Gerenciamento de Memória

Aleardo Manacero Jr.

Introdução

- Até agora examinamos estruturas considerando apenas sua organização

- Com isso estudamos os algoritmos para a manipulação básica dos elementos dentro de cada estrutura

- Nesse momento é preciso examinar um outro aspecto relacionado às estruturas de dados, seu armazenamento

Introdução

- Na prática quando definimos uma dada estrutura de dados estamos dizendo ao compilador duas coisas:
  - Quais operações são aplicáveis à estrutura
  - Como se deve reservar e manipular espaços na memória para a mesma

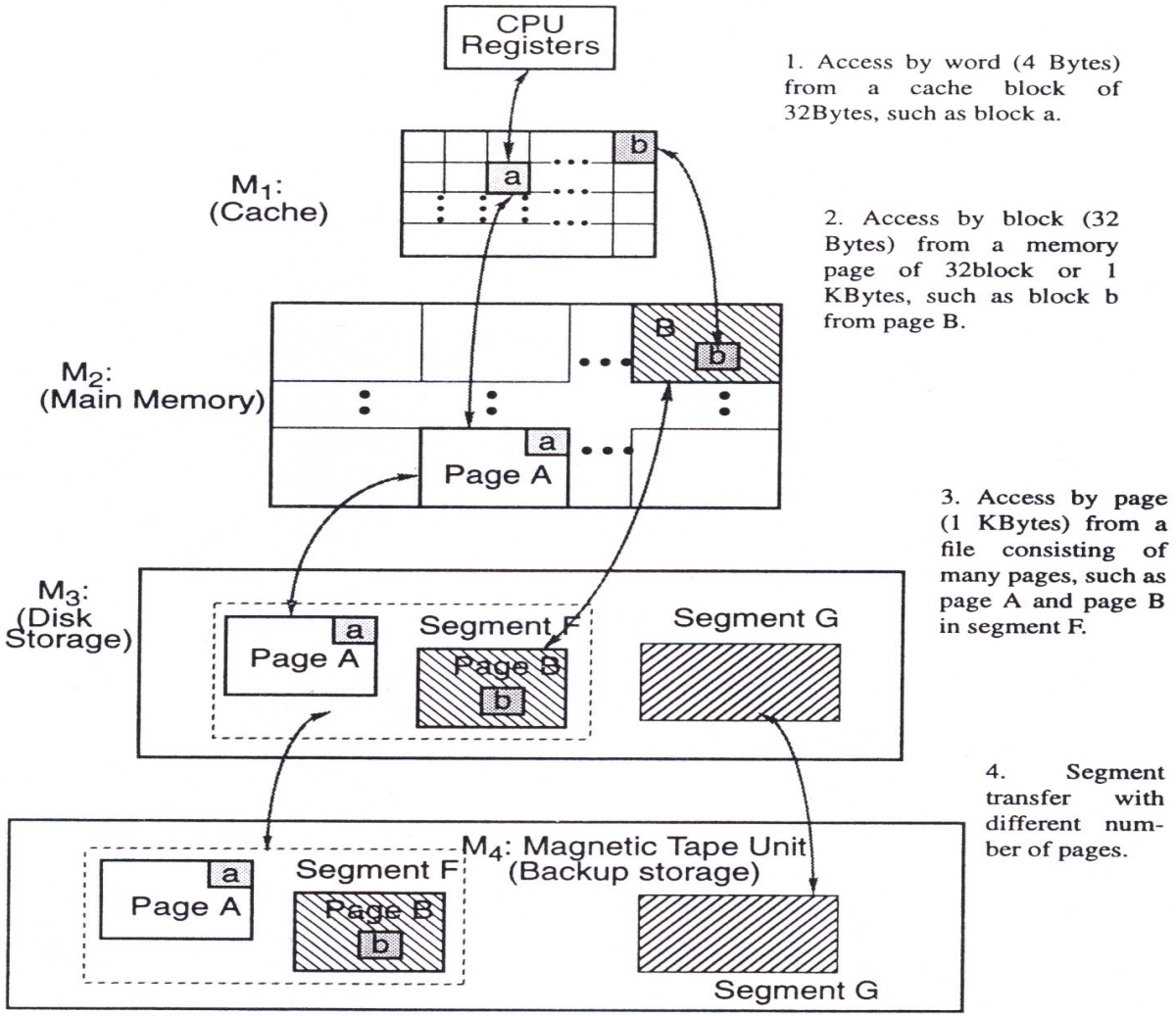
- Temos trabalhado apenas com a primeira delas. Examinaremos a segunda agora

