Universidad del Valle de Guatemala
Departamento de Matemática
Licenciatura en Matemática Aplicada

Estudiante: Rudik Roberto Rompich
Correo: rom19857@uvg.edu.gt
Carné: 19857

Análisis y diseño de algoritmos - Catedrático: Tomás Galvéz
19 de marzo de 2023

Parcial - Parte Casa

Problema 1. *Considérese el Travelling Salesman Problem (TSP):*

```
def tsp(G, start, node, visited):
    costs = []
    nextone = -1
    minactual = 10000000
    for edge in range(len(G[node])):
      if (edge not in visited):

        costs.append(G[node][edge]+tsp(G, start, edge, visited + [edge]))

        mintemporal = min(costs)
        if (minactual >= mintemporal):
          nextone = edge
          minactual = mintemporal

      if (len(costs)>0):

        visited.append(nextone)

        return min(costs)

      else:

        return G[node][start]
```

Entonces,

- *Divide-Conquer-Combine:*

- ***Divide***: *Es la parte en el que el problema se divide en subproblemas. En este caso tenemos al* ***for*** *que itera a cada* ***edge*** *y el condicional*

```
if (edge not in visited)
```

el cual subdivide el problema a los edges no visitados.

- ***Conquer***: *En esta parte es donde se ejecutan las llamadas recursivas. En este caso particular es cuando se ejecuta*

```
tsp(G, start, edge, visited + [edge])
```

- ***Combine***: *Aquí es donde se juntas las llamadas recursivas, es decir, en este algoritmo es la parte en donde se agregan a la lista de costos:*

```
costs.append(G[node][edge]+tsp(G, start, edge, visited + [edge]))
```

- *Relación de recurrencia de su tiempo de ejecución:*

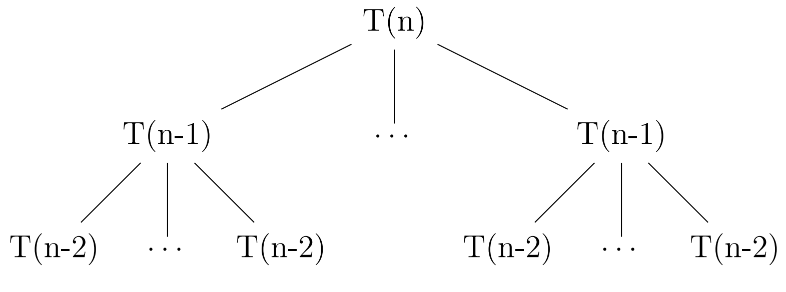
  - *Aquí notamos que hay diversas parte del algoritmo que debemos tomar en cuenta, tenemos n problemas (nodos) en el grafo completo, y cada problema (nodo) tiene $n-1$ subconjuntos posibles de problemas (nodos), esto podría expresarse como $T(n-1)$. Además, la recurrencia está dentro de un ciclo* ***for***. *Las demás partes del algoritmo o son $\Theta(1)$ o $\Theta(n)$ podemos concluir que la recurrencia de este algoritmo es*

$$T(n) = nT\left(n-1\right) + \Theta(n)$$

Siendo más formalistas, tenemos

$$T(n) = \begin{cases} \Theta(1), & n = 1 \\ T(n) = nT(n-1) + \Theta(n), & n > 1 \end{cases}$$

Ahora, usando la técnica del árbol de recursión, tenemos:



De esto, obtenemos:

$$T(n) = cn + cn(n-1) + cn(n-1)(n-2) + \cdots cn!$$

Con lo que podemos concluir que el algoritmo tiene complejidad

$$O(n!)$$

Problema 2. *Considere Merge Sort:*

```
Merge_Sort(A, p, r):
    if p < r:
        q= floor((p+r)/2)
        Merge-Sort(A,p,q)
        Merge-Sort(A,q+1,r)
        Merge(A,p,q,r)
```

Entonces:

- *Divide-Conquer-Combine:*
  - *Divide: La parte donde se divide el algoritmo:*

    `q= floor((p+r)/2)`

  - *Conquer: Las llamadas recursivas, es decir:*

    ```
    Merge-Sort(A,p,q)
    Merge-Sort(A,q+1,r)
    ```

  - *Combine: Donde se juntan las llamadas recursivas*

    `Merge(A,p,q,r)`

- *Para este caso, la relación de recurrencia ya la habíamos encontrado en la clase, la cual es:*

$$T(n) = \begin{cases} \Theta(1) & \textit{if } n = 1 \\ 2T(n/2) + \Theta(n) & \textit{if } n > 1 \end{cases}$$

(O sea la respuesta que da el PDF del parcial, es correcta) y con Master Method, se había comprobado que la solución era $\Theta(n \log_2 n)$. Ahora bien, usando el método de substitución, tenemos:

- *Para $T(n) = O(n \log_2 n)$*
  - *Paso inductivo, sea $T(n) \leq cn \log_2 n \implies T(n/2) \leq c\,(n/2) \log_2(n/2)$. Por lo tanto,*

$$\begin{aligned} T(n) = 2T(n/2) + cn &\leq 2\left(c\,(n/2)\log_2(n/2)\right) + cn \\ &= cn \log_2\left(\frac{n}{2}\right) + cn \\ &= cn(\log_2 n - \log_2 2) + cn \\ &= cn \log_2 n, \quad c \geq 2 \end{aligned}$$

  - *Paso base, a partir de $n \geq n_0 = 2$, $T(2) = 4$ y la propiedad se cumple:*

$$T(2) \leq c2 \log_2(2) = 2c, \quad c \geq 2$$

- *Para* $T(n) = \Omega(n \log_2 n)$
  - *Paso inductivo, sea* $T(n) \geq cn \log_2 n \implies T(n/2) \geq c\,(n/2) \log_2(n/2)$. *Por lo tanto,*

$$\begin{aligned} T(n) = 2T(n/2) + cn \geq 2\,(c\,(n/2) \log_2 (n/2)) + cn \\ &= cn \log_2 \left(\frac{n}{2}\right) + cn \\ &= cn(\log_2 n - \log_2 2) + cn \\ &= cn \log_2 n, \quad 0 < c \leq 1 \end{aligned}$$

  - *Paso base, a partir de* $n \geq n_0 = 2$, $T(2) = 4$ *y la propiedad se cumple:*

$$T(2) \geq c2 \log_2(2) = 2c, \quad 0 < c \leq 1$$