Estructuras Discretas

Tarea 5

Grafos / Gráficas

Oscar Andrés Rosas Hernandez

Diciembre 2018

Índice

1. Pregunta 2

Prueba que hay 11 gráficas no isomorfas con 4 vértices.

Demostración:

Vamos a llamar a los vértices $1, 2, 3, 4$, ahora vamos a iterar sobre las aristas: Supongamos que tenemos:

- Con 0 aristas solo podemos tener un grafo, en el que todos estan desconectados.
- Con 1 aristas solo podemos tener un grafo
  - Por ejemplo:$(1, 2)$, $(2, 3)$, $(3, 4)$
- Con 2 aristas solo podemos tener dos grafos
  - $(1, 2)$ y $(2, 3)$
  - $(1, 2)$ y $(3, 4)$
- Con 3 aristas solo podemos tener 3 grafos
  - $(1, 2), (2, 4)$ y $(2, 3)$
  - $(1, 2), (2, 3)$ y $(1, 3)$
  - $(1, 2), (2, 3)$ y $(3, 4)$
- Con 4 aristas solo podemos tener dos grafos
  - $(1, 2), (2, 3), (3, 4)$ y $(1, 4)$
  - $(1, 2), (2, 3), (1, 3)$ y $(2, 4)$
- Con 5 aristas solo podemos tener 1 grafo
  - $(1, 2), (2, 3), (3, 4), (1, 4)$ y $(1, 3)$
- Con 6 aristas solo podemos tener 1 grafo
  - $(1, 2), (2, 3), (3, 4), (1, 4), (1, 3)$ y $(2, 4)$

Recuerda que un grafo completo tendrá $\frac{n(n-1)}{2}$ es decir 6 aristas, por lo tanto no puede un grafo tener más aristas.

Ahora si sumamos: $1 + 1 + 2 + 3 + 2 + 1 + 1 = 11$

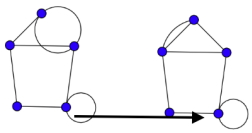
2. Pregunta 4

Muestra que las siguientes gráficas no simples no son isomorfas.

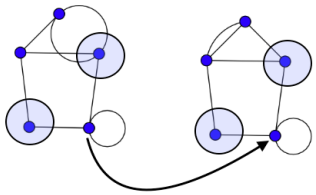
Conste que dice muestra y no prueba en si, así que esto será mas bien un bosquejo.

Primero, si es que fueran isomorfas entonces tendría que existir un mapeo entre vértices nos daría como resultado dos grafos iguales.

Ahora, enfoquemos en ese nodo que va hacia si mismo. Creo que es obvio que el hecho de que existe un solo de este tipo de vértices solo nos deja un mapeo que parece ser valido:



Ahora veamos porque este mapeo esta mal, veamos los vértices adjacentes a ese nodo. Ahi es donde notamos una diferencia, en los grados de dichos vértices, por lo tanto ese mapeo tampoco puede ser valido



Por lo tanto no son isomorfas

3. Pregunta 6

Demostración:

Basta con saber que si se tienen n vértices entonces se tendrá a lo máximo $\frac{n(n-1)}{2}$ aristas.

Usando un sencillo codigo en Python podemos para n vértices cuantas aristas máximas pueden tener:

```
    for vertices in range(1, 12):
        print (f"Vertices {vertices}: ")
        print (int((vertices * (vertices - 1)) / 2 ))
```

Así vemos que si tienes 10 vertices entonces puedes tener máximo 45 aristas y seguir siendo un grafo simple, ahora, si tienes 11 vertices entonces puedes tener hasta 55 aristas, por lo tanto la respuesta es 11.

4. Pregunta 7

Demostración:

Basta con recordar un teorema que pronto demostraré que dice que la suma del los grados de todos los vertices de un grafo es el doble de las aristas del mismo, por lo tanto:

$$aristas = \frac{4+3+3+2+2}{2} = 7$$

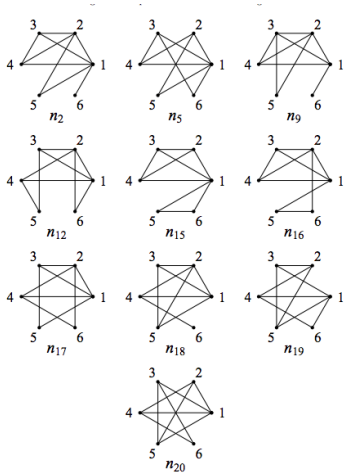
5. Pregunta 9

Demostración:

Esta esta sencilla, lo que en escencia se esta haciendo es preguntar si se puede hacer un nodo que tiene 6 vertices y 9 aristas tal que el grado de cualquier nodo es 3. Y claro que es posible, por el mismo argumento que en la anterior pregunta, si sumo 6 veces 3 entonces tengo 18 que entre dos nos da 9, así que todo perfecto.

