Particionando tarefas de uma linha de produção

Para este trabalho, considere um linha de produção com as tarefas $1, 2, \ldots, n$. Um dos objetivos de um projeto de uma linha de produção é particionar o conjunto de tarefas em subconjuntos disjuntos de forma a definir estações de trabalho. Dessa forma, cada estação de trabalho, é responsável por parte das tarefas, aumentando a eficiência da linha.

No entanto, uma das dificuldades desse particionamento, é o fato de haver dependências entre as tarefas. Por exemplo, em uma linha de produção de sucos engarrafados, a tarefa *tampar a garrafa* não deve ser realizada antes da tarefa *encher a garrafa*. O conjunto de dependências entre as tarefas pode ser expresso por um grafo – chamado de *grafo de precedências*.

1 Grafo de precedências

Um grafo de precedências $G(V, E)$, é um grafo direcionado topologicamente ordenado composto por um conjunto de vértices $V = \{1, 2, \ldots, n\}$, representando um conjunto de tarefas e há uma aresta (i, j) entre as tarefas i, j se j não ser iniciada antes de i ser finalizada, ou seja, há uma ordem de precedência entre as tarefas i e j. O conjunto de precedências define, portanto, o conjunto de arestas E de um grafo de precedências.

A Figura 1 mostra um exemplo de grafo de precedências. Na figura, é possível observar, por exemplo, que a tarefa 2, precisa ser finalizada antes que as tarefas 3 e 7 iniciem. Também é possível observar que há duas tarefas finais, 14 e 15, indicando que a linha de produção é responsável pela produção de dois tipos de itens. A Figura 2 ilustra o resultado desse procedimento aplicado ao grafo da Figura 1. Note, por exemplo, que quando a estação T_4 termina as tarefas 4, 8 e 12, a estação T_5 pode iniciar as tarefas 5, 9 e 13.

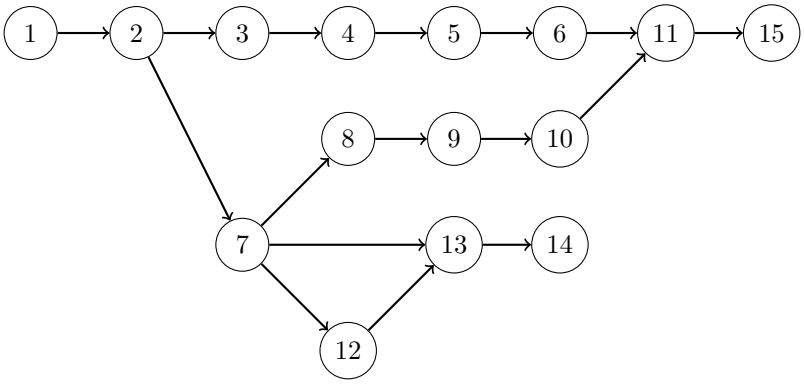

Figura 1: Exemplo de grafo de precedências.

2 Particionando o grafo em estações de trabalho

Uma linha de produção é organizada em estações de trabalho. As linhas simples são organizadas de forma sequencial, isto é, como uma sequência linear de estações de trabalho $T_1, T_2, \ldots, T_m$. Cada estação de trabalho é responsável pela realização de um subconjunto de tarefas. Um dos desafios desse particionamento é garantir a ordem de precedência entre as tarefas.

Uma primeira estratégia para estimar limitante superior para o número de estações de trabalho consiste em particionar as tarefas de acordo com sua "distância" relação à primeira

tarefa do grafo de precedência. Nessa estratégia, a tarefa 1 é alocada na primeira estação, T_1. Todas as tarefas que são liberadas para iniciar imediatamente após a finalização da tarefa 1 são candidatas para estarem na estação T_2 e o procedimento se repete para cada uma das novas tarefas liberadas. Um exemplo pode ser visto na Figura 2.

