Міністерство освіти і науки України
Національний університет «Львівська політехніка»
Кафедра систем штучного інтелекту

Звіт

про виконання розрахунково-графічних робіт блоку № 7

з дисципліни: «Основи програмування»

до:

ВНС Розрахунково-графічних робіт № 1-4

Практичних Робіт до блоку № 7

Виконав(ла):

Студент(ка) групи ШІ-12

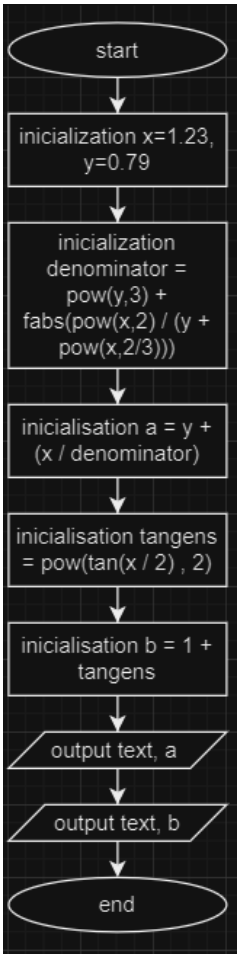
Олійник Божена Орестівна

Львів 2024

VNS practice work task №1

```cpp
#include <stdio.h>
#include <iostream>
#include <cmath>

using namespace std;

int main()
{
    float x = 1.23, y = 0.79;

    double denominator = pow(y, 3) + fabs(pow(x, 2) / (y + pow(x, 2 / 3)));

    double a = y + (x / denominator);

    double tangens = pow(tan(x / 2), 2);

    double b = 1 + tangens;

    cout << "a = " << a << endl;

    cout << "b = " << b << endl;
}
```

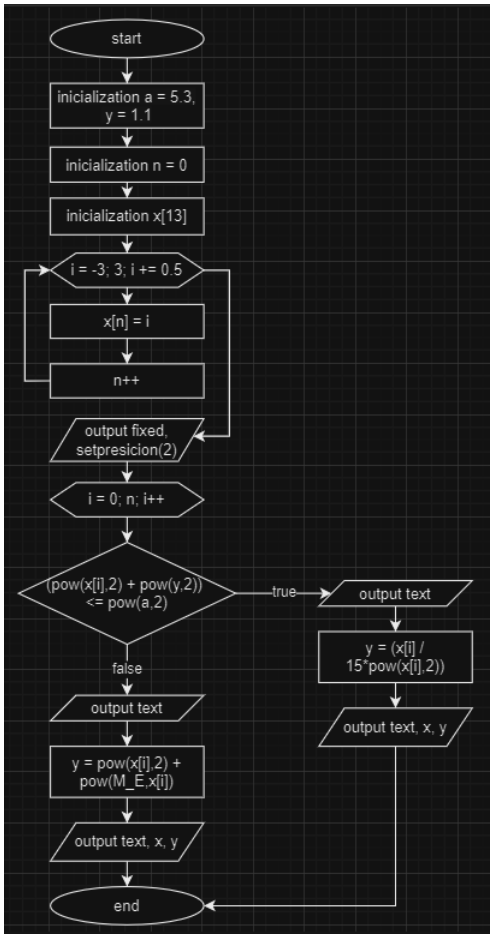
```
a = 1.70912
b = 1.49898
```



VNS practice work task №2

```cpp
#include <stdio.h>
#include <iostream>
#include <iomanip>
#include <cmath>

using namespace std;

int main()
{
    double a = 5.3, y = 1.1;
    int n = 0;
    double x[13];
    for (double i = -3; i <= 3; i += 0.5){
        x[n] = i;
        n++;
    }

    cout << fixed << setprecision(2);

    for (int i = 0; i < n; i++){
        if ((pow(x[i],2) + pow(y,2)) <= pow(a,2)){
            cout << "First case:" << endl;
            y = (x[i]/(15*pow(x[i],2)));
            cout << "x = " << x[i] << "    " << '\t' << "y = " << y << endl;
        } else {
            cout << "Second case:" << endl;
            y = pow(x[i],2) + pow(M_E,x[i]);
            cout << "x = " << x[i] << "    " << '\t' << "y = "  << y << endl;
        }
    }

}
```

