
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.
- Não é permitida nenhuma comunicação com outras pessoas durante a prova.
- Não é permitido o uso nenhuma ferramenta online, incluindo buscadores como o Google, durante a prova.

1. Uma sequência binária é dita "par por blocos", se ela pode ser particionada em trechos contíguos de forma que

   (a) Cada trecho possui apenas zeros ou apenas uns;

   (b) Cada trecho tem comprimento par.

   Por exemplo a sequência `00001100` é par por blocos, uma vez que pode ser particionada em três trechos satisfazendo as propriedades (a) e (b).

   Por outro lado, `000011100` não é par por blocos. Note, entretanto, que se considerarmos a subsequência contígua contendo os primeiros 6 elementos, ela é par por blocos. Similarmente, a subsequência contígua contendo apenas os últimos 4 elementos também é par por blocos.

   Neste problema, você deve resolver o seguinte problema: dado um vetor b, contendo n dígitos binários, determine qual a maior subsequência contígua par por blocos. Dica: este problema admite soluções de tempo linear.

2. Faltam 4 semanas o final do semestre e, com isso, muitos estudos, TP's, leituras e exercícios devem ser feitos, para um bom aproveitamento. Você resolveu fazer um planejamento e dividir suas tarefas em dois grupos: aquelas que você realizará na próxima quinzena, e aquela que você realizará na quinzena seguinte. A i-ésima tarefa das n tarefas que você precisa realizar, possui duração d_i. Como as durações podem ser diferentes, a carga de trabalho das duas semanas pode ficar bem distinta, dependendo das tarefas escolhidas. Assim, você notou que uma boa estratégia seria dividir as tarefas de forma que a duração total das tarefas selecionadas para cada uma das duas quinzenas seja a mais próxima possível. Mais formalmente, você quer particionar o conjunto $\{1, \ldots, n\}$ de tarefas em dois conjuntos, Q_1 e Q_2, de forma a minimizar o valor de

$$\left| \sum_{i \in Q_1} d_i - \sum_{i \in Q_2} d_i \right|.$$

   A primeira ideia que você teve foi um algoritmo guloso, com o seguinte funcionamento:

   - Ordenar as tarefas em ordem não crescente;
   - Considerando a ordem obtida, para cada tarefa, inseri-la em Q_j se, até aquele momento,

$$\sum_{i \in Q_j} d_i < \sum_{i \in Q_{3-j}} d_i.$$

     Em caso de empate, a tarefa pode ser adicionada em qualquer um dos dois conjuntos.

   (a) Execute o algoritmo para o caso em que $n = 7$ e o vetor d possui elementos `1,11,8,5,7,3,11`. Quais os conjuntos Q_1 e Q_2 ao final da execução do algoritmo? Podemos concluir que esta partição é ótima?

   (b) Encontre uma entrada tal que a solução fornecida pelo algoritmo não é ótima. Mostre qual seria uma solução ótima para esta entrada.

3. Já que a estratégia gulosa não foi suficiente para resolver o problema anterior com sucesso, façamos outra tentativa.

   Seja $S(i, j, k)$ uma variável booleana que é verdadeira se e somente se existe uma partição das i primeiras tarefas em dois conjuntos de forma que a soma dos elementos do primeiro conjunto é exatamente j e a soma dos elementos do segundo conjunto é exatamente k.

   (a) Mostre como resolver o problema corretamente, caso saibamos calcular $S(i, j, k)$ para quaisquer triplas (i, j, k) que desejarmos. Ou seja, mostre como determinar o menor valor de

   $$\left| \sum_{i \in Q_1} d_i - \sum_{i \in Q_2} d_i \right|$$

   que pode ser obtido dentre todas as possíveis partições (ou seja, determine o valor obtido em uma partição ótima).

   Dica: note que se $S(i, j, k) = \texttt{Verdadeiro}$, então $\sum_{r=1}^{i} d_r = j + k$, mas não necessariamente o contrário.

   (b) Escreva uma relação de recorrência para calcular $S(i, j, k)$ corretamente. Não se esqueça do caso base.

   (c) Em qual complexidade pode ser computada a solução do problema, usando a relação de recorrência fornecida no item (b), e a estratégia do item (a)?

4. **(Extra)** Forneça um algoritmo $\mathcal{O}(n \times \sum_{i=1}^{n} d_i)$ para o problema tratado nas duas questões anteriores.