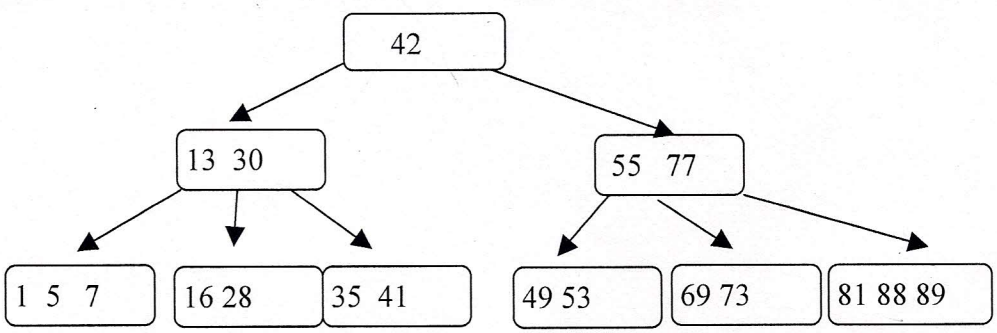
Universidade Federal do Rio de Janeiro
Instituto de Matemática - DCC
Organização de Dados I **30/11/2011**

1.1 – Na árvore B de ordem 2 abaixo, remova a chave 42, e mostre como a arvore fica.



1.2- Escreva o algoritmo de busca de uma chave **x** em uma arvore B, com raiz **ptRaiz**; Considere **nelem**, **filhos[]** e **chaves[]** em cada no da arvore. Considere que o vetor filhos e' utilizado a partir da posicao 0, enquanto o vetor chaves e' utilizado a partir da posicao 1.

2.1- Insira em uma heap de fibonacci inicialmente vazia (com avaliação tardia) as prioridades 1, 5, 8, 9, 3, 42, 7 e 2, nesta ordem.

2.2- Da heap obtida, remova a menor prioridade.

2.3- Na heap obtida, diminua a prioridade 42 para 3. Mostre a heap após cada passo.

3- Construa uma arvore de huffman para a seguinte expressao: "mostly harmless". Qual o tamanho desta expressao se codificada utilizando sua arvore ? (não esqueca do espaco !)

4.1- Insira as seguintes chaves: 12, 42, 13, 37 e 7 em uma tabela hash com encadeamento externo. Utilize a função de hash h(x) = x mod 6.

4.2- Insira as mesmas chaves em uma tabela hash com encadeamento interno, sem area de colisao, e com coalescencia.

5- Considere 10 conjuntos inicialmente unitarios, com elementos 1, 2, 3, ... 10. Realize as seguitnes operacoes, com compressao de caminhos e uniao por tamanho. Se 2 conjuntos tiverem o mesmo tamanho, e estivermos na operacao une(a,b), faça a apontar para b. Une(1,2), une(1,5), une(3,4), une(4,5), une(6,7), une(8,9), une(6,9), une(1,9), une(6,10). **Lembre-se que as operacoes de uniao incluem operacoes find (com compressão de caminhos !).**

6. Considere as seguintes operacoes:
Push(x) → insere x em uma pilha;
Pop() → retira o elemento do topo da pilha se houver;
multi-pop(k) → retira k elementos do topo da pilha. Se a pilha tiver menos de k elementos, então retira todos os elementos da pilha.

Faca uma análise amortizada mostrando que todas estas operacoes tem custo amortizado constante.