Montaño Pérez Joshua Said

Analisis de Algoritmos
Tarea06 Ordenamiento I

Fecha de entrega:
03/11/2022

1. El Problema de Selección consiste en encontrar el $k-ésimo$ elemento más pequeño de un conjunto de n datos, $k \leq n$. Considere los algoritmos de ordenamiento:
   (a) Bubble Sort;
   Respuesta
   No se puede, ya que en el peor de los casos el elemento más pequeño puede que se encuentre al final y no seria necesario ordenar el arreglo ó lista, ya que el menor de los elementos estaría al principio.
   (b) Insertion Sort;
   Respuesta
   Si se puede, ya que tenemos un arreglo ó lista ordenada de la cuál solo nos importa un segmento de esta y luego tener otra lista o arreglo en desorden, en la cual solo tendríamos que comparar el fragmento del arreglo ó la lista que llegue a tener el $k-esimo$ elemento que estamos buscando y a su vez devolviéndolo.
   (c) Selection Sort;
   Respuesta
   Depende, ya que dependiendo si el arreglo ó lista empiece a iterar en cero, recordando que en este aquí utilizamos un índice auxiliar, el cuál nos ayuda a saber donde va el elemento más pequeño, es decir, se pone al principio de la lista ó arreglo y va incrementando hasta llegara los n elementos, pero en el caso de que $k < n$ y este se repetiría hasta que $k \leq$indice pero esto depende si no se cumple la condición del que el arreglo ó lista empiece a iterar en cero
   (d) Shell Sort;
   Respuesta
   No se puede, ya que cuando dividimos dependemos siempre del estado inicial del arreglo o lista, es decir podemos inicializar le tamaño de la lista ó arreglo y seguir la secuencia de los $k-elementos$ pero esto no garantizaría que sea el $k-esimo$ elemento de la lista ó arreglo original.
   (e) Local Insertion Sort.
   Respuesta
   No se puede, ya que el algoritmo crea una bilista auxiliar y esta se va construyendo conforme se va interando la lista desde el principio y con esto el $k-esimo$ elemento de la lista ordenada podría ser el inicial y tomar su lugar en la bilista al final de esta.
   Pregunta: ¿Cuáles de las estrategias usadas por los algoritmos anteriores, nos ayudan a resolver el Problema de Selección **sin** tener que ordenar toda la secuencia? Suponga k tal que $1 < k < n$. Justifique, para cada inciso, sus respuestas.

2. Problema Φ: Suponga que tiene n intervalos cerrados sobre la recta real: $[a(i), b(i)]$, con $1 \leq i \leq n$. Encontrar la máxima k tal que existe un punto x que es cubierto por los k intervalos.
   (a) Proporcione un algoritmo que solucione el problema Φ.
   Respuesta
   La estrategia del algoritmo es encontrar cuantas veles se intersectan los intervalos, tomando una lista ordenada donde los intervalos $v = (x, y)$, donde $x, y \in l$, además de que tambien necesitamos de del principio y fin del intervalo, es decir, $v = (x, y)$ con $v = (x, inicio)$ y $v = (y, final)$, con esto podremos identificar cual es el valor máximo entre los intervalos que se intersectan.
   (b) Justifique que su propuesta de algoritmo es correcta.
   Respuesta

```
1
2  //Precondiciones: Una secuencia V con los intervalos (x,y)
3  //Ordenar un algoritmo de complejidad O(nlogn) a l, donde el criterio comparable
4  //es primero x y después p , donde inicio < final , en (x,p) El
5  //Postcondiciones : k es el maximo números de intervalos.
6
7  Transformamos cada elemento v=(x,y) in V en v ( (x,inicio),(y,final)) y se guardan en V
8  Por cada elemento de v en V , guardamos v,w en L donde v=(u,w).
9
10     Definimos k y contador = 0 .
11         while cada elemento (x.p) en l:
12             if p = inicio , contador incrementa en 1 , en otro caso
13                 if  p = fin , contador decrementa en 1
14                     K = max(k , contador)
15         Return k
16
17
```

(c) Calcule, con detalle, la complejidad computacional de su propuesta

Respuesta

El algoritmo propuesto ordena en $O(logn)$, ya que la gran mayoria del tiempo hace opreaciones elementales, como en el contador y k, además de que estas opereaciones cuestan $O(n)$ por lo que la complejidad no aumenta.

