MC102 - Algoritmos e Programação de Computadores
Turmas QRSTWY
Instituto de Computação - Unicamp
Professores: *Hélio Pedrini e Zanoni Dias*
Monitores: *Andre Rodrigues Oliveira, Gustavo Rodrigues Galvão, Javier Alvaro Vargas Muñoz e Thierry Pinheiro Moreira*

Lab 15b - Ó o CR!

Prazo de entrega: 29/06/2015 às 13h59m59s
Peso: 10

Em um dos primeiros laboratórios, vimos que o *Coeficiente de Rendimento* (CR) é o índice que mede o desempenho acadêmico de um aluno ao longo de seu curso. Por essa razão, o CR é geralmente o principal fator considerado pela universidade para preenchimento das vagas oferecidas durante o processo de matrícula, classificação do aluno em sua turma e até seleção de candidatos para realizar programas de intercâmbio.

Justamente por ser um dos principais fatores em boa parte dos processos internos da universidade, os funcionários da DAC (Diretoria Acadêmica) constumam se referir aos processos que dependem do CR como processos "Ó o CR". Essa expressão resulta da simplificação fonética de "Olha o CR" e remete ao fato de que, para tomar decisões, é necessário "olhar" o CR dos alunos. Além disso, essa expressão faz alusão à sigla OCR (*Optical Character Recognition*), que refere-se a tecnologia de reconhecimento de caracteres a partir de uma imagem. Tal alusão tem suas raízes no fato de que, para calcular o CR de um aluno, é preciso levar em consideração tanto as notas obtidas nas disciplinas cursadas na Unicamp quanto, eventualmente, as notas obtidas nas disciplinas cursadas em outra universidade. As notas da Unicamp são preenchidas pelos professores, porém as notas de outras universidades tem que ser processadas pelos funcionários da DAC. Esse processamento envolve digitalizar o histórico escolar do aluno oriundo de outra universidade e utilizar um OCR para "textualizar" o histórico digitalizado.

Um OCR nem sempre textualiza um documento digitalizado da maneira correta (inclusive, a imperfeição dos OCRs serve de motivação para o projeto reCAPTCHA do Google). Uma das incorreções observadas é o embaralhamento de blocos de texto, chamados *zonas*. Para entender melhor, observe o exemplo abaixo.

Texto correto: `Pinda` `monha` `ngaba`

Texto do OCR: `ngaba` `monha` `Pinda`

Neste exemplo, temos que a palavra "Pindamonhangaba" está divida em três zonas de 5 letras cada. Além disso, temos que o OCR embaralhou as zonas, invertendo a ordem correta. Em casos como esse, as correções devem ser feitas manualmente usando um editor de texto.

Seja uma correção o ato de mover, no texto errado, uma *substring* do texto correto para outra posição. Nós temos que, no exemplo anterior, são necessárias duas correções. Por exemplo, primeiro podemos mover a *substring* "Pinda" para o começo, obtendo o texto "Pindangabamonha". Depois, podemos mover a *substring* "ngaba" para o final, obtendo o texto correto. O número mínimo de correções necessárias para corrigir um texto produzido por um OCR é chamado de *métrica de zoneamento*. Essa

métrica pode ser utilizada para medir a qualidade de um OCR: quanto maior a métrica de zoneamento, menor a qualidade de um OCR.

A métrica de zoneamento pode ser calculada resolvendo-se um problema combinatório chamado Problema da Ordenação de Blocos. Dada uma permutação de número inteiros π, uma sequência contígua maximal de elementos $\pi_i\ \pi_{i+1}\ \ldots\ \pi_j$, com $1 \le i \le j \le n$, tal que $\pi_k = \pi_{k+1} - 1$ para todo $k = i, i+1, \ldots, j-1$ é chamada de bloco. Por exemplo, na permutação $\pi = (4\ 2\ 3\ 9\ 5\ 6\ 7\ 8\ 1)$, os blocos aparecem sublinhados. O Problema da Ordenação de Blocos consiste em encontrar o menor número de movimentos de blocos necessários para transformar uma permutação π na permutação identidade $\iota_n = (1\ 2\ \ldots\ n)$. Este número é chamado de *distância de bloco* da permutação π e é denotado por $d(\pi)$.

