Departamento de Ciência de Computadores FCUP

Algoritmos (CC4010)

2019/20

Teste (18.12.2019) *duração: 2h30 (+30')*

Cotação: (0.5+2+2), 2, 3, (2.5+1), (1,1.5.1.5+1,1,1)

N.º Nome

1. Recorde o problema *unit task scheduling* dado nas aulas, onde cada tarefa tem duração 1.

a) Explique o critério de *independência* (ou compatibilidade) para um subconjunto A de tarefas, dado pela condição "$N_k(A) \leq k$, para todo $k \geq 0$", onde $N_k(A)$ designa o número de tarefas de A que têm prazo limite não superior a k.

b) Considere a instância seguinte, sendo d_i o prazo limite para execução da tarefa i sem penalização e p_i o montante a pagar se executar a tarefa após esse prazo.

i	1	2	3	4	5	6	7	8	9	10	11	12
d_i	5	4	4	2	2	7	3	6	2	3	1	9
p_i	10	15	17	20	20	25	5	2	20	25	3	1

Apresente as tarefas que serão **realizadas dentro do prazo** pela ordem em que são escolhidas no algoritmo (em caso de empate, opte pela tarefa com identificador menor) . Indique o calendário para execução das 12 tarefas, com penalização mínima, que resulta de alocar cada tarefa o mais tarde possível, à medida que **o algoritmo** as vai processando, sem ultrapassar 12 slots de tempo . Indique a penalização .

c) Escreva o algoritmo referido acima em pseudocódigo e indique a sua complexidade temporal no pior caso e no melhor caso . Justifique sucintamente a complexidade nos dois casos.

2. Usando a definição de ordens de grandeza, justifique a veracidade ou falsidade de: "Se $h(n) \in \Omega(n^2)$ então $h(n) \in \Omega(35 + 10n + n^2)$, qualquer que seja a função $h : \mathbb{Z}^+ \to \mathbb{R}_0^+$".

3. Apresente os passos principais do algoritmo de aproximação dado nas aulas para o problema TSP métrico, com fator de aproximação 2 e a justificação de que garante essa aproximação. Explique em que passo a desigualdade triangular é relevante.

4. Considere o problema *Minimum Edit Distance*, em que os custos das operações de inserção e de remoção são iguais a c, mas o custo de uma substituição é $2c$, sendo $c > 0$ um inteiro (fixo).

a) Apresente (em pseudocódigo) a sua resolução por programação dinâmica, começando por indicar a recorrência em que se baseia.

N.º Nome

b) Determine o custo ótimo e uma solução ótima para conversão de **CADA** em **SALA**, para $c = 1$.

5. No problema **Load balancing** existem n tarefas que terão de ser realizadas por máquinas do mesmo tipo. Cada tarefa é processada sem interrupções por uma máquina. Cada máquina só pode estar a realizar uma tarefa em cada instante. A tarefa j tem duração d_j, para $j = 1, \ldots, n$. Existem m máquinas idênticas. As tarefas podem ser realizadas por qualquer ordem. Pretendemos atribuir tarefas às máquinas de modo a *minimizar a carga máxima* L das máquinas, definida por $L = \max_i C[i]$, sendo $C[i]$ a soma das durações das tarefas atribuídas à máquina i, para $i = 1, \ldots, m$. Admita que os valores d_j são inteiros positivos.

No problema **Partition** são dados n inteiros $a_1, \ldots, a_n$ e pretendemos saber se existe um subconjunto J de $\{1, \ldots, n\}$ tal que $\sum_{j \in J} a_j = \sum_{j \notin J} a_j$. Sabe-se que *Partition* é um problema **NP**-completo.

a) Prove que qualquer instância de *Partition* se pode reduzir (trivialmente) polinomialmente ao problema de decidir se uma instância de *Load balancing* com $m = 2$ admite uma solução com $L = \sum_j d_j/2$.

b) Justifique que o problema de decidir se uma qualquer instância de *Load Balancing* tem uma solução com $L \leq k$, para k arbitrário, é **NP**-completo, para qualquer $m \geq 2$ (com $n \gg m$).

c) Considere o algoritmo apresentado à direita, em que POSMIN(C, m) retorna o índice da primeira ocorrência do mínimo do *array* C, fazendo pesquisa linear.

1. Justifique que se trata de um algoritmo polinomial e que implementa uma estratégia *greedy* para atribuição das tarefas às máquinas.

2. Comente a veracidade das afirmações, supondo que $\mathbf{P}{\neq}\mathbf{NP}$: (i) "O algoritmo não obtém uma solução ótima para alguma instância de Load Balancing"; (ii) "Qualquer que seja a instância dada, sendo L o valor obtido pelo algoritmo e $L^\star$ o seu valor ótimo, tem-se $L \geq \alpha L^\star$, para algum $\alpha > 1$."

```
LOADBALANCING(d, n, m, S, C)
1.  L ← 0;
2.  Para i ← 1 até m fazer
3.      C[i] ← 0;
4.  Para j ← 1 até n fazer
5.      k ← POSMIN(C, m);
6.      S[j] ← k;
7.      C[k] ← C[k] + d[j];
8.      Se C[k] > L então L ← C[k];
9.  retorna L;
```

d) Justifique que qualquer que seja a instância dada se tem $L^\star \geq \max_j d_j$ e $L^\star \geq \frac{1}{m}\sum_j d_j$.

e) Seja L o valor retornado pelo algoritmo. Seja k' uma máquina que fica com carga L na solução (ou seja, $L = C[k']$). Seja j' a última tarefa atribuída à máquina k'. Tendo em conta a estratégia que o algoritmo implementa, justifique que $L - d_{j'} \leq C[i]$, para todas as máquinas i, considerando as cargas no momento em que processou j' no algoritmo e as cargas finais. Usando esse facto, conclua que, na linha 9, se tem $L - d_{j'} \leq \frac{1}{m}\sum_i C[i] = \frac{1}{m}\sum_i d_i \leq L^\star$, e que o algoritmo apresentado produz uma aproximação de fator 2, pelo que *Load Balancing* pertence à classe APX.

(Fim)