Análisis y diseño de algoritmos

Tarea 03

Diego Guadalupe Rodríguez Prieto

Professor/Dr. Eduardo A. Rodriguez Tello

1. Implemente en el lenguaje de programación C++ un programa que permita calcular de forma recursiva el promedio de un conjunto de n números enteros (almacenados en un vector). El programa recibe como parámetro desde la línea de comandos el nombre de un archivo de texto que contiene dos líneas. La primera con un número que corresponde al tamaño del vector (n), la segunda con n valores enteros separados por un espacio. Una vez que tenga el programa implementado responda los siguientes puntos:

   a) ¿Cuál es el resultado de llamar a su programa con los siguientes vectores: $\{1, 5, 10, 23, 56, 100\}$, $\{51, 35, 140, 323, 566, 1600, 231, 99, 1, 662\}$? Proporcione capturas de pantalla.

   - Para el vector $\{1, 5, 10, 23, 56, 100\}$: El resultado es **32.5**

```
diegordz08@dgrdz08:~/Descargas/Tarea03-ADA$ ./prog primervector.txt
Promedio: 32.5
```

   - Para el vector $\{51, 35, 140, 323, 566, 1600, 231, 99, 1, 662\}$: El resultado es **370.8**

```
diegordz08@dgrdz08:~/Descargas/Tarea03-ADA$ ./prog segundovector.txt
Promedio: 370.8
```

   b) ¿Cuál es la operación básica de su algoritmo? La suma

   c) Proporcione una expresión matemática (relación de recurrencia), en función del tamaño de la entrada de su algoritmo, que permita calcular cuántas veces se ejecuta la operación básica y resuélvala mediante sustitución hacia atrás detallando los pasos que realice.

   Determinamos el caso base y el caso de recursión

$$M(n) = M(n-1) + 1 \quad \text{para } n > 1$$
$$M(1) = 1 \quad \text{(caso base)}$$

   Con sustitución hacia atrás con $M(n-1)$

$$M(n) = M(n-1) + 1$$
$$M(n-1) = M(n-2) + 1$$

   Sustituyendo en la ecuación original

$$M(n) = (M(n-2) + 1) + 1$$
$$= M(n-2) + 2$$

   Repitiendo el proceso hasta llegar a k iteraciones tenemos:

$$M(n) = M(n-k) + k$$

   Dado que $n - k = 1$, tenemos el caso base

$$M(1) = 1 \quad \text{(caso base)}$$

   Tenemos que $k = n - 1$:

$$M(n) = 1 + (n-1) = n$$

   d) Indique cuál es la clase de eficiencia a la que pertenece su algoritmo.

$$O(n)$$

2. Implementar en el lenguaje de programación C++ un programa que permita calcular de forma recursiva la expresión a^b. El programa recibe como parámetro desde la línea de comandos el valor de a y b (dos enteros positivos). Por ejemplo, si su programa recibiera los valores $a = 3$ y $b = 4$ deberá entregar como resultado 81.
   Una vez que tenga el programa implementado responda los siguientes puntos:

   a ¿Cuál es el resultado de llamar a su programa con los valores $a = 5$ y $b = 8$? Proporcione capturas de pantalla.

```
diegordz08@dgrdz08:~/Descargas/Tarea03-ADA$ ./prog 5 8
5^8 = 390625
```

   b ¿Cuál es la operación básica del algoritmo? La multiplicación

   c Plantee una relación de recurrencia para calcular el número total de veces que la operación básica del algoritmo se ejecuta y resuélvala (por el método que desee) detallando los pasos que realice. Determinamos el caso base y el caso de recursión

$$M(n) = M(n-1) + 1 \quad \text{para } n > 1$$
$$M(0) = 0 \quad \text{(caso base)}$$

   Con sustitución hacia atrás con $M(n-1)$

$$M(n) = M(n-1) + 1$$
$$M(n-1) = M(n-2) + 1$$

   Sustituyendo en la ecuación original

$$M(n) = (M(n-2) + 1) + 1$$
$$= M(n-2) + 2$$

   Repitiendo el proceso hasta llegar a k iteraciones tenemos:

$$M(n) = M(n-k) + k$$

   Dado que $n - k = 0$, tenemos el caso base

$$M(1) = 1 \quad \text{(caso base)}$$

   Tenemos que $k = n$:

$$M(n) = M(0) + n$$
$$= 0 + n = n$$

   d ¿Cuál es la clase de eficiencia a la que pertenece este algoritmo?

