Algoritmos

Força Bruta

Lista de exercícios

1. Escreva um texto explicando o processo de construção de um algoritmo através da técnica de força bruta.

2. Escreva e explique o algoritmo global de força bruta. Detalhe para que serve cada uma das partes/funções do algoritmo.

3. Escreva um algoritmo que receba dois números e mostre o MMC - Mínimo Múltiplo Comum - entre eles. Use o algoritmo global de força bruta descrito na questão anterior. Implemente em **Linguagem C/C++** o algoritmo definido, realize um conjunto de testes e faça um gráfico com o tempo de execução dos testes. Verifique e descreva se os testes confirmam a análise de complexidade que você fez.

4. Descreva e implemente um algoritmo que leia dois números inteiros a e b e mostre se eles são primos entre sí. A implementação deve ser feita em **Linguagem C/C++**. Descreva a complexidade do seu algoritmo usando a notação O.

5. Descreva, de forma textual, uma proposta inicial para a solução do problema **Festa de formatura**, em anexo. Determine, usando a técnica de força bruta, qual o **espaço de solução**. Explique como você chegou a essa solução.

   DICA: De quantas formas diferentes eu posso colocar os alunos nas cadeiras?

6. Descreva um algoritmo de força bruta para determinar se existe um par de números em um *array* cuja soma seja S.

   (a) Escreva um breve texto explicando seu algoritmo, com exemplo.

   (b) Escreva um programa em **Linguagem C** que leia um *array* de N elementos ($1 \leq N \leq 10^6$) e informe se existe um par de números cuja soma seja S.

   (c) Escreva um conjunto de casos de teste e gere uma tabela com o tempo de execução em função do tamanho da entrada.

   (d) Faça um gráfico e verifique se a taxa de crescimento corresponde ao que você descreveu no algoritmo de força bruta.

   Exemplo: Para $S = 15$, no *array* { 1 4 9 13 18 20 32 45 } não existe um par de números no *array* cuja soma seja 15. Já no *array* { 10 20 9 1 63 6 5 12 } existem dois pares: $9 + 6 = 15$ e $10 + 5 = 15$.

7. Escreva um algoritmo de força bruta que leia um *array* A de n ($3 \leq n \leq 10^5$) números inteiros a_i ($-10^8 \leq x \leq 10^8$) e mostre se existem 3 números, a, b e c, que pertencem ao *array* e formam os lados de um triângulo retângulo. Para ser retângulo, os lados devem possuir a seguinte relação:

   $$a^2 = b^2 + c^2$$

   onde a é o maior lado.

   (a) Descreva a complexidade usando a notação *big-Oh*.

   (b) Implemente, em **linguagem C**, o algoritmo e realize testes para verificar se o tempo de execução se comporta de acordo com a complexidade identificada.

   (c) Faça um gráfico e explique o comportamento do tempo de execução em função do tamanho da entrada.

8. Problema **Sub-lista contígua de soma máxima**.

   Considere um *array* A de n elementos. Exemplo: $A = 10, 5, -17, 20, 50, -1, 3, -30, 10$

   O problema consiste em encontrar a maior subsequencia cuja soma seja a maior possível. No caso do vetor A a maior soma é 72, que é a soma dos elementos entre os índices 3 e 6 : $20 + 50 + -1 + 3$.

10	5	-17	20	50	-1	3	-30	10

(a) Determine, usando a técnica de força bruta, o desempenho do algoritmo para encontrar a maior subsequência.

Dicas:

- Quais são todas as possíveis sub-listas de A?
- Não esqueça de incluir o próprio vetor A nas possíveis listas.

(b) Escreva uma função, em *Linguagem C*, para encontrar a maior sub-lista de um vetor A de n elementos ($-10^4 \leq a_i \leq 10^4$ e $1 \leq n \leq 10^5$). Faça um programa para testar a função. Considere o exemplo os exemplos de entrada e saída a seguir:

Exemplo de entrada 1	**Exemplo de saída 1**
`9` `10 5 -17 20 50 -1 3 -30 1`	`72`

Escreva em forma de comentário no código do seu program a justificativa para a criação de todas as variáveis e seu respectivo tipo.

