**Assigment 1**

**-problem2-**

Student: Wojcicki Andrei-Cristian

Grupa: CR 1.3B

Specializare: CR

**Enunt:**

Se da o secventa de 2n numere reale ca date de intrare. Sa se faca un algoritm care partitioneaza numerele in n perechi cu proprietatea ca partitia minimizeaza maximul sumei perechii. De exemplu, se dau numerele (1,6,3,7), partitile posibile sunt ( (1,6) (3,7) ), ( (1,3) (6,7) ), ( (1,7) (6, 3) ). Suma perechilor pentru aceste partiti sunt: (7, 10) (4, 13) si (8, 9). Astfel, a 3-a partitie contine numarul 9 care este maximul sumei si este minimul dintre cele 3 partiti.

**Pseudocod:**

Partition (vector,begin,end)

1. pivot=vector[begin]

2. iterator1=begin-1

3. iterator2=end+1

4. while(TRUE)

5. repeat

6. iterator2=iterator2-1

7. until vector[iterator2] <= pivot

8. repeat

9. iterator1=iterator1+1

10. until vector[iterator]>=pivot

11. if iterator1<iterator2

12. vector[iterator1] vector[iterator2]

13. else

14. return iterator2

QUICK\_SORT(vector,begin,end)

1. if begin<end

2. pivot=PARTITION(vector,begin,end)

3. QUICK\_SORT(vector,begin,pivot)

4. QUICK\_SORT(vector,pivot+1,end)

MAIN

1. While citeste number

2. vector[counter]=number

3. counter=counter+1

4. QUICK\_SORT(vector,0,counter-1)

5. for iterator=0, iterator<counter/2

6. if maxim<vector[iterator]+vector[counter-1-iterator]

7. maxim=vector[iterator]+vector[counter-1-iterator]

GENERARE

1. for iterator=0, iterator<100000

2. random\_number = rand () % 1000000 // random\_number primeste un numar la intamplare din intervalul [0,1000000]

3. afiseaza random\_number

**Descrierea programului:**

**Ex:**

Pentru input-ul (1,6,3,7) => (1,6) (3,7) => suma (7,10)

(1,3) (6,7) => suma (4,13)

(1,7) (6,3) => suma (8,9)

* 9 reprezinta suma maxima care este cea mai mica dintre sumele celorlalte partiti.

Algortimul propus de mine, ordoneaza numerele crescator si apoi face suma dintre primul numar si ultimul, dupa al 2-lea numar si penultimul, etc. Suma cea mai mare, va reprezenta suma dorita, adica maxima dintre parechi care este minima dintre partiti. De exemplu:

1. Pentru input-ul (1,6,3,7) => (1,3,6,7) => 1 + 7 = 8

3 + 6 = 9

8 < 9 => 9 reprezinta suma ceruta.

2. Pentru input-ul (23, 30, 1, 39, 22, 40)

Partitile posibile sunt:

(23,30) (1,39) (22,40) => (53,40,62)

(23,30) (1,22) (39,40) => (53,23,79)

(23,30) (1,40) (39,22) => (53,41,61)

(23,1) (30,39) (22,40) => (24,69,62)

(23,1) (30,22) (39,40) => (24,52,79)

(23,1) (30,40) (39,22) => (24,70,61)

(23,39) (30,1) (22,40) => (62,31,62)

(23,39) (30,22) (1,40) => (62,52,41)

(23,39) (30,40) (1,22) => (62,70,23)

(23,22) (30,1) (39,40) => (55,31,79)

(23,22) (30,39) (1,40) => (55,79,41)

(23,22) (30,40) (1,39) => (55,70,40)

(23,40) (30,1) (39,22) => (63,31,61)

(23,40) (30,39) (1,22) => (63,69,23)

(23,40) (30,22) (1,39) => (63,52,40)

* 61 este suma dorita.

Pentru algoritmul propus de mine:

Pentru input-ul (23, 30, 1, 39, 22, 40) => (1, 22, 23, 30, 39 ,40)

1 + 40 = 41

22 + 39 = 61

23 + 30 = 53

Rezulta maximul este 61, iar partita corespunzatoare este ( (1,40) (22,39) (23, 30) )

Programul functioneaza pe un set de 100.000 date generat automat cu numere random cuprinse in intervalul [1,1000000].

Complexitatea algoritmului :

Best-case scenario : O(nlogn)

Avarage-case scenario: O(nlogn)

Worst-case scenario: O(n^2) cand numerele sunt deja ordonate.

T1 reprezinta testul 1 care are 100.000 de numere.

T2 reprezinta testul 2 care are 90.000 de numere.

T3 reprezinta testul 3 care are 80.000 de numere.

T4 reprezinta testul 4 care are 70.000 de numere.

T5 reprezinta testul 5 care are 60.000 de numere.

T6 reprezinta testul 6 care are 50.000 de numere.

T7 reprezinta testul 7 care are 40.000 de numere.

T8 reprezinta testul 8 care are 30.000 de numere.

T9 reprezinta testul 9 care are 20.000 de numere.

T10 reprezinta testul 10 care are 10.000 de numere.