“Um servidor de impressão é um aplicativo (software) ou dispositivo (hardware) capaz de controlar todas as tarefas de impressão enviados de qualquer computador ligado à uma rede. Sua principal função é definir as prioridades e gerenciar as filas de impressão, de modo que os trabalhos sejam distribuídos da melhor forma possível.”

**Questão 1 – FIFO (2,0)**. Considere um serviço de impressão em linguagem C com 5 níveis de prioridade (0-usuário, 1-coordenador, 2-gerente, 3-administrador e 4-super usuário).

Estrutura de dados:

typedef struct noh{

int trabalho;

int prioridade;

struct noh \*prox;

} \*IMPRESSORA;

Escreva uma função/procedimento, em linguagem C, para: Adicionar um trabalho na fila de impressão. A função/procedimento deverá receber o trabalho a ser impresso (representado por um valor inteiro) e sua prioridade. O trabalho deverá ser posicionado na fila relativa a sua prioridade (veja exemplo).

Exemplo:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Fila🡪 | 23 | 🡪 | 12 | 🡪 | 32 | 🡪 | 11 | 🡪 | 19 |  |
| 1 | 2 | 2 | 2 | 3 |

Inserção do elemento 33 com prioridade 3.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Fila🡪 | 23 | 🡪 | 12 | 🡪 | 32 | 🡪 | 11 | 🡪 | 33 | 🡪 | 19 |  |
| 1 | 2 | 2 | 2 | 3 | 3 |

**RESPOSTA:**

void inserirFILAIMPRESS(IMPRESSORA \*f,int trab,int prio){

IMPRESSORA a=\*f,b=a, novo = (IMPRESSORA) malloc(sizeof(struct nohFQ1));

if(novo){

novo->trabalho=trab;

novo->prioridade=prio;

if(\*f){

if((\*f)->prox){

b=a;a=\*f;

while(a&&a->prioridade<prio){

b=a;

a=a->prox;

}

b->prox=novo;

if(a){

novo->prox=a;

} else {

novo->prox=NULL;

}

}else{

if(prio<(\*f)->prioridade){

novo->prox=\*f;

} else {

(\*f)->prox=novo;

novo->prox=NULL;

}

}

}

else {

\*f = novo;

novo->prox=NULL;

}

}

}

**Questão 2 – LLSE, LIFO, FIFO (2,0)**. Dada uma lista simplesmente encadeada de caracteres formada por uma sequência alternada de letras e dígitos, construa uma função/procedimento, em linguagem C, que retorne uma lista na qual as letras são mantidas na sequência original e os dígitos são colocados na ordem inversa.

OBS: Considere a existência das funções/procedimentos para manipulação de pilhas e filas

Exemplos:

A1E5T7W8G→AETWG8751

3C9H4Q6→CHQ6493

**RESPOSTA:**

LLSE organizarLLSE(LLSE l){

FILA f=NULL;PILHA p=NULL;LLSE n=NULL;

while(l){

if(ehDigito(l->info)){

inserirFILA(&f, l->info);

}else{

Push(&p, l->info);

}

l=l->prox;

}

while(!estaVaziaFILA(f)){

insInicioLLSE(&n, elemDispFILA(f));

removerFILA(&f);

}

while(!Empty(p)){

insInicioLLSE(&n, Top(p));

Pop(&p);

}

return n;

}

int ehDigito(char c){

if(c>47&&c<59) return 1;

else return 0;

}

void insInicioLLSE(LLSE \*l,int vr){

LLSE novo = (LLSE) malloc(sizeof(struct noh));

if(novo){ // if(novo != NULL)

novo->info = vr;

novo->prox = \*l;

\*l = novo;

}

}

**Questão 3 – ABB (2,0).** Escreva uma função/procedimento, em linguagem C, para determinar se uma árvore binária é:

1. [ ] estritamente binária
2. [ ] completa

Responda apenas uma das opções. Marque sua escolha.