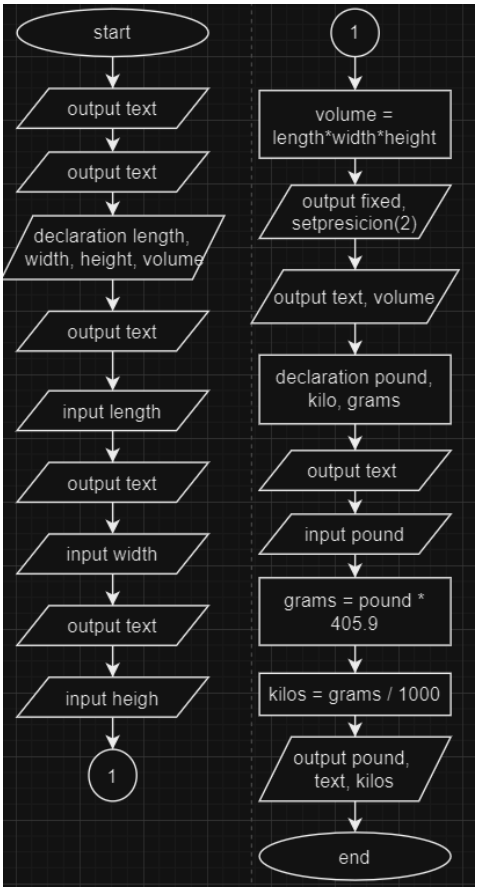
```
First case:
x = -3.00       y = -0.02
First case:
x = -2.50       y = -0.03
First case:
x = -2.00       y = -0.03
First case:
x = -1.50       y = -0.04
First case:
x = -1.00       y = -0.07
First case:
x = -0.50       y = -0.13
First case:
x = 0.00        y = nan
Second case:
x = 0.50        y = 1.90
First case:
x = 1.00        y = 0.07
First case:
x = 1.50        y = 0.04
First case:
x = 2.00        y = 0.03
First case:
x = 2.50        y = 0.03
First case:
x = 3.00        y = 0.02
```



VNS practice work task №3

```cpp
#include <iostream>
#include <iomanip>

using namespace std;

int main() {
    cout << "Обчислення об'єму паралелепіпеда." << endl;
    cout << "Введіть початкові дані:" << endl;

    double length, width, height, volume;
    cout << "Довжина (см) = ";
    cin >> length;
    cout << "Ширина(см) = ";
    cin >> width;
    cout << "Висота(см) = ";
    cin >> height;

    volume = length * width * height;

    cout << fixed << setprecision(2);
    cout << "Об'єм: " << volume << " куб.см." << endl;

    double pound, kilos, grams;
    cout << "Введіть вагу у фунтах: ";
    cin >> pound;

    grams = pound * 405.9;
    kilos = grams / 1000;

    cout << pound <<  " фунтів дорівнюють " << kilos << " кілограмам" << endl;

    return 0;
}
```

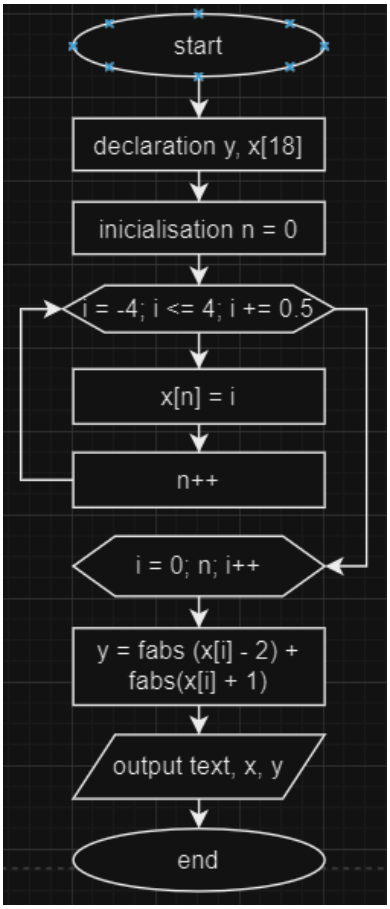
```
Обчислення об'єму паралелепіпеда.
Введіть початкові дані:
Довжина (см) = 9
Ширина(см) = 7.5
Висота(см) = 5
Об'єм: 337.50 куб.см.
Введіть вагу у фунтах: 30
30.00 фунтів дорівнюють 12.18 кілограмам
```



