EDPAA 2025/2 - Laboratório 07 - Divisão e Conquista
Prof. Luis Souza

O projeto de muitos algoritmos eficientes é baseado no método da divisão e conquista. Esse método (ou estratégia de projeto de algoritmos) consiste no seguinte:

- a instância dada do problema é dividida em duas ou mais instâncias menores,
- cada instância menor é resolvida usando o próprio algoritmo que está sendo definido,
- as soluções das instâncias menores são combinadas para produzir uma solução da instância original.

A segunda fase é implementada por uma chamada recursiva. Essa é a fase da conquista.

O método da divisão e conquista produz um algoritmo eficiente se a fase de divisão e a fase da combinação forem suficientemente rápidos.

Exemplos:

- Algoritmo Altura-DC para o problema do segmento de soma máxima,
- Busca binária: divide a instância em duas menores e resolve uma delas; a fase de combinação é vazia.
- Mergesort: divide a instância em duas menores (essa fase é muito rápida) e resolve as duas instâncias menores; a fase de combinação é a que consome mais tempo.
- Quicksort: a fase da divisão é lenta e produz duas instâncias menores; a fase de combinação é muito rápida.
- Algoritmo da mediana: a fase da divisão, que produz duas instâncias menores, é lenta; a fase da conquista resolve uma dessas instâncias; a fase de combinação é muito rápida.
- Algoritmo de Karatsuba: a fase da divisão é muito rápida e produz três instâncias menores; a fase de combinação consiste em algumas operações aritméticas.

Exercícios:

1. Aplique a estratégia da divisão e conquista ao problema de calcular a soma dos elementos de um vetor A[1..n] de números inteiros. Estime o consumo

de tempo do algoritmo. Compare o resultado com o consumo de tempo do algoritmo trivial de soma.

2. Escreva e implemente um algoritmo de divisão e conquista para encontrar o valor de um elemento máximo de um vetor A[p..r] de números inteiros. Seja n o número $r-p+1$ e C(n) o número de comparações que seu algoritmo faz entre elementos do vetor. Escreva a recorrência que C(n) satisfaz. Resolva a recorrência utilizando o teorema mestre.
3. Distância τ de Kendall. Suponha dadas duas permutações, digamos A[1..n] e B[1..n], de um mesmo conjunto de números. A distância τ de Kendall entre A e B é o número de pares de elementos do conjunto que estão em ordem diferente em A e B, ou seja, o número $|X - Y|$ onde X é o conjunto de todos os pares (A[i], A[j]) tais que i < j e Y é o conjunto de todos os pares (B[i], B[j]) tais que i < j. (A definição não é assimétrica pois $|X - Y| = |Y - X|$.) Escreva uma função eficiente que calcule a distância τ de Kendall entre A e B.

   A distância τ de Kendall entre duas permutações A e B pode ser calculada usando divisão e conquista ao transformarmos o problema em uma contagem de inversões. A ideia é:

   - Construir, a partir de B, um vetor pos tal que pos[x] é a posição do elemento x em B.
   - Reescrever A substituindo cada elemento A[i] por pos[A[i]].
   - Isso produz um vetor C que indica a ordem dos elementos de A segundo a ordem em B.
   - A distância de Kendall é exatamente o número de inversões de C.
   - Contamos inversões usando merge sort modificado, em O(n log n).

4. Dado um conjunto de pontos no espaço bidimensional, você deve encontrar a distância entre os dois pontos mais próximos. **Entrada:** O arquivo de entrada consiste de vários casos de teste. Cada começa com um inteiro N ($0 \leq N \leq 10000$), que denota o número de pontos no teste. As próximas N linhas contém as coordenadas dos N pontos bidimensionais. O primeiro inteiro denota a coordenada X e o segundo a coordenada Y de cada ponto. A entrada é terminada por um caso de teste com N = 0. O valor de cada coordenada deverá ser sempre maior ou igual a zero e menor que 40000. **Saída:** Para cada caso de teste deve ser impressa uma única linha contendo um ponto flutuante (com quatro casas decimais) representando a menor distância entre dois pontos. Se não houver nenhum par de pontos com distância menor que 10000, devem ser impressos os caracteres "INFINITY".

   A solução:

   - Ordena os pontos por x.
   - Divide o conjunto em duas metades.
   - Resolve recursivamente.

- Combina os resultados considerando apenas pontos próximos da linha divisória.
- É O(n log n).

Exemplos de Entrada	Exemplos de Saída
3 0 0 10000 10000 20000 20000	INFINITY
5 0 2 6 67 43 71 39 107 189 140 0	36.22