**RESPOSTA A:**

int estritamenteBinariaABB(ARV a){

if(!a)

return 0;

else {

if((!a->esq&&a->dir)||(a->esq&&!a->dir))

return 0;

else

if(a->esq&&a->dir)

return estritamenteBinariaABB(a->esq)&&estritamenteBinariaABB(a->dir);

else return 1;

}

}

**RESPOSTA B:**

int altura(ARV a){

if(!a)

return 0;

else {

int he=altura(a->esq);

int hd=altura(a->dir);

return 1+((he>hd)?he:hd);

}

}

int contarNos(ARV a){

if(!a) return 0;

else return 1+contarNos(a->esq)+contarNos(a->dir);

}

int ehCompletaABB(ARV a){

return (contarNos(a)==(pow(2,altura(a))-1));

}

}

**Questão 4 – ABB (1,0).** Responda

1. Quantos antecedentes tem um nó no nível n em uma árvore binária? Prove sua resposta.
2. Uma árvore estritamente binária com n nós folhas contém quantos nós?

**RESPOSTA:**

1. N ANTECEDENTES.
2. N+(N-1) NÓS

**Questão 5 – HUFFMAN (1,0).** Considere a seguinte sequência de chaves e suas respectivas frequências em um texto:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A | B | C | D | E | F | G | H | I | J |
| 15 | 6 | 10 | 12 | 13 | 13 | 7 | 5 | 11 | 8 |

Responda:

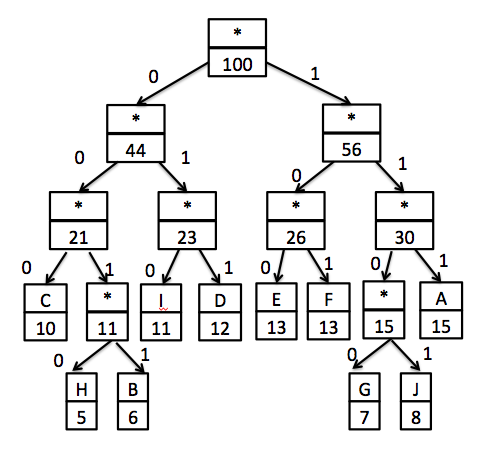
1. Qual a chave possui o menor código de huffman? Justifique.
2. Qual a taxa de compactação de huffman obtida?

**RESPOSTA:**

ORDENAR FREQUÊNCIAS:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| H | B | G | J | C | I | D | E | F | A |
| 5 | 6 | 7 | 8 | 10 | 11 | 12 | 13 | 13 | 15 |

CONSTRUÇÃO DA ÁRVORE DE HUFFMAN:



CÓDIGOS DE HUFFMAN:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CHAVE | FREQUENCIA | HUFFMAN | GASTO | GASTO COMPACTADO |
| A | 15 | 111 | 8\*15=120 | 3\*15=45 |
| B | 6 | 0011 | 8\*6=48 | 4\*6=24 |
| C | 10 | 000 | 8\*10=80 | 3\*10=30 |
| D | 12 | 011 | 8\*12=96 | 3\*12=36 |
| E | 13 | 100 | 8\*13=104 | 3\*13=39 |
| F | 13 | 101 | 8\*13=104 | 3\*13=39 |
| G | 7 | 1100 | 8\*7=56 | 4\*7=28 |
| H | 5 | 0010 | 8\*5=40 | 4\*5=20 |
| I | 11 | 010 | 8\*11=88 | 3\*11=33 |
| J | 8 | 1101 | 8\*8=64 | 4\*8=32 |
| TOTAIS | 100 | --- | 800 | 326 |

1. AS CHAVES A,C,D,E,F,I POSSUEM OS MENORES CÓDIGOS. AMBAS ESTÃO NO MESMO NÍVEL DA ÁRVORE DE HUFMAN POR POSSUIREM APROXIMADAMENTE A MESMA FREQUENCIA DE OCORRÊNCIA NO TEXTO.
2. TAXA DE COMPACTAÇÃO = (1-(326/800))\*100 = 59,25%

**Questão 6 – LIFO (1,0).** Transforme as expressões abaixo para as suas formas pré-fixas e pós-fixas.

1. A + (B \* C) / D – (E ^ F)
2. A \* B + ((C + D) – (E / F) ^ G)

**RESPOSTA:**

A. PRÉ: -+A/\*BCD^EF PÓS: ABC\*D/+EF^-

B. PRÉ: +\*AB-+CD^/EFG PÓS: AB\*CD+EF/G^-+

**Questão 7(1,0).** Escolha uma entre as três primeiras para valer 3,0 pontos.

**Boa Prova.**