VNS practice work task №4

```cpp
#include <iostream>
#include <string>
#include <iomanip>
#include <cmath>
using namespace std;

int main()
{
    double y, x[18];
    int n = 0;
    for (double i = -4; i <= 4; i += 0.5)
    {
        x[n] = i;
        n++;
    }

    for (int i = 0; i < n; i++)
    {
        y = fabs(x[i] - 2) + fabs(x[i] + 1);
        cout << "x = " << x[i] << "    " << '\t' << "y = " << y << endl;
    }
}
```

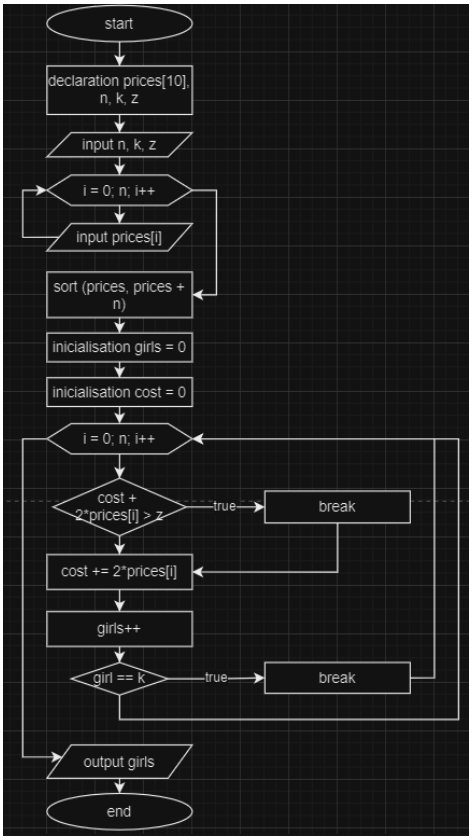
```
x = -4          y = 9
x = -3.5        y = 8
x = -3          y = 7
x = -2.5        y = 6
x = -2          y = 5
x = -1.5        y = 4
x = -1          y = 3
x = -0.5        y = 3
x = 0           y = 3
x = 0.5         y = 3
x = 1           y = 3
x = 1.5         y = 3
x = 2           y = 3
x = 2.5         y = 4
x = 3           y = 5
x = 3.5         y = 6
x = 4           y = 7
```



Algotester problem №0522

```cpp
#include <iostream>
#include <algorithm>

using namespace std;

int main() {
    int prices[10];
    int n, k, z;

    cin >> n >> k >> z;
    for (int i = 0; i < n; i++) {
        cin >> prices[i];
    }

    sort(prices, prices + n);
    int girls = 0;
    int cost = 0;

    for (int i = 0; i < n; i++) {
        if (cost + 2 * prices[i] > z) {
            break;
        }
        cost += 2 * prices[i];
        girls++;
        if (girls == k) {
            break;
        }
    }

    cout << girls << endl;

    return 0;
}
```

