# Instruções sobre o Exercício 4

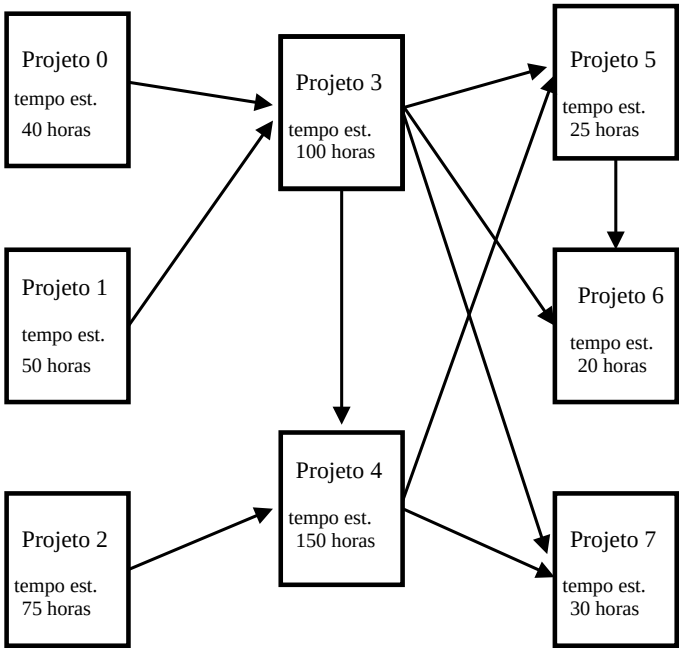
1000608/1001323 — Algoritmos em Grafos

Cândida Nunes da Silva

2º Semestre de 2020 – ENPE

## 1 Problema – Gerente de Projetos

Um gerente de projetos acabou de receber uma excelente notícia de sua chefe: assim que entregar todos os projetos sob sua responsabilidade no momento será promovido e seu salário dobrará! Imediatamente o gerente de projetos ficou ansioso para saber quando será a promoção e pediu (para ontem) uma estimativa, de cada equipe de cada projeto, do número de horas necessárias para concluir seu projeto. Suas equipes responderam rapidamente com suas estimativas, no entanto, lembraram que existem algumas relações de dependência entre os projetos gerenciados por ele, de forma que cada equipe só consegue começar quando todos os projetos de que dependem estiverem concluídos. A figura abaixo mostra um exemplo de um conjunto de projetos com os respectivos tempos estimados para conclusão e os projetos dos quais dependem. No exemplo da figura o tempo mínimo para a conclusão completa dos projetos (e para o gerente conseguir a promoção) é de 345 horas, dado pela soma dos tempos de conclusão dos projetos 1, 3, 4, 5 e 6 que dependem sequencialmente um do outro, nessa ordem. Qualquer outra sequência de dependências leva a um tempo total de conclusão menor.




Ao perceber que determinar o tempo mínimo até a promoção é mais complicado do que imaginou inicialmente, o gerente de projetos foi lamentar com seus outros colegas gerentes sua ansiedade, e

descobriu que todos eles estavam na mesma situação. Eles resolveram então se juntar e contratar você para determinar esse tempo mínimo para cada um deles.

## 2 Entrada

A entrada deve ser lida da entrada padrão (teclado). A primeira linha de cada caso de teste contém dois inteiros $N$ e $M$, separados por um espaço em branco, que representam, respectivamente, quantos são os projetos ($1 \leq N \leq 2.000$) e quantas são as relações de dependência ($1 \leq M \leq 5.000$). Cada projeto é representado por um inteiro entre 0 e $N-1$. A segunda linha de cada caso de teste contém $N$ inteiros $t_i$ ($1 \leq t_i \leq 10.000$) que representam os tempos estimados de conclusão de cada projeto. Cada uma das $M$ linhas subsequentes de cada caso de teste contém dois inteiros $u$ e $v$ ($0 \leq u, v \leq N-1$), separados por um espaço em branco, indicando que o projeto $v$ depende do projeto $u$ para poder ser concluído.

