Exercicios do capitulo:
resolva as equacoes abaixo:
a)1024
b)10
c)4,08
d)5
e)4

GRAFICO VAI NO FIM DO PDF, MAS JA TEM UM NO PDF DO TT04, CASO EU NAO MANDE NESSE

Contagem de operacoes:
3)realiza 2n subtracoes no pior caso e n comparacoes no melhor caso, 2 pra cada par e uma pra cada impar. Caso medio seria 3n/2
4)n-3 subtracoes
5)log(n) + 1 subtracoes
/*
Esse problema do caixeiro viajante eh aquele que faz com dijkstra ne?
https://www.aedb.br/seget/arquivos/artigos09/224_224_224_Artigo_Seget.pdf
*/

Otimizacao do compilador:
O primeiro codigo eh mais clean, mais legivel, e com certeza seria a opcao escolhida (pelo menos para arrays grandes) dentro de um ambiente de trabalho, ja o seugndo codigo, em compensacao, eh mais eficiente, pois nao realiza comparacoes e somas extras, apenas atribui o valor a posicao do array, mas ai ja entra uma questao que vai alem da otimizacao espaco-tempo que geralmente eh tratada na computacao, Ate que ponto vale a pena otimizar um algoritmo? nesse caso, eh preferivel uma sintaxe limpa em troca dep uma performance microscopicamente menor.

Para as aulas de AEDs2, a menos que dito o contrario, consideramos o pior caso
nao entendi o que significa E/S.

Funcao da complexidade de tempo:
mede o tempo (numero de execucoes da operacao relevante) de execucao do algoritmo para um problema de tamanho n.

Funcao de complexidade de espaco:
mede a quantidade de memoria necesaria para executar um algoritmo de tamanho n.

1) complexidade vai ser O(n), vai ter que dar uma volta no array
2) procurar um por um, pois a complexidade vai ser O(n), agora se for ordenar, vai ter que dar 2 voltas no array e a complexidade ja aumenta.

notacoes:
11) nao vou escrever uma por uma aqui, mas a gente tem que olhar sempre o maior termo do polinomio.

O mesmo vale para as outras notacoes:

O(1) | O(log(n) | O(n) | O(n*log(n)) | O(n^2) | O(n^3) | O(n^5) | O(n^20)
f(n) = log(n) X
f(n) = n * log(n) X
f(n) = 5n + 1 X
f(n) = 7n^5 - 3n^2 X
f(n) = 99n^3 - 1000n^2 X
f(n) = n^5 - 99999n^4 X