1) Analisando a tabela, partindo do bloco 4 temos :
   a) 4 ->86 ->76 ->91->38 ->13->11->10->12->Eof
   b) São 9 blocos de 2048 bytes, sendo assim temos 9x2048=18432 bytes;
   c) 9 blocos, sendo eles os 9 blocos da sequência mencionada na letra A

2) São 3 os critérios de análise que utilizamos para comparar as técnicas, sendo eles :

- **Rapidez :** Que seria a rapidez do acesso aos dados do arquivo, tanto no acesso sequencial como no acesso aleatório;
- **Robustez :** Que seria como a estratégia se comporta frente a erros, como por exemplo, dados corrompidos ou blocos de disco defeituosos;
- **Flexibilidade :** Que seria a flexibilidade oferecida por cada estratégia para a criação, modificação e exclusão de arquivos.

Realizando uma análise bem por cima, apenas para fins de exemplificação, temos que:

- **Estratégia contígua** : Possui um acesso rápido, seja sequencial ou aleatório, uma boa robustez frente a falhas de disco, porém possui pouca flexibilidade.
- **Alocação Encadeada(Simples) :** Seu acesso sequencial é rápido, o seu acesso aleatório é bastante prejudicado pela estratégia em si mas, esta técnica é bastante flexível.
- **Indexada(Multinível) :** Bastante rápida, tanto para acessos sequenciais como para acessos aleatórios, possui boa robustez, possui uma flexibilidade similar a alocação encadeada.

3) Falando um pouco da **encadeada,** temos em cada bloco um campo contendo informações para o próximo bloco, um ponteiro, a partir disso podemos observar algumas vantagens e desvantagens sobre essa técnica, sendo elas:

Desvantagens : Justamente por acessar o bloco seguinte a partir do bloco atual, caso haja algum problema que termine por corromper os dados do bloco atual, o acesso para os blocos seguintes não ocorre.

Vantagem: Como vantagem temos a sua alta flexibilidade, já que não há necessidade de se definir o tamanho máximo do arquivo, visto que qualquer bloco livre pode ser usado para armazenar ou estender o tamanho do arquivo, eliminando a necessidade de fragmentação externa.

Entrando agora na **indexada,** nela temos uma estrutura em forma de vetor que contém um índice de blocos, ou seja, cada entrada desse índice corresponde a um bloco do arquivo e aponta para a posição desse bloco no disco. Essa estrutura é denominada de index node ou i-node.

Desvantagens: Os i-nodes têm tamanho fixo, sendo assim, o número de entradas no índice de blocos de um arquivo é limitado, com isso temos uma limitação quanto ao tamanho do arquivo que pode ser armazenado.

Vantagens: Já que as informações para o bloco seguinte estão armazenadas no i-node e não no bloco em si, defeitos que podem ocorrer em um bloco de dados não afetam os demais blocos, diferentemente da **encadeada.** Além disso possui uma alta flexibilidade já que os arquivos podem ser criados em qualquer local do disco, sem risco de fragmentação externa, tendo apenas uma limitação quanto ao tamanho devido ao i-node.

4) Temos o bloco onde se inicia b0 = 7 , o byte que estamos procurando i = 8000 e o tamanho dos blocos lógicos B = 4096, para acharmos o número do bloco(**bi**) e a posição em que o byte i se encontra (**oi**) realizaremos duas operações :

bi = b0 + (i ÷ B) -> onde i ÷ B é uma divisão do tipo inteira.
oi = i % B -> % é o símbolo na programação que representa uma divisão por módulo(do tipo que retorna o resto)

Calculando, temos :
bi = 7 + (8000 ÷ 4096) = 7 + 1 = **8**
oi = 8000 % 4096 = **3904**
return(8;3904)

Logo, vemos que o byte i se encontra no bloco de número 8, na posição 3904