MC458 Projeto e Análise de Algoritmos I - 2016/02

Laboratório 4

Docente: Cid Carvalho de Souza

Monitora PED: Amanda Resende

Monitor PAD: Allan Sapucaia Barboza

Descrição

A empreiteira Ódmin está planejando a construção de duas torres no centro financeiro de NobodySLand. Ela já fechou contratos de venda com N empresas. O contrato garante à empresa i a ocupação de A_i andares em exatamente uma das torres (isso evita que os funcionários da empresa i tenham que se deslocar entre os prédios durante o trabalho).

NobodySLand é famosa pela corrupção dos agentes públicos, mas justamente o setor responsável pelo licenciamento da obra é dirigido por um funcionário incorruptível chamado John Doe. Apesar de todas as pressões da Ódmin sobre o governo e os políticos, com um amplo apoio da imprensa local, John Doe conseguiu fazer prevalecer pelo menos a regra de que o empreendimento só sairia do papel se fosse provado que a **altura da torre mais alta seria a menor possível**, já considerando os contratos fechados entre as N empresas e a empreiteira.

Os técnicos da Ódmin tiveram então que se debruçar sobre o seguinte problema. Dadas as quantidades de andares $A_1, A_2, \ldots, A_N$ já contratados pelas N empresas, os quais devem ser alocados consecutivamente em uma mesma torre, pergunta-se: qual é o menor número de andares que pode ter a torre mais alta do empreendimento?

Os técnicos sabem que não adianta trapacear porque John Doe é muito esperto e contratou **você** para implementar um algoritmo que calcula a resposta correta para este problema.

Neste laboratório você deve implementar o algoritmo que desenvolveu para o John (obviamente, sendo um profissional ético e comprometido com o bem estar da sociedade, você jamais iria fazer este trabalho para a Ódmin).

Claro que a sua responsabilidade era tremenda e, a princípio, você se sentiu um pouco inseguro: "será que eu sei resolver este problema?" Assim, estando do lado bom da *força*, você foi visitar o Mestre Yoda para pedir um auxílio. Sempre eningmático, ele lhe respondeu assim (note que as inversões da fala do Mestre foram desfeitas aqui para facilitar a redação do enunciado):

> Meu caro Jedi,
> Suponha que você soubesse responder com um simples SIM ou NÃO à seguinte pergunta: "É possível construir as duas torres T_1 e T_2 quando o número máximo de andares nestes prédios é limitado a L_1 e L_2, respectivamente?" Você saberia agora como resolver o problema de John Doe?

Com esta resposta ao mesmo tempo intrigante e sábia, você se pôs a trabalhar.

Entrada

O arquivo de entrada consiste em 2 linhas. A primeira contém um inteiro positivo N ($2 \leq N \leq 100$) que corresponde ao número de empresas. A segunda linha é composta por N inteiros positivos ($\sum_{i=1}^{N} N_i \leq 400$) que correspondem ao número de andares contratados por cada empresa (lembre-se que cada empresa ocupa **exatamente uma** das torres).

Saída

A saída do programa é um inteiro positivo que corresponde a menor altura possível da maior torre.

Exemplo 1

3 */* entrada */*

1 4 3 */* entrada */*

4 */* saída */*

Exemplo 2

4 */* entrada */*

1 3 2 3 */* entrada */*

5 */* saída */*

Exemplo 3

5 */* entrada */*

1 2 20 3 1 */* entrada */*

20 */* saída */*

Especificações

O programa deve ser implementado em C++.

No Susy serão fornecidos 2 arquivos: *main.cpp* e *solution.hpp*. O arquivo main.cpp lida com as entradas e saídas dos dados e é responsável por calcular a resposta a partir da lista de empresas em uma das torres. Por fim, a função *solution* (*solution.hpp*) tem como entrada um vetor de inteiros com N elementos contendo o número de andares contratados por cada empresa. Ela deve retornar um vetor de inteiros correspondente aos índices das empresas de apenas uma das torres (obviamente supõe-se que as outras empresas terão seus andares alocados à outra torre). A ordem em que os índices das empresas estão armazenados no vetor retornado é irrelevante.

O arquivo *main.cpp*, **não precisa e não deve ser alterado**. Você deve apenas implementar a função *solution* que está dentro do arquivo *solution.hpp*, portanto, o **único**

arquivo que deverá ser alterado e submetido no Susy será o *solution.hpp*. Podem ser criadas novas funções auxiliares (além da *solution*) no arquivo *solution.hpp* desde que elas não infrinjam nenhuma das regras. Todos os arquivos estão disponíveis na aba "Arquivos auxiliares" do Susy, incluindo o *Makefile*.

Avaliação

Haverá 20 testes abertos e 20 testes fechados. A nota do projeto **deverá ser** proporcional ao número de testes bem-sucedidos. Mais precisamente, se T é o número de testes bem-sucedidos do seu programa, então sua nota deverá ser próxima de $(\frac{T}{40}) \times 10 = \frac{T}{4}$. Contudo, caso a complexidade do seu algoritmo seja abaixo daquela esperada, sua nota ainda poderá sofrer um decréscimo, o qual será tanto maior quanto pior for a complexidade do seu algoritmo (NOTA: aqui estamos falando de complexidade e não de tempo de execução).

IMPORTANTE: É **proibido** usar procedimentos recursivos neste laboratório. O uso de tais procedimentos será considerado uma fraude (ver regras da disciplina a este respeito).

Prazo de submissão

O programa pode ser submetido até a seguinte data:

- 08:00h de 22 de novembro (terça-feira), portanto, em um prazo de 24 horas a contar da divulgação do enunciado.

Observações

- O número máximo de submissões é 10.
- Para a avaliação será considerada apenas a última versão do programa submetido.
- Para submissão no Susy utilize o número do seu RA em **Usuário** e a senha utilizada na DAC em **Senha**.