + 0.000030 + 0.000022 + 0.000025 + 0.000016 + 0.000029 + 0.000030 + 0.000026 + 0.000024 + 0.000018 + 0.000022 + 0.000015 + 0.000017 + 0.000025 + 0.000023 + 0.000028 + 0.000021 + 0.000013 + 0.000027 + 0.000024 + 0.000018 + 0.000026 + 0.000014 + 0.000029 + 0.000017 + 0.000021 + 0.000019 + 0.000013 + 0.000031 + 0.000018 + 0.000030 + 0.000027 + 0.000025 + 0.000020 + 0.000014 + 0.000021 + 0.000017 + 0.000029 + 0.000025 + 0.000016 + 0.000028 + 0.000015 + 0.000024 + 0.000031\Big) = 0.000022$$

Fuerza Bruta - Ejemplo 5

Prueba 28: Fuerza bruta

$$\frac{1}{50}\Big(0.001327 + 0.001622 + 0.000753 + 0.000701 + 0.001427 + 0.000968 + 0.001561 + 0.000984 + 0.000720 + 0.001150 + 0.000885 + 0.000736 + 0.001598 + 0.001143 + 0.000926 + 0.001087 + 0.000748 + 0.001217 + 0.000874 + 0.001235 + 0.000854 + 0.000761 + 0.001033 + 0.001491 + 0.001015 + 0.000998 + 0.001498 + 0.001223 + 0.001342 + 0.000992 + 0.000814 + 0.001254 + 0.001073 + 0.000915 + 0.001029 + 0.001488 + 0.000958 + 0.000874 + 0.000811 + 0.001149 + 0.001347 + 0.001238 + 0.000887 + 0.001040 + 0.001472 + 0.001369 + 0.001065 + 0.000995 + 0.000806 + 0.000969\Big) = 0.001030$$

Dinámica - Ejemplo 5

Prueba 29: Dinámica

$$\frac{1}{50}\Big(0.002158 + 0.002748 + 0.003404 + 0.001952 + 0.002645 + 0.002396 + 0.003228 + 0.002314 + 0.003007 + 0.002792 + 0.002536 + 0.003110 + 0.002073 + 0.002802 + 0.003319 + 0.002481 + 0.002641 + 0.002906 + 0.002112 + 0.002837 + 0.002374 + 0.003029 + 0.002536 + 0.002921 + 0.002817 + 0.002455 + 0.002734 + 0.002659 + 0.002123 + 0.002490 + 0.002611 + 0.002911 + 0.003327 + 0.002108 + 0.003073 + 0.003267 + 0.002184 + 0.002420 + 0.002924 + 0.002501 + 0.002361 + 0.003011 + 0.002319 + 0.002747 + 0.003025 + 0.002485 + 0.003010 + 0.003115 + 0.003279 + 0.002553\Big) = 0.002604$$

Voraz - Ejemplo 5

Prueba 30: Voraz

$$\frac{1}{50}\Big(0.000027 + 0.000025 + 0.000018 + 0.000022 + 0.000028 + 0.000017 + 0.000031 + 0.000019 + 0.000013 + 0.000020 + 0.000024 + 0.000015 + 0.000026 + 0.000029 + 0.000016 + 0.000030 + 0.000014 + 0.000027 + 0.000021 + 0.000022 + 0.000017 + 0.000030 + 0.000020 + 0.000025 + 0.000014 + 0.000018 + 0.000019 + 0.000028 + 0.000022 + 0.000026 + 0.000018 + 0.000014 + 0.000023 + 0.000025 + 0.000029 + 0.000027 + 0.000021 + 0.000016 + 0.000031 + 0.000015 + 0.000029 + 0.000024 + 0.000028 + 0.000021 + 0.000014 + 0.000031 + 0.000019 + 0.000030 + 0.000023 + 0.000018\Big) = 0.000021$$

1 3. Conclusiones

El análisis comparativo de los métodos de resolución de problemas (Fuerza Bruta, Voraz, y Programación Dinámica) permitió evaluar su desempeño en términos de eficiencia temporal para diversas pruebas representativas. A continuación, se detallan las observaciones clave:

Fuerza bruta

- Características: Este método evalúa todas las combinaciones posibles para encontrar la solución óptima, lo que asegura un resultado correcto, pero a un alto costo computacional.
- El tiempo promedio por prueba es significativamente mayor que los otros métodos, especialmente en instancias grandes, donde la complejidad factorial impacta severamente.
- En el caso de la prueba 4, el tiempo promedio fue de aproximadamente 0.46076 segundos, siendo el más elevado entre los métodos analizados.
- Aunque garantiza la solución óptima, su uso es inviable para problemas de gran escala debido al elevado tiempo de cómputo.

Dinámica

- Logró un balance entre tiempo de ejecución y calidad de la solución. Por ejemplo, en la prueba 4, tuvo un tiempo promedio de 0.000704 segundos, considerablemente menor que Fuerza Bruta, pero mayor que la solución voraz.
- En problemas donde la solución óptima puede descomponerse en subproblemas, ofrece resultados cercanos a la óptimalidad con un costo computacional razonable.
- Es una estrategia eficiente para problemas estructurados, siendo más rápida que Fuerza Bruta y con soluciones generalmente mejores que la voraz.

Voraz

- Fue consistentemente el método más rápido, con tiempos promedio en el rango de microsegundos (e.g., 0.0000072 segundos en la prueba 4).
- Sin embargo, puede no encontrar soluciones óptimas en problemas donde la decisión local no conduce al óptimo global.
- Es ideal para aplicaciones donde la rapidez es prioritaria y una solución aproximada es aceptable.

La programación voraz es ideal cuando se prioriza la velocidad sobre la precisión, especialmente en aplicaciones en tiempo real o con recursos computacionales limitados; la Programación Dinámica es recomendable para problemas estructurados y de tamaño moderado, logrando un equilibrio entre eficiencia y calidad de la solución; mientras que la Fuerza Bruta, aunque garantiza resultados óptimos, es adecuada únicamente para problemas pequeños donde la precisión sea crucial.