MAC 5711 – Análise de Algoritmos

Segundo Semestre de 2002

Terceira Prova – 25 de novembro

1. [2,0 pontos]

   (a) Explique para que serve a estrutura de dados conhecida como `union-find`. Entre outras coisas, diga o que se armazena em tal estrutura de dados, cite um exemplo de aplicação e explique o que fazem as rotinas `union` e `find`. Inclua na sua explicação a lista de parâmetros de tais rotinas.

   (b) Explique a implementação mais eficiente vista em aula para tal estrutura de dados. Em particular, explique o que são as heurísticas dos tamanhos e da compressão de caminhos. Fale sobre o tempo consumido por tal implementação.

2. [1,5 pontos]
   Descreva um algoritmo que recebe um grafo conexo como entrada e determina se tal grafo tem ou não um circuito ímpar. Seu algoritmo deve ser o mais eficiente possível. Explique porque ele funciona e analise a sua complexidade de tempo, determinando-a em função do número de vértices e do número de arestas do grafo.

3. [2,5 pontos]
   As descrições dos algoritmos abaixo não devem ser em pseudo-código. Explique cada algoritmo coloquialmente, porém de forma precisa o suficiente para que uma pessoa que saiba grafos, mas não conheça tais algoritmos, seja capaz de simulá-los a partir da sua descrição. No item (c), as letras m e n denotam, como de costume, o número de arestas e de vértices do grafo em questão, respectivamente.

   (a) Que problema resolve o algoritmo de Kruskal? Descreva o algoritmo de Kruskal, conforme acima. Nesse item, não fale de implementação.

   (b) Que problema resolve o algoritmo de Prim? Descreva o algoritmo de Prim, conforme acima. Nesse item, não fale de implementação.

   (c) Descreva em detalhes **ou** uma implementação $O(m \log n)$ do algoritmo de Kruskal **ou** uma implementação $O(m \log n)$ do algoritmo de Prim. Escreva em pseudo-código a implementação descrita. Assuma dadas rotinas básicas de manipulação de estruturas de dados vistas em aula. Analise a sua implementação, dizendo quanto tempo consome cada parte do código. Dê delimitações tão precisas quanto possível.

4. [2,0 pontos]
   Modifique o algoritmo KMP para que ele resolva o seguinte problema: dadas duas cadeias de caracteres, P e T, representando uma palavra e um texto respectivamente, determine uma posição do texto T em que aparece o maior prefixo da palavra P. Seu algoritmo deve consumir tempo $O(m + n)$ onde m é o número de caracteres da palavra P e n o número de caracteres em T. Suponha dada a função `calcula_função_prefixo`, usada no algoritmo KMP. Escreva o seu algoritmo em pseudo-código, explique onde ele difere do algoritmo KMP original e analise-o.

5. [2,0 pontos]

   (a) Explique informalmente o que significa um problema ser NP-completo. Do ponto de vista prático, qual é a relevância de se determinar que um certo problema é NP-completo?

   (b) Para cada uma das afirmações abaixo, diga se ela é *verdadeira*, *falsa*, *verdadeira se* $P \neq NP$ ou *falsa se* $P \neq NP$. Nos dois últimos itens, suponha que A e B são problemas em NP.

      1. Não há problemas em P que são NP-completos.
      2. Existe apenas algoritmo exponencial para o problema da parada.
      3. Existem problemas em P que estão em NP.
      4. Se A pode ser polinomialmente reduzido a B e B está em P então A está em P.
      5. Se A pode ser polinomialmente reduzido a B e B é NP-completo então A é NP-completo.

      Dê uma justificativa **curta** para cada resposta.