Comenzado el	domingo, 24 de septiembre de 2023, 21:09
Estado	Finalizado
Finalizado en	domingo, 24 de septiembre de 2023, 21:11
Tiempo empleado	1 minutos 38 segundos
Puntos	17/17
Calificación	**10** de 10 (**100**%)

Pregunta **1**

Correcta

Se puntúa 2 sobre 2

En esta consigna se propone el desarrollo y prueba de un programa para resolver el conocido problema del *Conteo de Frecuencias*: se tiene un conjunto de números de forma que cada número puede venir repetido varias veces, y se desea un programa que cuente cuántas veces apareció cada número (en este desafío, el conjunto de entrada estará dado por un arreglo de números que los alumnos deberán crear con funciones provistas por la Cátedra). Aunque no se sabe cuántos números diferentes vendrán, es posible que se conozca la cantidad total de números *n* que entrarán, y en algunas ocasiones se sabe también si esos números están en algún rango conocido.

En esta primera consigna, suponga que sabemos la cantidad total *n* de números que tiene el arreglo de entrada, pero no sabemos en qué rango vienen esos números (por ejemplo, podemos saber que vendrán *n = 1000* números, pero no sabemos si esos números están entre 1 y 1000 o entre 1 y 100000... o en ningún intervalo conocido de antemano). Para esta primera consigna, *el problema consiste en determinar cuántos números diferentes entraron* (aunque saber cuántas veces entró cada uno también será necesario para las consignas 2 y 3).

En general, es obvio que para llevar esa cuenta necesitamos un contador por cada *número diferente* que se detecte (no sabemos cuántos números diferentes vendrán, pero sí que cada uno que venga puede entrar varias veces) ¿Cuántos contadores necesitaremos, si no sabemos la cantidad de números distintos? Un rápido análisis sugiere que podrían usarse dos arreglos unidimensionales paralelos *num* y *cont*, de forma que en cada componente *num[i]* se guarde el *número* que nos interesa controlar y en el componente *cont[i]* se proceda a *contar* cuántas veces fue visto el número guardado en *num[i]*. El algoritmo general podría ser el siguiente:

```
0.)  Cree los dos arreglos num y cont inicialmente vacíos (sin casilleros).
1.)  Para cada número x del arreglo de entrada:
      1.1) Buscar x en el arreglo num y determinar el índice i de la casilla donde está
almacenado.
      1.2) ¿Existe x en el arreglo num (o sea: el valor de i es diferente de -1)?
          Sí:  Sumar 1 al casillero cont[i].
          No:  Agregar al final del arreglo num el número x, y agregar un 1 al final del arreglo
cont.
2.)  Mostrar el tamaño final del arreglo num (para saber cuántos numeros diferentes aparecieron).
```

Para cumplir con esta primera consigna, se provee un módulo *soporte.py* (que puede descargar haciendo click en el nombre del módulo o también haciendo click *aquí*) ya programado por los docentes de la Cátedra, conteniendo un par de funciones que los estudiantes obligatoriamente deberán usar para generar los arreglos que servirán como datos para las consignas de este desafío.

Su tarea:

a.) Cree un proyecto nuevo en PyCharm para desarrollar los programas que necesitará en este desafío.

b.) Descargue y descomprima el archivo *soporte.py* provisto por la Cátedra, e inclúyalo en el proyecto que acaba de crear.

c.) Cree un programa dentro del proyecto, e importe el módulo *soporte.py* en ese programa.

d.) Su programa debe usar la función *soporte.vector_unknown_range()* para crear y obtener el arreglo que debe procesar. El arreglo debe generarse con *n = 300000* (trescientos mil) elementos, y la instrucción que le permitiría hacerlo es simplemente la siguiente:

```
v = soporte.vector_unknown_range(300000)
```

e.) La instrucción anterior, asignará en v una referencia a un arreglo con *n = 300000* números mayores o iguales a 0, que pueden venir repetidos, pero de forma que se desconoce en qué intervalo o rango están. La invocación *v = soporte.vector_unknown_range(300000)* creará exactamente el mismo arreglo cada vez que la ejecute, por lo que no debe preocuparse: sus datos serán siempre los mismos, siempre y cuando el parámetro sea 300000. Con otros valores de ese parámetro, el arreglo generado será diferente: asegúrese de invocar a la función pasando el valor 300000 como parámetro.

f.) En el mismo programa, implemente el algoritmo de conteo descripto en el pseudocódigo de más arriba, usando un arreglo de registros de conteo, y obtenga como primera medida, *la cantidad de números diferentes que detecte en el arreglo v de entrada*.

