Práctica 5

Juan Llinares Mauri
74011239E

1. Práctica 4 - Complejidad temporal: Cálculo analítico (II)

1.1. Ejercicio 3

```
1 bool palind(string &pal, int pri, int ult) {
2     if (pri >= ult)
3         return true;
4     else
5         return (pal[pri] == pal[ult]) && palind(pal, pri + 1,
              ult - 1);
6 }
```

El tamaño de problema de este método depende de la palabra que se pase como parámetro. Asumimos que las variables `pri` y `ult` siempre serán `pri<ult` en la primera llamada. Tiene mejor y peor caso los cuales dependen del `and` dentro del `return` del `else`, pues si la primera comparación ya es `false`, no se harán las llamadas recursivas.

Mejor caso Cuando `pal[pri]==pal[ult]` sea `false` en la primera comparación, el algoritmo tendrá una complejidad temporal de $\Omega(1)$, pues sólo realizará un `return`.

Peor caso Cuando `pal[pri]==pal[ult]` sea siempre `true`, el algoritmo realizará todas las llamadas recursivas posibles sobre una palabra n, dejando la relación de recurrencia siguiente:

$$T(n) \begin{cases} 1 & n \leq 0 \\ 1 + T(n-2) & n > 0 \end{cases}$$

Mediante sustitución:

$$f(n) = 1 + T(n-2) = 2 + T(n-4) = 3 + T(n-6) = ... = k + T(n - \sum_{i=1}^{k} 2i)$$

Tendremos entonces que $1 = n - 2k \rightarrow k = \frac{n}{2}$, entonces $T(n) = \frac{n}{2} + T(n - 2(\frac{n}{2})) = \frac{n}{2} + T(n-n) = \frac{n}{2} + 1$, siendo $\Theta(n)$ la complejidad temporal del algoritmo en el peor caso.