Taller02.cpp > ...

```cpp
#include <iostream>
#include <fstream>

using namespace std;

struct persona
{
    int id;
    string nombre;
    double sueldo;
};

bool lectura(persona arr[], int &n);

class maxHeap
{
private:
    static const int MAX_tam = 1200;
    persona heap[MAX_tam];
    int tam;

    int padre(int index)
    {
        return (index - 1) / 2;
    }

    int hijoIzq(int index)
    {
        return 2 * index + 1;
    }

    int hijoDer(int index)
    {
        return 2 * index + 2;
    }

    bool tieneHijoIzq(int index)
    {
        return hijoIzq(index) < tam;
    }

    bool tieneHijoDer(int index)
    {
        return hijoDer(index) < tam;
    }

    bool tienePadre(int index)
    {
        return padre(index) >= 0;
    }

    void swap(int indexUno, int indexDos)
    {
        persona temp = heap[indexUno];
        heap[indexUno] = heap[indexDos];
        heap[indexDos] = temp;
    }
```

$C_1$ $C_2$ $C_3$ $\} C_a$

$C_4$ $\} C_4(C_1) = C_b$

$C_5$ $C_6$ $C_7$ $\} C_s$

$C_8$ — $C_c$

$C_9$ — $C_d$

$C_{10}$ $C_{11}$ $C_{12}$ $\} C_e$

$C_{13}$ $C_{14}$ $\} C_{13}(C_{14}) = C_f$

$C_{15}$ $C_{16}$ $\} C_{15}(C_{16}) = C_g$

$C_{17}$ $C_{18}$ $\} C_{17}(C_{18}) = C_h$

$C_{19}$ $C_{20}$ $\} C_{19}(C_{20}) = C_i$

$C_{21}$ $C_{22}$ $\} C_{21}(C_{22}) = C_j$

$C_{23}$ $C_{24}$ $\} C_{23}(C_{24}) = C_k$

$C_{25}$ $C_{26}$ $C_{27}$ $C_{28}$ $\} C_{25}[C_{26} + C_{27} + C_{28}] = C_l$

```
60      void agruparUp() ———————————————————— C1
61      {
62          int index = tam - 1; ———————————— C2
63          while (tienePadre(index) && heap[padre(index)].sueldo < heap[index].
            sueldo) ———————————————————— C3 ———————— log n
64          {                                                    Σ (Xi + 1)
65              swap(padre(index), index); ———— C4 }            i=1
66              index = padre(index); ———————— C5 } Xi — log n
67          }                                                   Σ Xi
68      }                                                       i=1
```

$$\sum_{i=1}^{n} x_i = \frac{n(n+1)}{2}$$

$$\underbrace{C_1 + C_2}_{a} + C_3 \sum_{i=1}^{\log n}(x_i + 1) + \underbrace{(C_4 + C_5)}_{b} \sum_{i=1}^{\log n} x_i$$

$$a + C_3\left[\sum_{i=1}^{\log n} x_i + \sum_{i=1}^{\log n} 1\right] + b\sum_{i=1}^{\log n} x_i$$

$$a + C_3\left[\frac{\log n(\log n+1)}{2} + \log n - 1\right] + b\left[\frac{\log n(\log n+1)}{2}\right]$$

$$a + C_3\left[\frac{(\log n)^2 + \log n}{2} + \log n - 1\right] + b\left[\frac{(\log n)^2 + \log n}{2}\right]$$

$$\cancel{a}^{c} + \cancel{C_3\left(\tfrac{1}{2}\right)}^{c} + \cancel{C_3}^{c}(\log n)^2 + \cancel{C_3}^{c}(\log n) + \cancel{C_3}^{c}(\log n - 1) + \cancel{b}^{c}\left(\tfrac{1}{2}\right) + \cancel{b}^{c}(\log n)^2 + \cancel{b}^{c}\log n$$

$$(\log n)^2 + \log n + \log n + (\log n)^2 + \log n$$

$$O(2(\log n)^2)$$

```
71      void agruparDo()
72      {
73          int index = 0;
74          while (tieneHijoIzq(index))
75          {
76              int hijoMayorIndex = hijoIzq(index);
77              if (tieneHijoDer(index) && heap[hijoDer(index)].sueldo > heap
                [hijoMayorIndex].sueldo)
78              {
79                  hijoMayorIndex = hijoDer(index);
80              }
81
82              if (heap[index].sueldo > heap[hijoMayorIndex].sueldo)
83              {
84                  break;
85              }
86
87              swap(index, hijoMayorIndex);
88              index = hijoMayorIndex;
89          }
90      }
```