Registre ahora (en el casillero que sigue) la cantidad de números diferentes del arreglo de entrada provisto:

Respuesta: 548 ✔

Pregunta **2**

Correcta

Se puntúa 3 sobre 3

En esta segunda consigna, se pide programar un algoritmo que calcule el *valor modal* o *moda* del mismo vector de números que tuvo que procesar para el problema del conteo de frecuencias.

El enunciado del problema es simple: dada una *colección lineal v*, el *valor modal* o *moda* de esa colección es el valor *x* que pertenece a *v* que se *repite con mayor frecuencia*. Si dos o más valores ***diferentes*** *se repiten con la misma frecuencia máxima*, entonces se dice que esa colección *no tiene valor modal*. Por ejemplo, para el conjunto {3, 5, 3, 5, 3, 6, 3, 8} el valor modal es 3, ya que es el que más aparece, con una frecuencia de 4. Pero para el conjunto {3, 5, 3, 5, 3, 5, 6, 8} *no hay valor modal*, ya que hay más de un número (el 3 y el 5) con la misma frecuencia máxima de aparición (3 veces cada uno).

Su trabajo es tomar los arreglos *num* y *cont* que generó para la consigna anterior, y en base a ellos *determinar el valor modal del vector v original*. No debería tener que volver a generar el vector en un programa separado, sino simplemente modificar su programa anterior para agregar ahora esta funcionalidad. Tenga en cuenta además, que en la consigna 3 se le pedirá que informe también la *frecuencia* del valor modal, así que puede ganar tiempo e ir calculando ambos valores en la misma corrida.

Existen diversas estrategias algorítmicas que pueden plantearse para resolver este problema, pero confiamos en que los alumnos sabrán discutir ventajas y desventajas y finalmente aportar una solución eficiente en cuanto al tiempo de ejecución. Considere que es *perfectamente posible* obtener el valor modal desde los arreglos *num* y *cont*, *con una sola pasada a través de ellos.*

Si al calcular el valor modal descubre que el mismo no existe (por ejemplo, por tener más de un número con la misma frecuencia máxima), entonces *informe un 0(cero)* cuando se le pida el valor modal.

Registre ahora (en el casillero que sigue) el valor modal del vector v de entrada provisto (o un 0(cero) si no existe valor modal):

Respuesta: 547 ✔

Pregunta **3**

Correcta

Se puntúa 3 sobre 3

Para el valor modal que calculó en la consigna 2 anterior, se pide simplemente que informe su frecuencia de aparición. Si al calcular el valor modal descubre que el mismo no existe (por ejemplo, por tener más de un número con la misma frecuencia máxima), entonces *informe un 0(cero)* cuando se le pida la frecuencia de aparición del valor modal.

Registre ahora (en el casillero que sigue) la frecuencia de aparición del valor modal del vector de entrada provisto (o un 0(cero) si no existe valor modal):

Respuesta: 9798 ✔

Pregunta **4**

Correcta

Se puntúa 1 sobre 1

Es muy posible que el programa que acaba de ejecutar para el cálculo de las frecuencias de aparición de cada valor en un conjunto de *n* números que vienen en un rango desconocido haya demorado bastante tiempo y es de esperar que ese tiempo aumente a medida que a su vez *n* sea mayor. La *operación crítica* (esto es: la operación que más tiempo insume) en este algoritmo es la búsqueda de cada número x en el vector usado para almacenar los números que aparecieron: cada uno de los *n* números debe buscarse secuencialmente en ese vector, y el vector presenta (en el peor caso) *k* números en cada búsqueda (con $k <= n$). ¿Cuál de las siguientes estimaciones describe mejor la cantidad total de comparaciones en el peor caso, que se hacen para buscar los *n* números y terminar el proceso?

