2. (3.0pt) Dado um grafo conexo e não direcionado $G = (V, E)$, explique como seria um algoritmo para decidir se existe ou não uma aresta em E tal que sua remoção não desconecta o grafo. Analise a complexidade do algoritmo proposto. Solucões mais eficientes terão maior pontuação.

5.(2.5pt) Responda as perguntas abaixo.

a) Em um grafo bipartido completo, todos os vértices de uma partição são ligados a todos vértices da outra. Qual é a altura da árvore gerada por uma BFS em um grafo bipartido completo? Qual é a altura da árvore gerada por uma DFS em um grafo bipartido completo?

b) Seja G um grafo direcionado e sejam u, v dois vértices de G. Assuma que ao executar uma DFS obtemos $pre(u) < pre(v) < post(v) < post(u)$. Podemos afirmar que existe um caminho entre u e v no grafo? Por que?

4. (2.0) Seja G um grafo direcionado com pesos nas arestas, seja s um vértices de G e seja k um inteiro positivo. Como podemos encontrar os k vértices mais próximos de s? Com qual complexidade? Análise em função de k, m, n.

3. (3.0pt) Um site de música na Internet decidiu criar um ranking com as melhores canções da última década, a partir de n canções, $s_1, \ldots, s_n$, pré-selecionadas. Durante um mês, sempre que um usuário acessava o site, duas canções escolhidas aleatoriamente eram exibidas, e o usuário devia marcar qual das duas ele preferia. Ao término deste processo, temos um conjunto de triplas $S = \{(s_i, s_j, d_{ij}) | 1 \leq i < j \leq n\}$, aonde $d_{ij} = i$ se a maioria das pessoas prefere s_i à s_j, $d_{ij} = j$ se a maioria das pessoas prefere s_j à s_i e $d_{ij} = 0$ se não há uma preferência entre as canções.

Um ranking R é consistente com a lista S se e somente se para todo par de canções s_i e s_j a seguinte condição é válida: se s_i vem antes da canção s_j no ranking R, então a maioria das pessoas prefere s_i à s_j ou não há uma preferência entre tais canções.

a) Descreva um algoritmo eficiente para verificar se é possível criar um ranking de canções consistente com a pesquisa. Note que o algoritmo deve responder SIM ou NÃO. Analise a complexidade de pior caso do algoritmo proposto em função de n. Explique as estruturas de dados utilizadas para obter tal complexidade.

b) Em algumas situações é possível existir mais de um ranking consistente com S. Dados dois rankings R e R' para uma lista S, dizemos que R *domina* R' se e somente se na primeira posição que R difere de R', o índice da canção de R é menor que o índice da canção de R'. Por exemplo, se $R = s_1 s_3 s_2 s_4$ e $R' = s_1 s_3 s_4 s_2$, então R domina R' já que os rankings diferem pela primeira vez na terceira posição e s_2 tem índice menor que s_4. Descreva como seria um algoritmo para determinar um ranking consistente com a lista S que não é dominado por nenhum outro ranking consistente com S. Análise a complexidade do algoritmo proposto em função de n. Explique as estruturas de dados utilizadas para obter tal complexidade.

3. (3.0pt) Seja um grafo não direcionado $G = (V, E)$. Uma cobertura para G é um conjunto de vértices $C \subseteq V$ tal que toda aresta de E tem pelo menos uma extremidade em C. Considere o seguinte algoritmo guloso.

$C \leftarrow \emptyset$; $G' \leftarrow G$
Enquanto G' tem arestas
    $v \leftarrow$ vértice de G' com maior grau
    $C \leftarrow C \cup v$
    $G' \leftarrow G' - v$ [1]
Fim Enquanto
Devolva C

a) Este algoritmo sempre devolve uma cobertura? Por que?

b) Este algoritmo sempre devolve a cobertura com o número mínimo de vértices? Por que?

c) Explique como implementar o algoritmo acima de forma eficiente. Soluções mais eficientes terão maior pontuação.

2.19. *A k-way merge operation.* Suppose you have k sorted arrays, each with n elements, and you want to combine them into a single sorted array of kn elements.

(a) Here's one strategy: Using the `merge` procedure from Section 2.3, merge the first two arrays, then merge in the third, then merge in the fourth, and so on. What is the time complexity of this algorithm, in terms of k and n?

(b) Give a more efficient solution to this problem, using divide-and-conquer.