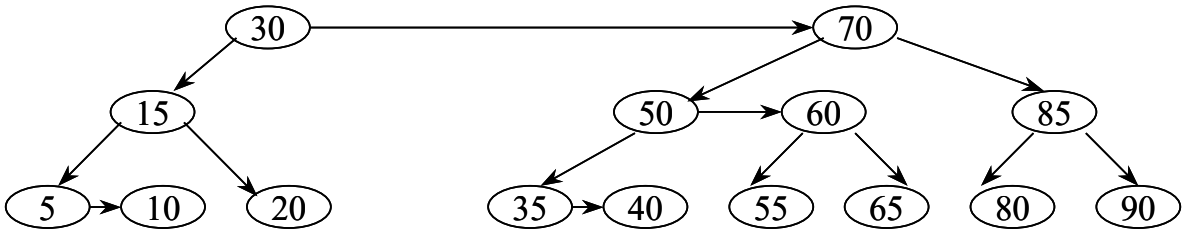
CT-234

QUESTÕES DE PROVAS ANTIGAS

1) Preencha a tabela abaixo com $\in$ ou $\notin$:

	$\omega(\log n)$	$\Theta(n)$	$O(n \log n)$	$\Omega(n^2)$	$o(n^3)$
$6n^3 + 12n^2 + 12.\log n + 3$					
$4n + 3n.\log n + \log n$					

2) Dada a árvore binária abaixo, escreva os seus percursos à esquerda: pré-ordem, in-ordem e pós-ordem.



3) Utilizando as funcionalidades de uma pilha (**`top`**, **`push`**, **`pop`** e **`isEmpty`**), escreva um algoritmo que leia um número inteiro positivo e imprima a sua correspondente representação binária.

4) Dado o algoritmo recursivo abaixo, calcule o seu <u>tempo exato</u> de execução.

```
Rec(n) {
    if (n==1) return 1;
    return Rec(n-1) + Rec(n-1) + Rec(n-1) + 1;
}
```

Importante: considere que as operações básicas (multiplicações, subtrações, comparações, etc.) gastam tempo constante.

5) Prove por indução: $5^{n+1} + 2.3^n + 1$ é divisível por 8, para $n \geq 0$.

6) William G. Horner sugeriu uma maneira alternativa de exprimir polinômios. Por exemplo, o polinômio $p(x) = 3x^3+8x^2-4x+9$ teria o seguinte formato de Horner: $p(x) = 9+x(-4+x(8+x.3))$.

Dado um polinômio p(x) de grau n e um parâmetro x, calcule o número de multiplicações e adições necessárias para avaliá-lo:

a) através da sua expressão convencional;

b) através do formato de Horner.

7) Escreva algoritmos recursivos que recebam como parâmetro um número inteiro x e calculem o polinômio $p(x) = \sum a_k x^k$, $0 \leq k \leq n$:

a) através da sua expressão convencional;

b) através do formato de Horner.

Suponha que os coeficientes a_k estejam armazenados em um vetor de dimensão n+1.

8) Descreva detalhadamente o que é um *heap*.

9) Explique as ideias do algoritmo *QuickSort* e indique a ordem do pior caso.

10) Explique as ideias do algoritmo *MergeSort* e indique a ordem do pior caso.

11) Descreva os passos da ordenação crescente do algoritmo *RadixSort* para os seguintes números: 454, 901, 15, 23, 188, 901, 0, 675, 227, 6.

12) Dê um exemplo de função que seja simultaneamente:

a) $O(n^2)$ e $\Omega(\log^2 n)$

b) $\omega(\log n)$ e $o(n)$

c) $\Theta(n)$ e $\Omega(n^3)$

d) $o(2^n)$ e $\Omega(n^{50})$

e) $O(n^2)$ e $\omega(n.\log n)$

Se em algum caso não existir tal função, explique por quê.

13) Foi visto que uma **fila de prioridades** (de máximo) é uma estrutura com as seguintes funcionalidades: *Max(), ExtractMax(), Insert(x)* e *Modify(k,x).* Também vimos que pode ser eficientemente implementada com um *heap*. Por outro lado, se quiséssemos implementá-la com uma lista ligada, explique como elaboraria essas 4 funcionalidades e qual seriam as suas complexidades de tempo.

14) Sem utilizar o comando `for(;;)` (e nenhum outro comando de repetição), escreva um algoritmo recursivo que calcule o produto de duas matrizes quadradas A e B de ordem n, armazenando o resultado numa matriz C. Suponha que essas matrizes sejam variáveis globais, e que C esteja inicialmente zerada.

