

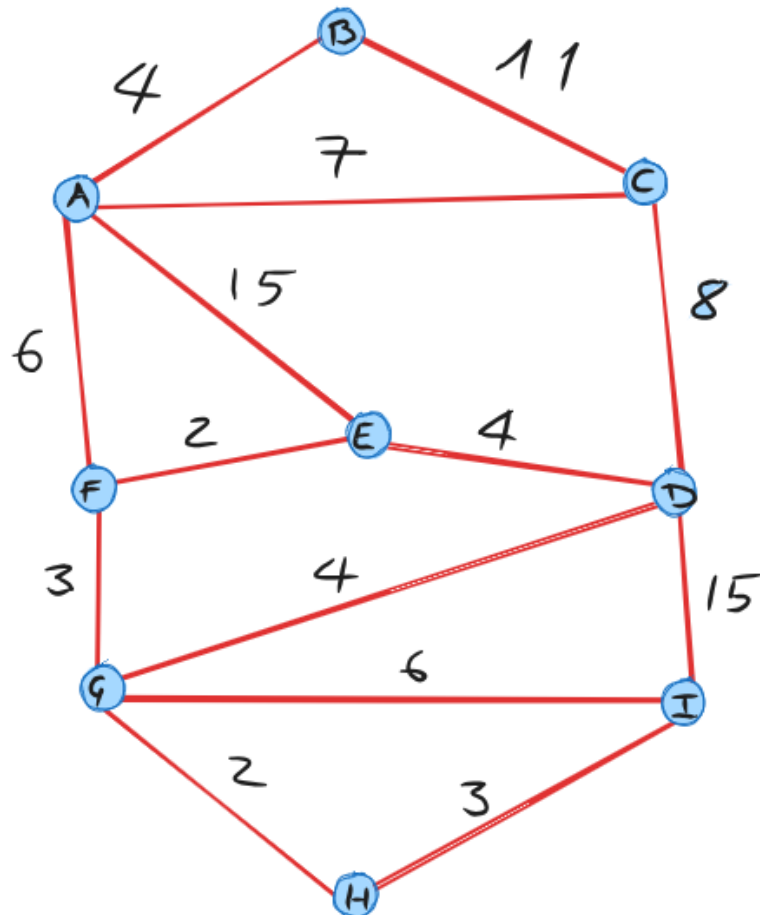
# Práctico 1 TEOCOMP: Algoritmos codiciosos.

Mauricio Velasco

1. *Scheduling avaricioso* Suponga que tenemos  $N = 100$  trabajos que deben compartir un mismo recurso. La longitud y la prioridad del  $t$ -ésimo trabajo están dadas por  $\ell_t := (t - 1)^2 + 4$  y  $w_t = (t - 5)^3$ .
  - a) Escriba una implementación de un algoritmo que calcule el mínimo tiempo total requerido para cumplir esas tareas.
  - b) Escriba tablas con los tiempos mínimos requeridos y las listas de las primeras 5 y últimas 5 tareas para  $N = 10, N = 100, N = 1000$ .
2. Implemente una clase `Grafo_no_dirigido` que represente un grafo  $G$  como lista de adyacencia. La clase debe recibir sólo el número de vértices del grafo e implementar las operaciones `G.nueva_arista(i,j)`, `G.nuevo_vertice()` y `G.print()`.
  - a) Escriba el código de su implementación.
  - b) Cuánta memoria (como función de  $n$ ) requiere su clase para representar:
    - 1) Un grafo completo  $K_n$ .
    - 2) Un grafo bipartito completo  $K_{n,n}$
    - 3) Un ciclo de longitud  $n$ .
    - 4) Un árbol con  $n$  vértices.
3. (*Árboles*) Recuerde que un **árbol** es un grafo conexo y acíclico. Demuestre las siguientes afirmaciones:
  - a) Todo árbol con  $n$  vértices tiene exactamente  $n - 1$  aristas.
  - b) Las siguientes tres afirmaciones son equivalentes para todo grafo no dirigido  $G$ :
    - 1)  $G$  es un árbol.
    - 2)  $G$  es minimal conexo (es decir  $G$  es conexo y quitarle cualquier arista lo vuelve desconexo).