Ahora, la idea es saber si es unica, lo que hice fue buscar todas las graficas de 6 vertices y 9 aristas y ver si habia dos que tuviera grado 3 en todos sus vértices. Y la respuesta es ... no.

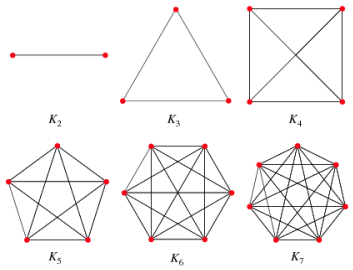
n_1	G161	H_{10}^9	{{6,2},{6,1},{5,2},{5,1},{4,2},{4,1},{3,2},{3,1},{2,1}}
n_2	G156		{{4,3},{4,2},{4,1},{5,2},{5,1},{6,1},{3,2},{3,1},{2,1}}
n_3	G162	H_9^9	{{3,4},{3,2},{3,1},{6,2},{6,1},{5,2},{5,1},{4,1},{2,1}}
n_4	G170	H_8^9	{{4,3},{4,2},{4,1},{6,2},{6,1},{5,2},{5,1},{3,2},{3,1}}
n_5	G164		{{4,3},{4,2},{4,1},{6,3},{6,1},{5,2},{5,1},{3,1},{2,1}}
n_6	G163	H_1^9	{{6,3},{6,2},{5,3},{5,1},{4,2},{4,1},{3,2},{3,1},{2,1}}
n_7	G166	H_2^9	{{4,2},{4,3},{4,1},{6,2},{6,1},{5,3},{5,1},{2,3},{2,1}}
n_8	G158	H_0^9	{{5,3},{5,2},{5,1},{4,3},{4,2},{4,1},{6,1},{3,1},{2,1}}
n_9	G157		{{4,3},{4,2},{4,1},{5,3},{5,2},{6,1},{3,2},{3,1},{2,1}}
n_{10}	G155		{{5,3},{5,2},{5,1},{4,3},{4,2},{4,1},{3,2},{3,1},{2,1}}
n_{11}	G159	H_6^9	{{5,2},{5,3},{5,1},{4,2},{4,3},{4,1},{6,1},{2,3},{2,1}}
n_{12}	G168		{{4,5},{4,2},{4,1},{3,5},{3,2},{3,1},{6,2},{6,1},{2,1}}
n_{13}	G173	H_4^9	{{5,3},{5,2},{5,1},{4,3},{4,2},{4,1},{6,2},{6,1},{3,1}}
n_{14}	G175	H_2^9	{{6,3},{6,2},{6,1},{5,3},{5,2},{5,1},{4,3},{4,2},{4,1}}
n_{15}	G165		{{4,3},{4,2},{4,1},{6,5},{6,1},{3,2},{3,1},{5,1},{2,1}}
n_{16}	G169		{{4,3},{4,2},{4,1},{6,5},{6,2},{3,2},{3,1},{5,1},{2,1}}
n_{17}	G167		{{4,6},{4,2},{4,1},{3,5},{3,2},{3,1},{6,2},{5,1},{2,1}}
n_{18}	G160		{{5,3},{5,2},{5,1},{4,6},{4,2},{4,1},{3,2},{3,1},{2,1}}
n_{19}	G172		{{5,3},{5,2},{5,1},{4,6},{4,2},{4,1},{3,2},{3,1},{6,1}}
n_{20}	G171		{{5,3},{5,2},{5,1},{4,2},{4,6},{4,1},{3,6},{3,1},{2,1}}
n_{21}	G174	H_3^9	{{6,4},{6,3},{6,2},{5,3},{5,2},{5,1},{4,2},{4,1},{3,1}}



6. Pregunta 11

Demostración:

Primero hay que recordar el teorema de que si en una grafica hay un o mas vertices que no tienen grado par entonces esa gráfica no tiene un circuito euleriano.



Ahora es obvio que K_4 NO tiene pero K_5 si.

