Agenda

1)Grafos (Ejercicio)

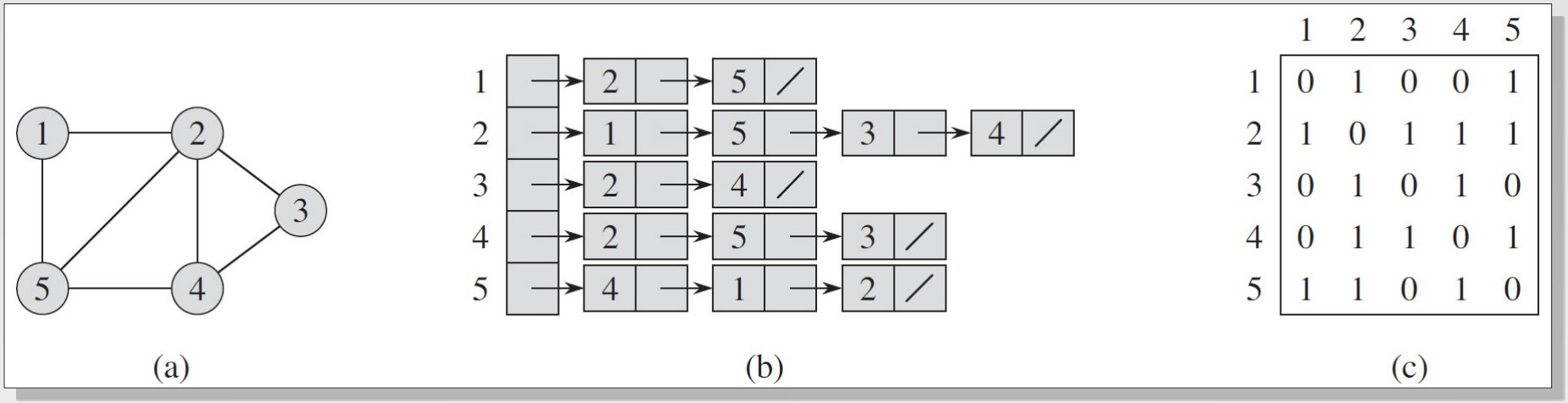
2)Archivos (Ejercicio)

3)Recordatorio:

1)Mañana defensa del proyecto

2)La próxima semana Tercer Quiz, exposición tarea extra (miércoles) y Examen final (jueves).

Representación de los Grafos



a)Representación de Imagen: grafo no dirigido de 5 vértices y 7 aristas.

b)Representación en Lista de adyacencia.

c)Representación en Matriz de adyacencia.

Algoritmos utilizados en Grafos

1)Recorrer el grafo:
   1)BFS – en amplitud (cola)
   2)DFS – en profundidad (pila)
2)Búsqueda del camino más corto:
   1)Dijkstra
   2)Bellman-Ford
   3)Floyd-Warshall
   4)Algoritmo de Johnson
3)Para encontrar un árbol recubridor mínimo (MST - Minimum Spanning Tree):
   1)Prim
   2)Kruskal

Ejercicio

A partir de la representación en Matriz de adyacencia del siguiente grafo no dirigido, realice:

a)La representación en imagen.

b)Recorra el grafo en amplitud (BFS) y en profundidad (DFS) (generar los árboles iniciando desde A).

c)Encuentre el árbol recubridor mínimo (MST) utilizando el algoritmo de Prim y Kruskal.

