PROVA DE MC448 - PROJETO E ANÁLISE DE ALGORITMOS I IC-UNICAMP
PROFESSOR: FLÁVIO K. MIYAZAWA

SEGUNDA PROVA 10/05/2012

Obs.: Você pode usar algoritmos de ordenação, seleção (i-ésimo), busca binária, heap. Caso use, você deve dar as propriedades dos algoritmos e estruturas de dados usadas bem como a complexidade de tempo destes algoritmos. A soma total dos pontos desta prova é de 11 pontos. Caso a nota desta prova passe de 10, ela será truncada.

1. (2.5 pts) Considere um vetor $v = (v_1, \ldots, v_n)$ contendo n números com muitas duplicações (um valor pode aparecer várias vezes no vetor). A quantidade de números diferentes neste vetor é $\lfloor \log n \rfloor$. Projete um algoritmo de ordenação para ordenar este vetor usando $O(n \log \log n)$ comparações no pior caso. Prove a corretude do seu algoritmo e sua complexidade de tempo.

2. (2.5 pts) Apresente um algoritmo que, dado um vetor $v = (v_1, \ldots, v_n)$ contendo n números distintos, imprime os $k = \lceil \log n \rceil$ menores números do vetor em ordem crescente. Seu algoritmo deve ter complexidade de tempo linear. Prove a corretude do seu algoritmo e sua complexidade de tempo.

3. (3 pts) Seja $S = \{a_1, \ldots, a_n\}$ um conjunto de n atividades que podem ser executadas em um mesmo local. Cada atividade tem um tempo de início e fim dado por um intervalo de tempo $T_i = [s_i, f_i)$, onde $s_i < f_i$. Cada tarefa gera um benefício $b_i > 0$, caso seja executada. Um conjunto de atividades R é dito ser compatível se para quaisquer duas atividades distintas a_i e a_j de R temos $T_i \cap T_j = \emptyset$. O objetivo é encontrar um conjunto de atividades compatíveis $R \subseteq S$ para serem executadas e que tenham benefício total máximo (isto é, o valor $\sum_{i \in R} b_i$ é máximo). Faça um algoritmo que encontre o valor do benefício total máximo que pode ser obtido de S (não é preciso apresentar o conjunto de atividades que produz este benefício máximo). Seu algoritmo deve ter complexidade de tempo $O(n^3)$. Prove a corretude do seu algoritmo e sua complexidade de tempo. Obs.: Note que aqui não necessariamente teremos uma solução com cardinalidade (quantidade de tarefas) máxima.

4. (3 pts) Dado um conjunto S de n números com valores $S = \{v_1, v_2, \ldots, v_n\}$, onde $0 < v_i < 1$, para $i = 1, \ldots, n$. Um número v_i pode formar um par com o número v_j, para $i \neq j$, se $v_i + v_j \leq 1$. Um emparelhamento em S é um conjunto de pares de números de S, onde cada número é pareado no máximo uma vez. Faça um algoritmo que encontre um emparelhamento de cardinalidade máxima. Seu algoritmo deve ter complexidade de tempo $O(n \log n)$. Prove a corretude do seu algoritmo e sua complexidade de tempo.