7. Pregunta 12

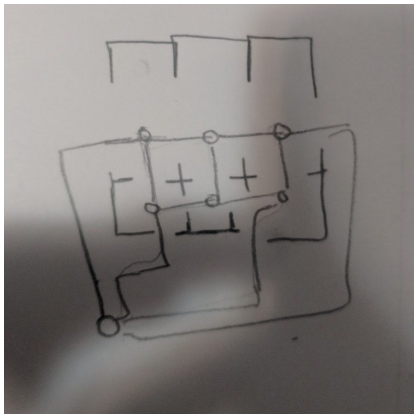
Demostración:

Es seguir la idea K_n tiene un circuito euleriano si es todos sus vértices tiene grado par. Por lo tanto K_n tiene un circuito euleriano si n es impar.

8. Pregunta 13

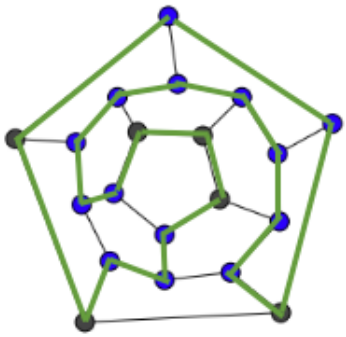
Demostración:

No, basta modelarla y ver que lo que se nos pide es un circuito euleriano, que no puede existir por el grado impar de muchos vertices.



9. Pregunta 14

Demostración:



10. Pregunta 16

Demostración:

Queremos probar que $2|E| = \sum_{v \in V} deg(v)$ para graficas simple, es decir sin loops.

Sea n el número de aristas de nuestro grafo $G(V, E)$.

La prueba será por inducción sobre el número de aristas:

- El caso base si $n = 0$, pues si no hay aristas entonces es obvio que la suma da cero. y $2 * 0 = 0$ que es el resultado correcto.
- Ahora supongamos cierto para una n cualquiera
- Ahora si tuviera un grafo con $n + 1$ aristas entonces tenemos que podemos pensar así:
  Si quitamos un arista entonces volvemos a tener un grafo $G'(V', E')$ que por la hipotesis de inducción tiene que pasar que $2n = \sum_{v \in V'} deg(v)$ ahora si añado de regreso la arista vamos a añadir 1 al grado de 2 vértices es decir: $2n + 2 = \sum_{v \in V} deg(v) = 2(n + 1)$, que es justo lo que queríamos.

11. Pregunta 15

Un grafo tiene un circuito euleriano si y solo si es que todos sus vértices tienen grado par y es conexo

Demostración:

Un circuito euleriano es un recorrido de todas las aristas de un gráfico simple una y solo una vez, iniciando en un vértice y terminando en el mismo vértice.

Se puede repetir los vértices tantas veces como queramos, pero nunca se puede repetir una arista una vez que la recorremos.

El grado de un vértice es el número de aristas que inciden en este vértice.

Entonces, sea G una gráfica que tenga un circuito euleriano. Cada vez que llegamos a un vértice durante nuestro recorrido de G, entramos por una arista y salimos por otra.

Por lo tanto, debe haber un número par de aristas incidentes en cada vértice. Por lo tanto, cada vértice de G tiene grado par.

Siendo más formales:

- Supongamos primero que tenemos una gráfica con un circuito euleriano que va como $v_0, v_1, \ldots, v_k$ donde $v_0 = v_k$.

  Ya que pasamos por cada arista una vez entonces la longitud de circuito es a cantidad de aristas.

  Ahora podemos definir de otra manera el grado de un nodo como la cantidad de veces que aparecen en la secuencia $v_0, v_1, \ldots, v_{k-1}$ multiplicado por dos, multiplicamos por dos porque si $u = v_i$ entonces v_{i-1}, v_i y v_i, v_{i+1} son aristas que inciden en el nodo u. Y si $u = v_0$ entonces v_{k-1}, v_k y v_0, v_1 son aristas que inciden sobre v_0. Por lo tanto todos los vértices tienen grado par.

- Ahora hay que ver porque es que si el grado de cada nodo es par y el grafo es conexo entonces tiene que existir un circuito euleriano si o si.

  Para hacerlo definimos a $W := v_0, v_1, \ldots, v_k$ com el camino mas grande en nuestro grafo que no pasa por mas una arista mas de una vez.

