Universidade Federal do Rio de Janeiro

IM – Instituto de Matemática

DCC – Departamento de Ciência da Computação

Disciplina: Projeto de Teste de Software

Exercício: Modelo de Teste Baseado em Grafo

Professor(a): Anamaria Martins Moreira

Alunos: Igor Fonseca – DRE 112214399

Matheus Pinheiro – DRE: 112191208

Rio de Janeiro, RJ.

Maio 2017

SUMÁRIO

1 BANCO

1.1 DESCRIÇÃO DO PROBLEMA

http://olimpiada.ic.unicamp.br/pratique/programacao/nivel2/2012f2p2_banco

A legislação em vigor obriga os bancos a iniciarem o atendimento a um cliente em no máximo 20 minutos após a entrada do cliente na fila única da agência bancária. A fila é única, assim um caixa livre solicita ao primeiro cliente da fila que venha ao seu guichê para ser atendido. (Vamos ignorar aqui o problema dos clientes prioritários, idosos, gestantes, portadores de necessidades especiais, etc.) Estamos supondo também que nenhum caixa atende dois clientes ao mesmo tempo.

Inicialmente todos os caixas estão vazios, já que a agência acabou de abrir. Seu problema é determinar o número de clientes que esperarão mais de 20 minutos para ter seu atendimento iniciado.

1.1.1 Entradas

A primeira linha da entrada contém dois inteiros separados por um espaço em branco. O primeiro, *C*, é o número de caixas ativas na agência bancária. O segundo, *N*, o número de clientes que procurarão atendimento na agência naquele dia.

As próximas *N* linhas terão cada uma informações sobre um cliente, consistindo de dois inteiros, *T* e *D*, separados por um espaço em branco. O inteiro *T* fornece o momento em que o cliente entra na fila, em minutos, a partir do instante de abertura da agência. O inteiro *D* fornece, em minutos, o tempo necessário para atender o cliente.

As linhas estão ordenadas por entrada dos clientes na fila.

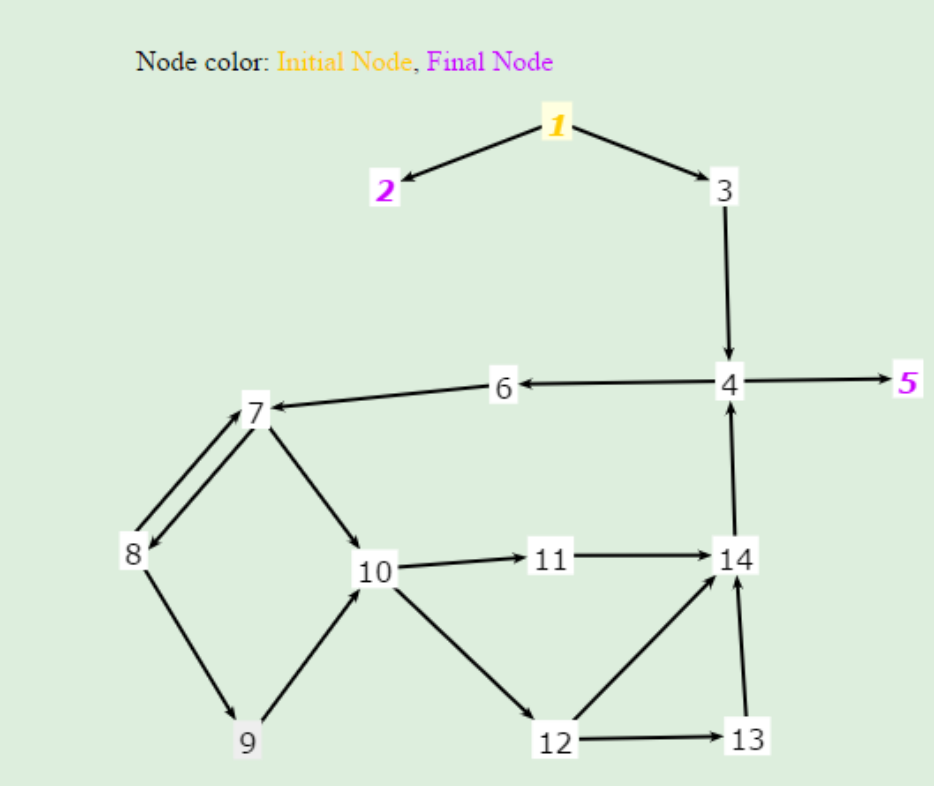
1.1.2 Saída

A saída deverá conter apenas uma linha, contendo um único inteiro, o número de clientes cujo atendimento será iniciado mais do que 20 minutos após sua entrada na fila.