A memória

- Podemos enxergar a memória de vários pontos de vista:
  - Sistema operacional
  - Arquitetura de computadores
  - Aplicações

- Faremos um exame rápido dos dois primeiros e depois nos concentraremos no estudo da memória a partir de suas aplicações

Níveis arquitêtonicos



Memória e o SO

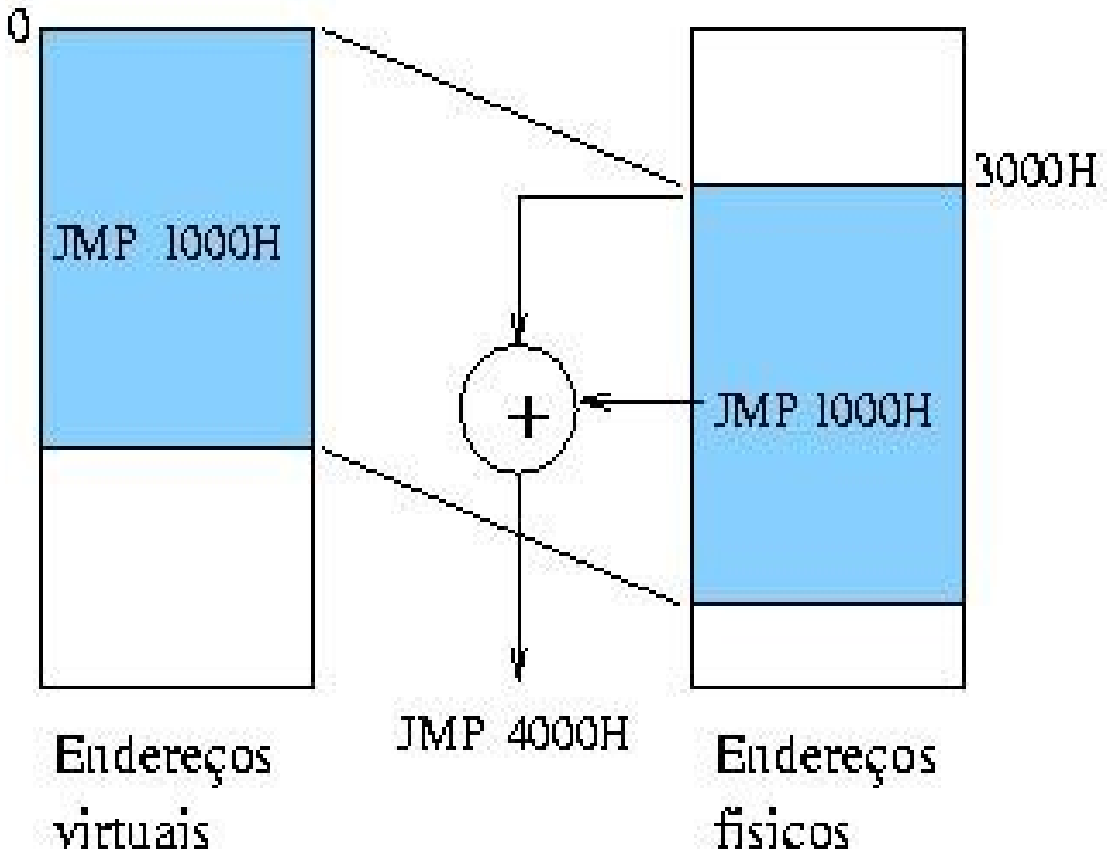
- O SO é responsável pela alocação de espaços na memória e pelos mecanismos de endereçamento de conteúdos

- Parte de seu trabalho envolve também a manipulação de *cache* e de memória virtual

- Nos dois casos (MV e *cache*) a base de seu trabalho é o princípio da localidade

