PARCIAL 1

Preguntas:

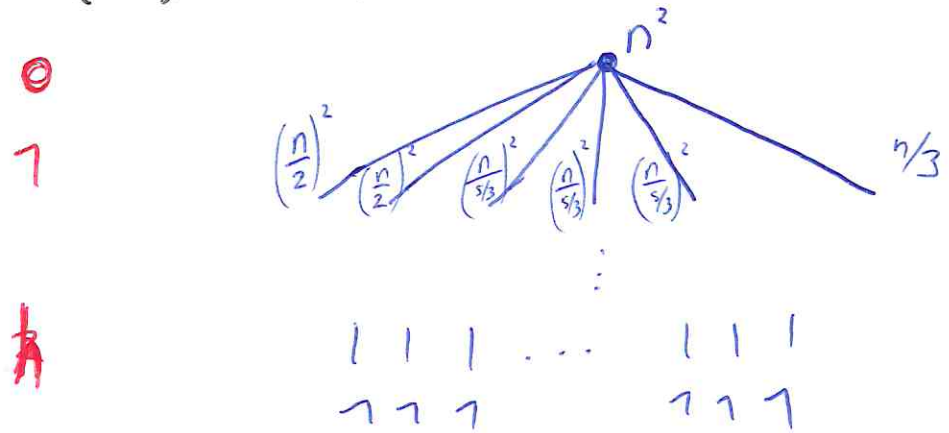
1) "Deben incluirse conceptos como: costos, cantidad de ejecuciones, orden de magnitud, mejor y peor de los casos, recurrencia, resolución de recurrencias, algoritmos de ordenamiento ..."

2) Orden de magnitud hace referencia a cantidad de ejecuciones mientras que orden de Convergencia hace referencia a velocidad de acercamiento a un dato.

3)
- rutas de aviones: para decidir asignación de pistas...
- habitaciones en hospital: asignación de vacentes...
- ascensor: decidir ruta de pisos...
- puertobus: asignación de estaciones...
- bancos: para control de transacciones...
- restaurantes: para ubicación de órdenes a cocina...
- aulas UCA: para asignar a materias...
- Uber: asignación de rutas y clientes...
- Torneos deportivos: administración de canchas.
- Fabricación de autos: automatizar ensamblaje...

Ejercicios:

1) $2T(n/2) + 3T(3n/5) + T(n/3) + O(n^2)$



Costos

- dato: C
- nodo: $C\frac{n^2}{(\frac{5^2}{3^2})^k} = C\frac{n^2}{(\frac{25}{9})^k}$
- nivel: $6^k\left[C\frac{n^2}{(\frac{25}{9})^k}\right] = C\frac{n^2}{(\frac{25}{54})^k}$

$= Cn^2\left(\frac{54}{25}\right)^k$

- árbol: $\sum_{k=0}^{h} cn^2 \left(\frac{54}{25}\right)^k = cn^2 \sum_{k=0}^{h} \left(\frac{54}{25}\right)^k$

Hacerlo hasta el infinito diverge

$\frac{n^2}{\left(\frac{25}{9}\right)^h} = 1 \rightarrow n^2 = \left(\frac{25}{9}\right)^h \Rightarrow \log_{25/9} n^2 = h \rightarrow h = 2\log_{25/9} n$

$\sum_{k=0}^{2\log_{25/9} n} \left(\frac{54}{25}\right)^k = cn^2 \frac{\left(\frac{54}{25}\right)^{2\log_{25/9} n + 1} - 1}{\frac{54}{25} - 1} = cn^2 \frac{\left(\frac{54}{25}\right)\left(\left(\frac{54}{25}\right)^2\right)^{\log_{25/9} n} - 1}{c_1}$

$= cn^2 \frac{c_2 n^{\log_{25/9}\left(\left(\frac{54}{25}\right)^2\right)}}{c_1} - \frac{1}{c_1} = c_3 n^{\log_2 4 - + 2} + c_4 n^2$

$\approx c_3 n^2 + c_4 n^2 = O(n^4)$

$2T(n/2) + 3T(3n/5) + T(n/3) + O(n^2) \leq cn^4$

$2\frac{n^4}{2^4} + 3\left(\frac{3^4 n^4}{5^4}\right) + \frac{n^4}{3^4} + c_1 n^2 \leq cn^4$

$\frac{n^2}{2^3} + \frac{3^5}{5^4} n^2 + \frac{1}{3^4} n^4 + c_1 n^2 \leq cn^4$

$O(n^4)$

$c \geq \frac{1}{2^3} + \frac{3^5}{5^4} + \frac{1}{3^4} + c_1$

2) a) $5T(n/3) + 4n^2$

$f(n) = 4n^2$ $\quad n^{\log_b a} = n^{\log_3 5} = n^{1.-}$

$n^{1.- + \varepsilon} = n^2 \rightarrow$ Caso 3

$\varepsilon > 0$

$af(n/b) \leq cf(n)$

$5\left(4\left(\frac{n}{3}\right)^2\right) \leq c(4n^2) \rightarrow \frac{20}{9} n^2 \leq 4cn^2 \rightarrow \frac{5}{9} n^2 \leq cn^2$

