Observação: Não é necessário responder todas perguntas para alcançar 10 *pontos.*

Prova

Questão 1 (Union-find, 2.5 pt)
Explique duas estruturas de dados para manter um conjunto de conjuntos sobre um universo de elementos que fornece as operações `make-set(e)` (cria o conjunto $\{e\}$), `find(e)` (dado um elemento e encontra o elemento que representa o conjunto de e) e `union(e,f)` (unir dois conjuntos dado os seus elementos representantes e e f).
(Busca apresentar estruturas eficientes: os pontos dependem da complexidade das operações.)

Questão 2 (k máximos, 2 pt)
Dado um array de n números queremos determinar os k números maiores. Propõe um algoritmo que resolve este problema em espaço adicional $O(k)$ e tempo $O(n \log k)$ (2pt), ou tempo $O(n \log n)$ (1pt).

Questão 3 (Fluxo máximo, 2.5pt)
Seja $G = (V, A)$ um grafo direcionado com capacidades positivas c_a nos arcos $a \in A$, com fonte $s \in V$ e destino $t \in V$. Decide se as seguintes duas afirmações são verdadeiras ou falsas. Para uma afirmação verdadeira: dá uma breve justificativa. Para uma afirmação falsa: dá um contra-exemplo.

a) Caso f é um fluxo s-t máximo em G, então f satura todo arco sainte de s (i.e. temos um fluxo igual a capacidade nestes arcos).

b) Seja (A, B) um corte mínimo em G. Supõe que vamos aumentar a capacidade de cada arco por 1. Então (A, B) continua sendo um corte mínimo em G com essas novas capacidades.

Questão 4 (O viajante medroso, 2.5pt)
Temos um grafo não-direcionado $G = (V, A)$, com distâncias d_a nas arestas $a \in A$. Um viajante medroso quer viajar nessa grafo. Ele tem medo de visitar um subconjunto $M \subseteq V$ dos vértices, mas aceita visitar no máximo uma cidade em M por viagem. Por isso ele quer saber o caminho mais curto entre todas pares de cidades, tal que o caminho contém no máximo um vértice em M. (Ele não está interessado no caminho mais curto entre pares de cidades em M). Encontra um algoritmo que resolve este problema. (Os pontos dependem da eficiência.)

Questão 5 (Um problema de agendamento, 2.5pt)
Considere o problema de agendamento $1 \mid p_j = 1 \mid \sum U_j$. Propõe um algoritmo que resolve-o, mostra a sua corretude e fornece uma analise de complexidade dele. (0.6 pt: complexidade $\Omega(2^n)$, 1.3 pt: complexidade $O(2^n)$, 2.5pt: complexidade $n^{O(1)}$.)
(No problema $1 \mid p_j = 1 \mid \sum U_j$ temos n tarefas, cada um com tempo de execução 1 e com um prazo ("due date") d_j, $j \in [n]$. Uma solução do problema é uma atribuição de tempos de início $S_j \geq 0$ às tarefas, tal que não tem sobreposição na execução (porque temos uma máquina única). Uma tarefa tem que ser executada sem interrupção. Logo, sabendo os tempos de início S_j também temos tempos de término $C_j = S_j + 1$. Uma tarefa é atrasada caso $C_j > d_j$ e no prazo caso contrário. A função

$$U_j = \begin{cases} 0 & C_j \leq d_j \\ 1 & C_j > d_j \end{cases}$$

contabiliza isso. O objetivo do problema é minimizar $\sum_{j \in [n]} U_j$, i.e. o número de tarefas atrasadas.

Questão 6 (Busca informada, 2pt)
Explique as ideias principais e o funcionamento do algoritmo A^*. Comenta sobre corretude, garantias, e complexidade.

Sucesso!