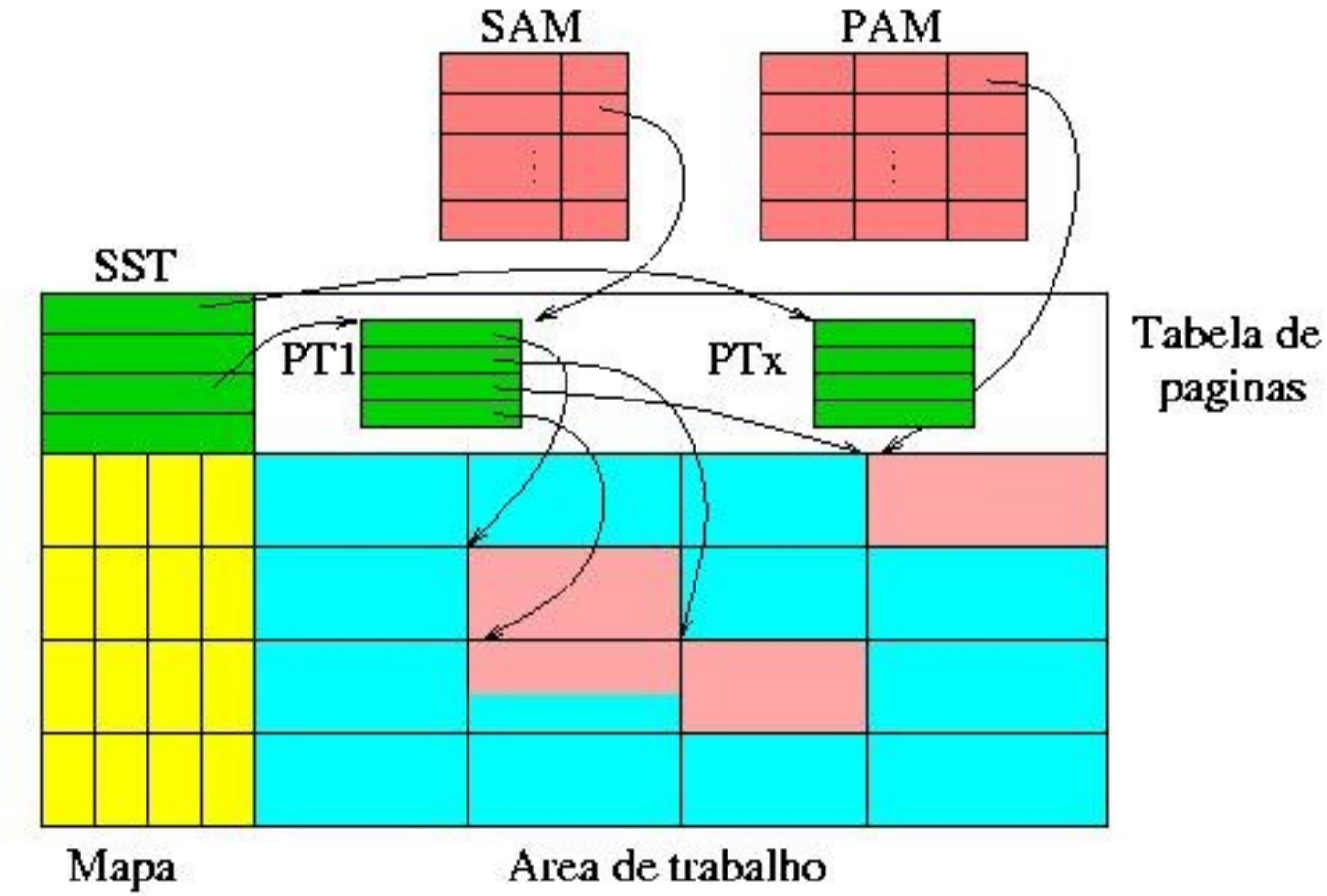
Memória e o SO

- Endereçamento por relocação dinâmica



Memória e o SO

- Funcionamento de memória em blocos



Princípio da localidade

- O gerenciamento de memória apenas apresenta bons resultados quando se atende o Princípio da Localidade

- O Princípio da Localidade diz que os acessos às páginas de memória ocorrem quando sempre em endereços relativamente vizinhos (próximos)

Princípio da localidade

- A eficiência da memória está, portanto, dependente do atendimento do princípio da localidade

- Esse atendimento depende, na prática, da forma como arranjamos os dados e como os mesmos são manipulados

- Isso nos leva de volta às estruturas de dados

Memória e ED

- A interação entre uma estrutura de dados e a memória ocorre em dois momentos:
  - Criação da estrutura, ou alocação de espaço para a mesma
  - Manutenção dos espaços alocados

- Não examinaremos aqui as formas antigas de alocação de memória (em espaços contíguos), nos dedicando apenas ao conceito de memória paginada

Memória e ED

- Dentro desse contexto, o aspecto mais importante passa a ser a manutenção dos espaços alocados

- Essa operação recebe o nome de coleta de lixo, do inglês garbagge collection

- A correta aplicação de coleta de lixo aumenta a garantia de atendimento do princípio da localidade

