Algo

venerdì 25 marzo 2022 19:53

```
F_y(n):
        Z=n, T=0,                       2C
        While z > 0:                    Ctwhile
                X = z mod 2             Ctw
                Z = z div 2             Ctw
                If x == 0:              Ctw
                        For i = 1 to n: Ctifwhile
                                T++     C*n*tifwhile
        Return(t)                       C
```

$T_{F_y}(n) = 3c + 4ct_w + ct_{ifwhile} + c * n * t_{ifwhile}$

Caso migliore:

- Twhile = 0 -> z <= 0 -> n <= 0 --> Non è cosi tanto facile, non farlo durante l'esame lol

Analizzando:
Noi dividiamo sempre, il nostro z si continua a dividere e,
Se la nostra divisione non da resto, aumentiamo T.
Noi non vogliamo entrare nell'if, quindi vogliamo sempre avere un resto
1 / 2 -> dà resto, okay
2 / 2 -> non dà resto, non lo vogliamo
3 / 2 -> Dà resto -> 1 / 2 -> dà resto, okay
5 / 2 -> Dà resto -> 2/2 -> non da resto, non va bene.
Continuando così, possiamo notare un pattern:
7 / 2 -> Dà resto -> 3/2 -> Dà resto -> 1/2 -> Dà resto
E quindi, tutti i numeri che nella loro rappresentazione binaria
Sono tutti 1.
Mentre, non vogliamo i valori che sono sempre pari, quindi
1000000

Ora, dobbiamo capire questo Twhile che valore abbia. Noi continuiamo a dividere, quindi,

$\frac{n}{2} \to \frac{\frac{n}{2}}{2} \to \frac{\frac{\frac{n}{2}}{2}}{2}$

E così via. Quindi

$\frac{n}{2^i} \to i = \ln_2 n = t_w$

Caso migliore:

- Tutti i bit sono a 1
- Tif = 0

$t_{F_y}(n) = 3c + 4c * \ln n = \Omega(\ln n)$

Caso peggiore:

- Tutti bit 0 tranne più significativo
- Tif = n - 1 -> $\ln(n-1)$

$3c + 4c \ln n + c(\ln(n-1)) + cn * \ln n = O(n \ln n)$

Per migliorarlo: t += n
In questo modo $O = \ln n$
E quindi $\theta = \ln n$