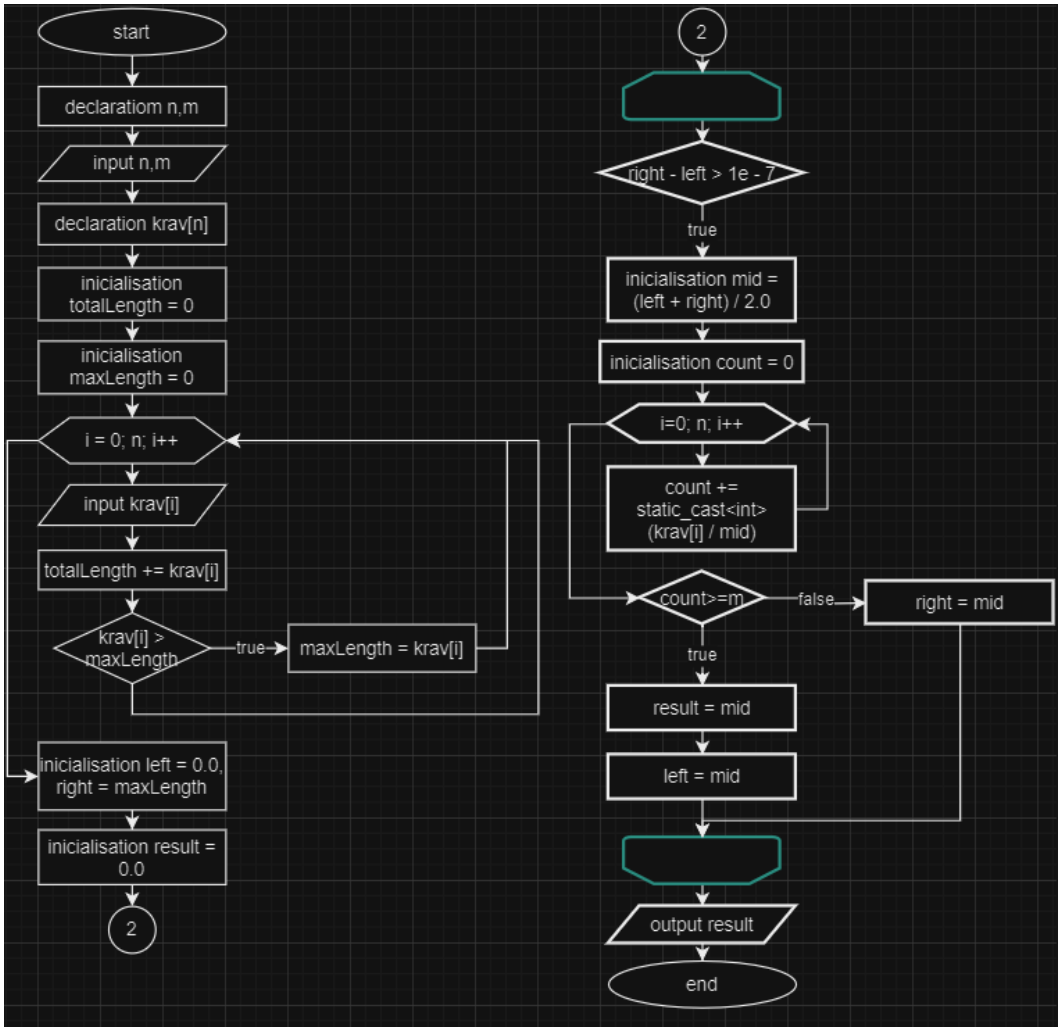
```
4 7 44
4 47 7 74
2
```



Algotester problem №0023

```cpp
#include <iostream>
#include <iomanip>

using namespace std;

int main() {
    int n, m;
    cin >> n >> m;

    int krav[n];
    int totalLength = 0;
    int maxLength = 0;

    for (int i = 0; i < n; i++) {
        cin >> krav[i];
        totalLength += krav[i];
        if (krav[i] > maxLength) {
            maxLength = krav[i];
        }
    }

    double left = 0.0, right = maxLength;
    double result = 0.0;

    while (right - left > 1e-7) {
        double mid = (left + right) / 2.0;
        int count = 0;

        for (int i = 0; i < n; i++) {
            count += static_cast<int>(krav[i] / mid);
        }

        if (count >= m) {
            result = mid;
            left = mid;
        } else {
            right = mid;
        }
    }

    cout << fixed << setprecision(7) << result << endl;

    return 0;
}
```