A coleta de lixo

- Os procedimentos de coleta de lixo atuam, basicamente, em duas etapas

- Primeiro são marcados os elementos da estrutura que estejam em uso

- Depois os demais elementos são removidos e reagrupados os que estão em uso de forma a compactar os elementos da estrutura

Técnicas de marcação

- O procedimento de marcação é similar ao de caminho em árvores

- Nele podemos tratar a memória como sendo uma lista de listas, em que cada elemento pode ser um átomo ou outra lista

- Durante a marcação percorremos essa lista, marcando os átomos em uso (“apontados” por alguém

Técnicas de marcação

- Pode ser feito recursivamente (usando a pilha de execução) ou iterativamente (usando ou não uma pilha explícita)

- No processo recursivo garante-se que todo o espaço foi percorrido através da manutenção da pilha de execução das chamadas recursivas

- Seu problema é o estouro dessa pilha numa lista complexa

Técnicas de marcação

- Os algoritmos iterativos podem ou não fazer uso de pilhas

- Se o uso de pilha for explícito podemos ter também problemas no estouro de pilha

- Isso ocorre pois temos que usar estruturas estáticas no processo

Técnicas de marcação

- Algoritmos que não usam pilhas operam com a inversão de ponteiros ao percorrer sub-listas (para guardar o caminho de volta)

- Um exemplo é o algoritmo de Schorr e Waite (também conhecido como Deutsch Schorr, Waite ou outro DSW)

Regeneração de espaço

- Após a marcação dos elementos em uso o processo parte para a eliminação (compactação) dos elementos não mais em uso

- Isso pode ser feito através de duas listas ou através de cópia para um novo espaço

Regeneração de espaço

- Uma técnica faz uso de uma lista para os elementos marcados e outra para os vazios

- Nesse mecanismo o que se faz é percorrer simultaneamente as duas listas, movendo-se os elementos marcados cada vez que se identifica um espaço vazio

Regeneração de espaço

- Nos métodos de cópia o que se faz é copiar os espaços marcados para uma nova região

- Nessa operação todos os espaços marcados passam a ficar armazenados de forma contígua, eliminando os vazios existentes

Coletores incrementais

- Os métodos de coleta de lixo do SO trabalham a partir de momentos críticos de execução, quando a memória se torna escassa

- Com isso, durante o processo de coleta, os processos ficam bloqueados até que a memória recupere um estado confiável

Coletores incrementais

- Uma técnica para diminuir esse problema é fazer a coleta incremental

- Na coleta incremental, as operações de marcação e regeneração intercaladas, permitindo que os programas fiquem bloqueados por um menor intervalo de tempo

Compressão de dados

Compressão de dados

- A coleta de lixo serve para economizar o uso de memória por parte dos programas

- Isso, entretanto, não significa uma redução no espaço ocupado por dados quando armazenados em discos ou transmitidos ao longo de redes de computadores

- Para essa redução é preciso o uso de compressão de dados

Compressão de dados

- A idéia por trás da compressão de dados é usar códigos diferenciados para os símbolos a serem transmitidos

- Esses novos códigos devem considerar a freqüência de cada símbolo, fazendo com que símbolos frequentes tenham códigos menores

Compressão de dados

- A estruturação dos símbolos segundo sua probabilidade de ocorrência permite o cálculo da entropia do código, dada por:

$$L_{médio} = P(m_1).L(m_1) + \ldots + P(m_n).L(m_n)$$

- Em que P(m) é a probabilidade de um símbolo e L(m) o comprimento de seu código

Compressão de dados

- A compressão de dados deve atender algumas restrições:

1. Cada símbolo tem exatamente um código
2. Não se deve exigir olhar à frente no processo
3. O comprimento de um código mais freqüente não pode ser maior do que o de um menos freqüente
4. Deve-se esgotar os códigos possíveis de um comprimento antes de usar um código maior

Exemplos de códigos

- Huffman
- Shannon-Fano
- Liv-Zempel

Exemplos de códigos

- Os códigos de Huffman e Shannon-Fano necessitam de informações sobre as freqüências de ocorrências de cada símbolo antes da codificação

- Já o código de Liv-Zempel pode construir essas freqüências durante a codificação da informação

Código de Huffman

- Trabalha criando uma árvore com os símbolos, ordenados de forma crescente de probabilidade

