Quick Sort

- Mais rápido algoritmo de Ordenação (não paralelo)
- Baseia-se na ordenação de sub-listas divididas por um pivô
- O pivô é um elemento escolhido ao acaso, que é colocado na sua posição correta quando a lista estiver ordenada.
- Implementação complexa, não muito melhor para listas de pequeno tamanho (<100 elementos)

Exemplo

- Lista Inicial (Desordenada)
  - 25 48 37 12 57 86 33 92
  - 1a. Iteração:
    - Pivô: 25
    - Elemento da Esquerda: 48
    - Elemento da Direita: 12
  - 25 12 37 48 57 86 33 92
    - Elemento da Esquerda: 37
    - Elemento da Direita: 12 (Cruzamento de direita e esquerda)
    - Troca de Pivô por elemento da direita

Exemplo (Continuação)

- 2a. Iteração:
  - 12 25 37 48 57 86 33 92
  - Antigo pivô (Elemento 25) está na posição correta.
  - Basta ordenar sub-listas:
    - 12 (1 elemento = lista já ordenada)
    - 37 48 57 86 33 92

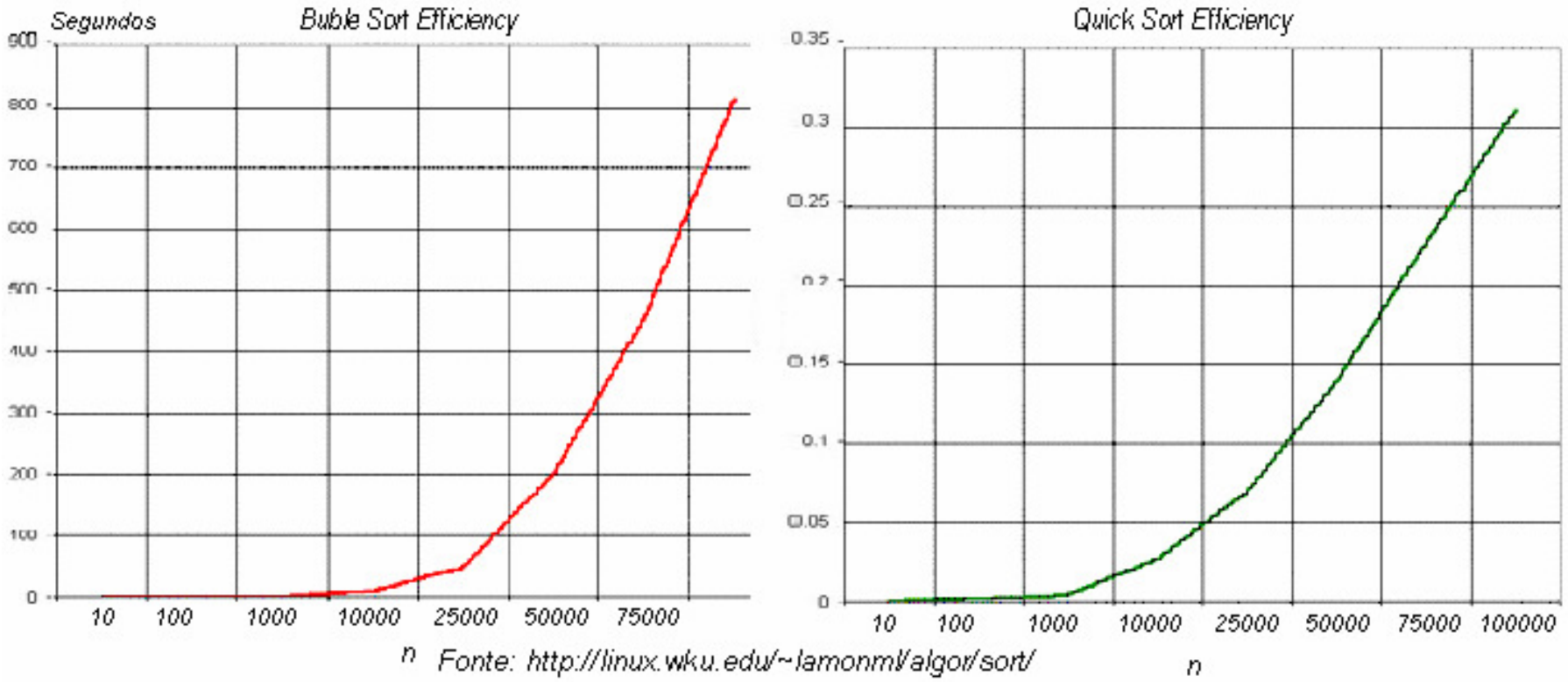
Exemplo (Continuação 2)