  Ahora, sabemos que W tiene que pasar por cada una de las aristas que llegan a v_k sino podriamos extender a W para que sea mas grande.

  Ahora, tiene que pasar que $v_0 = v_k$ y por lo tanto que W es un circuito porque sino entonces v_k tendría un grado impar. Ahora podría ir por contradicción y decir que W no es euleriano porque no pasa por todas las aristas (como G es conexo tenemos que podemos siempre encontrar una arista que no esta en W pero si que incide a un vértice que esta en W), llamemoslo (u, v_i), pero, entonces sería muy fácil extender a una W' que si que lo hiciera como: $W' = u, v_i, v_{i+1}, \ldots, v_k, v_1, v_2, \ldots, v_i$.

  Por lo tanto W tiene que ser un circuto euleriano.

12. Pregunta 17

Demostrar que si G es bipartita entonces todos los ciclos que contiene G tienen una cantidad par de vértices

Demostración: Una dirección es un sencilla: Si es que G es bipartito con dos conjuntos V_1, V_2, entonces cada paso a lo largo de un ciclo te manda o bien de V_1 a V_2 o de V_2 a V_1, para acabar donde empezaste, por lo tanto tienes que tomar un numero par de pasos.

Por otro lado, suponte que cada ciclo dentro de nuestro grafo tiene una longitud par, para cada vértice v_0 en el mismo componente C_0 que v_0, sea $d(v)$ sea la longitud del camino más corto de v_0 a v.

Colorea de rojo cada vértice en C_0 cuya distancia desde v_0 sea par, y colorea los otros vértices de C_0de azul. Hacemos lo mismo para cada componente del grafo, ahora ve que si nuestro grafo tuviera una arista entre dos vértices rojos o entre dos vértices azules, tendríamos un ciclo impar.

Por lo tanto, G es bipartito, con los vértices rojos y azules siendo las dos partes.

13. Pregunta 18

Demostrar que la cantidad de vértices de grado impar de una grafo G es par

Demostración:

La suma de todos los grados es igual al doble del número de aristas. Dado que la suma de los grados es par y ya que vamos a separar a las suma en otras 2 sumas, amabas tienen que ser o impar o par. Pero sabemos que la suma de los grados de los vértices con grado par es par, así que la suma de los grados de vértices con grado impar debe ser par.

Si la suma de los grados de vértices con grado impar es par, debe haber un número par de esos vértices, pues solo sumando una cantidad par de números impares obtenemos par.

14. Pregunta 19

Demostrar que el máximo número de aristas en una grafo G es $\frac{n(n-1)}{2}$ con n numero de vertices

Demostración:

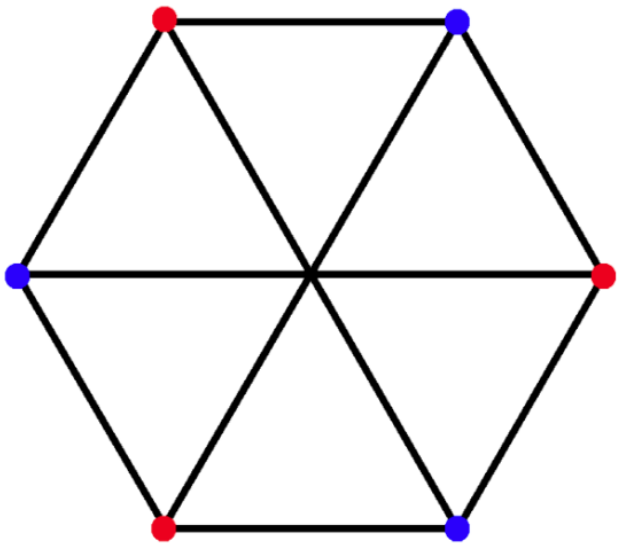
Una idea sencilla, el grafo que mas aristas podría tener dejando fijo en numero de vertices sería un grafo completo, es decir que cada vértice está conectado con cada otro vértice. Si tomamos un vértice, por lo tanto tiene $n-1$ aristas saliendo de el.