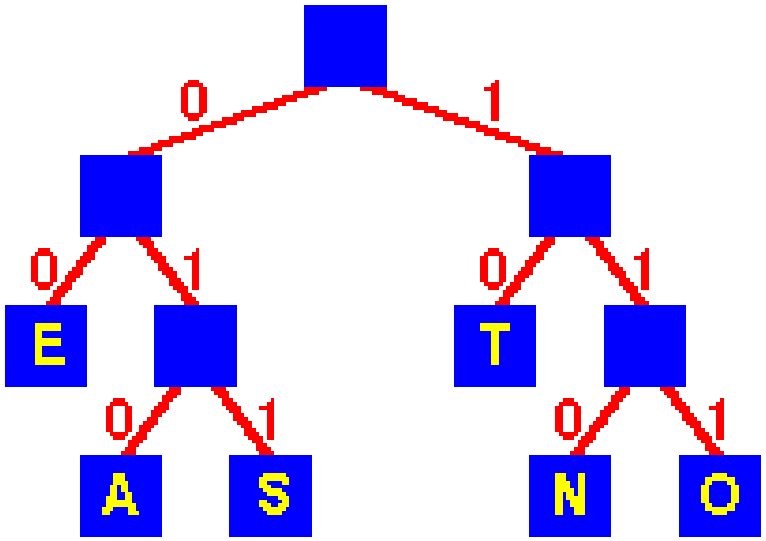
- Associa então os bits 0 e 1 para cada ramo da árvore, sendo cada símbolo codificado pela leitura dos bits a partir da raiz até ele

Código de Huffman

- Codificação
  - Use uma árvore
  - Codifique da raiz para folhas
  - Como em
    - E é 00
    - S é 011
  - Caracteres freqüentes

