Práctica 2: Algoritmos Divide y Vencerás

Daniel Bolaños Martínez, José María Borrás Serrano,
Santiago De Diego De Diego, Fernando De la Hoz Moreno

ETSIIT

Introducción

Código Divide y Vencerás

Código secuencial

Listing 1: Función unimodal DyV

```
int unimodal(vector<int> v)
{
        bool fin=false;
        int indice=1;

        while(!fin)
        {
                if(v.at(indice+1)<v.at(
                    indice))
                        fin=true;
        else
                indice++;
        }

        return indice;
}
```

Listing 2: Función main

```
int main(int argc, char* argv[])
{
        vector<int> array;
        int valor = -1;
        int v_size = atoi(argv[1]);
        array.resize(v_size);
    int p = 1 + rand() %(v_size-2);
    array.at(p) = v_size-1;

    for (int i=0; i<p; i++)
        array.at(i)=i;

    for (int i=p+1; i<v_size; i++)
        array.at(i)=v_size-1-i+p;

        clock_t tantes;
        clock_t tdespues;
        tantes=clock();
        valor = unimodal(array);
        tdespues=clock();

        for(int i=0; i < v_size; i++)
        cout << array.at(i) << endl;

    cout << "Maximo: " << array.at(valor) << endl;
        cout << v_size <<" "<< (double)(tdespues - tantes) / CLOCKS_PER_SEC <<
             endl;
}
```

Código secuencial

Listing 3: Función unimodal Secuencial

```
int unimodal_secuencial(vector<int> v)
{
        bool fin=false;
        int indice=1;

        while(!fin)
        {
                if(v.at(indice+1)<v.at(
                    indice))
                        fin=true;
        else
                indice++;
        }

        return indice;
}
```

Listing 4: Función main

```
int main(int argc, char* argv[])
{
        vector<int> array;
        int valor = -1;
        int v_size = atoi(argv[1]);
        array.resize(v_size);
        int p = 1 + rand() %(v_size-2);
        array.at(p) = v_size-1;

    for (int i=0; i<p; i++)
        array.at(i)=i;

    for (int i=p+1; i<v_size; i++)
        array.at(i)=v_size-1-i+p;

        clock_t tantes;
        clock_t tdespues;
        tantes=clock();
        valor = unimodal_secuencial(array);
        tdespues=clock();

        for(int i=0; i < v_size; i++)
                cout << array.at(i) << endl;

    cout << "Maximo: " << array.at(valor) << endl;
        cout << v_size <<" "<< (double)(tdespues - tantes) / CLOCKS_PER_SEC <<
             endl;
}
```

Comparación de la eficiencia

Eficiencia teórica

Figura: Pie de imagen

Eficiencia empírica

Eficiencia híbrida

Conclusión