DCC/ICEx/UFMG
Projeto e análise de algoritmos
2º semestre de 2020

Prof. Vinicius Fernandes dos Santos
Prova - Módulo Paradigmas

Instruções gerais: A clareza e concisão das respostas também é objeto de avaliação. A **complexidade** dos algoritmos fornecidos **será levada em consideração** durante a avaliação, assim, quanto mais eficiente seu algoritmo, melhor.

1. Gabriel gosta muito de números. Entretanto, os números que ele realmente gosta são os números pares. Ele gosta tanto de números pares que mesmo quando um número é par, se ele tem muitos dígitos ímpares consecutivos, Gabriel para de gostar do número. Sempre que isso acontece, ele começa a trocar dígitos ímpares do número por dígitos pares, até ficar safisfeito. Mas ele percebeu que estava trocando dígitos demais dos números com os quais lidava e pediu sua ajuda para escrever um programa que conte o menor número de dígitos que devem ser trocados para que não haja k dígitos ímpares consecutivos.

   Nesta questão você deve escrever um algoritmo que receba como parâmetros um vetor v de n posições, cada uma preenchida com um dígito no intervalo 0-9, e também um inteiro k, satisfazendo $n \geq k \geq 2$. Seu algoritmo deve retornar o menor número de dígitos que devem ser modificados de forma que não hajam k dígitos ímpares consecutivos. Determine a complexidade de seu algoritmo. Justifique por que o valor retornado está correto.

2. Mesmo em momentos difíceis, surgem novas ideias interessantes para toda a sociedade. Durante a pandemia de COVID-19, atividades online se popularizaram, como seminários e transmissões ao vivo, as chamadas "lives". Embora sejam um substituto às suas versões físicas, elas acabam tornando possível o acesso de pessoas que por motivos financeiros ou geográficos não poderiam usufruir de tais atividades de outra forma, mesmo em tempos normais.

   Um certo professor se encontra com uma dificuldade relacionada a essa disponibilidade de conteúdo: com suas férias chegando, ele quer usar suas 720 horas de férias da melhor maneira possível. Entretanto, dentre todo o conteúdo ao vivo que ele gostaria de assistir, entre lives, seminários de algoritmos, tutoriais de ukulele e vídeos de Magic: The Gathering, muitos deles têm conflitos de horário. Ainda assim, ele gostaria de assistir à maior quantidade possível deles, mas sem assistir vídeos de maneira incompleta: para cada vídeo, ele quer assisti-lo totalmente ou não simplesmente não assistir. Mais do que isso, ele quer "maratonar", ou seja, assistir uma sequência de vídeos sem intervalos entre eles. Sua tafera é ajudar o professor a selecionar quais vídeos assistir. Ele não possui restrições quanto ao horário de início do primeiro vídeo, ou do horário de término do último, desde que não haja intervalo entre vídeos consecutivos e que o número total de vídeos seja o maior possível.

   Para facilitar o trabalho, ele já eliminou algumas possibilidades. Na lista atual de vídeos que ele considera assistir, todos os vídeos possuem duração de um número inteiro de horas, sempre começando e terminando em uma hora "redonda". Além disso, em cada hora só termina exatamente um vídeo.

   Escreva um algoritmo de tempo **linear**, que receba como entrada um vetor `v[]` e um inteiro `n` e retorne um inteiro, correspondendo ao maior número de vídeos que podem ser assistidos, de acordo com as restrições acima. O vetor `v` contém n números inteiros, indexados de `v[1]` a `v[n]`, e o i-ésimo elemento contém a hora de início do i-ésimo vídeo. A hora de término do i-ésimo vídeo é exatamente i. Por exemplo, se $n = 5$ e se o vetor contém os elementos $\langle 0, 0, 1, 0, 2 \rangle$, então os vídeos têm os seguintes intervalos: $[0, 1], [0, 2], [1, 3], [0, 4], [2, 5]$. Neste caso, algumas soluções seriam, assistir os vídeos 1 e 3, assistir os vídeos 2 e 5, ou assistir o vídeo 4. Note que esta última não é uma solução ótima, pois há soluções possíveis com uma quantidade maior de vídeos. Note também que os vídeos 2 e 3 não formam uma solução válida, pois no intervalo [1,2] seria necessário assistir os dois vídeos simultaneamente.

3. Escreva um algoritmo que receba um vetor e retorne o seu terceiro maior elemento. Você deve utilizar divisão e conquista. Analise a complexidade do seu algoritmo.