1.1.3 Restrições

- $1 \leq C \leq 10$
- $1 \leq N \leq 1000$
- $0 \leq T \leq 300$
- $1 \leq D \leq 10$

1.2 REPRESENTAÇÃO DO ALGORITMO EM GRAFO



1.3 ESTRATÉGIAS

1.3.1 Estratégia de Nós: (14)

TR = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14 }

1.3.2 Estratégia de Arcos: (18)

TR = { (1,2), (1, 3), (3, 4), (4, 5), (4, 6), (6, 7), (7, 8), (7, 10), (8, 7), (8, 9), (9, 10), (10, 11) ,(10, 12), (11, 14), (12, 13), (12, 14), (13, 14), (14, 4) }

1.3.3 Estratégia de Pares de Arcos: (25)

TR = { (1,2), (1, 3, 4), (3, 4, 5), (3, 4, 6), (4, 6, 7), (6, 7, 8), (6, 7, 10), (7, 8, 9), (7, 8, 7), (7, 10, 11), (7, 10, 12), (8, 7, 10), (8, 7, 8), (8, 9, 10), (9, 10, 11), (9, 10, 12), (10, 11, 14), (10, 12, 13), (10, 12, 14), (11, 14, 4), (12, 13, 14), (12, 14, 4), (13, 14, 4), (14, 4, 5), (14, 4, 6) }

1.3.4 Estratégia de Caminhos Primos: (72)

TR = { (1, 2), (1, 3, 4, 5), (1, 3, 4, 6, 7, 10, 12, 14), (1, 3, 4, 6, 7, 10, 11, 14), (1, 3, 4, 6, 7, 10, 12, 13, 14), (1, 3, 4, 6, 7, 8, 9, 10, 11, 14), (1, 3, 4, 6, 7, 8, 9, 10, 12, 14), (1, 3, 4, 6, 7, 8, 9, 10, 12, 13, 14), (4, 6, 7, 10, 12, 14, 4), (4, 6, 7, 10, 11, 14, 4), (4, 6, 7, 8, 9, 10, 12, 14, 4), (4, 6, 7, 8, 9, 10, 11, 14, 4), (4, 6, 7, 8, 9, 10, 12, 13, 14, 4), (4, 6, 7, 10, 12, 13, 14, 4), (6, 7, 10, 12, 14, 4, 6), (6, 7, 10, 12, 14, 4, 5), (6,

