Esercitazione 2

Codice:

#include <iostream>

using namespace std;

#define L 4

#include <algorithm>

void sum( int A []){

//inizializzo la somma globale al minimo del tipo senno non funziona con tutti numeri negativi

int sumg=INT\_MIN;

int suml=0;

//somma locale nel caso misto

int sumln=0;

//somma locale nel caso di tutti negativi o tutti positivi

for (int i=0;i<L;i++){



//il for mi gira l'array

suml+=A[i];



sumln=suml;



//somma locale si aggiorna e ha memoria del giro precedente del for la somma locale negativa cambia a ogni giro

//effettuo la somma locale man mano che scorro l'array in avanti

if(suml<0){

suml=0;

//inizializzazione necessaria se ho numeri positivi nell'array

}

cout<<"la somma locale mista è "<<suml<<endl;

cout<<"la somma locale negativa o positiva è "<<sumln<<endl;

if(sumg<sumln ){

// se la somma locale negativa è maggiore della somma globale la nuova somma globale è la locale negativa

sumg=sumln;

}

cout<<"la somma globale è "<<sumg<<endl;

}

}

int main(){

int A[L];

int B[L];

int C[L];

// first case test

A[0]=-1;

A[1]=-3;

A[2]=4;

A[3]=2;

//second case test

B[0]=-1;

B[1]=2;

B[2]=-5;

B[3]=7;

//third case test

C[0]=-1;

C[1]=-2;

C[2]=-5;

C[3]=-7;

cout<<"start algorithm for array A"<<endl;

sum(A);

cout<<"start algorithm for array B"<< endl;

sum(B);

cout<<"start algorithm for array C"<< endl;

sum(C);

return 0;

}

Spiegazione: L’algoritmo cerca il sottoarray in cui la somma tra i numeri adiacenti è massima, per effettuare la risoluzione l’algoritmo vengono impiegate 3 variabili una variabile somma locale, una variabile somma locale negativa e una variabile somma globale. La variabile somma locale viene inizializzata come somma di se stessa più gli elementi dell’array, mentre la somma locale negativa viene inizializzata ogni volta come uguale alla somma locale. Ora se la somma locale è minore di zero la reinizializzo a 0 perché significa che ci sono numeri negativi nell’array. La somma locale negativa non viene inizializzata ma viene paragonata alla somma globale che è inizializzata al valore più piccolo possibile ad ogni giro la somma che tiene memoria del giro precedente è la positiva la negativa non ha memoria. Se l’array è composto di soli negativi i subarray massimo è quello composto da un singolo elemento. Nel caso in cui l’array è composto da soli positivi il sotto array con la somma massima è la somma dell’array iniziale.

Studio della complessità: La complessità dell’algoritmo è O(n) quindi l’algoritmo è lineare in tutti i casi, poiché esegue un singolo for n volte e poiché tutte le altre operazioni vengono eseguite n-1 volte, tutte le altre operazioni sono costanti. Un punto negativo di questo algoritmo è l’impiego di 3 variabili per effettuare le operazioni necessarie. Forse con le giuste condizioni si potrebbe levare la variabile di somma locale negativa e ricorrere a due sole variabili.

Codice:

#include <iostream>

using namespace std;

void triangle(int d){

for(int i=1;i<=d;i++){



//il for permette di tornare a capo una volta finita una riga

int number=1;

//primo numero da inserire è 1

for (int j=1;j<=i;j++){



//il for scorre e crea la riga

cout<<" "<<number<<" ";

number=number\*(i-j)/j;

//equazione matematica del triangolo di tartaglia

}

cout <<endl;

}

}

int main(){

int a=3;

int b=5;

cout<<"starting triangle a"<<endl;

triangle(a);

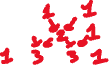
cout<<"starting triangle b"<<endl;

triangle(b);

return 0;

}

Spiegazione:L’ algoritmo punta costruire il triangolo di tartaglia usando due for per realizzare le righe e gli elementi del triangolo. Prendo come input l’indice del primo che mi permette di inizializzare il secondo e di costruire la stampa tramite il number.



Studio della complessità: ovviamente la complessità in ogni caso di esecuzione è di O(n^2) poiché ho due for innestati ogni for esegue n volte e quindi la complessità non può essere ridotta in nessun caso.

Codice:

#include <iostream>

#include "math.h"

#include <cmath>

using namespace std;

void solve(double p,double q,double r, double s,double t,double u){

if((p>=0) && (r<=20) && (q>=-20) && (s<=0) && (t<=0)){

//check del dominio dell'equazione

cout<<"l'input non viola il dominio "<<endl;

double y=1;

double x=0;

while (y!=0 && x<1){

y=((p\*exp(x))+(q\*cos(x))+(s\*tan(x))+(t\*x\*x)+u);

//equazione da risolvere in cui sostituisco i termini

float n=ceil(y\*100)/100;

//funzione per abbassare le cifre decimali che vedo della y

x+=0.0001;

//faccio avanzare x fino alla quarta cifra decimale

if(n==0){

cout<<"x è "<<x<<endl;

//caso in cui l'equazione ha soluzione nel dominio

break;

}

if(x>=1 && n!=0){

cout<<"nessuna soluzione nel dominio"<<endl;

}

}

}

else{

cout<<"dominio violato"<<endl;

}

}

int main(){

double p=0;

double q=0;

double r=0;

double s=0;

double t=-2;

double u=1;

cout<<"starting first case"<<endl;

solve(p,q,r,s,t,u);

cout<<"starting second case"<<endl;

p=0;

q=1;

r=0;

s=0;

t=-1;

u=2;

solve(p,q,r,s,t,u);

return 0;

}