Estrutura de Dados - 1o. período de 2010

Gabarito da Segunda Avaliação à Distância

1. (1,0) Descreva um algoritmo que imprima, para cada nó v de uma árvore binária T, o número de nós que existem na sub-árvore de T enraizada por v (incluindo o próprio v).

   Resposta: Seja $f(v)$ o número de nós da sub-árvore enraizada por v.

   procedimento $totalNos(v)$
   - *filho-esq* $:= v \uparrow .esq$; $f($*filho-esq*$) := 0$
   - *filho-dir* $:= v \uparrow .dir$; $f($*filho-dir*$) := 0$
   - se *filho-esq* $\neq \lambda$ então
     - $f($*filho-esq*$) := totalNos($*filho-esq*$)$
   - se *filho-dir* $\neq \lambda$ então
     - $f($*filho-dir*$) := totalNos($*filho-dir*$)$
   - *total* $:= f($*filho-esq*$) + f($*filho-dir*$) + 1$
   - imprimir (Total de nós de v: *total*)
   - retornar *total*

   Chamada externa: $f(ptraiz) := totalNos(ptraiz)$

2. (1,0) Prove ou dê contra-exemplo: Uma árvore binária pode ser construída, de forma única, a partir das seguintes informações:(i) percurso em pré-ordem e (ii) número de nós em cada subárvore.

   Resposta: A afirmativa é falsa. Considere duas árvores binárias T_1 e T_2, onde cada uma delas contém apenas dois nós A e B de forma que:
   - em T_1, B é filho esquerdo de A;
   - em T_2, B é filho direito de A.

   Para ambas as árvores acima, o percurso em pré-ordem é AB, a subárvore enraizada por A tem 2 nós e por B tem 1 nó. No entanto, elas são distintas.

3. (1,5) Uma árvore k-ária, para um inteiro $k \geq 2$ fixo, é definida como uma árvore em que cada nó tem no máximo k filhos. Escreva um algoritmo que calcule as alturas de todos os nós de uma árvore k-ária. Para cada nó v, suponha a existência de campos $F_1(v), F_2(v), \ldots, F_k(v)$ que são ponteiros que apontam para os k filhos de v (sendo que alguns destes campos podem ser nulos, no caso de v ter menos do que k filhos).

   Resposta:

procedimento altura(pt)

se ($pt \uparrow .F1 \neq \lambda$) então $h_1 := altura(pt \uparrow .F1)$
senão $h_1 := 0$
se ($pt \uparrow .F2 \neq \lambda$) então $h_2 := altura(pt \uparrow .F2)$
senão $h_2 := 0$
...
se ($pt \uparrow .Fk \neq \lambda$) então $h_k := altura(pt \uparrow .Fk)$
senão $h_k := 0$
retornar $1 + max\{h_1, h_2, \cdots, h_k\}$

Chamada externa: $altura(ptraiz)$

4. (2,0) Determinar a árvore binária de custo mínimo relativa às seguintes frequências: $f_1 = 1, f_2 = 0, f_3 = 2, f'_0 = 0, f'_1 = 3, f'_2 = 1, f'_3 = 2$.

Resposta:

As matrizes do algoritmo de cálculo da árvore ótima são:

Matriz dos custos $c[i, j]$:

0	4	9	18
-	0	4	12
-	-	0	5
-	-	-	0

Matriz dos valores $F[i, j]$:

0	4	5	9
-	3	4	8
-	-	1	5
-	-	-	2

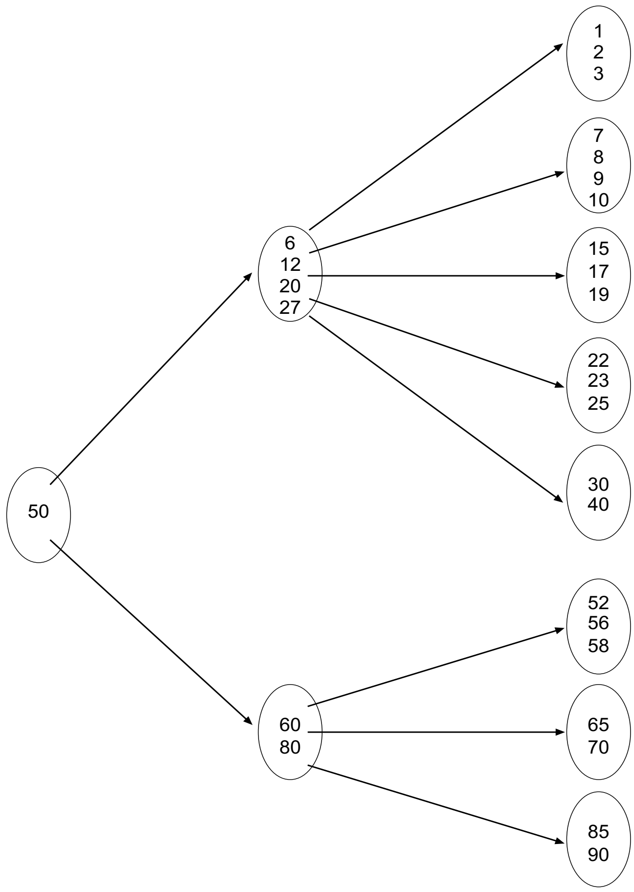
Matriz dos valores minimizantes k:

-	1	1 (2)	2 (3)
-	-	2	3
-	-	-	3
-	-	-	-

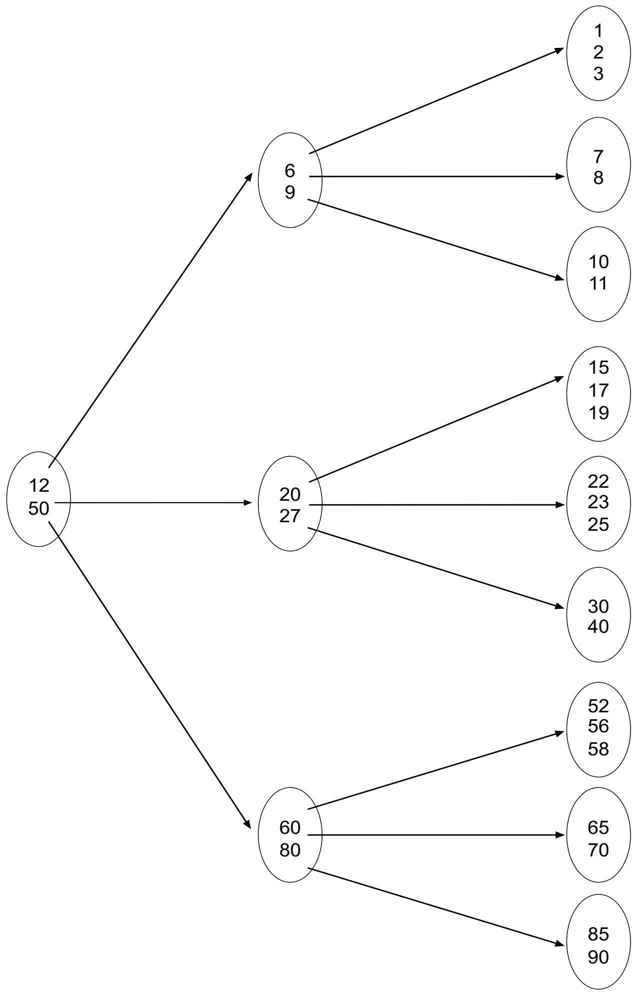
Da última matriz acima, obtemos três possíveis árvores ótimas:
- raiz s_2, filho esquerdo s_1 e direito s_3;
- raiz s_3, sendo s_1 filho esquerdo de s_3, e s_2 filho direito de s_1;
- raiz s_3, sendo s_2 filho esquerdo de s_3, e s_1 filho esquerdo de s_2.

5. (1,5) Desenhe uma árvore B de ordem $d = 2$ com três níveis. (Os valores nos nós ficam à sua escolha.) A seguir, escolha uma nova chave de forma que a sua *inserção* exija uma cisão propagável. Desenhe a árvore B resultante após a inserção.

Resposta:



Inserindo a chave 11, temos uma cisão propagável, que resulta na seguinte árvore B:



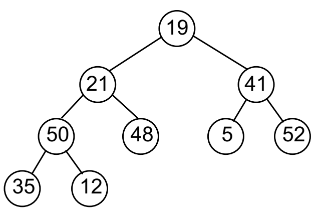
6. (1,0) Descreva como são as árvores-rubro negras com número máximo de nós negros em relação aos rubros.

   CANCELADA.

7. (2,0) Aplique o algoritmo Heapsort aos seguintes valores: $19, 21, 41, 50, 48, 5, 52, 35, 12$. Desenhe os passos intermediários do algoritmo, desenhando cada passo em formato de árvore.

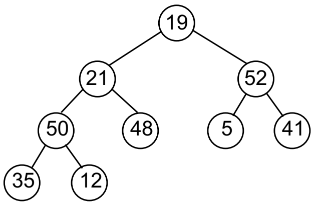
   Resposta:

   Início:

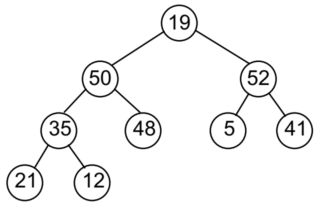


   Construção do heap: (comando *arranjar*(n))

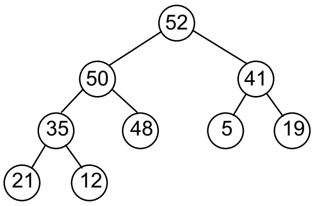
Descer 50:
Descer 41:



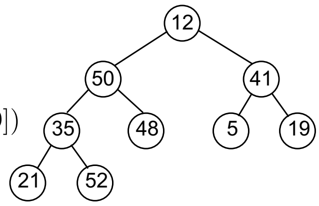
Descer 21:



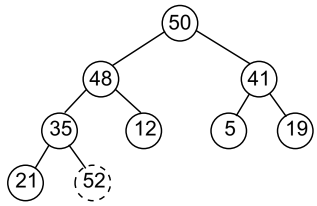
Descer 19:
Heap
obtido:



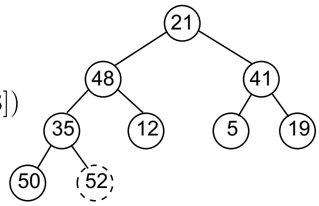
$m := 9$
trocar(TB[1],TB[9])



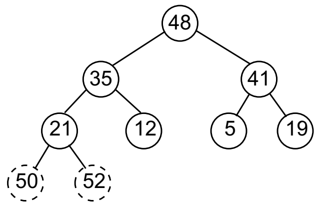
$m := 8$
descer(1,8)



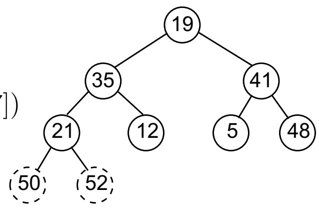
trocar(TB[1],TB[8])



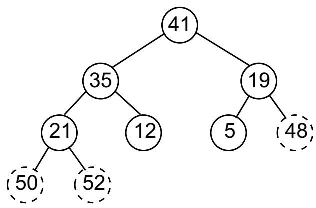
$m := 7$
descer(1,7)



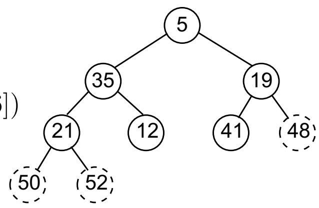
trocar(TB[1],TB[7])



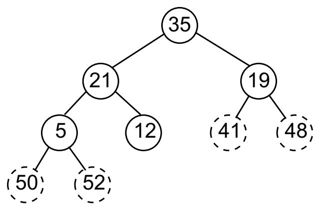
$m := 6$
descer(1,6)



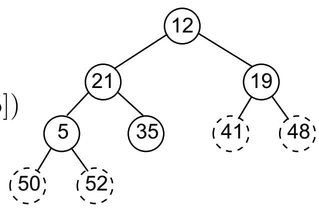
trocar(TB[1],TB[6])



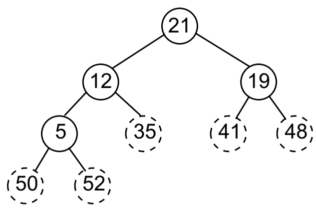
$m := 5$
descer(1,5)



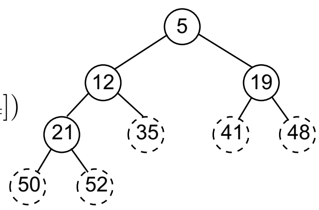
trocar(TB[1],TB[5])



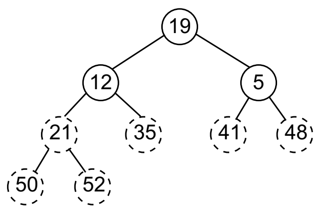
$m := 4$
descer(1,4)



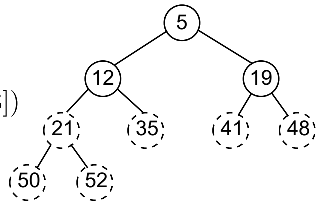
trocar(TB[1],TB[4])



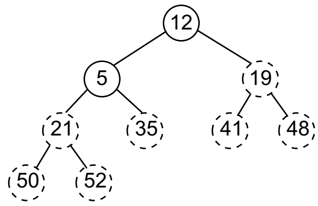
$m := 3$
descer(1,3)



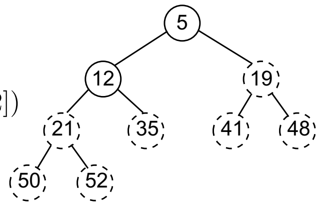
trocar(TB[1],TB[3])



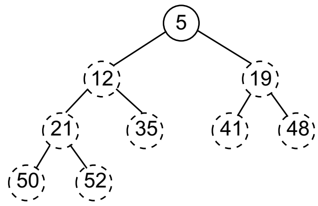
$m := 2$
descer(1,2)



trocar(TB[1],TB[2])



$m := 1$
descer(1,1)



8. (1,0) Descrever um algoritmo de inserção em uma tabela de dispersão por encadeamento exterior.

   Resposta:

   Suponha que x é a chave a ser incluída na tabela $T[0 \ldots m-1]$, de tamanho m.

   $end := h(x)$ mod m
   $pont := T[end]$ % ponteiros para percorrer a lista
   $achou := falso$
   enquanto $pont \neq \lambda$ faça
       $ant := pont$
       se $pont \uparrow .chave = x$
           então $achou := verdadeiro$; $pont := \lambda$ % x já está na lista
           senão $pont := pont \uparrow .prox$
   se $achou = falso$ então
       $ocupar(pt)$
       $pt \uparrow .chave := x$
       $pt \uparrow .prox := \lambda$
       se $T[end] = \lambda$
           então $T[end] := pt$
           senão $ant \uparrow .prox := pt$