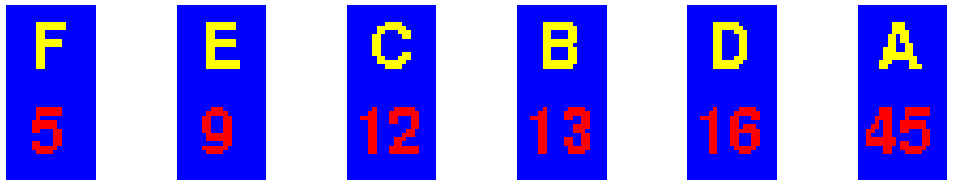
    E, T 2 bits
  - Outros

    A, S, N, O 3 bits



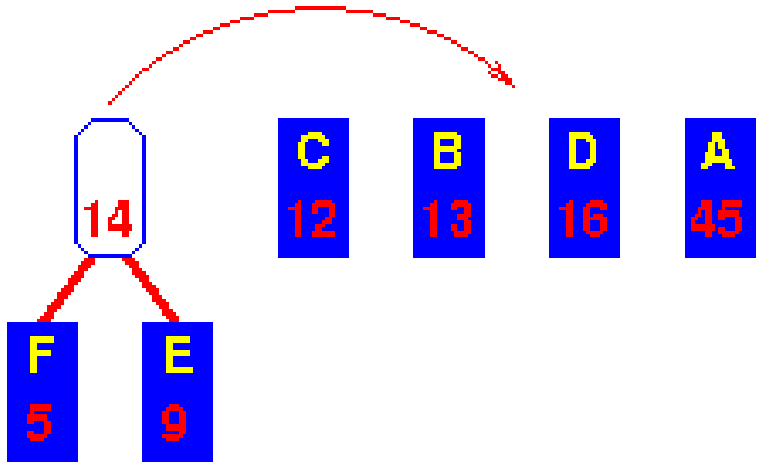
Código de Huffman - Operação

Seqüência inicial ordenada por freqüência



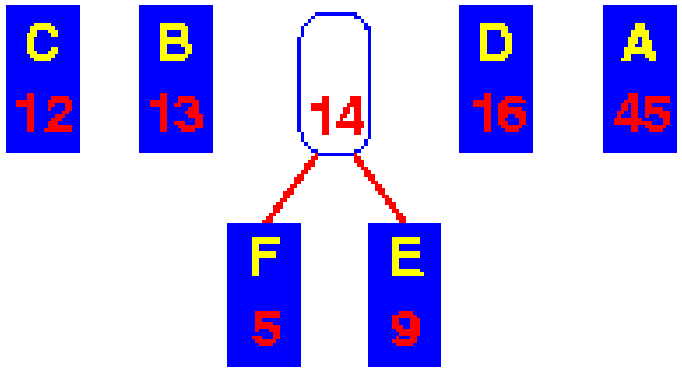
Combine mais baixas em uma sub-árvore

Mova para o lugar correto



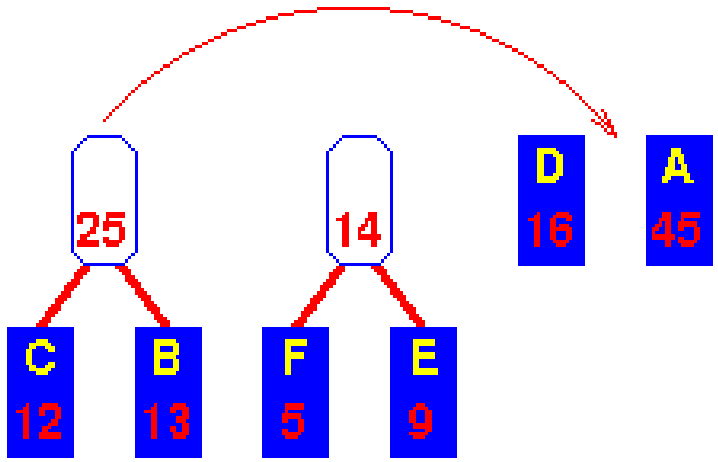
Código de Huffman - Operação

Deslocando a sub-árvore para seu lugar correto



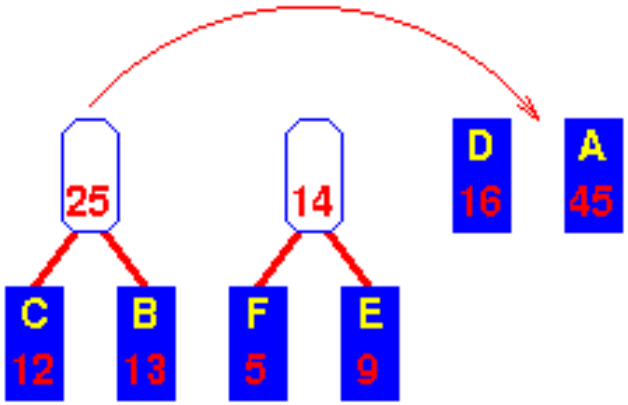
Combine próximo par mais baixo