Abaixo, você irá encontrar uma das possíveis soluções para o Problema da Ordenação de Blocos para o caso em que $\pi = (8\ 7\ 6\ 5\ 10\ 4\ 3\ 9\ 2\ 1)$. Os números sublinhados indicam qual bloco foi movido para gerar a próxima permutação da sequência. Note que $d(\pi) = 8$ nesse caso.

```
(8 7 6 5 10 4 3 9 2 1)
(8 7 6 5 9 10 4 3 2 1)
(7 6 5 8 9 10 4 3 2 1)
(6 5 7 8 9 10 4 3 2 1)
(5 6 7 8 9 10 4 3 2 1)
(4 3 2 1 5 6 7 8 9 10)
(3 2 1 4 5 6 7 8 9 10)
(2 1 3 4 5 6 7 8 9 10)
(1 2 3 4 5 6 7 8 9 10)
```

Dada uma permutação π com n elementos, nós podemos estendê-la com outros dois elementos 0 e $n+1$ tal que $\pi = (0\ \pi_1\ \pi_2\ \ldots\ \pi_n\ n+1)$. Note que, por simplicidade, a permutação estendida também é denotada por π. Nós dizemos que um *breakpoint* ocorre entre um par de elementos π_i e π_{i+1}, com $0 \le i \le n$, se $\pi_{i+1} - \pi_i \neq 1$ (isto é, dois elementos que deveriam ser consecutivos na identidade mas não são na permutação π). Na permutação $\pi = (0\ *\ 3\ *\ 2\ *\ 4\ 5\ *\ 1\ *\ 6)$, por exemplo, os breakpoints aparecem marcados com $*$. Com isso, podemos dizer alternativamente que um bloco é uma subsequência maximal $\pi_i,\ldots,\pi_j$ de π, com $1 \le i \le j \le n$, tal que os elementos da subsequência não formam *breakpoints*. Como a permutação identidade é a única sem *breakpoints*, transformar uma permutação π na permutação identidade é equivalente a remover os *breakpoints* da permutação π.

O número de *breakpoints* de uma permutação π é denotado por $b(\pi)$. Uma propriedade importante do Problema da Ordenação de Blocos é que não é possível ordenar uma permutação π qualquer com menos do que $b(\pi)/3$ movimentos de bloco. Isso ocorre porque cada movimento de bloco pode remover no máximo 3 *breakpoints* de uma permutação. Por isso, dizemos que o limite inferior da distância de bloco de uma permutação π é $b(\pi)/3$, isto é, a distância de qualquer permutação será sempre um valor maior ou igual a $b(\pi)/3$. Outra propriedade importante do Problema da Ordenação de Blocos é que é sempre possível ordenar uma permutação π qualquer com no máximo $b(\pi)$ movimentos de bloco. Isso ocorre porque sempre existe pelo menos um movimento de bloco que remove no mínimo um *breakpoint* de uma permutação (caso ela não seja a identidade). Por isso, dizemos que o limite superior da distância de bloco de uma permutação π é $b(\pi)$, isto é, a distância de qualquer permutação será sempre um valor menor ou igual a $b(\pi)$.

Para entender de que maneira a distância de bloco é equivalente à métrica de zoneamento, voltemos ao primeiro exemplo. Nós temos que o texto correto pode ser representado pela permutação identidade $(1\ 2\ 3)$, ou seja, a zona `Pinda` é o elemento 1, a zona `monha` é o elemento 2 e a zona `ngaba` é o elemento 3. Deste modo, note que o texto do OCR pode ser representado pela permutação $(3\ 2\ 1)$. A distância de bloco desta permutação é 2, igual à métrica de zoneamento calculada anteriormente.

Os funcionários da DAC gostariam de determinar qual dos OCRs disponíveis produz os melhores resultados. Para isso, eles pediram que você os ajudassem implementando um programa que calcula a distância de bloco de uma permutação qualquer. Caso você esteja desmotivado(a), os funcionários pediram que lhe fosse dado o seguinte aviso: ó o CR!

Observação: Neste link, na seção *Search Rearrangement Distance*, você pode descobrir não só a distância de bloco como também a sequência de permutações intermediárias da ordenação de qualquer permutação com tamanho menor ou igual a 12. Para permutações com tamanho 13, é possível apenas obter a distância. Basta digitar no campo *Permutation* a permutação desejada, escolher *Strip Move* no campo *Rearrangement model* e selecionar a opção *Solution*.

Entrada

- A primeira linha da entrada contém um único número inteiro `n`, `2 ≤ n ≤ 10`, que indica o tamanho da permutação π.
- A próxima linha contém `n` inteiros positivos entre `1` e `n`, que representam a permutação π.

Saída

Seu programa deve imprimir um único número `d` que indica o valor da distância de bloco da permutação π.

Exemplos

#	Entrada	Saída
1	3 3 2 1	2
2	2 1 2	0
3	4 2 1 4 3	2
4	5 5 4 3 2 1	4
5	6 5 4 2 1 6 3	4