15) O produto interno de dois vetores n-dimensionais $u = (u_1, ..., u_n)$ e $v = (v_1, ..., v_n)$ é definido como $\sum u_i.v_i$. Escreva um algoritmo recursivo que, dados dois vetores n-dimensionais, calcule o seu produto interno.

16) Explique para que serve uma árvore AVL. Dê um exemplo de aplicação prática.

17) Prove por indução matemática: $2^n < n!$, para $n > 3$.

18) Suponha que um dado vetor de n elementos seja "tetratônico", isto é, seja formado por 4 subsequências ordenadas, cujos comprimentos são desconhecidos.

Veja um exemplo a seguir, onde n = 13 (as setas indicam a ordenação):

57 32 -21	32 45 98 98 233	202 -12	45 67 109

Para vetores desse estilo, descreva um algoritmo de ordenação que seja ótimo. Justifique.

19) Calcule as complexidades de tempo e de espaço do programa abaixo, cujo corpo principal consiste apenas na chamada de **`Rec(n)`**:

```
Rec(n) {
    if (n==1) return 1;
    return 3*Rec(n-1) + Funcao(n);
}

Funcao(n) {
    for (i=0; i<n; i++)
         for (j=i; j>0; j--)
             v[i,j] = 4*i*i+j;
    return v[n-1,n-1];
}
```

Importante:

a) Considere que as operações básicas (multiplicações, subtrações, comparações, etc.) gastam tempo constante.

b) O vetor **`v`** tem dimensões **`[0..n-1, 0..n-1]`** e é uma variável global.

20) O algoritmo abaixo realiza a busca binária de **`chave`** em um vetor **`v`** de índices **`0`** a **`N-1`**, ordenado crescentemente:

```
int PesquisaBinaria (int v[], int chave, int N)
{
     int inf, sup, meio;
     inf = 0;
     sup = N-1;
     while (inf <= sup) {
          meio = (inf+sup)/2;
          if (chave == v[meio]) return meio;
          if (chave < v[meio]) sup = meio - 1;
              else inf = meio + 1;
     }
     return -1;   /* não encontrado */
}
```

Com o uso de recorrências, demonstre que a sua complexidade de pior caso é $\Theta(\log \mathbf{N})$.

Observação: basta considerar o caso em que **`N`** é potência de 2.

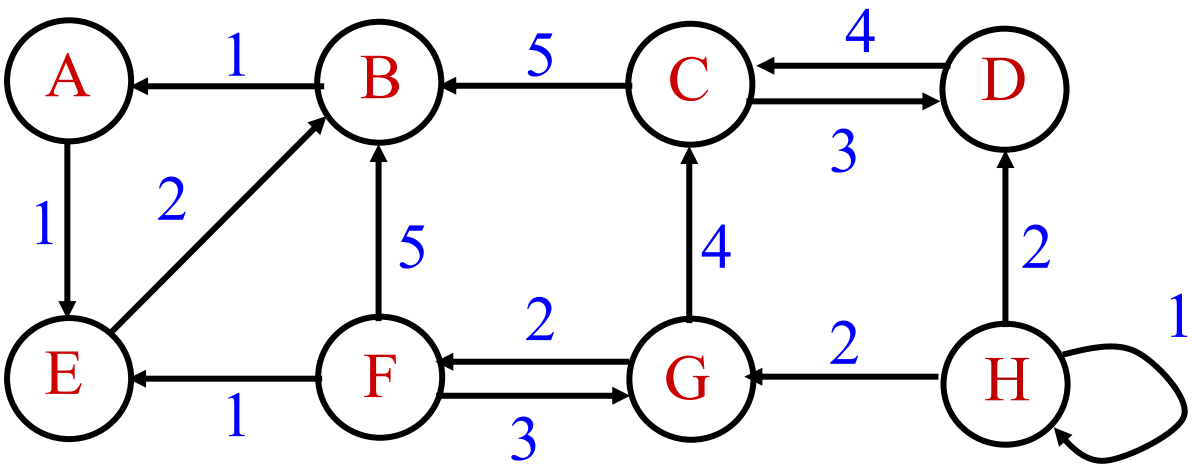
21) Dado um vetor não ordenado de tamanho n, apresente um algoritmo que encontre o maior e o menor elementos, mas realizando no máximo $3n/2$ comparações.

22) Suponha um vetor de n posições cuja primeira metade está em ordem crescente. Para completar a ordenação desse vetor, os elementos da segunda metade são inseridos na parte ordenada, um a um, utilizando-se busca binária. Essa ordenação pode ser sempre terminada em tempo $O(n.\log n)$? Justifique.

