Notas de Aula - AED1 – Algoritmos de Ordenação (parte 3)
Prof. Jefferson T. Oliva

Neste material são apresentados mais dois algoritmos de ordenação avançada: Shell sort e intercalação (*mergesort*)

Shell sort

Extensão do insert sort

Contorna o principal problema do insertion sort possibilitando troca de registros que estão distantes um do outro.

Tem como objetivo aumentar o passo de movimento dos elementos ao invés das posições adjacentes.

Consiste em classificar sub-arranjos do original.

Esses sub-arranjos contêm todo h-ésimo elemento do arranjo original.

O valor de h é chamado de incremento.

Por exemplo, se h é 5, o sub-arranjo consiste dos elementos x[0], x[5], x[10], etc
- Sub-arranjo 1: x[0], x[5], x[10]
- Sub-arranjo 2: x[1], x[6], x[11]
- Sub-arranjo 3: x[2], x[7], x[12]
- Sub-arranjo 4: x[3], x[8], x[13]

Após a ordenação dos sub-arranjos:
- Define-se um novo incremento menor que o anterior
- Gera-se novos sub-arquivos
- Aplica-se novamente o método da inserção

O processo é realizado repetidamente até que h seja igual a 1

O valor de h pode ser definido de várias formas, por exemplo
- $h(s) = 3h(s - 1) + 1$, para $s > 1$
- $h(s) = 1$, para $s = 1$

Implementação

```
void shellsort(int v[], int n){
        int h = 1;
        int x, i, j;

        while (h < n)
                h - 3 * h + 1;

        h /= 3;

        while (h >= 1){
                for (i = h; i < n; i++){
                        x = v[i];
                        j = i;

                        while ((j >= h) && (x < v[j - h])){
                                v[j] = v[j - h];
                                j -= h;
                        }

                        v[j] = x;
                }

                h /= 3;
        }
}
```

Ver exemplo no slide 8.

Um problema com o shell sort ainda não resolvido é a escolha dos incrementos que fornecem os melhores resultados.

É desejável que ocorra o maior número possível de interações entre as diversas cadeias.

Ótima opção para arquivos de tamanho moderado.

Tempo de execução sensível à ordem dos dados.

O algoritmo não é estável.
Custo de tempo não é conhecido, mas estima-se aproximadamente:
- Pior caso: O(n^2)
- Melhor caso: O(n ^{1,25})

Ordenação por Intercalação (Mergesort)

Baseado no paradigma de projeto de algoritmos divisão e conquista, o mergesort divide o arranjo em dois até que cada um tenha tamanho 1. Em seguida, os sub-arranjos são intercalados até a ordenação completa.

Se o vetor tiver o tamanho igual a 1, a função apenas retorna o elemento

Caso o tamanho do vetor seja maior que 1
1 - Divisão: divida o vetor ao meio
2 - Conquista 1: ordene a primeira metade recursivamente
3 - Conquista 2: ordene a segunda metade recursivamente
4 - Combinação: intercale as duas metades

Implementação

```
void merge(int v[], int esq, int meio, int dir){
   int i, j, k;
   int n1 = meio - esq + 1;
   int n2 = dir - meio;
   int L[n1 + 1];
   int R[n2 + 1];

   for (i = 0; i < n1; i++)
      L[i] = v[esq + i];

   for (j = 0; j < n2; j++)
      R[j] = v[meio + j + 1];

   L[n1] = INT_MAX;
   R[n2] = INT_MAX;

   i = 0;
   j = 0;

   for (k = esq; k <= dir; k++)
      if (L[i] <= R[j]){
         v[k] = L[i];
         i++;
      }else{
         v[k] = R[j];
         j++;
      }
}
void mergesort(int v[], int esq, int dir){
   int meio;

   if (esq < dir){
      meio = (esq + dir) / 2;
      mergesort(v, esq, meio);
      mergesort(v, meio + 1, dir);
      merge(v, esq, meio, dir);
   }
```

}

Ver exemplo entre os slides 15 e 33.

A complexidade do mergesort é O(n log n) no pior caso

Referências

Cormen, T. H., Leiserson, C. E., Rivest, R. L., Stein, C. Introduction to Algorithms. Third edition, The MIT Press, 2009.

Horowitz, E., Sahni, S. Rajasekaran, S. Computer Algorithms. Computer Science Press, 1998.

Rosa, J. L. G. Métodos de Ordenação. SCE-181 - Introdução à Ciência da Computação II. Slides. Ciência de Computação. ICMC/USP, 2018.

Ziviani, N. Projeto de Algoritmos - com implementações em Java e C++. Thomson, 2007.