Ahora, si tenemos n vértices en total, sería lógico pensar que hay $n(n-1)$ aristas en total. Pero este método cuenta cada arista dos veces, porque cada arista que sale de un vértice es una arista que va a otro vértice. Por lo tanto, hay que dividir entre dos, dejandonos con: $\frac{n(n-1)}{2}$

15. Pregunta 21

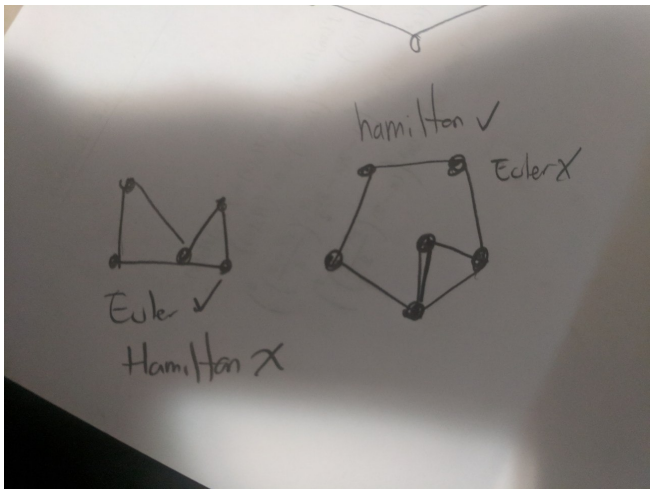
Construye una gráfica bipartita hamiltoniana y no euleriana.

Este esta sencillo, no puede ser Euleriano porque todos sus vertices tienen grado impar, pero el camino hamiltoniano es super sencillo es simplemente caminar por la parte exterior.



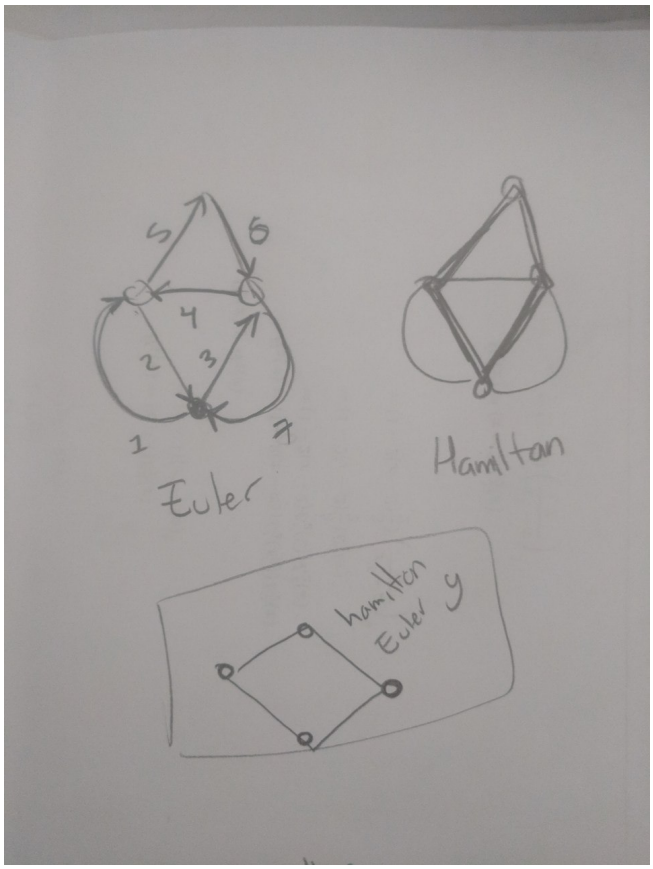
16. Pregunta 23

Proporcione una gráfica que contenga ciclo eulerino y no tenga ciclo hamiltonino, y una con ciclo hamiltoniano pero no tenga ciclo euleriano



17. Pregunta 24

De una gráfica que contenga un ciclo euleriano que también sea un ciclo hamiltoniano y otra donde sean distintos



18. Pregunta 25

Muestra que si n >2 la gráfica completa de n vértices Kn tiene un ciclo hamiltoniano

Demostración:

Llamemos a los vertices $v_1, v_2, \ldots, v_n$ entonces si empezamos el ciclo en v_1 como nuestro grafo es k_n es completo por lo que estamos seguros de que existe una arista a v_2 de ahi tomamos la arista que seguro tambien tiene que existir y así hasta que estemos en v_n de ahi de nuevo porque nuestro grafo es completo podemos asegurar de que existe una arista de regreso a v_1 con eso cerrando el ciclo.