7, 10, 11, 14, 4, 5), (6, 7, 10, 11, 14, 4, 6), (6, 7, 8, 9, 10, 12, 14, 4, 5), (6, 7, 8, 9, 10, 11, 14, 4, 6), (6, 7, 8, 9, 10, 11, 14, 4, 5), (6, 7, 8, 9, 10, 12, 14, 4, 6), (6, 7, 10, 12, 13, 14, 4, 6), (6, 7, 10, 12, 13, 14, 4, 5), (6, 7, 8, 9, 10, 12, 13, 14, 4, 5), (6, 7, 8, 9, 10, 12, 13, 14, 4, 6), (7, 8, 7), (7, 8, 9, 10, 12, 13, 14, 4, 6, 7), (7, 10, 12, 14, 4, 6, 7), (7, 10, 11, 14, 4, 6, 7), (7, 10, 12, 13, 14, 4, 6, 7), (7, 8, 9, 10, 12, 14, 4, 6, 7), (7, 8, 9, 10, 11, 14, 4, 6, 7), (8, 7, 8), (8, 7, 10, 11, 14, 4, 5), (8, 7, 10, 11, 14, 4, 6), (8, 7, 10, 12, 14, 4, 5), (8, 7, 10, 12, 14, 4, 6), (8, 7, 10, 12, 13, 14, 4, 5), (8, 7, 10, 12, 13, 14, 4, 6), (8, 9, 10, 12, 13, 14, 4, 6, 7, 8), (8, 9, 10, 11, 14, 4, 6, 7, 8), (8, 9, 10, 12, 14, 4, 6, 7, 8), (9, 10, 12, 13, 14, 4, 6, 7, 8, 9), (9, 10, 11, 14, 4, 6, 7, 8, 9), (9, 10, 12, 14, 4, 6, 7, 8, 9), (10, 11, 14, 4, 6, 7, 10), (10, 12, 14, 4, 6, 7, 10), (10, 12, 13, 14, 4, 6, 7, 10), (10, 12, 14, 4, 6, 7, 8, 9, 10), (10, 11, 14, 4, 6, 7, 8, 9, 10), (10, 12, 13, 14, 4, 6, 7, 8, 9, 10), (11, 14, 4, 6, 7, 10, 11), (11, 14, 4, 6, 7, 10, 12, 13), (11, 14, 4, 6, 7, 8, 9, 10, 11), (11, 14, 4, 6, 7, 8, 9, 10, 12, 13), (12, 14, 4, 6, 7, 10, 12), (12, 14, 4, 6, 7, 10, 11), (12, 13, 14, 4, 6, 7, 10, 11), (12, 13, 14, 4, 6, 7, 10, 12), (12, 14, 4, 6, 7, 8, 9, 10, 11), (12, 14, 4, 6, 7, 8, 9, 10, 12), (12, 13, 14, 4, 6, 7, 8, 9, 10, 12), (12, 13, 14, 4, 6, 7, 8, 9, 10, 11), (13, 14, 4, 6, 7, 10, 12, 13), (13, 14, 4, 6, 7, 8, 9, 10, 12, 13), (14, 4, 6, 7, 8, 9, 10, 12, 13, 14), (14, 4, 6, 7, 8, 9, 10, 12, 14), (14, 4, 6, 7, 8, 9, 10, 11, 14), (14, 4, 6, 7, 10, 12, 13, 14), (14, 4, 6, 7, 10, 11, 14), (14, 4, 6, 7, 10, 12, 14) }

1.4 CASOS DE TESTE

- CT1 - C = 2, N = 1, tempos = [“1 1”]
    - Resultado esperado: 0
    - Resultado obtido: 0
    - Caminho: `[1-2]`
- CT2 - C = 3, N = 4, tempos = [“1 1”, “1 10”, “3 10”, “3 1”]
    - Resultado esperado: 0
    - Resultado obtido: 0
    - Caminho: `[1-3-4-6-7-10-11-14-4-6-7-8-7-10-11-14-4-6-7-10-12-14-4-5]`
- CT3 - C = 3, N = 10, tempos = [“0 10”, “0 10”, “10 10”, “10 10”, “10 10”, “10 10”, “10 10”, “10 10”, “10 10”, “10 10”]
    - Resultado esperado: 0
    - Resultado obtido: 0
    - Caminho: `[1-3-4-6-7-10-11-14-4-6-7-8-7-8-9-10-11-14-4-6-7-10-11-14-4-6-7-10-12-14-4-6-7-10-12-14-4-6-7-10-12-14-4-6-7-10-12-14-4-6-7-10-12-13-14-4-6-7-10-12-13-14-4-5]`
- CT4 - C = 1, N = 2, tempos = [“0 10”, “10 10”]
    - Resultado esperado: 0

- Resultado obtido: erro (NoSuchElementException)
- Caminho: [1-3-4-6-7-8-9-10-12-erro

1.5 CONSIDERAÇÕES QUANTO AO CÓDIGO

Inicialmente, passamos o código que estava escrito em Python para Java, para seguir igual nossos últimos trabalhos. Porém, ao executar os testes, vimos que nunca estava passando por um nó que era para passar. Investigamos através do depurador e percebemos que estávamos utilizando a função indexOf() ao invés do get(), com isso, tinhamos índices negativos no array e que não retornavam nenhum elemento válido.

Corrigimos o erro para ficar igual ao código em Python e, apesar dos testes preencherem os requisitos, descobrimos alguns defeitos do código. Por exemplo, o nó 10, referente ao “if (caixas > 1)”, deveria ser “>=”.

Outro ponto a se destacar é que o requisito de par de arco (3, 4, 5) nunca irá acontecer, pois seria necessário que o número de clientes fosse maior que o número de caixas e que o nó 4, referente ao loop for, náo fosse executado nenhuma vez. Porém, o N teria que ser igual a 1 e, como o número mínimo válido de caixas também é 1, o programa terminaria a execução logo nos primeiros nós (1 e 2).