$c \geq \frac{5}{9} \rightarrow c < 1 \rightarrow T(n) = \Theta(n^2)$

b) $2T(n/4) + n^2 - 2n + 4$

descartable

$f(n) = n^2$ $\quad n^{\log_b a} = n^{\log_4 2} = n^{0.-}$

$n^{0.- + \varepsilon} = n^2 \rightarrow$ Caso 3

$\varepsilon > 0$

$af(n/b) \leq cf(n)$

$2\left(\frac{n^2}{4^2}\right) \leq cn^2 \rightarrow \frac{n^2}{2^3} \leq cn^2 \rightarrow c \geq \frac{1}{8}, c < 1$

$\hookrightarrow \boxed{T(n) = \Theta(n^2)}$

c) $8T(n/q) + n^3 \log_2 n$

$f(n) = n^3 \log_2 n$ $\quad n^{\log_q 8} = n^{0.-}$

$n^3 \log_2 n = n^{0.- + \varepsilon} \rightarrow$ Caso 3

$\varepsilon > 0$

$af(n/b) \leq cf(n)$

$8\left(\frac{n}{q}\right)^3 \log_2\left(\frac{n}{q}\right) \leq cn^3 \log_2 n$

$\frac{8}{q^3} n^3 \left[\log_2 n - \log_2 q\right] = \frac{8}{q^3} n^3 \log_2 n - \frac{8}{q^3} n^3 \log_2 q \leq \frac{8}{q^3} n^3 \log_2 n$

$\leq cn^3 \log_2 n$

$c \geq \frac{8}{q^3}, c < 1$

$\boxed{T(n) = \Theta(n^3 \log_2 n)}$

```cpp
#include <iostream>
#include "math.h"

using namespace std;

typedef struct Nodo {
    int dato;
    struct Nodo *ant=NULL, *sig=NULL;
} Nodo;

Nodo *ini = NULL;

int suma(Nodo **C, int n){
    int s = 0;
    for(int i=0;i<n;i++)
        s += C[i]->dato;
    return s;
}

bool verificar(int m){
    if(ini->dato != m) return false;
    Nodo *temp = ini->sig;
    while(temp != ini){
        if(temp->dato != m) return false;
        temp = temp->sig;
    }
    return true;
}

void MAXHEAPIFY(Nodo**C, int n, int i){
    int L = 2*i;
    int R = 2*i + 1;
    int largest;
    if(L <= n && C[L]->dato > C[i]->dato)
        largest = L;
    else
        largest = i;
    if(R <= n && C[R]->dato > C[largest]->dato)
        largest = R;
    if(largest != i){
        Nodo* temp = C[i];
        C[i] = C[largest];
        C[largest] = temp;
        MAXHEAPIFY(C, n, largest);
    }
}

void BUILDMAXHEAP(Nodo **C, int n){
    int heapsize = n-1;
    for(int i=floor((n-1)/2);i>=0;i--)
        MAXHEAPIFY(C, heapsize, i);
}

void HEAPSORT(Nodo **C, int n){
    int heapsize = n-1;
    BUILDMAXHEAP(C, n);
    for(int i = n-1; i > 0; i--){
        Nodo* temp = C[0];
        C[0] = C[i];
        C[i] = temp;
        heapsize--;
        MAXHEAPIFY(C, heapsize, 0);
    }
}

void daruno(Nodo *N){
    if(N->ant->dato <= N->sig->dato)
```

```
68              N->ant->dato += 1;
69          else
70              N->sig->dato += 1;
71          N->dato -= 1;
72      }
73
74      void decidir(Nodo **C, int n){
75          HEAPSORT(C, n);
76          daruno(C[n-1]);
77      }
78
79      int Ej3(Nodo **C, int m, int n){
80          int cont = 0;
81          while(verificar(m) != true){
82              decidir(C, n);
83              cont++;
84          }
85          return cont;
86      }
87
88      int main()
89      {
90          int n;
91          cin >> n;
92
93          Nodo* C[n];
94          for(int i=0;i<n;i++){
95              C[i] = (Nodo*) malloc(sizeof(Nodo));
96              cin >> C[i]->dato;
97          }
98
99          C[0]->ant = C[n-1]; C[0]->sig = C[1];
100         C[n-1]->ant = C[n-2]; C[n-1]->sig = C[0];
101         for(int i=1;i<n-1;i++){
102             C[i]->ant = C[i-1];
103             C[i]->sig = C[i+1];
104         }
105
106         ini = C[0];
107
108         int m = suma(C,n)/n;
109
110         int res = Ej3(C,m,n);
111
112         cout << res;
113
114         return 0;
115     }
```