23) Prove por indução que uma árvore binária completa com $n = 2^k$ folhas tem altura $k = \log_2 n$.

24) Prove por indução que a representação binária de um número inteiro $n > 0$ tem exatamente $\lfloor \lg n \rfloor + 1$ bits.

O digrafo abaixo será utilizado nas três próximas questões.



25) Através do algoritmo de Dijkstra, encontre o caminho mínimo entre os vértices H e A. Deixe indicado os passos que seguiu.

26) Através do algoritmo de Tarjan, encontre as componentes fortemente conexas. Deixe indicados os passos que seguiu.

27) Através do algoritmo de Prim, encontre uma árvore geradora de custo mínimo a partir do vértice A. Deixe indicado os passos que seguiu.

Importante: ignore a orientação dos arcos.

28) Escreva um algoritmo *Divisão-e-Conquista* para encontrar o valor máximo do vetor A[1..n] de inteiros. Calcule a sua complexidade de tempo.

29) Através de *Programação Dinâmica*, escreva um algoritmo que, dado n, monte uma matriz de ordem n com os valores do triângulo de Pascal. Calcule a sua complexidade de tempo.

Exemplo para n = 6:

1					
1	1				
1	2	1			
1	3	3	1		
1	4	6	4	1	
1	5	10	10	5	1

30) Monte o autômato de busca do padrão "bacabacb".

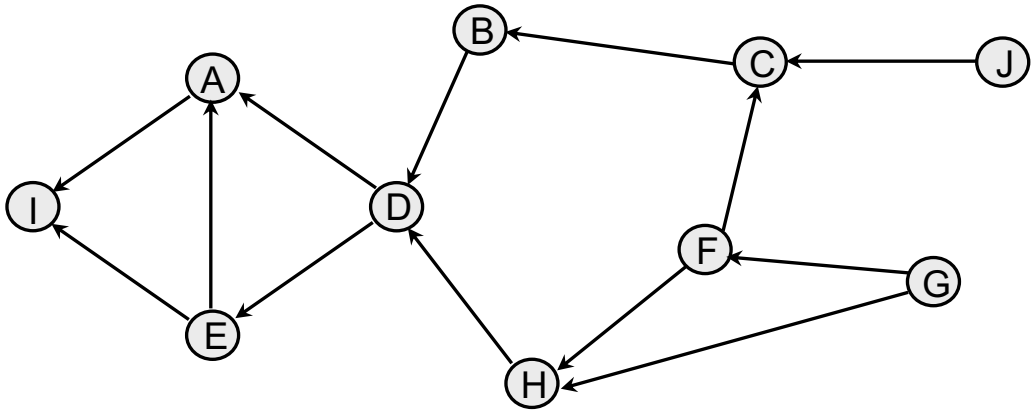
31) Dê a classificação de *Flynn* (1966) e descreva em que consistem cada um dos seus itens. Apresente uma nomenclatura atual equivalente.

32) Dado o texto "**`Faturei com folga na prova de complexidade de algoritmos`**" e o padrão "**`algo`**", indique (e justifique) o *número de comparações* que os algoritmos abaixo realizam até encontrarem esse padrão no texto:

a) Knuth-Morris-Pratt;

b) Boyer-Moore.

33) Considere o digrafo abaixo:



a) Supondo armazenamento por listas de adjacências, onde os vértices estão em ordem lexicográfica, aplique o algoritmo de Tarjan a partir do vértice G, classificando os arcos em 4 conjuntos diferentes: *Árvore*, *Avanço*, *Retorno* e *Cruzamento*. Algum desses conjuntos ficou vazio? Por quê?

b) Encontre uma ordenação topológica para ele, caso exista. Justifique.

34) Escreva a matriz de incidências do digrafo da questão anterior.

35) Ainda para o mesmo digrafo, desconsidere as orientações das arestas e aplique a variante do algoritmo de Tarjan a partir do vértice A para encontrar seus vértices e arestas de corte.

36) Através do algoritmo de *Programação Dinâmica* visto em aula, encontre a maior subsequência comum de <AACGTGGCCTA> e <CACGTTCCA>.

37) Dado o texto "**cababaabababa**" e o padrão "**abababa**", indique (e justifique) o *número de comparações* que os dois algoritmos abaixo realizam até encontrarem esse padrão no texto:

a) Knuth-Morris-Pratt;

b) Boyer-Moore.