Por último, temos que o requisito de par de arco (9, 10, 12) também é impossível de acontecer, pois o nó 9 representa uma condição em que o vetor “termina” é vazio e o nó 12 representa métodos de acesso aos índices desse vetor. Porém, por causa do defeito no código, o caso de teste CT4 acaba encontrando este erro, estourando um erro NoSuchElementException no console.

2 PALÍNDROMO

2.1 DESCRIÇÃO DO PROBLEMA

Um palíndromo é uma palavra, frase ou sequência de unidades que tem a propriedade de poder ser lida tanto da direita pra esquerda, como da esquerda pra direita. O problema consistem em, dado uma palavra ou frase de entrada, dizer se a mesma é ou não um palíndromo.

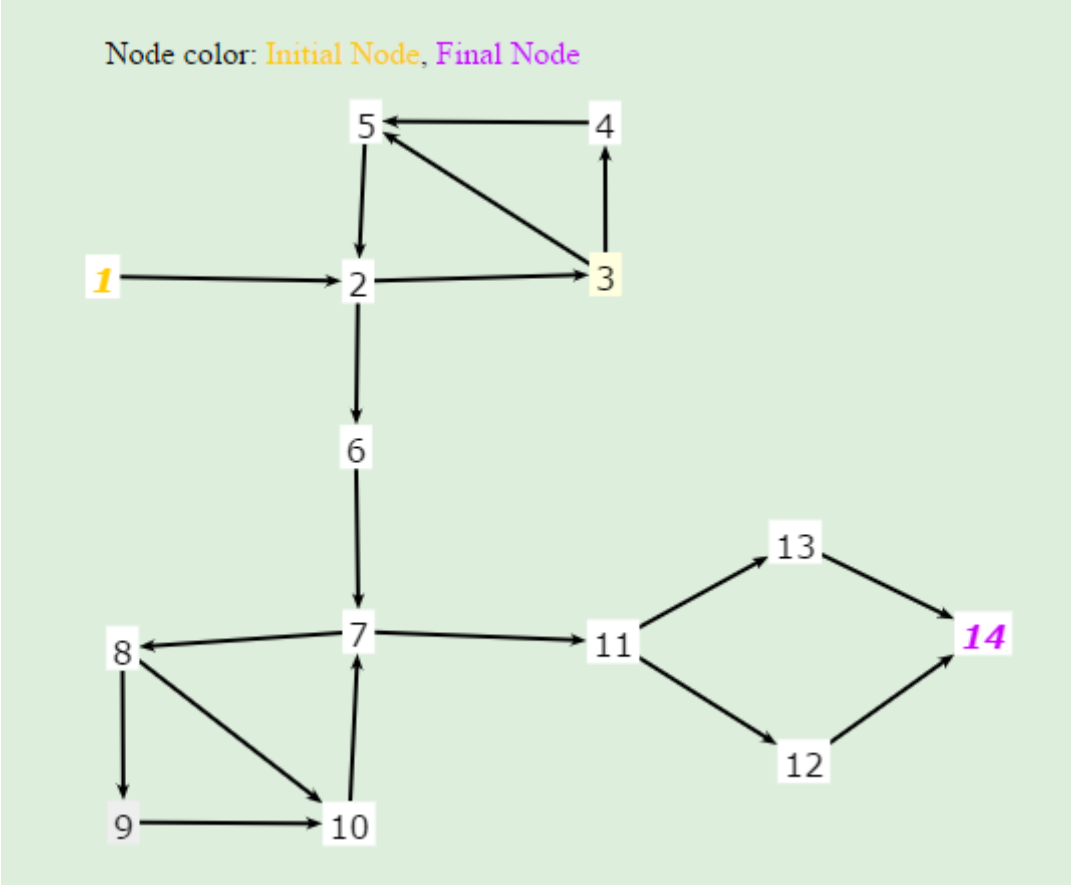
2.1.1 Entradas

A entrada será composta por uma string que deseja-se verificar se é ou não um palíndromo.

2.1.2 Saída

A saída será composta por uma string “SIM”, caso a palavra seja um palíndromo, ou “NAO”, caso a palavra não seja um palíndromo.