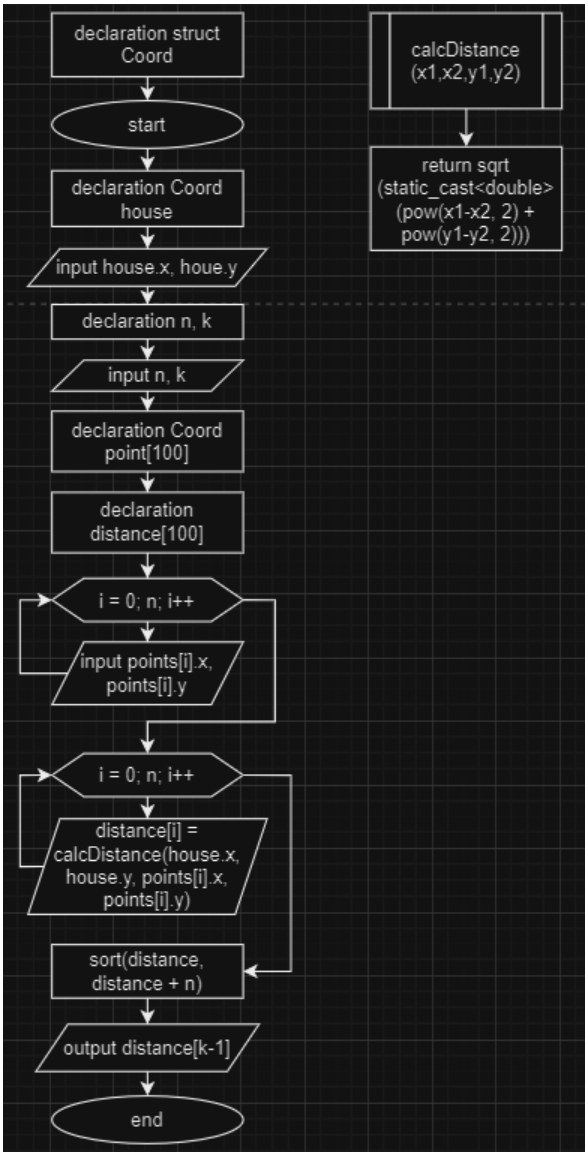
```
3 4
1 10 5
3.3333333
```



Algotester problem №0183

```cpp
#include <iostream>
#include <cmath>
#include <algorithm>
#include <iomanip>
using namespace std;

struct Coord {
    int x;
    int y;
};

double calcDistance(int x1, int y1, int x2, int y2) {
    return sqrt(static_cast<double>(pow(x1 - x2, 2) + pow(y1 - y2, 2)));
}

int main() {
    Coord house;
    cin >> house.x >> house.y;

    int n, k;
    cin >> n >> k;

    Coord points[1000];
    double distance[1000];

    for (int i = 0; i < n; i++) {
        cin >> points[i].x >> points[i].y;
    }

    for (int i = 0; i < n; i++) {
        distance[i] = calcDistance(house.x, house.y, points[i].x, points[i].y);
    }

    sort(distance, distance + n);
    cout << setprecision(13) << distance[k - 1];
    return 0;
}
```

```
5 6
4 2
3 4
8 11
6 5
17 25
2.828427124746
```