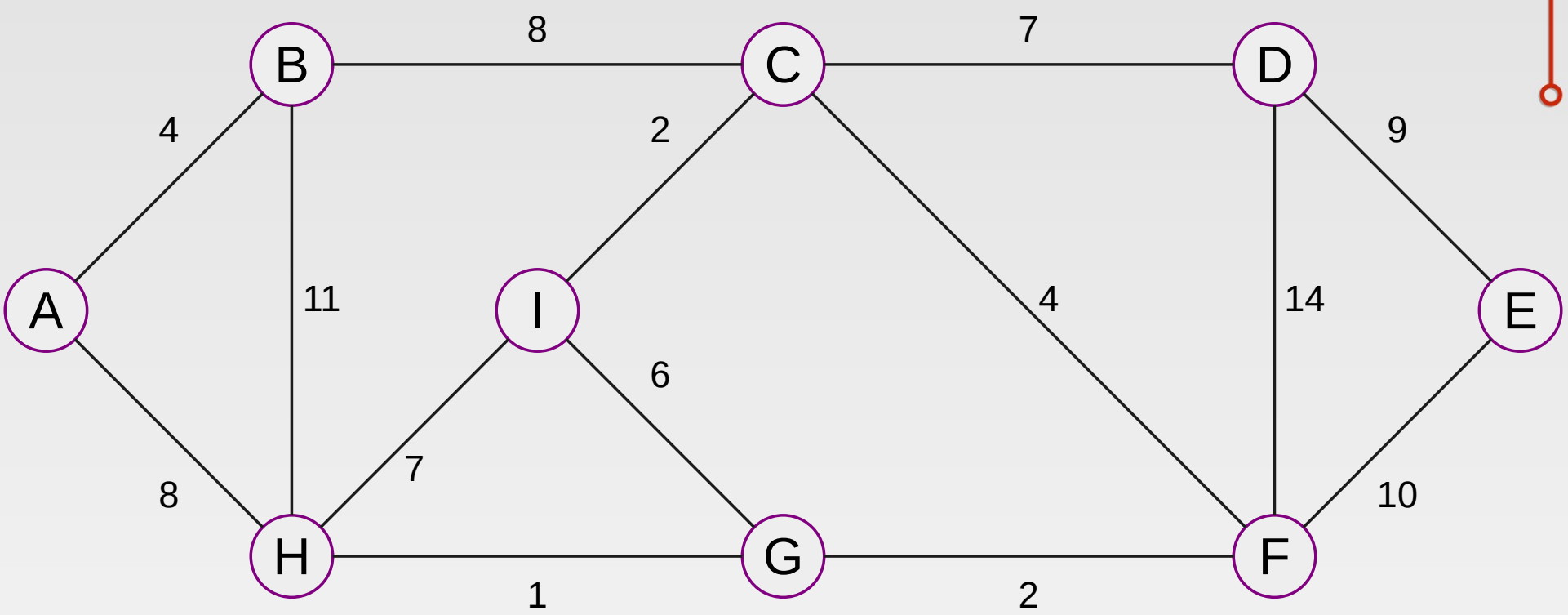
d)Calcular los pesos de los árboles según los algoritmos de Prim y Kruskal.

e)Opcional: Generar en lenguaje C++ una subrutina que encuentre el MST según el algoritmo de Kruskal.

Ejercicio – Matriz de adyacencia

	A	B	C	D	E	F	G	H	I
A	0	4	0	0	0	0	0	8	0
B	4	0	8	0	0	0	0	11	0
C	0	8	0	7	0	4	0	0	2
D	0	0	7	0	9	14	0	0	0
E	0	0	0	9	0	10	0	0	0
F	0	0	4	14	10	0	2	0	0
G	0	0	0	0	0	2	0	1	6
H	8	11	0	0	0	0	1	0	7
I	0	0	2	0	0	0	6	7	0

Solución – Grafo



Pseudocódigo BFS y DFS

```
BFS(G, s):
  for each vertex v in G:
    visited[v] = false
  queue Q
  Q.add(s)
  visited[s] = true
  while Q is not empty:
    u = Q.pop()
    for each neighbor v of u:
      if visited[v] == false:
        visited[v] = true
        Q.add(v)
```

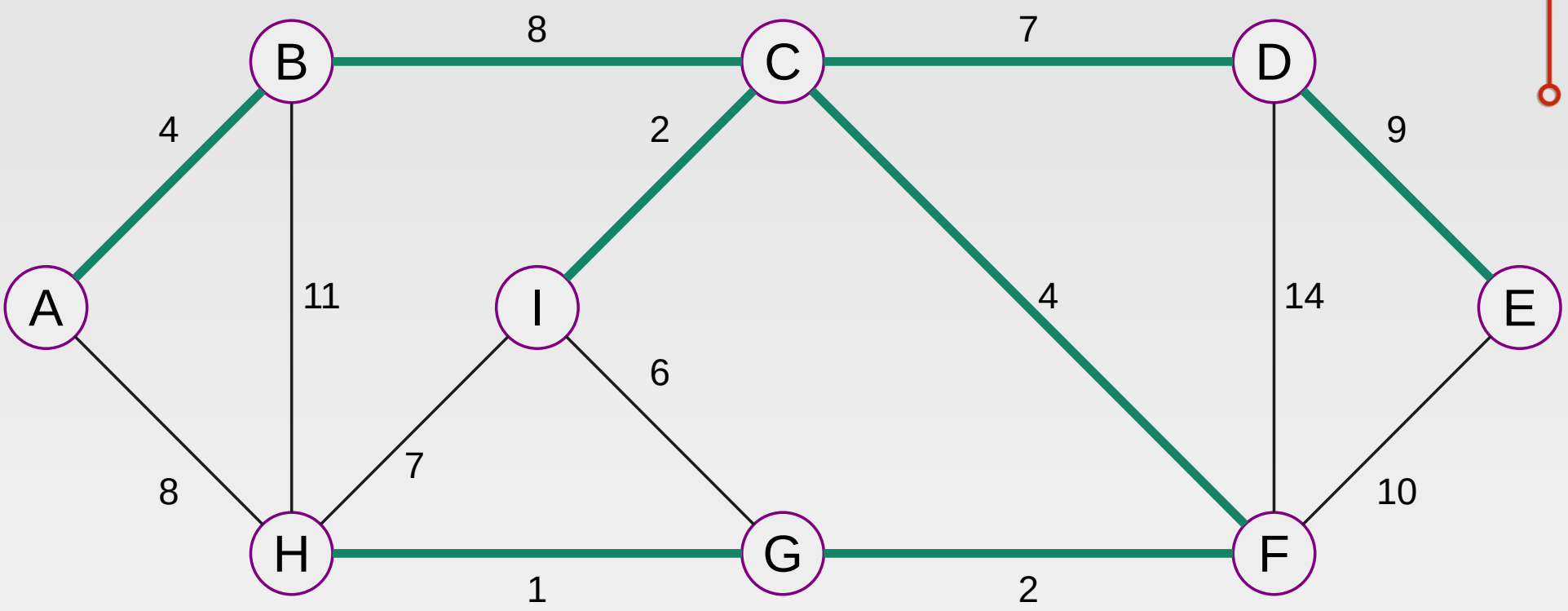
```
DFS(G, v):
  let S be a stack
  S.push(v)
  mark v as visited
  while S is not empty:
    t = S.pop()
    for each neighbor w of t:
      if w is not visited:
        mark w as visited
        S.push(w)
```

