Relación Divide y Vencerás

Rubén Calvo Villeretón - Algorítmica G.D

Ejercicios:

1) a[k] = k

```
int seCumple(const vector<int> &v, int ini, int fin){
    int pos = -1;
    int mitad = (ini + fin) / 2;
    if(v[ini] == ini){
        pos = ini;
    }
    if(ini != fin){
        int pos1 = seCumple(v, ini, mitad);
        int pos2 = seCumple(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 intercambio;
    int pos_pivote = primero;
    int valor_pivote = v[primero];
  for(int i = primero + 1; i <= ultimo; i++){
      if( v[i] < valor_pivote){
          pos_pivote++;
          intercambio = v[i];
          v[i] = v[pos_pivote];
          v[pos_pivote] = intercambio;
      }
  }
  intercambio = v[primero];
  v[primero] = v[pos_pivote];
  v[pos_pivote] = intercambio;

  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 OrdenaResultados(vector< vector<int> > res){
    for(int i = 0; i < res.size(); i++){
        QuickSort(res[i], 0, res[i].size());
    }
}
```

4) tornillos y tuercas

```
// Suponiendo implementada la función pivote
void Unir(vector<int> &v1, vector<int> &v2, int ini, int fin){
    if(fin - ini != 0){
        int piv = Pivote(v1, ini, fin);
        int elem, pos;
        bool enc = false;
        for(int i=0; i< v2.size() && !enc; i++)
            if(v2[i] == v1[piv]){
                pos = i;
                enc = true;
            }
        elem = v2[pos];
        v2[pos] = v2[ini];
        v2[ini] = elem;

        Pivote(v2, ini, fin);
        int mitad = (ini + fin)/2;
        Unir(v1, v2, ini, mitad);
        Unir(v1, v2, mitad+1, fin);
    }
}
```

5) Subsecuencia mayor de un vector

```
vector<int> SubsecuenciaMayor(const vector<int> &v, int ini,
                                                    int fin){
    vector<int> res;
    if(ini == fin){
        res.push_back(v[ini]);
    }else{
    vector<int> lenizq, lendch; int mitad, lencentro = 0, c;
        mitad = (ini+fin)/2;
        lenizq = SubsecuenciaMayor(v, ini, mitad);
        lendch = SubsecuenciaMayor(v, mitad+1, fin);
        if(v[mitad] < v[mitad+1]){
            c = mitad; lencentro = 2;
            while(c > ini && v[c] > v[c-1]){
                lencentro++;
                c--;
                res.push_back(v[c]);
            }
            c = mitad+1;
            while(c < fin && v[c] < v[c+1]){
                lencentro++;
                c++;
                res.push_back(v[c]);
            }
        }
    //Ordenar res (si la mayor secuencia está en el centro, res
                 guarda la secuencia pero en orden:)
    if(lenizq.size() > lencentro && lenizq.size() > lendch.size())
        return lenizq;
    if(lendch.size() > lencentro && lendch.size() > lenizq.size())
        return lendch;

    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 sumaIzq, sumaDch, mitad, sumaCentro = 0, c;
        mitad = (ini + fin)/2;

        sumaIzq = sumaMayor(v, ini, mitad);
        sumaDch = sumaMayor(v, mitad+1, fin);
        i = mitad+1; int valor = 0;
        while( i < fin && (valor + sumaCentro) > sumaCentro){
            sumaCentro += valor;
            valor = v[i];
            i++;
        }
        c = mitad; valor = 0;
        while (i > ini && (valor + sumaCentro) > sumaCentro){
            sumaCentro += valor;
            valor = v[i];
            i--;
        }
    }
    suma = sumaCentro;
    if(sumaIzq > sumaCentro && sumaIzq > sumaDch)
        suma = sumaIzq;
    if(sumaDch > sumaCentro && sumaDch > sumaIzq)
        suma = sumaDch;

    return suma;
}
```