Instruções sobre o Trabalho 3

481556 — Algoritmos e Complexidade

Cândida Nunes da Silva

1º Semestre de 2014

1 Pescaria

Junho é um mês de festas juninas, quermesse, comidas e brincadeiras típicas. Joaninha foi com seus pais na festa junina de sua escola e como toda criança ela queria participar das brincadeiras e concorrer a um prêmio.

A brincadeira consiste numa pescaria. O participante recebe uma vara e tem que pescar três peixinhos um de cada piscina. Os peixinhos possuem um número inteiro escrito em suas brânquias. Então ao retirar os três peixinhos, soma-se os seus valores e assim determina-se o prêmio. Os prêmios são expostos aos participantes juntamente com a soma que deve ser obtida para ganhá-lo.

Joaninha está desconfiada e acredita que existe trapaça na brincadeira. Ela acredita que existam prêmios (muito bons e caros) para os quais é impossível atingir sua soma. Então ela resolveu pedir a sua ajuda. Você deve construir um programa que recebe os valores de todos os peixinhos das três piscinas e o valor da soma associado ao prêmio que Joaninha desconfia que é impossível obter e determinar se existe uma combinação de peixinhos, um de cada piscina, que lhe permita ganhar o prêmio.

2 Entrada

A entrada deve ser lida da entrada padrão (teclado) e será composta por diversos casos de teste. A primeira linha de cada caso de teste contém dois números inteiros N ($1 \leq N \leq 5000$) e X ($-1000 \leq X \leq 5000$) que indicam respectivamente o número de peixinhos em cada piscina e a soma que deve ser obtida para que Joaninha ganhe o prêmio. As 3 linhas seguintes possuem cada uma N inteiros P ($-300 \leq P \leq 1000$) que indica o valor associado aos peixes de cada piscina.

O último caso de teste é seguido por uma linha contendo dois zeros separados por espaço.

3 Saída

A saída deve ser escrita na saída padrão (terminal). Para cada caso de teste a saída deve ter uma única linha escrito SIM caso seja possível ganhar aquele prêmio ou NÃO caso contrário.

4 Exemplo

Entrada	Saída
5 -107 -49 -158 272 278 -109 440 165 492 42 987 -127 -40 3 -9 -160 6 734 -97 212 -110 -79 -279 -67 672 393 336 485 745 228 -294 -101 -208 268 24 30 7 1208 101 -177 228 -298 -235 -165 263 536 425 669 115 94 629 501 -95 -104 -198 23 -82 266 237 0 0	SIM SIM NÃO

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 "t3-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 arquivos com diversas entradas (t3.in) e as respectivas saídas esperadas (t3.sol). É **imprescindível** que o **algoritmo** implementado esteja correto, isto é, retorne a solução esperada para **qualquer** entrada do arquivo de testes. 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. Será disponibilizado no moodle um trabalho modelo (trabalho 0) que faz a entrada e a saída da forma requerida.

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 "t3.in" seja o arquivo com os casos de teste, a linha de comando:

```
shell$ ./t3-nomesn < t3.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, "t3.my.sol". A respectiva linha de comando seria:

```
shell$ ./t3-nomesn < t3.in > t3.my.sol
```

Após a execução, a comparação pode ser feita usando o comando `diff` do Unix. Por exemplo, se o arquivo "t3.sol" contém as saídas esperadas, a linha de comando:

```
shell$ diff t3.sol t3.my.sol
```

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

7 Entrega e Prazos

A data para a entrega do projeto é dia 12 de junho. Cada aluno deve entregar via moodle seu único arquivo fonte com nome no padrão requerido até essa data. Lembre-se que é **imprescindível** que o código compile em ambiente Unix e que a entrada e a saída sejam feitas pela entrada e saída padrão.

8 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.