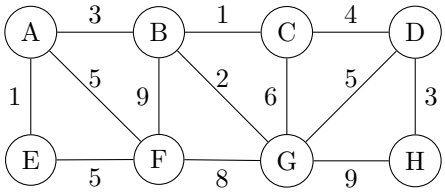
Exercícios: Algoritmos gulosos e divisão e conquista

Questão 1 (2.5 pt)
Discutimos em aula um algoritmo guloso para o problema $1 \mid\mid \sum C_j$. Propõe um algoritmo guloso que resolve o problema $1 \mid\mid \sum w_j C_j$ em tempo polinomial. Mostra a corretude do algoritmo, e analisa a sua complexidade.

Lembrança: no problema $1 \mid\mid \sum w_j C_j$ temos n tarefas, cada tarefa $j \in [n]$ com um tempo de processamento p_j e um peso w_j. O problema consiste em encontrar um agendamento das tarefas numa única máquina. Duas tarefas não podem ser executadas no mesmo tempo, e a execução de uma tarefa não pode ser interrompido, depois de iniciar ("non-preemptive"). Para um dado agendamento, seja C_j o tempo de término da tarefa j. O objetivo é encontrar um agendamento que minimiza $\sum_{j\in[n]} w_j C_j$.
Observe que um agendamento ótimo não possui tempo ocioso na máquina. Em outras palavras um agendamento ótimo pode ser especificado por uma permutação das tarefas. Para uma permutação $\pi = (\pi_1, \ldots, \pi_n)$ os tempos de término estão $C_j = \sum_{1\le i\le \pi^{-1}(j)} p_i$.

Questão 2 (2.5 pt)
A árvore geradora mínima encontra um subgrafo de menor peso total conectado. Supõe agora, que vamos relaxar a condição de conectividade: queremos encontrar um subgrafo de menor peso total, que possui no máximo dois componentes conexos. Encontra um algoritmo em tempo polinomial para este problema. Mostra a corretude do algoritmo, e analisa a sua complexidade.

Questão 3 (2.5 pt)
Suponhe que desejamos encontrar a árvore espalhada mínima do seguinte grafo.



a) Execute o algoritmo de Prim; sempre que houver uma escolha de nós, use a ordem alfabética (por exemplo comece pelo nó A). Desenhe uma tabela mostrando os valores intermediários dos custos (distâncias) dos nós.

b) Execute o algoritmo de Kruskal no mesmo grafo. Mostre como fica a coleção de conjuntos em cada passo do algoritmo.

Questão 4 (2.5 pt)
Discutimos em aula o algoritmo Mergesort, que ordena n números por divisão e conquista, ordenando num primeiro passo recursivamente os $\lfloor n/2 \rfloor$ primeiros números e os $\lceil n/2 \rceil$ últimos números, juntando eles depois para uma lista ordenada de n números.
Este exercício pede analisar a complexidade de tempo de dois casos diferentes do MergeSort com uma divisão alternativa.

a) Uma divisão em um lote de $\lfloor \epsilon n \rfloor$ números e um outro de $n - \lfloor \epsilon n \rfloor = \lceil (1-\epsilon)n \rceil$ números, para um parâmetro $0 < \epsilon < 1$ arbitrário. O entrelaçamento ("merge") das duas sequências fica igual.

b) Uma divisão em k lotes, com $k - k'$ lotes de tamanho $\lfloor n/k \rfloor$ e $k' = n \mod k$ lotes de tamanho $\lceil n/k \rceil$. Observe que o entrelaçamento tem que considerar k sequências agora.

Questão 5 (Quebra-cabeça, opcional)
Um algoritmo para computar uma árvore geradora mínimo que discutimos é o *reverse delete*. Ele processa as arestas do grafo em ordem de peso não-decrescente, e remove cada aresta, caso o grafo continua ser conectado após da remoção. Para isso temos que testar repetidamente se a remoção de

uma aresta desconecta o grafo. Um forma simples de fazer isso é em tempo $O(n+m)$ por uma busca em largura ou profundidade. Com isso o algoritmo precisa tempo $O(m(n+m)) = O(m^2)$. Encontra uma maneira mais eficiente de implementar o *reverse delete*.

Regras para listas de exercícios

1. Os exercícios podem ser resolvidos em colaboração com outros, mas a entrega é individual informando os eventuais colaboradores.
2. A entrega é eletrônica, não escrito a mão, em formato PDF.
3. Para receber pontos as respostas devem ser justificadas (i.e. provadas).
4. Somente entregam respostas que vocês sabem explicar pessoalmente.