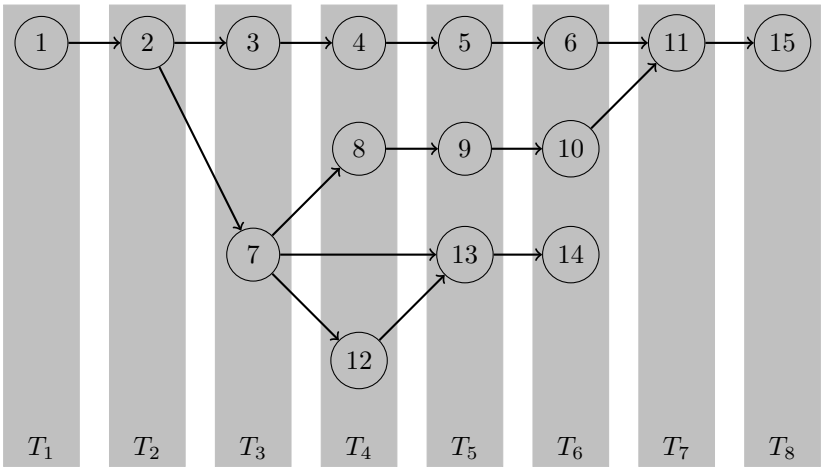

Figura 2: Particionamento das tarefas em estações de trabalho ilustrado no grafo de precedência.

Note que, a existência de uma aresta (i, j) implica que a tarefa j será alocada a uma estação a uma estação posterior à tarefa i. Respeitando tal regra, as tarefas são alocadas à estação mais próxima ao início da linha (T_1). Observe, como exemplo, as tarefas 7, 12 e 13 na Figura 2, é necessário que a tarefa 7 esteja em uma estação anterior a da tarefa 12 (pois existe a arestas (7, 12)), que, por sua vez, deve estar antes da tarefa 13 (aresta (12, 13)).

3 Tarefas

Usando os conceitos de orientação a objetos e encapsulamento, deve-se:

1. Implementar uma classe *Grafo* capaz de representar um grafo de precedências. As estruturas de dados utilizadas para representar o grafo devem estar *escondidas* do usuário da classe. Algumas funcionalidades que devem ser suportadas são: *adição de vértices*, *adição de arestas* e *coleção de vértices vizinhos a um vértice dado*.

2. Todas as estruturas utilizadas pelo classe *Grafo* devem ser de implementação própria (ou inspiradas pelo material fornecido na disciplina).

3. Implementar o algoritmo apresentado para a partição do grafo de precedências em estações de trabalho de forma determinar o número de estações necessárias na linha de produção. Esse algoritmo deve ser externo à classe *grafo*. Isto é, devem ser usados os métodos como *adição de vértice*, *adição de arestas* e *coleção de vértices vizinhos a um vértice dado* para implementar o algoritmo.

Para a implementação, pode-se assumir que a tarefa 1 é a inicial, ou seja, todas as tarefas dependem de a tarefa 1 ter sido realizada.

4 Formato de entrada e saída de dados

A entrada de dados é composta essencialmente pela definição do grafo de precedências. A primeira linha define o número de vértices do grafo e então há um número indefinido de linhas, cada linha definindo uma aresta. Uma aresta é definida pela cadeia de caracteres "i,j" para indicar que a existência da aresta (i, j) no grafo de precedências correspondente. A linha "-1,-1" indica o término da entrada. Por exemplo, o grafo das figuras 1 e 2 é representado da seguinte forma:

Entrada:

```
15
1,2
2,3
3,4
4,5
5,6
6,11
11,15
2,7
7,8
8,9
9,10
10,11
7,12
7,13
12,13
13,14
-1,-1
```

A saída esperada deve ser apenas um número inteiro correspondente ao número de estações de trabalho definido pelo algoritmo apresentado. Para o exemplo apresentado, a saída deve ser a seguinte.

Saída esperada:

```
8
```