Curso de Tecnologia em Sistemas de Computação
Disciplina: Estrutura de Dados e Algoritmos
AP2 - Primeiro Semestre de 2015

Nome -
Assinatura -

Observações:

1. Prova sem consulta e sem uso de máquina de calcular.
2. Use caneta para preencher o seu nome e assinar nas folhas de questões e nas folhas de respostas.
3. Você pode usar lápis para responder as questões.
4. Ao final da prova devolva as folhas de questões e as de respostas.
5. Todas as respostas devem ser transcritas nas folhas de respostas. As respostas nas folhas de questões não serão corrigidas.

---

1. Responda às seguintes questões:

   (1a) (1,0) Desenhe uma árvore B de ordem $d = 2$ e altura $h = 3$ com o menor número possível de chaves.

   Resposta: Pela definição de árvores B, sabemos que o número mínimo de elementos em uma árvore B de ordem d ocorre quando a página raiz possui apenas 1 elemento e todas as demais páginas possuem exatamente d elementos. A Figura 1 representa um exemplo de árvore B de altura 3 e ordem 2 com o mínimo número de elementos.

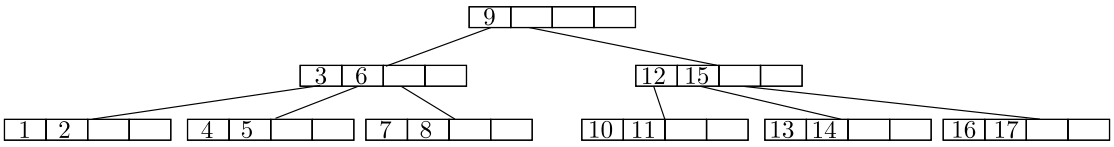

Figura 1: Representação de uma árvore B de ordem 2 e altura 3 com o menor número de elementos.

   (1b) (1,0) Desenhe uma árvore AVL de altura $h = 4$ com o menor número possível de nós.

   Resposta: Uma árvore AVL de altura h com mínimo número de elementos é obtida de modo que todos os nós internos possuam subárvores que diferenciam de exatamente uma unidade. A Figura 2 mostra um exemplo de árvore AVL com número mínimo de elementos.

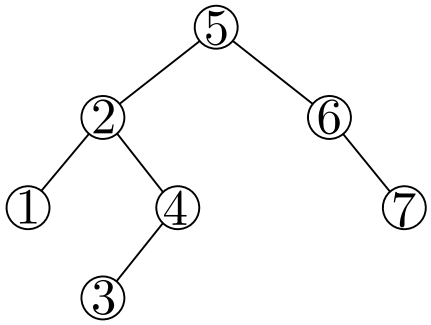

Figura 2: Representação de uma árvore AVL de altura 4 com o menor número de elementos.

2. Assinale V ou F, justificando:

   (2a) (1,0) Se T é uma árvore estritamente binária com 10 nós então T possui 4 nós no terceiro nível.

Resposta: Falso. Lembrando que uma árvore é estritamente binária se todos os seus nós possuem exatamente 0 ou 2 filhos, podemos verificar que não existe uma árvore estritamente binária com 10 nós. De fato, não existe árvore estritamente binária com um número par de elementos. Para mostrar esta afirmação, utilizamos um lema demonstrado na Videoaula 18, que diz que o número de subárvores vazias de uma árvore binária com n elementos é igual a $n + 1$. Dessa forma, como uma árvore estritamente binária possui um número par de subárvores vazias, já que todos os nós folhas devem possuir exatamente dois filhos vazios e todos os nós internos devem possuir exatamente dois filhos não vazios. Logo $n + 1$ deve ser par e, assim, n deve ser ímpar.

(2b) (1,0) Se T é uma árvore binária com 10 nós então $4 \leq h(T) \leq 10$, onde $h(T)$ é a altura de T.

Resposta: Verdadeiro. Como sabemos, a altura mínima de uma árvore binária se dá quando a mesma é uma árvore completa com apenas um nó no último (maior) nível e seu valor é dado por $h_{\min} = 1 + \lfloor \log n \rfloor$. Com isso, substituindo o valor de n por 10, obtemos que $h_{\min} = 4$. Além disso, sabemos que a altura máxima de uma árvore binária é obtida por uma árvore zigue-zague, ou seja, uma árvore em que cada nó não folha possui exatamente um filho. Logo a altura máxima $h_{\max} = 10$. Portanto, $4 \leq h(T) \leq 10$, para qualquer árvore binária T.

3. Dado um vetor V com n elementos, pede-se um algoritmo que verifica se V representa um *heap* (fila de prioridade), onde a raiz tem valor máximo de prioridade.

(3a) (0,5) Descrever a ideia geral do algoritmo, apenas em palavras.

