Algoritmos e Estruturas de Dados II

Exercícios – Caminho Crítico

1) Dado as atividades do projeto e os tempos requeridos para a execução destas tarefas:

- Qual é o caminho crítico deste projeto?
- Existem folgas no projeto? Em quais atividades?

a)

Atividade	**Duração**	**Dependência**
1	5	-
2	6	1
3	7	1
4	2	2, 3
5	6	4
6	5	4
7	3	6
8	1	5, 7

b)

Atividade	**Duração**	**Dependência**
A	5	-
B	4	-
C	3	A
D	2	A, B
E	5	B, C, D

2) Com base no algoritmo visto em aula para calcular o tempo inicial de cada tarefa em um sistema com paralelismo (código-fonte se encontra no Moodle), implemente:

a) A partir do código que calcula o caminho mais curto usando ordenação topológica (*AcyclicSP.java*), crie a classe *AcyclicLP.java*, que resolve o problema do caminho mais longo de um vértice até todos os demais. Para tanto, inicialize *distTo[]* com infinito negativo (*Double.NEGATIVE_INFINITY*) e troque o sinal de > por < no método *relax*.

b) Verifique que o algoritmo funciona, testando com o grafo "tinyEWDAG.txt".

c) A seguir, implemente o algoritmo de planejamento (na classe *EdgeWeightedDigraph.java*), gerando um grafo a partir da entrada de um arquivo com as tarefas, seus tempos e dependências. Lembre-se que você deve gerar dois vértices para cada tarefa: entrada e saída, mais a aresta que deve usar a duração como peso (slide 49 do material). O formato a ser utilizado para o arquivo está descrito a seguir:

```
10              # Total de tarefas
41.0  3  1 7 9  # Duração da tarefa 0, 3 dependem dela: 1, 7, 9
51.0  1  2      # Duração da tarefa 1, 1 depende dela: 2
50.0  0         # ...
36.0  0
38.0  0
45.0  0
21.0  2  3 8
32.0  2  3 8
32.0  1  2
29.0  2  4 6
```