Estrutura de Dados e Algoritmos
Projeto 01: k-ésimo menor
Ciência da Computação

Prof. Daniel Saad Nogueira Nunes

1 Contextualização

Astrogildo espalhou várias cartas distintas em diversos decks e te propôs um desafio. Se ele escolher k decks a serem unidos, você conseguiria achar a l-ésima menor carta nesta união de decks?

2 Especificação

O projeto deverá ser executado através da linguagem `C`.

A entrada deve ser lida da entrada padrão (`stdin`), enquanto a saída deverá ser impressa na saída padrão (`stdout`).

O programa deverá obedecer rigorosamente o formato de saída especificada, pois parte da correção será automatizada.

2.1 Entrada

A primeira linha da entrada possui um inteiro k ($1 \leq k \leq 10^5$), indicando o número de decks. As próximas k linhas descrevem, cada uma, um deck d_i. Cada linha começa com um inteiro n_i, indicando o tamanho do deck, e que é seguido por n_i inteiros, separados por um espaço, que descrevem o valor das cartas. O valor de cada carta está limitada ao intervalo $[1, 10^9]$. É garantido que não existem cartas repetidas entre os decks e que $\sum_{i=1}^{k} n_i \leq 10^5$.

Após a descrição dos decks, existe uma linha com um inteiro q ($1 \leq q \leq 10^5$), representando o número de perguntas a serem respondida. Cada uma das próximas q linhas possui 3 inteiros, separados por um espaço: a b ($1 \leq a \leq b \leq k$) e c ($1 \leq c \leq \sum_{i=a}^{b} n_i$), os quais indicam que deseja-se saber qual a c-ésimo menor carta considerando os decks d_i com $i \in [a, b]$.

2.2 Saída

Para cada umas das q perguntas, o seu programa deverá responder, em uma linha, o valor da c-ésima menor carta.

3 Exemplos

- Entrada :

```
5
4 9 1 5 3
3 2 8 7
4 22 21 17 15
3 27 29 28
2 24 23
3
1 1 3
```

```
1 3 6
2 5 10
```

- Saída:

```
5
8
27
```

3.1 Compilação

Um arquivo `Makefile` deve ser disponibilizado para compilação do projeto.

3.2 Limites de Tempo e Memória

Para cada caso de teste, será permitido a execução do programa por apenas 1 segundo com utilização máxima de 256 MB de memória. Caso o programa leve mais tempo ou memória do que isso, será considerado que o algoritmo empregado foi ineficiente.

3.3 Documentação

Junto do(s) código(s) necessário(s) para resolver o problema, deverá ser disponibilizado um arquivo README, identificando o autor do trabalho e especificando as instruções para compilação e execução do(s) código(s).

3.4 Critérios de Correção

Fazem partes dos critérios de correção:

- Eficiência do programa.
- Utilização de estruturas de dados adequadas.
- Documentação: além do arquivo README, o código deve estar bem documentado.
- Legibilidade.

3.5 Ambiente de Correção

Os projetos serão corrigidos em uma máquina com sistema `GNU/Linux` e compilador `gcc 10.2.0`.

Trabalhos que não compilarem não serão avaliados.

4 Considerações

- Este projeto deve ser executado individualmente.
- Os trabalhos que incidirem plágio serão avaliados automaticamente com nota 0 para os envolvidos. Medidas disciplinares também serão tomadas.
- O trabalho deve ser entregue dentro de uma pasta zipada com a devida identificação do(s) aluno(s) através da sala de aula virtual da disciplina na data estipulada no ambiente.