Mova a sub-árvore para seu lugar correto



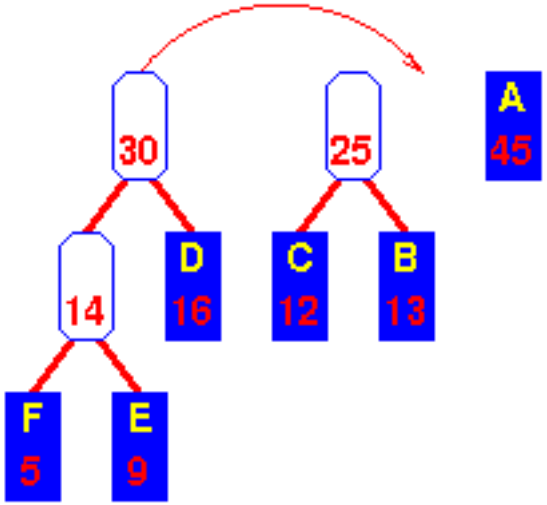
Código de Huffman - Operação

Mova a nova árvore para seu lugar correto

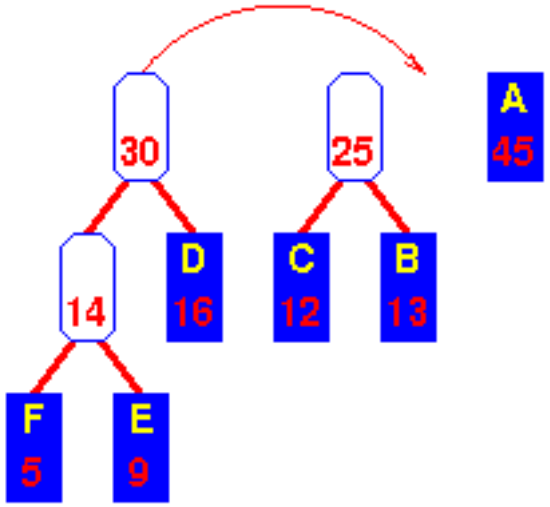


Agora os dois menores formam a sub-árvore “14” e D

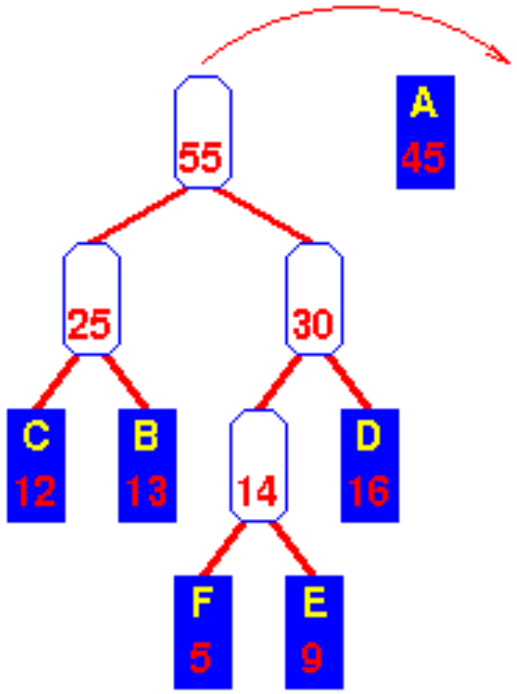
Combine e mova para o lugar correto



Código de Huffman - Operação



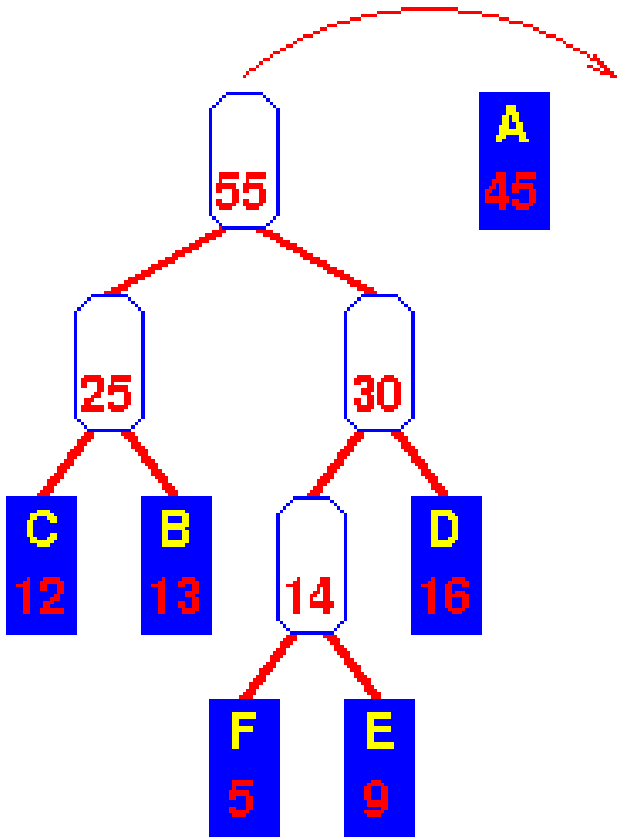
Mova a nova árvore para seu lugar correto



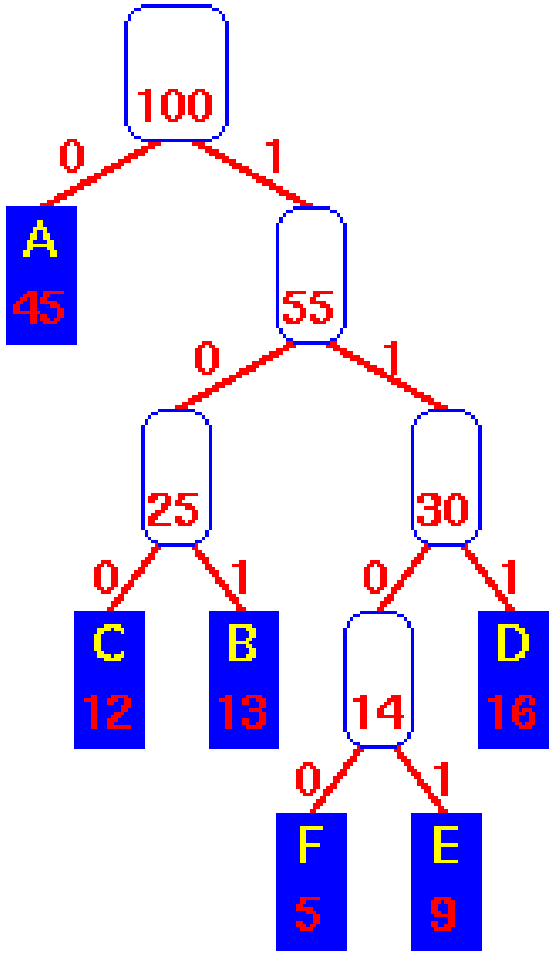
Agora os dois menores são as árvores “25” e “30”