(d) Proporcione un pseudo-código del algoritmo propuesto.

Respuesta


```
1
2  //Precondiciones: Una secuencia V con los intervalos (x,y)
3  //Ordenar un algoritmo de complejidad O(nlogn) a l, donde el criterio comparable
4  //es primero x y después p , donde inicio < final , en (x,p) El
5  //Postcondiciones : k es el maximo números de intervalos.
6
7  Transformamos cada elemento v=(x,y) in V en v ( (x,inicio),(y,final)) y se guardan en V
8  Por cada elemento de v en V , guardamos v,w en L donde v=(u,w).
9
10     Definimos k y contador = 0 .
11         while cada elemento (x.p) en l:
12             if p = inicio , contador incrementa en 1 , en otro caso
13                 if  p = fin , contador decrementa en 1
14                     K = max(k , contador)
15         Return k
16
17
```


3. Realice la siguiente modificación al algoritmo Insertion Sort: Para buscar la posición del nuevo elemento, el que está en revisión, usar Busqueda Binaria, en vez de hacerlo secuencialmente.
   a) Determine el desempeño computacional del algoritmo modificado.
   Respuesta
   Como ya no hacemos una búsqueda lineal, si no ahora hacemos una búsqueda binaria, hacemos menos comparaciones para insertar un elemento y pasamos de $O(n)$ a $O(logn)$. Además como el algoritmo solo necesita de una cantidad constante de espacio, solo tendríamos q desplazar e insertar el elemento deseado y este tendrá una complejidad de $O(logN)$.
   b) ¿Mejora el desempeño computacional del proceso total? Justifique.
   Respuesta
   Si, por que ya no hacemos tantas comparaciones innecesarias que hacemos de la forma lineal y este es muy eficiente a la hora de hacer muchas comparaciones debido a que es más "rápidoëncontrarlos, sin dejar de lado que la complejidad es mucho mejor a la hora de usar este algoritmo.

4. Indique si lo siguientes algoritmos de ordenamiento son estables o no y justifique por qué:

(a) Bubble Sort;
Respuesta
Es estable, ya que intercambiamos los elementos cuando estos son menores que el siguiente elemento y si estos son iguales no les hacemos nada.

(b)Insertion Sort;
Respuesta
Es estable, ya que vamos creando la lista ordenadamente, mientras que la otra lista esta desordenada, pero por la construcción de la primera es mayor los elementos de la lista ordenada que la otra.

(c) Selection Sort;
Respuesta
Es estable, ya que vamos iterando con el indicie auxiliar y vamos del elemento más pequeño de todos los elementos, así hasta terminar y construirlo.

(d) Shell Sort;
Respuesta
No es estable, ya que cuando dividimos la secuencia al mismo orden de la subsecuencia, una de las dos puede que estar ordenada, además que una de las subsecuencias inicial tenga un elemento menor al de la otra subsecuencia.

(e) Local Insertion Sort.
Respuesta
Es estable, ya que cuando vamos construyendo la lista que es bidireccional los elementos se van acomodando donde deben ir, esto mientras iteremos la lista.

5. **Opcional** Problema Γ: Dados n puntos en el plano, encontrar un polígono que tenga como vértices los n puntos dados.
   (a) Proporcione un algoritmo que solucione el problema Γ.
   (b) Justifique que su propuesta de algoritmo es correcta.
   (c) Calcule, con detalle, la complejidad computacional de su propuesta
   (d) Proporcione un pseudo-código del algoritmo propuesto.
   Respuesta