Seleccione una:

a. En el orden de n comparaciones en total [o sea, $O(n)$ comparaciones].

b. En el orden de n^2 comparaciones en total [o sea, $O(n^2)$ comparaciones]. ✔

c. En el orden de n^3 comparaciones en total [o sea, $O(n^3)$ comparaciones].

d. En el orden de $n * log(n)$ comparaciones en total [o sea, $O(n * \log(n))$ comparaciones].

Pregunta **5**

Correcta

Se puntúa 3 sobre 3

En esta consigna se propone resolver el mismo problema del *Conteo de Frecuencias*, pero ahora partiendo de un vector de números *que se sabe que están en un rango o intervalo conocido*, y que además ese intervalo es razonable para ser mapeado en una estructura de datos de memoria.

Si este es el caso (se conoce el rango o intervalo de los valores de entrada) entonces se puede plantear una solución MUCHO más rápida que la que vimos en las consignas 1, 2 y 3: en lugar de usar dos arreglos para almacenar y contar los números que van apareciendo, y tener que buscar secuencialmente cada número *x* en el arreglo de números, podemos usar (ahora sí...) un *arreglo de conteo directo*. La idea es crear un arreglo *c* de tantos elementos como números posibles haya en el rango de entrada (300000 casilleros en nuestro caso), y usar a cada casillero *c[x]* para contar la cantidad de veces que entró el número *x*. Esta solución requiere eventualmente más memoria (ya que podrían quedar casilleros sin usar en el arreglo *c*), pero a cambio es notablemente más veloz (y garantizamos que los alumnos notarán esto cuando comparen los tiempos de ejecución de las dos soluciones...)

El algoritmo básico general podría ser el siguiente, suponiendo que el rango o intervalo de los valores de entrada es [0..299999] (como dijimos) y lo que se pide es determinar cuántos números diferentes entraron:

```
0.)  Cree un arreglo c de 300000 casillas, inicializadas en cero.
1.)  Para cada número x del vector v de entrada:
         1.1) Incremente en uno la casilla c[x]
2.)  Mostrar la cantidad de casilleros diferentes de cero que quedaron en el arreglo c.
```

Para cumplir con las consignas que quedan, debe usar el mismo módulo *soporte.py* que ya se indicó para las consignas anteriores (programado por los docentes de la Cátedra) pero de forma que ahora se use la función que corresponda para generar un arreglo con valores en un intervalo conocido, de acuerdo a los siguientes pasos generales:

a.) En el mismo proyecto que ya ha creado para las consignas anteriores, en el que ya debería estar incluido el módulo *soporte.py*, cree otro programa e importe el módulo *soporte.py* (o añada la nueva funcionalidad al programa que ya desarrolló para las consignas anteriores). Haga lo que haga, sólo asegúrese de importar el módulo *soporte.py* como se indicó.

b.) Su programa debe usar ahora la función ***soporte.vector_known_range()*** (***no se confunda***: ahora se llama ***vector_known_range()*** en lugar de ***vector_unknown_range()***) para crear y obtener el vector *v* que debe procesar. El vector debe generarse con *n = 300000* (trescientos mil) elementos, y la instrucción que le permitiría hacerlo es simplemente la siguiente:

```
v = soporte.vector_known_range(300000)
```

c.) La instrucción anterior, asignará en *v* una referencia a un arreglo con *n = 300000* números mayores o iguales a 0, que pueden venir repetidos, pero de forma que ahora ***todos*** los números del vector están en el intervalo [0..299999]. La invocación *v = soporte.vector_known_range(300000)* creará exactamente el mismo arreglo cada vez que la ejecute, por lo que no debe preocuparse: sus datos serán siempre los mismos, siempre y cuando el parámetro sea 300000. Con otros valores de ese parámetro, el vector generado será diferente: asegúrese de invocar a la función pasando el valor 300000 como parámetro.

d.) En el mismo programa, implemente el algoritmo de conteo descripto en el pseudocódigo presentado más arriba, y obtenga como primera medida, *la cantidad de números diferentes que detecte en el vector v de entrada*.