Algoritmo de Kruskal

1)Sort all the edges in non-decreasing order of their weight.
2)Create a new empty set called "tree" that will store the edges of the minimum spanning tree.
3)For each edge (u, v) in the sorted edges list:
 1)Find the parent set of both vertices u and v using a union-find algorithm.
 2)If the parent set of both vertices are different, then add the edge (u, v) to the tree set and unite the parent sets of both vertices.
4)The set tree now contains the minimum spanning tree of the graph.

Restricción: No pueden existir ciclos

Solución – MST



Peso: 37

Ejercicio – Grafos y archivos

El problema consiste en leer un archivo de texto que contiene una lista de ciudades y las distancias en kilómetros entre ellas. Con esta información, se debe crear un grafo ponderado y no dirigido que represente estas ciudades y sus distancias.

Una vez creado el grafo, se debe implementar un algoritmo de búsqueda en profundidad (DFS) para recorrer todas las ciudades y calcular la distancia total recorrida.

El resultado final del programa debe ser un archivo de texto con el recorrido de todas las ciudades y la distancia total recorrida.

Se debe establecer el formato que tendrá el archivo, que le permita al programa interpretar las ciudades como vértices y las distancias en sus aristas.

Ejercicio – Lista de ciudades

1)Nueva York (NYC) a Londres (LON): 5.586 km

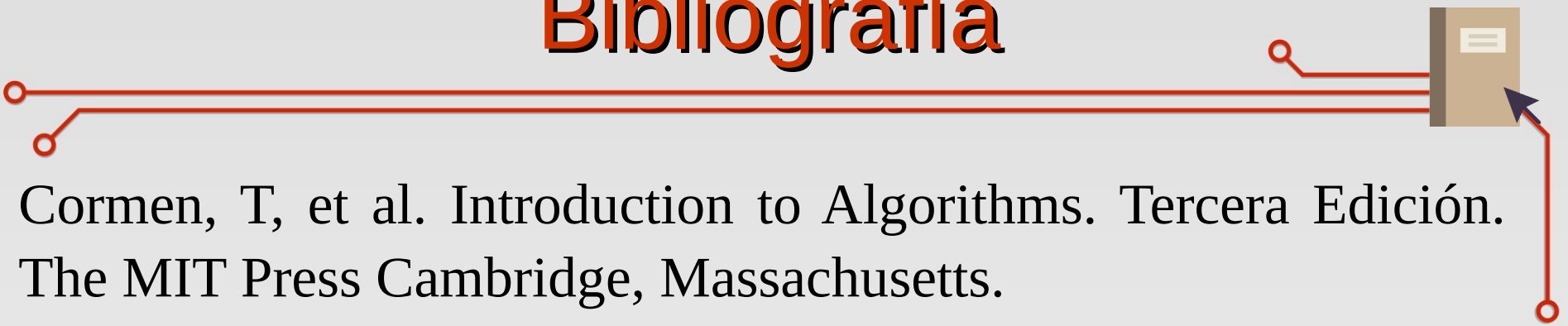
2)Nueva York (NYC) a Tokio (TYO): 11.986 km

3)Nueva York (NYC) a Sídney (SYD): 17.768 km

4)Londres (LON) a Tokio (TYO): 9.359 km

5)Londres (LON) a Sídney (SYD): 17.208 km

Bibliografía



- Cormen, T, et al. Introduction to Algorithms. Tercera Edición. The MIT Press Cambridge, Massachusetts.