Instruções sobre o Exercício 1

1001101/481556 — Projeto e Análise de Algoritmos

Cândida Nunes da Silva

2º Semestre de 2020

1 Problema

Dada uma seqüência $X = x_1, x_2, \ldots, x_n$ de números inteiros (não necessariamente positivos), queremos encontrar uma **subseqüência consecutiva** $Y = x_i, x_{i+1}, \ldots, x_j$ de X, onde $1 \leq i, j \leq n$, cuja soma de seus elementos seja máxima dentre todas as subseqüências consecutivas.

Implemente um programa em C **de complexidade** $O(n)$ que resolva uma versão simplificada deste problema em que retorna apenas o valor da soma dos elementos da subsequência consecutiva de soma máxima.

Supõe-se que uma sequência vazia tem soma zero, portanto tal soma é sempre não negativa.

2 Entrada

A entrada deve ser lida da entrada padrão. Cada caso de teste é dado em precisamente duas linhas. A primeira contém um inteiro N ($1 \leq N \leq 10^6$) que indica o tamanho da sequência de inteiros da entrada. A segunda contém N números inteiros X_i ($-10^3 \leq X_i \leq 10^3$) representando a sequência da entrada.

3 Saída

A saída deve ser escrita na saída padrão e deve conter uma única linha contendo um único inteiro que é o valor da soma dos elementos da subsequência consecutiva de soma máxima seguido de uma quebra de linha.

4 Exemplo

Entrada	Saída
8 3 0 -1 2 4 -1 2 -2	9

Entrada	Saída
3 -1 -2 0	0

Entrada	Saída
2 -1 -3	0

Entrada	Saída
10 1 -2 1 10 14 -7 -7 5 3 4	25

Entrada	Saída
20 10 12 -3 -4 -5 -6 -7 8 8 2 2 5 -9 10 2 3 9 -1 -2 4	41

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 "ex01-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 "ex01.in" seja o arquivo com um caso de teste, a linha de comando:

```
shell$ ./ex01-nomesn < ex01.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, "ex01.my.out". A respectiva linha de comando seria:

```
shell$ ./ex01-nomesn < ex01.in > ex01.my.out
```

Após a execução, a comparação pode ser feita usando o comando `diff` do Unix. Por exemplo, se o arquivo "ex01.out" contém as saídas esperadas, a linha de comando:

```
shell$ diff ex01.out ex01.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.