Combine-as e mova para o lugar correto

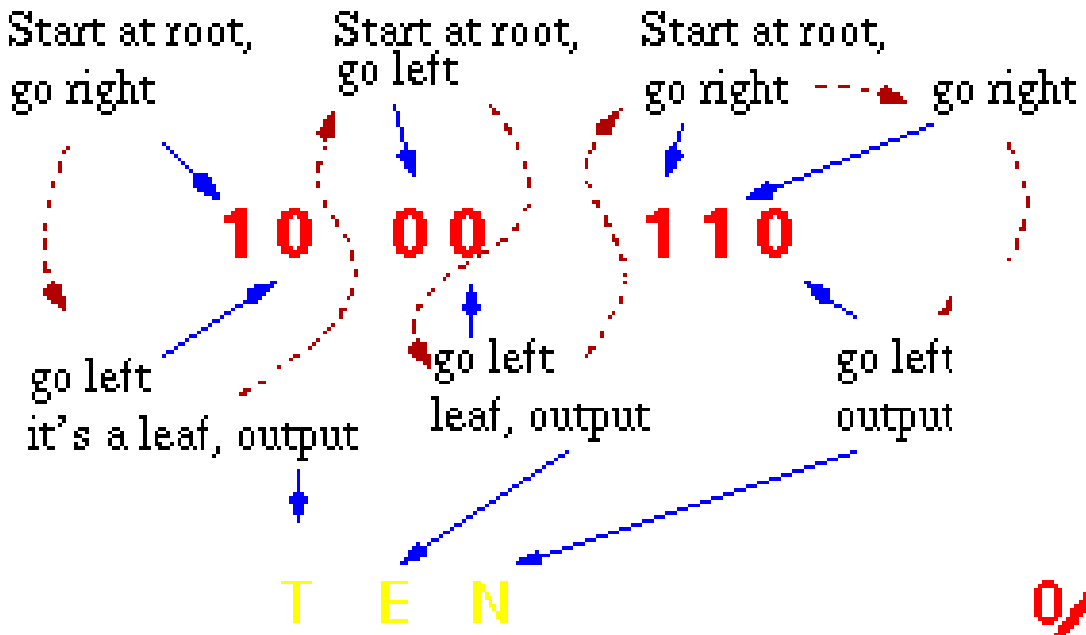
Código de Huffman - Operação

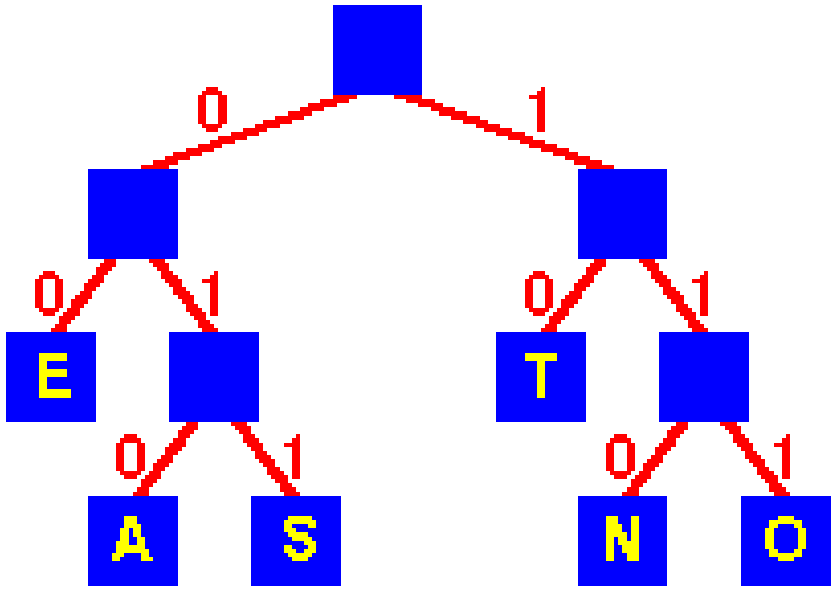


Combine as últimas sub-árvores



Código de Huffman - Decodificação





Código Liv-Zempel

- Os símbolos da informação são mantidos num buffer, com L1 posições iniciais contendo os símbolos mais recentemente codificados
- As L2 posições restantes contêm os símbolos a serem codificados

Código Liv-Zempel

- A cada passo procura-se na cadeia L2 uma sub-cadeia em L1
- Se encontrada, transmite-se um código composto pela posição do casamento encontrado, o comprimento da cadeia e o primeiro símbolo não casado
- Deslocam-se depois os símbolos no buffer