$$O(n)$$

3. Encuentre el orden de crecimiento de las siguientes relaciones de recurrencia utilizando el Teorema Maestro:

   a $T(n) = 4T\left(\frac{n}{2}\right) + n^2$ para $n > 1, T(1) = 1$

   Tenemos que ver a nuestra función con la formula

   $$T(n) = aT(n/b) + f(n).$$

   Donde

   - $a = 4$
   - $b = 2$
   - $f(n) = n^2$

   de modo que:

   $$T(n) = \Theta(n^{\log_b a})$$

   ya que se ignora el ϵ por ser una $f(n) > 0$, siendo este el caso 1 del teorema maestro

   $$\log_b a = \log_2 4 = 2$$

   Como podemos ver $f(n)$ es igual a $n^{\log_b a}$

   $$f(n) = n^2 \qquad n^{\log_b a} = n^2$$

   Por lo tanto, pasamos al caso 2 de Teorema Maestro.

   Si $f(n) = O(n^{\log_b a} \log n)$, entonces $T(n) = \Theta(n^{\log_b a} \log_2 n)$

   Así que:

   $$\therefore \; T(n) = \Theta(n^2 \log n)$$

   b $T(n) = T\left(\frac{n}{2}\right) + 1$ para $n > 1, T(1) = 0$

   Donde

   - $a = 1$
   - $b = 2$
   - $f(n) = 1$

   de modo que:

   $$T(n) = \Theta(n^{\log_b a})$$

   ya que se ignora el ϵ por ser una $f(n) > 0$, siendo este el caso 1 del teorema maestro

   $$\log_b a = \log_2 1 = 0$$

   Como podemos ver $f(n)$ es igual a $n^{\log_b a}$

   $$f(n) = 1 \qquad n^{\log_b a} = n^0 = 1$$

   Por lo tanto, pasamos al caso 2 de Teorema Maestro.

   $$\therefore \; T(n) = \Theta(\log n)$$

4. Lea los capítulos 1 y 2 del libro de Garey, M. R. y Johnson, D. S. (1979). *Computers and Intractability: A guide to the Theory of NP-Completeness*. W.H. Freeman and Company, New York. Posteriormente desarrolle un ejemplo, sobre un grafo particular que usted proponga, de la transformación polinomial explicada en la sección 2.5.

   Para esto debemos desarrollar un grafo Hamiltoniano G que se compone de:

   $$VerticesV = \{A, B, C, D, E\}$$
   $$AristasE = \{(A, B), (B, C), (C, D), (D, B), (D, E), (E, A)\}$$

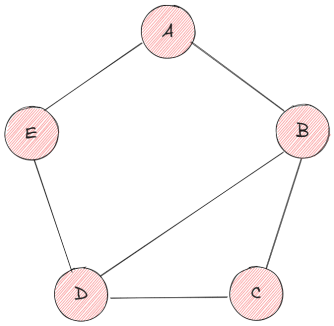

Figura 1. Grafo Hamiltoniano.

Tal como lo mencionado en el capitulo, se debe de convertir en un grafo completo y agregarle pesos, de modo en que se tomen las aristas de peso 1 como las mas cortas y de peso 2 como las no tan viables de tomar.

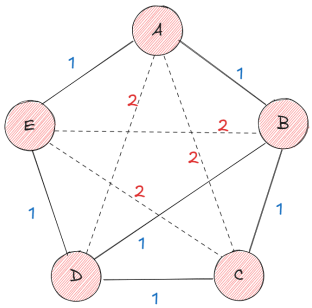

Figura 2. Grafo Hamiltoniano completo.