$C_1$ } $C_a$ — $\log n$ $\sum_{i=1}$ $(X_i + 1)$

$C_2$

$C_3$

$C_4$

$C_5$ } $Max(0,1)$

$C_6$

$C_7$ } $Max(0,1)$

$C_8$

$C_9$

$C_{10}$

$\log n$ $\sum_{i=1}$ $X_i$

$$C_a + (C_2(C_3)) \sum_{i=1}^{\log n} (x_i + 1) + (C_5 + C_6 + C_7 + C_8 + C_9 + C_{10}) \sum_{i=1}^{\log n} x_i$$

$C_a$ $C_b$

$$C_a \left[ \sum_{i=1}^{\log n} x_i + \sum_{i=1}^{\log n} 1 \right] + C_b \sum_{i=1}^{\log n} x_i$$

$$C_a \left[ \frac{\log n (\log n + 1)}{2} + \log n \right] + C_b \left[ \frac{\log n (\log n + 1)}{2} \right]$$

$$C_a \left(\frac{1}{2}\right)((\log n)^2 + \log n) + C_a(\log n) + C_b\left(\frac{1}{2}\right) + C_b((\log n)^2) + C_b$$

$$(\log n)^2 + \log n + \log n + (\log n)^2$$

$$O(\log x^2)$$

```
92  public:
93      maxHeap() : tam(0) {}
94
95      //agrega un nuevo dato al heap
96      void insertar(persona elemento)
97      {
98          if (tam == MAX_tam)
99          {
100             std::cout << "El heap está lleno, no se puede insertar más
                elementos." << std::endl;
101             return;
102         }
103         heap[tam] = elemento;
104         tam++;
105         agruparUp();
106     }
107     //=======================================================
```

$C_1$

$C_2$

$C_3$ — $X_i + 1$

$C_3$

$C_4$

$C_5$

$C_6$

$\max(0,1)$

$X_i$

$$C_1 + C_2 + \sum_{i=1}^{\log n} 1 + (C_3 + C_4 + C_5 + C_6) \sum_{i=1}^{\log n} X_i$$

$$C_a \sum_{i=1}^{\log n} 1 + \qquad C_b \sum_{i=1}^{\log n} Y_i$$

```
108     persona eliminarMax()
109     {
110         if (tam == 0)
111         {
112             std::cout << "El heap está vacío, no se puede eliminar el máximo."
                << std::endl;
113             persona p = {-1, "", 0.0};
114             return p;
115         }
116         persona maxElement = heap[0];
117         heap[0] = heap[tam - 1];
118         tam--;
119         agruparDo();
120         return maxElement;
121     }
122
```

```
123     void mostrarHeap()
124     {
125         cout << "Contenido del heap: " << endl;
126         for (int i = 0; i < tam; ++i)
127         {
128             cout << heap[i].id << " " << heap[i].nombre << " " << heap[i].
                sueldo << endl;
129         }
130         cout << endl;
131     }
```

```
    void headsort(persona arr[], int n)
    {
        maxHeap maxheap;
        for (int i = 0; i < n; ++i)
        {
            maxheap.insertar(arr[i]);
        }
        for (int i = 0; i < n; ++i)
        {
            arr[i] = maxheap.eliminarMax();
        }
    };
};
```

```cpp
int main()
{
    const int n = 1200;
    persona datos[n];
    int c = 0;

    if (!lectura(datos, c))
    {
        cout << "No se pudo abrir el archivo." << endl;
        return 1;
    }

    cout << "Número de elementos leídos: " << c << endl;

    if (c == 0)
    {
        cout << "No se leyeron datos del archivo." << endl;
        return 1;
    }

    maxHeap maxheap;
    maxheap.headsort(datos, c);

    cout << "Arreglo ordenado en orden descendente: " << endl;
    for (int i = 0; i < c; ++i)
    {
        cout << datos[i].id << " " << datos[i].nombre << " " << datos[i].
        sueldo << endl;
    }

    return 0;
}
```

```cpp
bool lectura(persona arr[], int &n)
{
    ifstream archivo;
    archivo.open("C:/Users/HP/Downloads/taller02_Algoritmos/
    taller02_Algoritmos/datos.txt", ios::in);

    if (archivo.fail())
    {
        return false;
    }

    n = 0;
    while (archivo >> arr[n].id)
    {
        archivo.ignore();
        getline(archivo, arr[n].nombre, '$');
        archivo >> arr[n].sueldo;
        archivo.ignore();

        n++;

        if (n >= 1200)
        {
            break;
        }
    }

    archivo.close();

    return true;
}
```