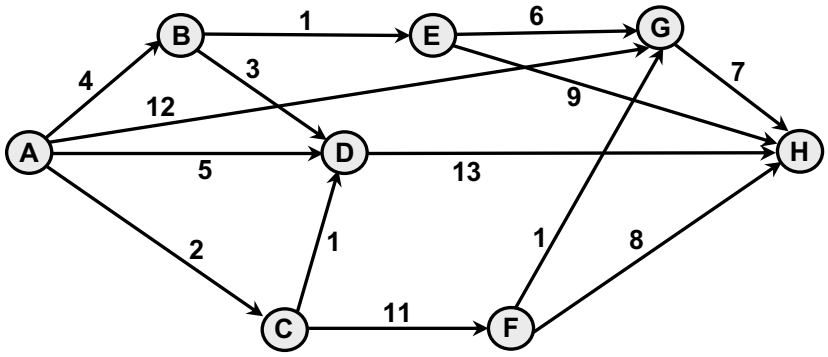
38) Em aula foi apresentada uma implementação do algoritmo de Prim que utiliza uma estrutura de *heap*. Se ao invés dessa estrutura utilizássemos uma lista ligada, na qual os vértices fossem mantidos em ordem crescente de custo, qual seria a complexidade de tempo do algoritmo? Justifique.

39) Um determinado algoritmo sequencial tem complexidade de tempo $O(n^2)$. Um aluno elaborou a seguinte paralelização escalável desse algoritmo, onde p é o número de processadores:

Complexidade de tempo	*Intervalo de processadores*
$O(n^2/p + \log n)$	$0 < p < \log n$
$O(n/p + \log^2 n)$	$\log n \leq p < n$
$O(n^2/p)$	$n \leq p \leq n^2$

A partir desses resultados, é possível verificar se o aluno cometeu algum erro? Qual? Explique.

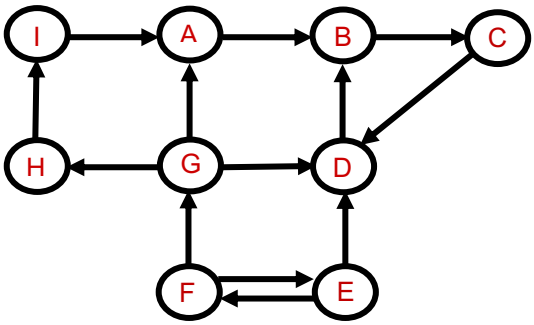
40) Através de algum algoritmo visto em aula, encontre uma ordenação topológica para o digrafo abaixo. Indique os passos que seguiu.



41) Para o digrafo da questão anterior, encontre as distâncias mínimas a partir do vértice A utilizando o algoritmo de Dijkstra. Indique os passos que seguiu.

42) Ainda nesse mesmo digrafo, desconsidere a orientação dos arcos e encontre uma árvore geradora de custo mínimo. Para isso, simule o algoritmo de Kruskal, indicando seus passos.

43) A partir do algoritmo de Tarjan visto em aula, classifique os quatro tipos de arcos do digrafo abaixo. Comece a exploração a partir do vértice I, e considere armazenamento em ordem lexicográfica.



44) Com o algoritmo de Tarjan a partir do vértice I, encontre as componentes fortemente conexas do digrafo anterior. Considere armazenamento em ordem lexicográfica.

45) Como é possível saber se o *Método Guloso* funciona ou não na resolução de um determinado problema? Explique.

46) Dado o texto abaixo, encontre uma codificação de Huffman:
AABCCDAABAEAEDBCCEEAAA

47) Dada o produto de matrizes abaixo, simule o algoritmo de *Programação Dinâmica* que encontra a melhor parentização:
$P = M_1 . M_2 . M_3 . M_4 . M_5$
Respectivas dimensões das matrizes M_i: (2x3), (3x7), (7x8), (8x4), (4x1).

48) Calcule o produto 34 x 1729 através do algoritmo de Karatsuba.
Importante: basta simular apenas o primeiro nível de recursão.

49) Para grafos esparsos (com pouquíssimas arestas), qual é a melhor estrutura de armazenamento? Justifique. E para grafos densos?

50) Sejam X[1..n] e Y[1..m] vetores de caracteres, onde n e m são inteiros positivos. Considere X vazio quando n=0, e Y vazio quando m=0.

a) Baseado no algoritmo de *Programação Dinâmica* visto em aula, escreva um pseudocódigo *memoization* que encontre o **tamanho da maior subsequência comum** de X e Y.

b) Neste problema, *memoization* pode ter melhor desempenho que *Programação Dinâmica*? Explique.