# Taller 2

**1)** $(n+a)^b = O(n^b)$

$$(n+a)^b = n^b + an^{b-1} + a^2 n^{b-2} + \dots + a^{b-1}n + a^b$$
$$= O(n^b) + O(n^{b-1}) + O(n^{b-2}) + \dots + O(n) + O(1)$$
$$= O(n^b)$$

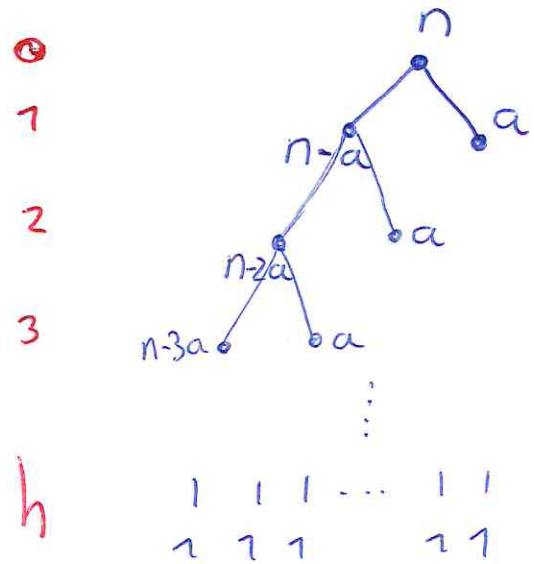
**2)** $2T(\lfloor n/2 \rfloor) + n$ es $\Omega(n \log_2 n)$

$$2\lfloor n/2 \rfloor \log_2 \lfloor n/2 \rfloor + n \geq c\, n \log_2 n$$

$$2\lfloor n/2 \rfloor \log_2 \lfloor n/2 \rfloor \geqslant 2\left(\tfrac{n}{2}-1\right)\log_2\left(\tfrac{n}{2}-1\right) = \cancel{2}\left(\tfrac{n-2}{\cancel{2}}\right)\log_2\left(\tfrac{n-2}{2}\right) = (n-2)\left[\log_2(n-2) - \cancel{\log_2 2}\right]$$

$$= n\log_2(n-2) - n - 2\log_2(n-2) + 2 \approx O(n\log_2 n) + O(n) + O(\log_2 n) + O(1)$$
$$= O(n \log_2 n)$$

**3)** $T(n) = T(n-a) + T(a) + cn$




Costos:

- dato: $c$
- nodo: $c(n-ka) = cn - kac = cn - kc_1$
- nivel: $2^k(cn - kc_1) = cn(2^k) - c_1 k(2^k)$
- árbol:

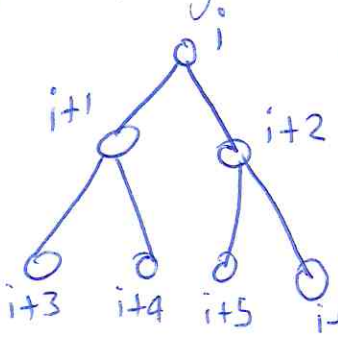
$$\sum_{k=0}^{h}\left[cn(2^k) - c_1 k(2^k)\right]$$

$$n - ha = 1$$
$$h = \frac{n-1}{a}$$

$$cn\sum_{k=0}^{\frac{n-1}{a}} 2^k - c_1 \sum_{k=0}^{\frac{n-1}{a}} k(2^k) \leq cn \sum_{k=0}^{\frac{n-1}{a}} 2^k$$

$$cn\sum_{k=0}^{\frac{n-1}{a}} z^k = cn\,\frac{2^{\frac{n-1}{a}+1}-1}{2-1} = 2cn\left(2^{\frac{n-1}{a}}\right) - cn \approx O(n(2^n))$$

4) Dado el siguiente árbol:




... Nivel 0
... Nivel k
... Nivel k+1

Hipótesis en Nivel k:

$i+1 \equiv 2*i$

$i+2 \equiv 2*i+1$

En nivel k+1:

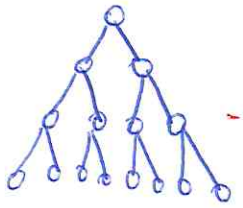
$i+3 = i+1+2 \equiv 2i+2 = 2(i+1)$ ✓

$i+4 = i+2+2 \equiv 2i+1+2 = 2(i+1)+1$ ✓

$i+5 = i+1+4 \equiv 2i+4 = 2(i+2)$ ✓

$i+6 = i+2+4 \equiv 2i+1+4 = 2(i+2)+1$ ✓

5) En un árbol:



.... Nivel k
... Nivel k+1

Hipótesis en Nivel k:

Total de nodos: $\sum_{k=0}^{k} 2^k = \frac{2^{k+1}-1}{2-1} = 2^{k+1}-1$

Punto medio: $\left\lfloor \frac{2^{k+1}-1}{2} \right\rfloor = \left\lfloor \frac{2^{k+1}}{2} - \frac{1}{2} \right\rfloor = \left\lfloor 2^k - \frac{1}{2} \right\rfloor$

Hojas están en:

$\{\lfloor 2^k - \frac{1}{2} \rfloor + 1 \ \ldots \ 2^{k+1}-1\}$

$\{\lfloor 2^k - \frac{1}{2} + 1 \rfloor \ \ldots \ 2^{k+1}-1\}$

$\{\lfloor 2^k + \frac{1}{2} \rfloor \ \ldots \ 2^{k+1}-1\}$

$\{2^k \ \ldots \ 2^{k+1}-1\}$

"Hojas comienzan en $2^k$, hasta el final"

En nivel k+1:

Total de nodos: $\sum_{k=0}^{k+1} 2^k = \frac{2^{k+2}-1}{2-1} = 2^{k+2}-1$

Punto Medio: $\left\lfloor \frac{2^{k+2}-1}{2} \right\rfloor = \left\lfloor \frac{2^{k+2}}{2} - \frac{1}{2} \right\rfloor = \left\lfloor 2^{k+1} - \frac{1}{2} \right\rfloor$

Hojas están en:

$\{ \lfloor 2^{k+1} - \frac{1}{2} \rfloor + 1 \ \ldots \ 2^{k+2}-1 \}$

$\{ \lfloor 2^{k+1} - \frac{1}{2} + 1 \rfloor \ \ldots \ 2^{k+2}-1 \}$

$\{ \lfloor 2^{k+1} + \frac{1}{2} \rfloor \ \ldots \ 2^{k+2}-1 \}$

$\{ \lfloor 2^{k+1} \rfloor \ \ldots \ 2^{k+2}-1 \}$

$\{ 2^{k+1} \ \ldots \ 2^{k+2}-1 \}$

↳ "Desde $2^{k+1}$, hasta el final"

Mantiene patrón

6) La porción de demostración establece que la parte no recursiva de la Recurrencia para un valor n cualquiera es proporcional a $n^{\log_b a}$. Luego, esta conclusión se utiliza para definir que el resto de la Recurrencia difiere de este valor en solamente un factor $\log_2 n$.

```cpp
#include <iostream>

using namespace std;

typedef struct Data{
    int altura;
    char gender;
} Data;

int PARTITION(Data *A, int p, int r){
    Data x = A[r], temp;
    int i = p - 1;
    for(int j=p; j<r; j++)
        if((A[j].altura < x.altura) || (A[j].altura == x.altura && A[j].gender ==
        'b' && x.gender == 'g')){
            i++;
            temp = A[i];
            A[i] = A[j];
            A[j] = temp;
        }
    temp = A[i+1];
    A[i+1] = A[r];
    A[r] = temp;
    return i+1;
}

void QUICKSORT(Data *A, int p, int r){
    if(p < r){
        int q = PARTITION(A, p, r);
        QUICKSORT(A, p, q-1);
        QUICKSORT(A, q+1, r);
    }
}

int main()
{
    int T;

    cin >> T;

    for(int i=0;i<T;i++){
        int M, N;
        cin >> M >> N;
        if(N < M){ cout << "NO"; continue; }
        Data people[M+N];
        for(int b=0; b<M; b++){
            cin >> people[b].altura;
            people[b].gender = 'b';
        }
        for(int g=0; g<N; g++){
            cin >> people[M+g].altura;
            people[M+g].gender = 'g';
        }

        QUICKSORT(people,0,M+N-1);

        for(int j=0;j<M+N;j++){
            if(j == 0 && people[j].gender == 'b'){ cout << "NO"; break; }
            if(people[j].gender == 'b' && people[j-1].gender != 'g'){ cout << "NO";
            break; }
            if(j == M+N-1) cout << "YES";
        }
    }

    return 0;
}
```