5. Investigue dos problemas de la clase NP-Complete distintos a los mencionados en las diapositivas de nuestra clase. Para cada uno de ellos proporcione la siguiente información:

   - **Problema 1:***Feedback arc set*

     a Su definición formal:

     Un conjunto de arcos de realimentación en un grafo dirigido D es un conjunto S de arcos en D tal que D S es acíclico. El problema de determinar un conjunto de arcos de realimentación con el menor número de arcos es NP - completo para un dígrafo general. Definimos el número de arcos de realimentación fuerte $\beta_s(G)$ para varias orientaciones de un grafo no dirigido G. [1]

     b Un ejemplo ilustrado de una instancia del problema y su solución.

     $$V = \{A, B, C, D\}$$
     $$E = \{(A \to B), (B \to C), (C \to D), (D \to B)\}$$

     Dado el anterior problema es requerido eliminar la menor cantidad de arcos posibles para lograr un grafo acíclico, eliminando en este caso el perteneciente a $(D \to B)$

     c Una lista de al menos tres algoritmos conocidos para resolverlo.

     - Algoritmo de karp-sipser
     - Algoritmos basados en programación lineal entera (ILP)
     - Algoritmo de eliminación de ciclos mediante heurísticas de búsqueda local

- **Problema 2**: *Problema de la partición*
  a Su definición formal:

    Si dado un multiconjunto de números enteros, puede este ser particionado en dos "mitades"tal que sumando los elementos de cada una, ambas den como resultado la misma suma. [3]
  b Un ejemplo ilustrado de una instancia del problema y su solución.

$$S = \{1, 5, 11, 5\}$$
$$S_1 = \{1, 5, 5\}$$
$$S_2 = \{11\}$$

    Dado el conjunto S, que se divide en dos subconjuntos, la suma de los componentes de este subconjunto dan el mismo resultado para ambos conjuntos
  c Una lista de al menos tres algoritmos conocidos para resolverlo.
    - Backtracking
    - Branch and bound
    - Algoritmo de Aproximación Greedy

6. Investigue en la literatura un par de problemas de decisión A y B tales que A sea polinomialmente reducible a B, i.e., $A \leq_p B$. Explique de forma detallada el proceso para reducir polinomialmente A en B.

   - Problema A.Subset Sum: Dado un conjunto S y un numero objetivo t, se espera comprobar que t existe en S
   - Problema B. Partition. Dado un conjunto S, se espera comprobar que existe un conjunto S_1 y S_2 donde la suma de sus componentes sea igual para ambos conjuntos.

   Sea (L, B) una instancia del problema de la suma de subconjuntos, donde L es una lista (multiconjunto) de números, y B es la suma objetivo. Sea $S = \sum L$. Sea L' la lista formada al agregar $S + B$ y $2S - B$ a L. Nótese que $\sum L' = 4S$. [2]
   ($\Rightarrow$) Si existe un subconjunto $M \subseteq L$ cuya suma es B, entonces L' puede ser particionado en dos partes iguales: $M \cup \{2S - B\}$ y $L \backslash M \cup \{S+B\}$. De hecho, la primera parte suma $B+(2S-B) = 2S$, y la segunda parte suma $(S - B) + (S + B) = 2S$.
   ($\Leftarrow$) Si L' puede ser particionado en dos partes iguales P_1 y P_2, entonces demostraremos que existe un subconjunto de L cuya suma es B.
   Recordemos que, dado que $\sum L' = 4S$, entonces $\sum P_1 = \sum P_2 = 2S$.
   Nótese que $(S + B) + (2S - B) = 3S \neq 2S$, por lo tanto, estos dos elementos deben estar en diferentes partes de la partición.
   Sin pérdida de generalidad, supongamos que $2S - B \in P_1$. Por lo tanto, el resto de los elementos en P_1 deben provenir exactamente de L y su suma debe ser B, lo que demuestra que hemos encontrado un subconjunto de L cuya suma es B, como queríamos demostrar.
   Funcionando para conjuntos de enteros positivos.

Referencias

[1] I. Rajasingh, B. Rajan y L. Joice, «Feedback Arc Set in Oriented Graphs», en *2009 International Conference on Computer Technology and Development*, vol. 1, 2009, págs. 569-573. DOI: 10.1109/ICCTD.2009.215.

[2] S. E. Community, *How can I reduce Subset Sum to Partition*, Question and answers on Stack Exchange, 2011. dirección: https://cs.stackexchange.com/questions/6111/how-can-i-reduce-subset-sum-to-partition.

[3] Wikipedia, *Problema de la partición*, Wikipedia, La enciclopedia libre, Consultado el 8 de octubre de 2024, 2024. dirección: https://es.wikipedia.org/wiki/Problema_de_la_partici%C3%B3n.