Registre ahora (en el casillero que sigue) la cantidad de números diferentes del nuevo archivo de entrada provisto:

Respuesta: 211927 ✔

Pregunta **6**

Correcta

Se puntúa 2 sobre 2

En esta quinta consigna, se pide programar un algoritmo que calcule el *valor modal* o *moda* del segundo vector de números que tuvo que procesar para el problema del conteo de frecuencias usando un arreglo de conteo directo.

Su trabajo es tomar el arreglo de conteo directo que generó para la consigna 5 anterior, y en base a ese arreglo *determinar el valor modal del segundo vector original*. No debería tener que volver a generar el vector en un programa separado, sino simplemente modificar su programa anterior para agregar ahora esta funcionalidad. Tenga en cuenta además, que en la consigna 7 se le pedirá que informe también la frecuencia del valor modal, así que puede ganar tiempo e ir calculando ambos valores en la misma corrida.

Existen diversas estrategias algorítmicas que pueden plantearse para resolver este problema, pero confiamos en que los alumnos sabrán discutir ventajas y desventajas y finalmente aportar una solución eficiente en cuanto al tiempo de ejecución. Considere que es *perfectamente posible* obtener el valor modal desde el arreglo de conteo directo, *con una sola pasada a través del mismo*.

Si al calcular el valor modal descubre que el mismo no existe (por ejemplo, por tener más de un número con la misma frecuencia máxima), entonces informe un 0(cero) cuando se le pida el valor modal.

Registre ahora (en el casillero que sigue) el valor modal del segundo vector de entrada provisto (o un 0(cero) si no existe valor modal):

Respuesta: 6 ✔

Pregunta **7**

Correcta

Se puntúa 2 sobre 2

Para el valor modal que calculó en la consigna 6 anterior, se pide simplemente que informe su frecuencia de aparición. Si al calcular el valor modal descubre que el mismo no existe (por ejemplo, por tener más de un número con la misma frecuencia máxima), entonces informe un 0(cero) cuando se le pida a frecuencia de aparición del valor modal.

Registre ahora (en el casillero que sigue) la frecuencia de aparición del valor modal del segundo vector de entrada provisto (o un 0(cero) si no existe valor modal):

Respuesta: 515 ✔

Pregunta **8**

Correcta

Se puntúa 1 sobre 1

Ahora debe haber notado que el programa que acaba de ejecutar para el cálculo de las frecuencias de aparición de cada valor en un conjunto de *n* números con rango conocido demoró ***muy poco*** tiempo en terminar, sobre todo si se compara con el tiempo que demoró el programa cuando el rango era desconocido y cada valor debía buscarse secuencialmente en un vector de registros de conteo. La *operación crítica* (esto es: la operación que más tiempo insume) en este algoritmo es la de *acceder al casillero x en el arreglo de conteo* usado para contar la frecuencia de aparición del número *x*: para cada uno de los *n* números se debe simplemente acceder en forma directa a su casilla de conteo e incrementarla en uno. ¿Cuál de las siguientes estimaciones describe mejor la ***cantidad total*** de accesos a casilleros de conteo en el peor caso, que se hacen para contar los *n* números y terminar el proceso?

Seleccione una:

- a. En el orden de n accesos en total [o sea: $O(n)$ accesos]. ✔
- b. En el orden de n^2 accesos en total [o sea: $O(n^2)$ accesos].
- c. En el orden de n^3 accesos en total [o sea: $O(n^3)$ accesos].
- d. En el orden de $n * log(n)$ accesos en total [o sea: $O(n * log(n))$ accesos].

◄ Cuestionario 19 [Temas: hasta Ficha 19]

Ir a...

Trabajo Práctico 03: Gestión de Cabinas de Peaje [3.0] ►