Resposta: Verificamos em ordem decrescente as posições do vetor verificando se o elemento $V[i] \leq V[\lfloor \frac{i}{2} \rfloor]$. Enquanto esta condição for verdadeira, decrementamos de uma unidade a posição definida pelo índice i corrente e passamos para a próxima, até que chegamos ao índice $i = 1$, onde verificamos que V representa um *heap*, ou que a condição não é satisfeita na posição $i > 1$, onde o algoritmo para e retorna não como resposta.

(3b) (1,0) Descrever uma implementação do algoritmo.

Algoritmo 1: Algoritmo de verificação de *heap*.

Entrada: Vetor V com n elementos.
Saída: *Sim* se V define um *heap* ou *Não*, caso contrário.

```
1 i ← n;
2 enquanto i > 1 faça
3     se V[i] > V[⌊i/2⌋] então
          Resultado: Não;
4     i ← i − 1
Resultado: Sim;
```

(3c) (0,5) Determinar a complexidade do algoritmo em função de n.

Resposta: Como o algoritmo percorre todo o vetor exatamente uma vez no pior caso, que ocorre quando V representa um *heap*, então temos que a complexidade de pior caso do Algoritmo 1 é $O(n)$.

4. É dada uma tabela de espalhamento T com 5 posições, numeradas de 0 a 4. Deseja-se armazenar em T as seguintes chaves: 15, 17, 23, 26, 34, 52, 74, 88. Supondo que T seja implementada utilizando o método de encadeamento exterior, responda as seguintes questões:

   (4a) (1,0) Determine um número $k_1 > 1$ tal que a função de espalhamento $h_1(x) = x \bmod k_1$ gere o menor número possível de colisões. Quantas colisões são geradas com esta escolha?

   Resposta: Observe que $k_1 < 8$, pois caso contrário temos que $7 \leq h_1(15) \leq 15$. Também podemos notar que $k_1 \neq 7$, pois $26 \mod 7 = 5$ e o mesmo ocorre se $k_1 = 6$, já que $17 \mod 6 = 5$. Com isso, temos que $k_1 = 5$ gera o número mínimo de colisões possíveis, onde obtemos 3 colisões.

   (4b) (1,0) Determine um número $k_2 > 1$ tal que a função de espalhamento $h_2(x) = x \bmod k_2$ gere o maior número possível de colisões. Quantas colisões são geradas com esta escolha?

   Resposta: Neste caso podemos notar que $k_2 = 2$ gera conflitos entre todos os números pares e entre todos os números ímpares entre si. Ou seja, geramos 6 colisões.

5. É dada a cadeia de caracteres $X = ababababab$. Deseja-se verificar se uma certa cadeia Y é subcadeia de X, utilizando-se o algoritmo de força bruta. Responda os itens abaixo.

   (5a) (1,0) Dê exemplo de uma cadeia Y com 5 caracteres que leve o algoritmo de força bruta a realizar o *menor* número possível de comparações. Quantas comparações são realizadas com esta escolha?

   Resposta: Seja $Y = ababa$. Como Y pertence ao início de X, ou seja, Y é um prefixo de X, temos um total de 5 comparações efetuadas pelo algoritmo de força bruta, onde é feita uma comparação para cada posição de Y e verifica-se que a cadeia foi encontrada e algoritmo para com resposta positiva na posição 1.

   (5a) (1,0) Dê exemplo de uma cadeia Y com 5 caracteres que leve o algoritmo de força bruta a realizar o *maior* número possível de comparações. Quantas comparações são realizadas com esta escolha?

   Resposta: Para obtermos uma cadeia Y que gere o máximo de comparações do algoritmo de força bruta, devemos escolher uma cadeia em que o algoritmo a percorra até sua última posição várias vezes, ou seja, a cadeia Y

sem seu último elemento é uma subcadeia de X, mas Y não é subcadeia de X, a menos que Y seja um sufixo de X, onde percorremos toda a cadeia X até sua última posição. Neste caso, podemos escolher a cadeia $Y = ababb$ ou a cadeia $Y = babaa$. Considerando $Y = ababb$, podemos ver que o algoritmo executa 5 comparações para cada posição de X em que existe um elemento a, onde os 4 primeiros elementos são compatíveis e apenas na última comparação temos a resposta negativa, forçando o algoritmo a proceder para a próxima posição de X em relação a iteração anterior. Além disso, cada posição de X com elemento b em que se inicia a verificação de Y como subcadeia, é feita apenas uma comparação que retorna valor negativo, já que Y inicia com a letra a, e o algoritmo segue para a próxima posição de X reiniciando a busca por Y. Dessa forma, temos um total de 5+1+5+1+5+1=18 comparações. Caso seja utilizada a cadeia $Y = babaa$, obtemos um total de 1+5+1+5+1+5=18 comparações. Note que se X seguisse o mesmo padrão de alternar caracteres a's e b's e X terminasse em um caractere igual a a, então a cadeia $Y = ababb$ geraria um número maior de comparações que a cadeia $Y = babaa$.