- Ordenar sub-lista
  - 37 48 57 86 33 92
    - Pivô: 37
    - Elemento da Esquerda:48
    - Elemento da Direita: 33
  - 37 33 57 86 48 92
    - Elemento da Esquerda: 57
    - Elemento da Direita: 33 (Cruzamento->Troca de Pivô)
  - 33 37 57 86 48 92 (Pivô na posição correta)]
    - Sub-listas:
      - 33 (1 elemento já ordenado)
      - 57 86 48 92 (ordenar seguindo o mesmo algoritmo)

Quick Sort em Vetores

```
void quick_sort(int vetor[],int left,int right){
    if (right<= left) return;
    else {
        int pivot = vetor[left];     int inicio=left;
        int fim=right;
        do {
                    while (left <=right && vetor[left] <= pivot) left++;
                    while (vetor[right] > pivot) right--;
                    if (left < right) { /* faz troca */
                                int temp = vetor[left];
                                vetor[left] = vetor[right];
                                vetor[right] = temp;
                                left++; right--;
                    }
        } while (left <= right);
        /* troca pivot */
        vetor[inicio] = vetor[right];
        vetor[right] = pivot;       pivot=right;
        /* ordena sub-vetores restantes */
        if(inicio<pivot)  quick_sort(vetor,inicio,pivot-1);
        if(pivot<fim)                   quick_sort(vetor,pivot+1,fim);
    }   }
```

Comparação de Desempenho



Fonte: http://linux.wku.edu/~lamonml/algor/sort/

Um pouco de desordem...

- Função para desordenar um vetor..
  - Útil em certos tipos de aplicação. Exemplo: jogos de carta,..

```
void desordena(int v[],int tam) {
randomize();
 int aux; int x;
 for(int i=0;i<tam;i++) {
    x=rand()%tam;
    aux=v[i];
    v[i]=v[x];
    v[x]=aux; }    }
```

Quick Sort em Listas Duplamente Encadeadas

```
void quick_sort(node* left,node* right){
                        if(!validos(left,right))
                                return ;
        cadeia pivot; strcpy(pivot,left->nome);  node* inicio=left;    node* fim=right;  node* aux;          int
    cruzou=0;
        do {
                        while (!cruzou  && strcmp(left->nome,pivot)<=0 ) {
                         if(left==right)
                                cruzou=1;
                         left=left->prox;
                        }
                        while (strcmp(right->nome,pivot)>0) {
                                if(left==right)
                                 cruzou=1;
                                right=right->prev;
                        }
                        if (!cruzou) { // faz troca /
                                swap3(left->prev,right->prev);
                                aux=left;       left=right;                     right=aux;
                        }
        } while (!cruzou);
```

Quick Sort em Listas Duplamente Encadeadas (Continuação)

```
    // troca pivot
     swap3(inicio->prev,right->prev);
     aux=inicio;     inicio=right;    right=aux;
     // ordena sub-vetores restantes
     if(inicio!=right)   quick_sort(inicio,right->prev);
     if(right!=fim)      quick_sort(right->prox,fim);
}
```

Quick Sort em Listas Duplamente Encadeadas (Versão Cópia)

```
void quick_sort_copia_dados(node* left,node* right){
                if(!validos(left,right))
                        return ;
        cadeia pivot; strcpy(pivot,left->nome);      node* inicio=left;
        node* fim=right; cadeia aux;        int cruzou=0;
        do {      while (!cruzou &&strcmp(left->nome,pivot)<=0 ) {
                   if(left==right)      cruzou=1;
                   left=left->prox;
                  }
                  while (strcmp(right->nome,pivot)>0) {
                    if(left==right)  cruzou=1;
                            right=right->prev;
                  }
        ..............
```

Quick Sort em Listas Duplamente Encadeadas (Versão Cópia) Continuação

```
.............
    if (!cruzou) { // faz troca /
                    strcpy(aux,left->nome);
                   strcpy(left->nome,right->nome);
                    strcpy(right->nome,aux);
                   }
          } while (!cruzou);
          // troca pivot
           strcpy(aux,inicio->nome);
           strcpy(inicio->nome,right->nome);
           strcpy(right->nome,aux);
          // ordena sub-vetores restantes
          if(inicio!=right)  quick_sort_copia_dados(inicio,right->prev);
          if(right!=fim)  quick_sort_copia_dados(right->prox,fim);
}
```

Verificação de Validade de Ponteiros Left e Right

```
int validos(node* left,node* right) {
  if(left==NULL || right==NULL || left==right)
     return 0;
  while(left->prox!=NULL) {
     left=left->prox;
     if(left==right)
          return 1;
  }
  return 0;
}
```