19. Pregunta 26

Este estuvo buena, tuve que programar Dijkstra, pero fue mas rapido que andar haciendo las cuentas a mano les dejo el codigo asi como ss del resultado, eso si, por coherencia cambie la x por una a para que todo quede bonito.

```cpp
#include <vector>
#include <set>
#include <iostream>

using namespace std;
#define INF 0x3f3f3f3f

struct DijkstraResult { vector<int> minDistanceFromSourceTo; vector<int> previousVertex;};

DijkstraResult Dijkstra(const vector < vector< pair<int,int> > >& graph, int source, int
    numberOfVertexs) {
    vector<int> minDistanceFromSourceTo(numberOfVertexs, INF);
    vector<int> previousVertex(numberOfVertexs, -1);

    minDistanceFromSourceTo[source] = 0;
    previousVertex[source] = source;

    set< pair<int,int> > distancesFromSource;
    distancesFromSource.insert({0, source});

    while (distancesFromSource.empty() == false) {
        auto top = distancesFromSource.begin();
        distancesFromSource.erase(top);
        int u = top->second;

        for (auto next: graph[u])    {
            int v = next.first, weight = next.second;

            if (minDistanceFromSourceTo[v] > minDistanceFromSourceTo[u] + weight) {
                auto previous = distancesFromSource.find( {minDistanceFromSourceTo[v], v});
                if (previous != distancesFromSource.end()) distancesFromSource.erase(previous);

                minDistanceFromSourceTo[v] = minDistanceFromSourceTo[u] + weight;
                distancesFromSource.insert({minDistanceFromSourceTo[v], v});
                previousVertex[v] = u;
            }
        }
    }

    return {minDistanceFromSourceTo, previousVertex};
}

int main() {
    int numberOfVertexs, numberOfEdges, source;
    cin >> numberOfVertexs >> numberOfEdges >> source;


    int u, v, weight;
    vector < vector< pair<int,int> > > graph(numberOfVertexs);

    for (int i = 0; i < numberOfEdges; i++) {
        //Input the starting vertex of the edge, the ending vertex and the cost of the edge.
        cin >> u >> v >> weight;
        graph[u].push_back({v, weight});
        graph[v].push_back({u, weight});
    }

    auto result = Dijkstra(graph, source, numberOfVertexs);
    auto minDistanceFromSourceTo = result.minDistanceFromSourceTo;
    auto previousVertex = result.previousVertex;

    for (int vertex = 0; vertex < numberOfVertexs; vertex++) {
        cout << "min distance to " << char('a' + vertex) << ": " << minDistanceFromSourceTo[vertex]
    << endl;

        int currentVertex = vertex;
        while (currentVertex != source) {
            cout << char('a' + currentVertex) << " <- ";
            currentVertex = previousVertex[currentVertex];
        }

        cout << char('a' + source) << endl;
    }

}
```

Y la entrada que presenta el grafo, la primera linea es el numero de nodos, de aristas y el nodo origen, despues son una lista de 3 elementos que representan aristas, que es inicio, final y peso.

```
8 14 0
0 1 3
0 2 2
1 2 2
1 3 5
1 5 9
2 4 4
2 5 1
3 4 2
3 5 1
3 6 4
4 5 4
4 6 3
4 7 3
5 7 5
6 7 5
```

```
➜  scr git:(master) ✗ g++ -std=c++14 Dijkstra.cpp
➜  scr git:(master) ✗ ./a.out < Input26.in
min distance to a: 0
a
min distance to b: 3
b <- a
min distance to c: 2
c <- a
min distance to d: 4
d <- f <- c <- a
min distance to e: 6
e <- c <- a
min distance to f: 3
f <- c <- a
min distance to g: 8
g <- d <- f <- c <- a
min distance to h: 8
h <- f <- c <- a
```

20. Pregunta 27

Aqui el input que usamos:

```
9 17 0
0 1 0
0 2 3
0 3 1
1 2 1
1 3 1
2 4 2
2 4 2
2 7 4
3 4 5
3 5 3
4 5 3
4 6 1
5 6 1
5 8 9
6 7 4
6 8 9
7 8 4
```

```
➜  scr git:(master) ✗ ./a.out < Input27.in
min distance to a: 0
a
min distance to b: 0
b <- a
min distance to c: 1
c <- b <- a
min distance to d: 1
d <- a
min distance to e: 3
e <- c <- b <- a
min distance to f: 4
f <- d <- a
min distance to g: 4
g <- e <- c <- b <- a
min distance to h: 5
h <- c <- b <- a
min distance to i: 9
i <- h <- c <- b <- a
```