

Universidade Federal de Itajubá

-UNIFEI-

-Itajubá-

-2015-

- Pedro Henrique Dias Scarpioni – 31075.

- Prof. Dr. Roberto Claudino – Algoritmos e Grafos

- Bacharelado em Sistemas de Informação.

- Instituto de Matemática e Computação (IMC).

- Universidade Federal de Itajubá (UNIFEI). – Itajubá – MG- Brasil.

===============================================

**1)**

void unint(int \*a,int \*b,int m,int n,int \*inter,int \*uniao,int \*k,int \*u){

int i,j,r,aux;

for(i = 0; i < n; i++){

r = busca\_binaria(a,0,m-1,\*(b+i));

if(r != -1){

\*(inter + \*k) = \*(a + r);

\*k += 1;

}

\*(uniao + \*u) = \*(b + i);

\*u += 1;

}

for(i = 0; i < m; i++){

\*(uniao + \*u) = \*(a + i);

\*u += 1;

}

}

Percorrendo o segundo vetor e fazendo uma busca binária no primeiro, comparando com o i-ésimo elemento do segundo vetor, conseguimos uma complexidade de **O(nlgm)**.

**2)**

Árvore de recursão do Mergesort para um vetor de 16 elementos:

**D E G H A B I K L M C N O F P J**

**/ \**

**D E G H A B I K L M C N O F P J**

**/ \ / \**

**D E G H A B I K L M C N O F P J**

**/ \ / \ / \ / \**

**D E G H A B I K L M C N O F P J**

**/ \ / \ / \ / \ / \ / \ / \ / \**

**D E G H A B I K L M C N O F P J**

**\/ \/ \/ \/ \/ \/ \/ \/**

**DE GH AB IK LM CN FO JP**

**\/ \/ \/ \/**

**DEGH ABIK CLMN FJOP**

**\/ \/**

**ABDEGHIK CFJLMNOP**

**\/**

**ABCDEFGHIJKLMNOP**

A programação dinâmica não é capaz de acelerar o Mergesort, devido a esta ser aplicável a problemas nos quais a solução ótima pode ser computada a partir da solução ótima previamente calculada e memorizada - de forma a evitar recálculo - de outros subproblemas que, sobrepostos, compõem o problema original.

O que um problema de otimização deve ter para que a programação dinâmica seja aplicável são duas principais características:

1. **subestrutura ótima**, é quando uma solução ótima para o problema contém em seu interior soluções ótimas para subproblemas.
2. **superposição de subproblemas**, que acontece quando um algoritmo recursivo reexamina o mesmo problema muitas vezes.

No caso da ordenação, há ocorrência de vários subproblemas diferentes entre si, ou seja, os elementos ordenados são diferentes, não ocorrendo assim um novo exame do mesmo problema.

**3)**

**a)**

Para a função A o cálculo da analise combinatória terá uma complexidade exponencial → **O(2n)** . Devido a esta representar uma recursão burra onde a superposição de problemas.

Já para a função B, também para o cálculo de análise combinatória, ele teria uma complexidade quadrática → **O(n2)**. Devido as duas iterações que ela realiza no pior caso.

**b)**

A função B é a mais eficiente, por ter uma complexidade menor que a função A, e pelo fato que na função A há problemas que poderiam ser reaproveitados se houvesse a aplicação da programação dinâmica, diminuindo-se assim a sua complexidade.

**4)**

para i de 0 até n faça

para j de 0 até n faça

para x de 0 até i faça

para y de 0 até j faça

aux[i][j] + = M[x][y]

y = y + 1

x = x + 1

j = j + 1

i = i + 1