Tarea 3. Análisis y diseño de Algoritmos.
Técnica de diseño: Divide y vencerás 3.
Fecha de entrega: 6 de octubre

28 de septiembre de 2022

1. Tarea

1. Sea S un conjunto de n puntos $S = \{(x_i, y_i)\}$. Suponemos que no hay dos puntos de S con coordenadas x o y iguales. Un punto q es *dominado* por otro punto p si ambas coordenadas de p son mayores: $p.x > q.x$ y $p.y > q.y$. Devuelve todos los puntos que no son dominados por nadie más en ese conjunto.

   a) Diseña un algoritmo que solucione el problema en $O(n \log(n))$.

   b) Demuestra que es correcto.

   c) Demuestra que tiene la complejidad solicitada.

   Tip: Considera dividir la instancia recursiva en dos subinstancias, con la misma idea que en el problema de la pareja de puntos más cercana. Además, usa también la idea de recibir como parámetros el conjunto de puntos ordenados por x y aparte los mismos puntos ordenados por y; de esa forma no tendrás que hacer ordenamientos en el cuerpo del algoritmo, y podrás mantenerlo con complejidad interna $O(n)$ para poder alcanzar la complejidad solicitada.

2. Has estado trabajando con algunos físicos que necesitan estudiar, como parte de su diseño experimental, las interacciones sobre un gran cantidad de partículas cargadas muy pequeñas.

   Básicamente, la ejecución funciona de la siguiente manera. Ellos tienen una estructura de rejilla inerte, y la utilizan para colocar las partículas cargadas a una distancia regular a lo largo de una línea recta. Así, podemos modelar su estructura como si consistiera en los puntos $\{0, 2, 3, \ldots, n-1\}$ en la línea recta; y en cada uno de esos puntos j, ellos tienen una partícula cargada q_j. (Cada carga puede ser positiva o negativa).

   Ellos quieren estudiar la fuerza total sobre cada partícula, midiéndola y comparándola con la predicción computacional. En esta parte computacional es donde necesitan tu ayuda. La fuerza neta total en una partícula j, por la ley de Coulomb, es igual a:

$$F_j = \sum_{i<j} \frac{cq_iq_j}{(j-i)^2} - \sum_{i>j} \frac{cq_iq_j}{(j-i)^2}$$

Ellos han diseñado un algoritmo para computar F_j para toda j, tal que $1 \leq j \leq n$, de la siguiente forma:

---

función $F_jIterativo(n)$:
  For $j = 0, 1, \ldots, n$
    Inicialize $F_j = 0$
    For $i = 1, \ldots, n$
      If i < j **then**
        Add $\frac{cq_iq_j}{(j-i)^2}$ a F_j
      Else if i > j **then**
        Add $-\frac{cq_iq_j}{(j-i)^2}$ a F_j
    Output F_j

---

a) Los físicos no están contentos con el tiempo que tarda en ejecutarse el algoritmo cuando el número n de partículas aumenta. La complejidad del algoritmo que diseñaron es de $O(n^2)$. Diseña un algoritmo que solucione este problema con una complejidad de $O(n \log(n))$.

b) Demuestra que tu solución es correcta, es decir, que es una alternativa al algoritmo diseñado por los físicos.

c) Argumenta la complejidad de tu solución.

3. Dada una secuencia de n caracteres $A = a_0, \ldots, a_{n-1}$, con $a_i \in \{a, b\}$ y un patrón $P = p_0, \ldots, p_{m-1}$ con $m < n$, y $p_i \in \{a, \_\}$. Devuelve todos los índices i para los cuales el patrón P, teniendo su primer caracter alineado con el caracter de índice i de A, tiene la propiedad de que todo caracter 'a' de P, está alineado con un caracter 'a' de A, los caracteres '_' de P pueden estar alineados con cualquier caracter de A.

   a) Describe un algoritmo de complejidad $O(n \log n)$ que resuelve el problema.

   b) Demuestra la corrección de tu algoritmo.

   c) Demuestra que tiene la complejidad solicitada.

4. Considera dos collares cada uno con n perlas, teniendo cada perla color rojo o azul; y representadas por las secuencias circulares $P = p_0, \ldots, p_{n-1}$ y $Q = q_0, \ldots, q_{n-1}$, cuyos

elementos son el caracter 'r', representando una perla roja, y 'b', representando una perla azul.

Para cada i, con $0 \leq i \leq n-1$ considera que Q rota i posiciones en sentido contrario al reloj, y sea c_i el número índices en los que las correspondientes parejas de perlas (una de P y una de Q en las mismas posiciones relativas), tienen el mismo color.

a) Describe un algoritmo de complejidad $O(n \log n)$ que devuelva todos los valores de i (posiciones de rotación de Q), para las cuales el valor c_i es máximo posible.

b) Demuestra la corrección de tu algoritmo.

c) Demuestra que tiene la complejidad solicitada.