

Relación Divide y vencerás

Rubén Celso Villalón - Algoritmos G.D

Ejercicios:

1) $a[k] = k$

```
int seComple(const vector<int> &v, int mi, int fin) {
    int pos = -1;
    int mitad = (mi + fin) / 2;
    if (v[mi] == mi) {
        pos = mi;
    }
    if (mi != fin) {
        int pos1 = seComple(v, mi, mitad);
        int pos2 = seComple(v, mitad + 1, fin);
        if (pos1 != -1)
            return pos1;
        if (pos2 != -1)
            return pos2;
    }
    return pos;
}
```

2) Ordenar resultados

```
int Ordena (vector<int> &v, int primero, int ultimo) {  
    int intercambie;  
    int pos-pivote = primero;  
    int valor-pivote = v[primero];  
    for (int i = primero + 1; i <= ultimo; i++) {  
        if (v[i] < valor-pivote) {  
            pos-pivote++;  
            intercambie = v[i];  
            v[i] = v[pos-pivote];  
            v[pos-pivote] = intercambie;  
        }  
    }  
    intercambie = v[primero];  
    v[primero] = v[pos-pivote];  
    v[pos-pivote] = intercambie;  
    return pos-pivote;  
}  
  
void QuickSort (vector<int> &v, int ini, int fin) {  
    if (ini - fin != 0) {  
        int pos = Ordena (v, ini, fin);  
        int mitad = pos;  
        QuickSort (v, ini, mitad - 1);  
        QuickSort (v, mitad + 1, fin);  
    }  
}
```

```
void OrderResults(vector<vector<int>> & res) {  
    for(int i = 0; i < res.size(); i++) {  
        QuickSort(res[i], 0, res[i].size() - 1);  
    }  
}
```

4) tornillos y tuercas

// Suponiendo implementada la función pivot

```
void Qsort(vector<int> &v1, vector<int> &v2, int ini, int fin){
```

```
    if (fin - ini != 0){
```

```
        int piv = Pivot(v1, ini, fin);
```

```
        int elem, pos;
```

```
        bool enc = false;
```

```
        for (int c = 0; c < v2.size() && !enc; c++){
```

```
            if (v2[c] == v1[piv]){
```

```
                pos = c;
```

```
            } else {
```

```
                elem = v2[pos];
```

```
                v2[pos] = v2[ini];
```

```
                v2[ini] = elem;
```

```
                Pivot(v2, ini, fin);
```

```
                int medio = (ini + fin) / 2;
```

```
                Qsort(v1, v2, ini, medio);
```

```
                Qsort(v1, v2, medio + 1, fin);
```

```
            }
```

```
        }
```

5) Subsequence mayor de un vector

```
vector<int> SubsequenceMayor(const vector<int> &v, int ini,
                             int fin) {
```

```
    vector<int> res;
```

```
    if (ini == fin) {
```

```
        res.push_back(v[ini]);
```

```
    } else {
```

```
        vector<int> lorig, lanch; int mited, lcentro = 0, c;
```

```
        mited = (ini + fin) / 2;
```

```
        lorig = SubsequenceMayor(v, ini, mited);
```

```
        lanch = SubsequenceMayor(v, mited + 1, fin);
```

```
        if (v[mited] < v[mited + 1]) {
```

```
            c = mited; lcentro = 2;
```

```
            while (c > ini && v[c] > v[c - 1]) {
```

```
                lcentro++;
```

```
                c--;
```

```
                res.push_back(v[c]);
```

```
            }
```

```
            c = mited + 1;
```

```
            while (c < fin && v[c] < v[c + 1]) {
```

```
                lcentro++;
```

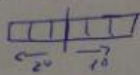
```
                c++;
```

```
                res.push_back(v[c]);
```

```
            }
```

```
        }
```

```
        // Ordenar res
```

(si la mayor secuencia está en el centro, res
guarda la secuencia pero en orden: 

```
        if (lorig.size() > lcentro && lorig.size() > lanch.size())
```

```
            return lorig;
```

```
        if (lanch.size() > lcentro && lanch.size() > lorig.size())
```

```
            return lanch;
```

```
    return res;
```


6) Suma mayor de secuencia

```
int sumaMayor(const vector<int> &v, int ini, int fin){
    int suma = 0;
    if (ini == fin)
        suma = v[ini];
    else{
        int sumadch, sumaezg, mitad, sumacentro = 0, c;
        mitad = (ini + fin) / 2;

        sumaezg = sumaMayor(v, ini, mitad);
        sumadch = sumaMayor(v, mitad + 1, fin);
        c = mitad + 1; int valor = 0;
        while (c < fin && (valor + sumacentro) > sumacentro){
            sumacentro += valor;
            valor = v[c];
            c++;
        }
        c = mitad; valor = 0;
        while (c > ini && (valor + sumacentro) > sumacentro){
            sumacentro += valor;
            valor = v[c];
            c--;
        }
        suma = sumacentro;
        if (sumaezg > sumacentro && sumaezg > sumadch)
            suma = sumaezg;
        if (sumadch > sumacentro && sumadch > sumaezg)
            suma = sumadch;
    }
    return suma;
}
```