Ordenação: Mergesort

Algoritmos e Estruturas de Dados II

Introdução

- Baseado em *merging*
  - Combinação de dois vetores ordenados em um vetor maior que também esteja ordenado

- Quicksort vs. Mergesort
  - **Quicksort**:
    - divide o vetor em vetores independentes
    - Indexação da posição do pivô + duas chamadas recursivas
  - **Mergesort**:
    - une dois vetores para criar um único
    - Duas chamadas recursivas + procedimento para unir vetores

Merging

- Dois vetores **a** e **b** ordenados para um vetor **c**
- **Ideia**
  - Escolhe para **c**, o menor de dos elementos que ainda não foram escolhidos dos vetores **a** e **b**.

```
mergeAB(Item c[], Item a[], int N, Item b[], int M ) {
int i, j, k;
    for (i = 0, j = 0, k = 0; k < N+M; k++) {
        if (i == N) { c[k] = b[j++]; continue; }
        if (j == M) { c[k] = a[i++]; continue; }
        if (a[i].chave < b[j].chave)
            c[k] = a[i++];
        else
            c[k] = b[j++];
    }
}
```

Merging

- **Problema**
  - Há dois testes no laço interno.
  - Dois vetores separados são passados (a e b)

- **Solução**
  - Para evitá-los, copia um dos vetores em ordem reversa e o percorre da direita para esquerda.
  - Passa vetor único, indicando o índice do último elemento do vetor da esquerda (variável m).

Merging

```
Item aux[maxN];

merge(Item a[], int e, int m, int d){
int i, j, k;

    /* copia a e b (reverso) para vetor auxiliar */
    for (i = 0; i <= m; i++)
        aux[i] = a[i];
    for (j = m+1; j <= d; j++)
        aux[d-m+j+1] = a[j];
    i = e; j = d;
    for (k = e; k <= d; k++)
        if (aux[i].chave <= aux[j].chave)
            a[k] = aux[i++];
        else
            a[k] = aux[j--];
}
```

Mergesort

```
void mergesort(Item a[], int e, int d)  {
int m = (d+e)/2;

    if (d <= e) return;
    mergesort(a, e, m);
    mergesort(a, m+1, d);
    merge(a, e, m, d);

}
```

Mergesort não Recursivo

```
#define min(A, B) (A < B) ? A : B

void mergesortBU(Item a[], int e, int d) {
int i, m;

    for (m = 1; m < d-e; m = m+m)
      for (i = e; i <= d-m; i += m+m)
        merge(a, i, i+m-1, min(i+m+m-1, d));
}
```

Considerações

- **Vantagens**
  - Ordena vetor com N elementos e tempo proporcional a NlogN, não importa a entrada.
  - Deve ser considerado quando alto custo de pior caso não pode ser tolerável.
  - Método de ordenação estável.

- **Desvantagens**
  - Requer espaço extra proporcional a N.
  - Não é adaptável (tempo de execução independe dos dados da entrada).