(c) Faça um gráfico e verifique se a taxa de crescimento corresponde ao que você descreveu no algoritmo de força bruta. Considere os seguintes tamanho: 10, 50, 100, 500 1000, 5000, 10000 e 50000.

9. Um *array* A de n elementos é considerado em ordem não decrescente se para todo i e j, onde $0 \leq i < j < n$, então $A_i \leq A_j$.

Para este problema considere um *array* A de n elementos, onde $0 \leq n \leq 10^5$ e $-10^9 \leq A_i \leq 10^9$.

Importante: Baixe o arquivo *zip* em anexo com um programa para teste das funções de ordenação. Instruções básicas estão no arquivo `README.txt`. Você deve implementar as funções no arquivo `ordenacao.c`. Estude o código fonte e faça pequenos testes antes de começar a implementar.

As tarefas são:

(a) Escreva uma função que receba um *array* A e determine se A está em ordem não decrescente. Esta função será usada como função de teste para verificar se os algoritmos de ordenação estão implementados corretamente.

Escrever os testes antes de implementar é uma boa estratégia para construir programas de computador com qualidade.

(b) **Ordenação por seleção**: Implemente em Linguagem C uma função que receba um *array* de n elementos e ordene os elementos usando o método de ordenação por seleção. Determine o desempenho da função usando a notação O (*(*big-Oh)).

(c) **Ordenação por inserção**: Implemente em Linguagem C uma função que receba um *array* de n elementos e ordene os elementos usando o método de ordenação por inserção. Determine o desempenho da função usando a notação O (*(*big-Oh)).

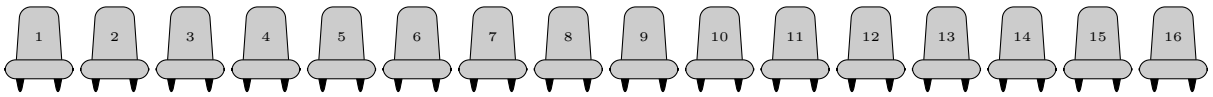
(d) Crie casos de testes com *arrays* de tamanho 10, 100, 1000, 10000 e 100000. Faça mais de um caso de teste para cada tamanho. Construa uma tabela com os tempos de execução dos casos de teste para cada um dos métodos de ordenação implementados.

(e) Construa um gráfico e verifique se a função se comporta de acordo com o desempenho encontrado. Escreva um texto fazendo uma análise do gráfico.

Festa de formatura

Ao final do curso, os alunos resolveram fazer uma grande festa de formatura e, cedo, começaram a planejar os detalhes para que tudo ocorresse bem. Durante o planejamento, vários problemas tiveram que ser resolvidos. Um problema em específico estava causando conflitos: onde cada formando iria se sentar durante os discursos dos professores. Este problema ocorreu porque alguns formandos definitivamente queriam se sentar ao lado de outros formandos, pois possuem mais afinidades. O motivo é que fiquem juntos durante a maior parte da cerimônia e tenham muitas fotos em comum.

Como eles estavam perdidos e não chegavam a um acordo, você foi contratado para determinar a posição de cada um dos formandos. Todos os formandos ficam sentados na fileira da frente do auditório, um ao lado do outro, como na figura a seguir:



Entrada

A primeira linha da entrada é composta de um número inteiro N ($2 \leq N \leq 16$), que indica a quantidade de formandos. Cada formando recebe um número entre 1 e N, inclusive. A segunda linha contém um inteiro R ($0 \leq R \leq N$), que indica a quantidade de restrições. As próximas R linhas contêm dois inteiro A e B ($0 \leq A, B \leq N$), indicando que o formando A quer sentar-se ao lado do formando B.

Saída

Seu programa deve imprimir uma única linha indicando a ordem dos alunos nas poltronas do auditório. Se houver mais de uma configuração possível, mostre qualquer uma delas. Caso não seja possível, o programa deve mostrar -1.

Exemplo de entrada 1	**Exemplo de saída 1**
5 3 1 2 3 4 1 4	2 1 4 3

Exemplo de entrada 2	**Exemplo de saída 2**
6 5 1 2 1 5 2 3 4 3 2 3	-1