## 3 Saída

A saída deve ser escrita na saída padrão (terminal). Para cada caso de teste seu programa deve imprimir uma única linha contendo um único inteiro que é o tempo mínimo para o gerente concluir seus projetos atuais e obter a sua promoção.

## 4 Exemplo

| Entrada | Saída |
|---|---|
| 8 10<br>40 50 75 100 150 25 20 30<br>0 3<br>1 3<br>2 4<br>3 4<br>3 5<br>3 6<br>3 7<br>4 5<br>4 7<br>5 6 | 345 |

| Entrada | Saída |
|---|---|
| 5 4<br>10 20 30 40 50<br>0 1<br>1 2<br>3 1<br>1 4 | 110 |

| Entrada | Saída |
|---|---|
| 6 8<br>20 70 60 30 100 10<br>0 1<br>0 3<br>1 2<br>1 4<br>2 5<br>3 4<br>4 2<br>4 5 | 260 |

| Entrada | Saída |
|---|---|
| 1 0<br>9384 | 9384 |

## 5 Desenvolvimento e Apresentação

Cada aluno deve implementar a sua solução individual. A implementação da solução do problema deve ser em C em arquivo único. O nome do arquivo deve estar na forma "ex04-nomesn.c", onde "nomesn" representa o primeiro nome do aluno seguido das iniciais de seu sobrenome. Note que todas as letras são minúsculas e o separador é "-" (hífen) e não "_" (underscore).

O juiz online verificará seu programa comparando para cada um dos casos de teste se a saída gerada pelo seu programa é igual à saída esperada. É **imprescindível** que o **algoritmo** implementado esteja correto, isto é, retorne a solução esperada para **qualquer** entrada. É **desejável** que a implementação seja eficiente.

## 6 Ambiente de Execução e Testes

O programa deve ser compilável em ambiente Unix com `gcc`. Sugere-se que os testes também sejam feitos em ambiente Unix. Deve-se esperar que a entrada seja dada na entrada padrão (teclado) e não por leitura do arquivo de testes. Da mesma forma, a saída deve ser impressa na saída padrão (terminal), e não em arquivo.

A motivação dessa exigência é apenas simplificar a implementação de entrada e saída, permitindo o uso das funções `scanf` e `printf` da biblioteca padrão para leitura e escrita dos dados, sem precisar manipular arquivos.

Por outro lado, é evidente que efetivamente entrar dados no teclado é muito trabalhoso. Em ambiente Unix, é possível usar redirecionamento de entrada na linha de comando de execução para contornar esse problema. Supondo que o nome do arquivo executável seja análogo ao arquivo fonte, e "ex04.in" seja o arquivo com os casos de teste, a linha de comando:

```
shell$ ./ex04-nomesn < ex04.in
```

executa o programa para todos os casos de teste de uma só vez, retornando todas as saídas em sequência para o teminal. Novamente, pode-se usar o redirecionamento de saída na linha de comando para escrever a saída em um arquivo de saída de nome, por exemplo, "ex04.my.out". A respectiva linha de comando seria:

```
shell$ ./ex04-nomesn < ex04.in > ex04.my.out
```

Após a execução, a comparação pode ser feita usando o comando `diff` do Unix. Por exemplo, se o arquivo "ex04.out" contém as saídas esperadas, a linha de comando:

```
shell$ diff ex04.out ex04.my.out
```

serve para comparar automaticamente os dois arquivos, retornando nada caso sejam idênticos e as linhas onde há discrepâncias caso contrário.

# 7 Notas

As notas serão baseadas na correção da solução implementada, clareza do código fonte e eficiência da solução.

Trabalhos que não atendam aos requisitos mínimos expressos neste documento de forma a invibializar o teste do programa receberão nota ZERO. Em particular, receberá nota ZERO todo programa que:

- não compila em ambiente Unix;
- dá erro de execução;
- não usa entrada e saída padrão para leitura e escrita.