Actividad 4:
Algoritmos de fuerza Bruta

Diego Javier Mora Cortés

11 de Noviembre 2022

CUCEI - INGENIERÍA INFORMÁTICA

Algoritmia

Profesor: David Alejandro Gómez Anaya

Índice

1. Algoritmo 1

Tres equipos de investigación tratan de resolver un mismo problema de forma independiente. Las probabilidades de fracasar son 0.4,0.6 y 0.8. Se desea minimizar la probabilidad de fracaso y se dispone de dos científicos más para reforzar los equipos. Para determinar a qué equipo asignarlos se elabora la tabla siguiente, que da la probabilidad de fracaso de cada equipo cuando es reforzado con 0,1 y 2 científicos. Diseñe un algoritmo de fuerza bruta que encuentra cuantos científicos debe tener cada equipo para minimizar la probabilidad de error total. El algoritmo recibe de entrada una matriz de 3 x 3 que describe las probabilidades de fracaso de los equipos con: 0, 1, 2 refuerzos y devuelve un vector de 3 espacios con el número de refuerzos que debe tener cada equipo (en este vector, la posición 0 representa al equipo 1, la posición 1 representa al equipo 2 y la posición 2 representa al equipo 3).

Entrada: $a[3][3]$
Salida: $indMin[3]$

Algorithm 1

```
1: min ← +∞
2: indMin[3] = {0, 0, 0}
3: for i ← 0, i < 3 , i++ do
4:    for j ← i, j < 3 , j++ do
5:       ref[3] ← {0, 0, 0}
6:       ref[i]++
7:       ref[j]++
8:       if a[ref[0]][0] + a[ref[1]][1] + a[ref[2]][2] < min then
9:          indMin[0] = ref[0]
10:         indMin[1] = ref[1]
11:         indMin[2] = ref[2]
12:         min = a[ref[0]][0] + a[ref[1]][1] + a[ref[2]][2]
13:      end if
14:   end for
15: end for
16: return indMin[]
```

2. Algoritmo 2

Utilizando únicamente operaciones básicas (+,-,* y /), diseñe un algoritmo de fuerza bruta que obtenga el valor aproximado a cuatro espacios decimales de raíces cuadradas de una entrada x. Ejemplos de entradas: $\sqrt{7}$, $\sqrt{26}$ y $\sqrt{50}$.
Entrada: $n \in \mathbb{R}^+$
Salida: $\approx \sqrt{n}$

Algorithm 2

```
1: q ← 0
2: while q^2 ≤ n do
3:    q ← q + 0,0001
4: end while
5: return q − 0,0001 {devuelve un aproximado menor o igual a la raiz}
```

3. Algoritmo 3

Escriba un algoritmo que dado un conjunto de puntos en el plano(x1 , y1),...,(xn , yn), encuentre la pareja de puntos más cercana entre sí. El algoritmo recibe de entrada un vector de coordenadas (x,y) representado por un arreglo bidimensional, y devuelve los índices de las coordenadas más cercanas del conjunto. PairIndex[2] ClosestPairPoints(P[2][n]).

Entrada: cordenadas c[n]
Salida: cordenadas minCor[2]

Algorithm 3

1: $min \leftarrow +\infty$
2: minCor[2]
3: **for** $i \leftarrow 0,\ i < n\ , i++$ **do**
4: **for** $j \leftarrow i+1,\ j < n\ , j++$ **do**
5: **if** $\left|c[i].x - c[j].x\right| + \left|c[i].y - c[j].y\right| < min$ **then**
6: $minCor[0] = c[i]$
7: $minCor[1] = c[j]$
8: $min = \left|c[i].x - c[j].x\right| + \left|c[i].y - c[j].y\right|$
9: **end if**
10: **end for**
11: **end for**
12: **return** $minCor[]$

4. Algoritmo 4

Diseñe un algoritmo de fuerza bruta que, dada una cadena de caracteres, imprima la subcadena más grande que cumple con ser palíndromo.
Entrada: $a[n]$
Salida: imprime subcadena mas grande que es palindromo

Algorithm 4

```
1: s ← 0 {Indice inicio de la subcadena}
2: f ← 0 {Indice final de la subcadena}
3: len ← 1
4: for i ← 0, i < n ,i++ do
5:    for j ← i + 1, j < n ,j++ do
6:       subLen ← j − i + 1
7:       if len < subLen then
8:          mid ← subLen/2 {truncates, mid-1 es el último indice de la primera mitad de la subcadena}
9:          cont ← 0
10:         while cont<mid do
11:            pila.enqueue(a[i+cont++])
12:         end while
13:         mid ← mid + subLen mod 2 {primer indice de la segunda mitad}
14:         esPalindromo = true
15:         cont = 0
16:         while esPalindromo and not pila.empty() do
17:            if not pila.dequeue() == a[mid + cont++] then
18:               esPalindromo = false
19:            end if
20:         end while
21:         if esPalindromo then
22:            s ← i
23:            f ← j
24:            len ← subLen
25:         end if
26:      end if
27:   end for
28: end for
29: imprime a[i:j+1] {Imprime la subcadena con indices i hasta j, como en Python }
```