Relatório Projeto 1 AED 2023-2024

Nome: Miguel Curto Castela
Nº Estudante: 2022212972
PL (inscrição): PL8

Tabela para as 3 soluções*

solução/nº de casos	20000	40000	60000	80000	100000
Algoritmo A: Solução exaustiva	4907054 µs	19507049 µs	43759709 µs	77789038 µs	121594995 µs
Algoritmo B: Solução c/ ordenamento	1256 µs	2643 µs	4116 µs	5668 µs	7077 µs
Algoritmo C: Solução elaborada	997 µs	2083 µs	2988 µs	3991 µs	4993 µs

Gráfico para a solução A

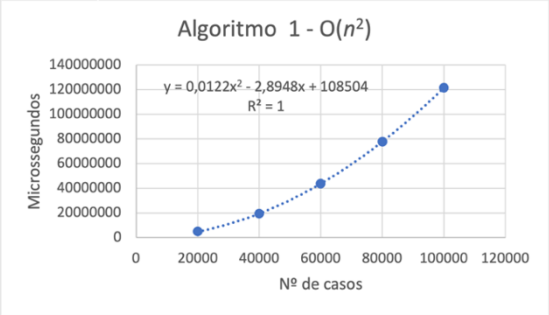


Gráfico para a solução B

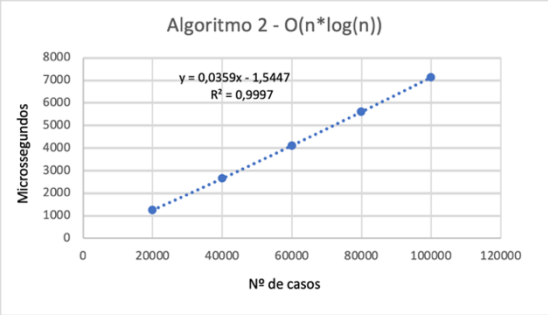
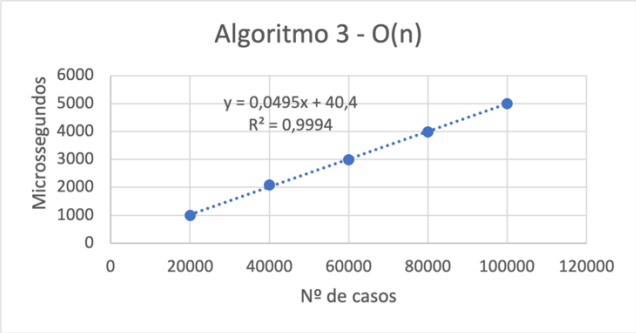


Gráfico para a solução C



Análise dos resultados tendo em conta as regressões obtidas e como estas se comparam com as complexidades teóricas:

A complexidade teórica do 1º algoritmo é O(n^2) uma vez que testo todas as combinações de pares possíveis. Para isto, é usado um algoritmo "brute force" com dois *for loops*, sendo que para cada iteração do primeiro ciclo que percorre os N elementos, o segundo também os percorre. A complexidade teórica do 2º algoritmo é O(nlog(n)), por usar o algoritmo *std::sort()* do C++, que tem esta complexidade. A pesquisa com 2 ponteiros a partir do *while* tem complexidade O(n), pois no pior dos casos percorre os N elementos do *Array.* Assim, a complexidade do algoritmo é igual à do *sort,* O(nlog(n)). A complexidade teórica do 3º algoritmo é O(n), pois cada elemento já visto vai sendo guardado num *Vector* (no índice correspondente ao valor), sendo apenas necessário verificar para um elemento i do *Array* se k-i (complemento) já foi visto e adicionado ao *Vector* (a pesquisa, inserção e criação do vetor têm complexidade O (1)). Assim, o *Array* é apenas percorrido uma vez e no máximo para N elementos. Avaliando o desempenho dos 3 algoritmos para amostras diferentes, obtemos as regressões $R^2 \approx 1.000$ para o 1º algoritmo, $R^2 \approx 0.9997$ para o 2º e $R^2 \approx 0.9994$ para o 3º. Todos estes valores indicam que as complexidades empíricas estão de acordo com as teóricas.