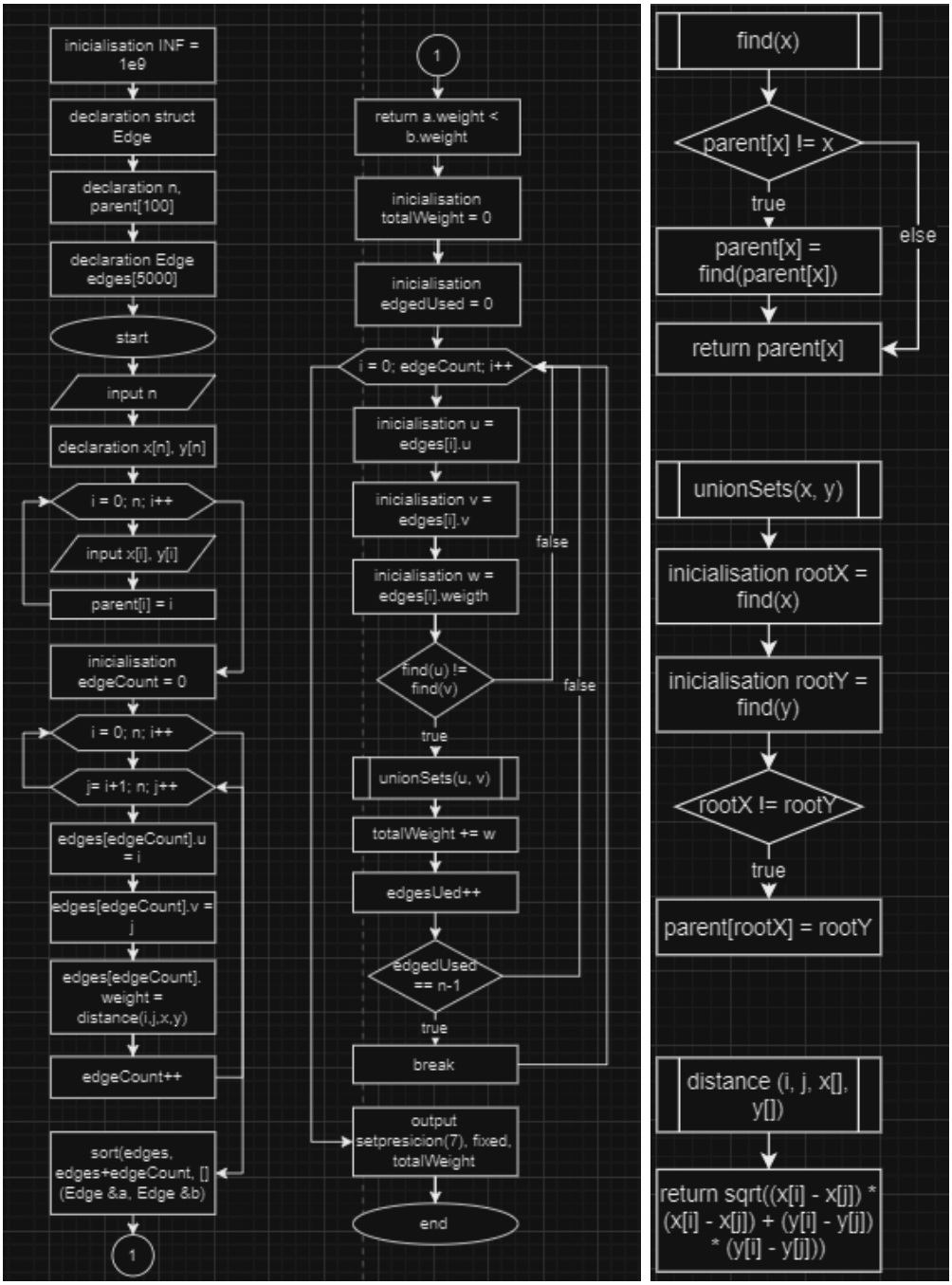


Algotester problem №0606

```cpp
#include <iostream>
#include <cmath>
#include <algorithm>
#include <cstring>

using namespace std;

const double INF = 1e9;

struct Edge {
    int u, v;
    double weight;
};

int n;
int parent[100];
Edge edges[5000];

int find(int x) {
    if (parent[x] != x)
        parent[x] = find(parent[x]);
    return parent[x];
}

void unionSets(int x, int y) {
    int rootX = find(x);
    int rootY = find(y);
    if (rootX != rootY)
        parent[rootX] = rootY;
}

double distance(int i, int j, int x[], int y[]) {
    return sqrt((x[i] - x[j]) * (x[i] - x[j]) + (y[i] - y[j]) * (y[i] - y[j]));
}

int main() {
    cin >> n;
    int x[n], y[n];

    for (int i = 0; i < n; i++) {
        cin >> x[i] >> y[i];
        parent[i] = i;
    }

```

```
    int edgeCount = 0;

    for (int i = 0; i < n; i++) {
        for (int j = i + 1; j < n; j++) {
            edges[edgeCount].u = i;
            edges[edgeCount].v = j;
            edges[edgeCount].weight = distance(i, j, x, y);
            edgeCount++;
        }
    }

    sort(edges, edges + edgeCount, [](Edge &a, Edge &b) {
        return a.weight < b.weight;
    });

    double totalWeight = 0;
    int edgesUsed = 0;

    for (int i = 0; i < edgeCount; i++) {
        int u = edges[i].u;
        int v = edges[i].v;
        double w = edges[i].weight;

        if (find(u) != find(v)) {
            unionSets(u, v);
            totalWeight += w;
            edgesUsed++;
            if (edgesUsed == n - 1) break;
        }
    }

    cout.precision(7);
    cout << fixed << totalWeight << endl;

    return 0;
}
```

```
4
0 1
1 0
-1 0
0 -1
4.2426407
```


inicialisation INF = 1e9
declaration struct Edge
declaration n, parent[100]
declaration Edge edges[5000]
start
input n
declaration x[n], y[n]
i = 0; n; i++
input x[i], y[i]
parent[i] = i
inicialisation edgeCount = 0
i = 0; n; i++
j= i+1; n; j++
edges[edgeCount].u = i
edges[edgeCount].v = j
edges[edgeCount].weight = distance(i,j,x,y)
edgeCount++
sort(edges, edges+edgeCount, [](Edge &a, Edge &b)
1
1
return a.weight < b.weight
inicialisation totalWeight = 0
inicialisation edgedUsed = 0
i = 0; edgeCount; i++
inicialisation u = edges[i].u
inicialisation v = edges[i].v
inicialisation w = edges[i].weigth
find(u) != find(v)
false
true
unionSets(u, v)
totalWeight += w
edgesUed++
edgedUsed == n-1
true
break
false
output setpresicion(7), fixed, totalWeight
end
find(x)
parent[x] != x
true
else
parent[x] = find(parent[x])
return parent[x]
unionSets(x, y)
inicialisation rootX = find(x)
inicialisation rootY = find(y)
rootX != rootY
true
parent[rootX] = rootY
distance (i, j, x[], y[])
return sqrt((x[i] - x[j]) * (x[i] - x[j]) + (y[i] - y[j]) * (y[i] - y[j]))