2.2 REPRESENTAÇÃO DO ALGORITMO EM GRAFO



2.3 ESTRATÉGIAS

2.3.1 Estratégia de Nós: (14)

TR = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14 }

2.3.2 Estratégia de Arcos: (18)

TR = { (1, 2), (2, 3), (2, 6), (3, 4), (3, 5), (4, 5), (5, 2), (6, 7), (7, 8), (7, 11), (8, 9), (8, 10), (9, 10), (10, 7), (11, 12), (11, 13), (12, 14), (13, 14) }

2.3.3 Estratégia de Pares de Arcos: (23)

TR = { (1, 2, 3), (1, 2, 6), (2, 3, 4), (2, 3, 5), (2, 6, 7), (3, 4, 5), (3, 5, 2), (4, 5, 2), (5, 2, 3), (5, 2, 6), (6, 7, 8), (6, 7, 11), (7, 8, 9), (7, 8, 10), (7, 11, 12), (7, 11, 13), (8, 9, 10), (8, 10, 7), (9, 10, 7), (10, 7, 8), (10, 7, 11), (11, 12, 14), (11, 13, 14) }

2.3.4 Estratégia de Caminhos Primos: (32)

TR = { (1, 2, 3, 5), (1, 2, 3, 4, 5), (1, 2, 6, 7, 8, 10),(1, 2, 6, 7, 8, 9, 10), (1, 2, 6, 7, 11, 13, 14),(1, 2, 6, 7, 11, 12, 14),(2, 3, 5, 2), (2, 3, 4, 5, 2), (3, 5, 2, 3), (3, 4, 5, 2, 3), (3, 4, 5, 2, 6, 7, 8, 9, 10), (3, 4, 5, 2, 6, 7, 11, 12, 14), (3, 4, 5, 2, 6, 7, 11, 13, 14), (3, 5, 2, 6, 7, 8, 9, 10), (3, 4, 5, 2, 6, 7, 8, 10), (3, 5, 2, 6, 7, 11, 12, 14), (3, 5, 2, 6, 7, 11, 13, 14), (3, 5, 2, 6, 7, 8, 10), (4, 5, 2, 3, 4), (5, 2, 3, 4, 5), (5, 2, 3, 5), (7, 8, 10, 7),(7, 8, 9, 10, 7), (8, 10, 7, 8),(8, 9, 10, 7, 8), (8, 9, 10, 7, 11, 12, 14), (8, 9, 10, 7, 11, 13, 14), (8, 10, 7, 11, 12, 14), (8, 10, 7, 11, 13, 14), (9, 10, 7, 8, 9), (10, 7, 8, 9, 10), (10, 7, 8, 10) }

2.4 CASOS DE TESTE

- CT1: “ arara ”
    - Resultado esperado: “SIM”
    - Resultado obtido: “SIM”
    - Caminho: `[1-2-3-5-2-3-4-5-2-3-4-5-2-3-4-5-2-3-4-5-2-3-5-2-6-7-8-10-7-8-10-7-11-12-14]`
- CT2: “teste”
    - Resultado esperado: “NAO”
    - Resultado obtido: “NAO”
    - Caminho: `[1-2-3-4-5-2-3-4-5-2-3-4-5-2-3-4-5-2-3-4-5-2-6-7-8-9-10-7-8-9-10-7-11-13-14]`
- CT3: “ ”
    - Resultado esperado: “SIM”

- Resultado obtido: “SIM”
- Caminho: [1-2-3-5-2-6-7-11-12-14]

- CT4: “”
  - Resultado esperado: “SIM”
  - Resultado obtido: “SIM”
  - Caminho: [1-2-6-7-11-12-14]
- CT5: “a b”
  - Resultado esperado: “NAO”
  - Resultado obtido: “NAO”
  - Caminho: [1-2-3-4-5-2-3-5-2-3-4-5-2-6-7-8-9-10-7-11-13-14]
- CT6: “ ”
  - Resultado esperado: “SIM”
  - Resultado obtido: “SIM”
  - Caminho: [1-2-3-4-5-2-6-7-11-12-14]
- CT7: “bcca”
  - Resultado esperado: “NAO”
  - Resultado obtido: “NAO”
  - Caminho: [1-2-3-4-5-2-3-4-5-2-3-4-5-2-3-4-5-2-6-7-8-9-10-7-8-10-7-11-13-14]
- CT8: “ab ”
  - Resultado esperado: “NAO”
  - Resultado obtido: “NAO”
  - Caminho: [1-2-3-4-5-2-3-4-5-2-3-5-2-6-7-8-9-10-7-11-13-14]