*De notar que nas três soluções defini o k como sendo 2*sz+1, forçando os algoritmos a correr até ao final e retornando False. Fiz isto para ter uma base mais sólida de tempos a registar. Para testar o normal funcionamento dos algoritmos para o problema em questão atribui-se um valor random ao k (k = rand() %(sz *2))

Os 3 algoritmos

```
#include <iostream>.
#include <vector>
#include <cstdlib>
#include <ctime>
#include <algorithm>
#include <chrono>

using namespace std;

//bruteforce
bool solucao1(vector<int> array, int k) {
    for (int i = 0; i < array.size(); ++i) {
        for (int j = 0; j < array.size(); ++j) {
            if (array[i] != array[j] && array[i] + array[j] == k) {
                return true;
            }
        }
    }
    return false;
}
//2 vector solution
bool solucao2(vector<int> array_ordenado, int k) {
    sort(array_ordenado.begin(), array_ordenado.end());
    int left = 0;
    int right = array_ordenado.size() - 1;
    while (left < right) {
        if (array_ordenado[left] + array_ordenado[right] < k) {
            left++;
        } else if (array_ordenado[left] + array_ordenado[right] > k)
{
            right--;
        } else {
            if (array_ordenado[left] != array_ordenado[right]) {
                return true; //isto porque o k nao pode ser formado
por 2 valores iguais(k/2+k/2)
            }
            return false;
        }
    }
    return false;
}
//memoization
bool solucao3(vector<int> array, int k) {
    vector<bool> visto(k + 1, false);
    for (int i = 0; i < array.size(); ++i) {
        int complemento = k - array[i];
        if (array[i] * 2 == k || complemento < 0) {
            continue; //isto porque o k nao pode ser formado por 2
valores iguais(k/2+k/2)
        }
        if (visto[complemento]) {
            return true;
        }
        visto[array[i]] = true;
    }
    return false;
}
```

Teste para diferentes samples/registo de tempos dos algoritmos

```
int main() {
    int numm = 0;
    srand(time(0));
    int samples = 10;
    vector<int> inp = {20000, 40000, 60000, 80000,
100000};
    for (int sz : inp) {
        auto start =
std::chrono::high_resolution_clock::now();
        auto end = std::chrono::high_resolution_clock::now();
        auto timeTaken =
std::chrono::duration_cast<std::chrono::microseconds>(e
nd - start);

        cout << "teste " << numm << endl;
        numm +=1;
        int tempo1 = 0;
        int tempo2 = 0;
        int tempo3 = 0;
        for (int i = 0; i < samples; ++i) {
            int k = sz * 2;
            vector<int> array(sz);
            for (int s = 0; s < sz; ++s) {
                array[s] = rand() % sz;
            }
            start = std::chrono::high_resolution_clock::now();
            solucao1(array, k);
            end = std::chrono::high_resolution_clock::now();
            timeTaken =
std::chrono::duration_cast<std::chrono::microseconds>(e
nd - start);
            tempo1 += timeTaken.count();
        }
        for (int i = 0; i < samples; ++i) {
            int k = sz * 2;
            vector<int> array(sz);
            for (int s = 0; s < sz; ++s) {
                array[s] = rand() % sz;
            }
            start = std::chrono::high_resolution_clock::now();
            solucao2(array, k);
            end = std::chrono::high_resolution_clock::now();
            timeTaken =
std::chrono::duration_cast<std::chrono::microseconds>(e
nd - start);
            tempo2 += timeTaken.count();
        }
        for (int i = 0; i < samples; ++i) {
            int k = sz * 2;
            vector<int> array(sz);
            for (int s = 0; s < sz; ++s) {
                array[s] = rand() % sz;
            }
            start = std::chrono::high_resolution_clock::now();
            solucao3(array, k);
            end = std::chrono::high_resolution_clock::now();
            timeTaken =
std::chrono::duration_cast<std::chrono::microseconds>(e
nd - start);
            tempo3 += timeTaken.count();
        }
        cout << "Para " << sz << " o tempo médio da
solução 1 é: " << tempo1 / samples << endl;
        cout << "Para " << sz << " o tempo médio da
solução 2 é: " << tempo2 / samples << endl;
        cout << "Para " << sz << " o tempo médio da
solução 3 é: " << tempo3 / samples << endl;

    }
    return 0;
}
```