Instruções sobre o Exercício 2

1001101/481556 — Projeto e Análise de Algoritmos
Cândida Nunes da Silva
2º Semestre de 2020

1 Problema - Pescaria

Nos tempos pré-pandêmicos, junho sempre foi um mês de festas juninas, quermesse, comidas e brincadeiras típicas. Ah, que saudades dessas aglomerações!

Para tentar aplacar a saudade dos alunos destes festejos, promover socialização e estimular brincadeiras possíveis, uma escola decidiu fazer a "pescaria online". A brincadeira demanda que um par de alunos colabore, sendo que cada um deve escolher um peixe em sua "piscina virtual". Cada peixinho está associado a um número inteiro não negativo, e a lista dos números inteiros que representam todos os peixinhos da piscina virtual de cada aluno é mostrada na parte superior da tela. Assim que cada aluno escolhe seu peixinho, é calculado o produto dos números inteiros relativos aos dois peixinhos, e esse produto define o prêmio que ambos alunos ganharão.

Os prêmios são expostos antecipadamente aos participantes, juntamente com o produto que deve ser obtido para ganhá-lo. Acontece que Vítor e Alice, colegas muito competitivos e desconfiados, acreditam que existe trapaça na brincadeira. Eles acreditam que existam prêmios (muito bons e caros) para os quais é impossível atingir seu produto. Então eles resolveram pedir a sua ajuda.

Você deve construir um programa que recebe os valores de todos os peixinhos das duas piscinas e o valor do produto associado ao prêmio que os alunos acreditam que é impossível obter e determinar se existe uma combinação de peixinhos, um de cada piscina, que lhes permita ganhar o prêmio. Seu programa **deve ter complexidade de pior caso** $O(n \log n)$**!**

2 Entrada

A entrada deve ser lida da entrada padrão. O caso de teste é dado em precisamente três linhas. A primeira linha contém dois números inteiros N ($1 \leq N \leq 5.000$) e X ($0 \leq X \leq 100.000.000$) que indicam respectivamente o número de peixinhos em cada piscina e o produto que deve ser obtido para ganhar o prêmio. As 2 linhas seguintes possuem cada uma N inteiros P ($0 \leq P \leq 10.000$) que indica o valor associado aos peixes de cada piscina.

3 Saída

A saída deve ser escrita na saída padrão (terminal) e deve ter uma única linha escrito SIM caso seja possível ganhar aquele prêmio ou NAO caso contrário. Deve haver também uma quebra de linha.

4 Exemplo

Entrada	Saída
5 1090 49 158 272 278 109 440 165 492 42 10	SIM

Entrada	Saída
10 734 672 393 336 485 745 228 12 15 77 128 94 101 208 268 24 30 100 23 45 99	NAO

Entrada	Saída
15 289212 101 177 228 298 235 165 263 536 425 669 115 94 629 501 33 95 104 198 23 82 266 237 1208 2345 15 22 865 231 8764 10	SIM

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 "ex02-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).

Serão disponibilizados vários arquivos com as entradas e as respectivas saídas esperadas. É **imprescindível** que o **algoritmo** implementado esteja correto, isto é, retorne a solução esperada para **qualquer** arquivo de testes. Também é necessário que a implementação tenha a **complexidade esperada**.

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 "ex02.in" seja o arquivo com um caso de teste, a linha de comando:

```
shell$ ./ex02-nomesn < ex02.in
```

executa o programa para o caso de teste, retornando a saída 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, "ex02.my.out". A respectiva linha de comando seria:

```
shell$ ./ex02-nomesn < ex02.in > ex02.my.out
```

Após a execução, a comparação pode ser feita usando o comando `diff` do Unix. Por exemplo, se o arquivo "ex02.out" contém as saídas esperadas, a linha de comando:

```
shell$ diff ex02.out ex02.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.
- apresentar sinais claros de cópia seja de outros colegas ou de código disponível na Internet.