- 3)  $G$  es maximal acíclico (es decir  $G$  no contiene ningún ciclo y adicionarle cualquier arista nueva hace aparecer al menos un ciclo en  $G$ ).
4. (*Cuántos árboles generadores tiene un grafo?*) El teorema de Birkhoff dice que el número de árboles generadores de un grafo no dirigido  $G$  es igual al producto  $(\lambda_1 \dots \lambda_{n-1})/n$  donde los  $\lambda_i$  denotan los valores propios diferentes de cero de la matriz  $L = D - A$  donde  $D$  es la matriz diagonal cuyas entradas son los grados de los vértices de  $G$  y  $A$  es la matriz de adyacencia de  $G$ . Realice los siguientes ejercicios:
- Si  $G = K_4$  es el grafo completo de 4 vértices, escriba la matriz  $L$ , calcule los valores propios y aplique la formula de Birkhoff. Dibuje todos los árboles generadores de  $K_4$  y verifique que todo sea consistente.
  - Como es la matriz  $L$  para el grafo  $K_n$  completo con  $n$  vértices?
  - Use el Teorema de Birkhoff y la matriz que calculo en la parte (b) para demostrar que  $K_n$  tiene  $n^{n-2}$  árboles generadores posibles (para  $n = 50$  esto es más que el número estimado de átomos en el universo).
  - Escriba el código en Python de un algoritmo que calcule el número de árboles generadores de un grafo de la clase `Grafo_no_dirigido` del punto (1). Coméntelo adecuadamente y verifíquelo para grafos completos con la fórmula del numeral anterior. Cuál es el  $n$  más grande al que puede llegar antes de que su computador sea incapaz de seguir?
5. Sea  $G$  un grafo no dirigido con pesos en las aristas. Escriba un algoritmo para construir un árbol generador de **máximo peso** para  $G$  que pueda ejecutarse en tiempo  $O((n + m) \log(n))$ . Justifique la validez de su respuesta.

6. (El algoritmo de Prim.) Sea  $G$  el grafo con costos en las aristas del siguiente dibujo:



- a) Ejecute a mano el algoritmo de Prim para encontrar el mínimo árbol generador para  $G$  con vértice inicial  $A$ . Escriba el vector  $X$  que representa el orden en el que se incluyen los vértices y el vector  $T$  de las aristas que lo constituyen. Dibuje el árbol generador obtenido.
- b) Ejecute a mano el algoritmo de Kruskal para encontrar el mínimo árbol generador para  $G$  con vértice inicial  $A$ .
7. (La propiedad de corte) Sea  $G$  un grafo no dirigido con costos en las aristas y suponga que los costos de todas las aristas son distintos. Un **corte** en  $G$  es una partición de los vértices en dos conjuntos disjuntos  $A$  y  $B$ . Una arista **cruza** el corte  $(A, B)$  si tiene un vértice en  $A$  y el otro en  $B$ .

- a) Demuestre la siguiente afirmación (llamada la propiedad de corte): Si una arista  $e$  es la más barata que cruza un corte  $(A, B)$  entonces  $e$  pertenece a todo árbol generador de mínimo costo de  $G$ .
  - b) Utilice la propiedad del corte para dar una demostración alternativa de que el algoritmo de Prim produce un árbol generador de mínimo costo.
8. (*Kruskal vs Prim*) El objetivo de este ejercicio es hacer una comparación empírica entre los algoritmos de Prim y Kruskal.
- a) Sea  $G_n$  el grafo completo con  $n$ -vértices con pesos  $1, 2, 3, \dots, \binom{n}{2}$  en las aristas.
  - b) Cuál es peso mínimo de un árbol generador en  $G_n$ ? La respuesta debe depender de  $n$ . Demuestre la validez de su respuesta usando el algoritmo de Prim.
  - c) Implemente los algoritmos de Prim y Kruskal.
  - d) Escriba tablas que midan el tiempo de ejecución y la memoria utilizada para encontrar el MST para los grafos  $G_n$  para diferentes valores de  $n$  (cuál es el máximo  $n$  en el que su algoritmo funciona?).
  - e) Escriba un párrafo que resuma sus conclusiones: Es alguno de los dos algoritmos mejor que el otro en algún aspecto? Argumente su respuesta.
9. (*Hacia un Kruskal eficiente*) Investigue los siguientes puntos:
- a) Describa de manera precisa la estructura de datos `union_find`.
  - b) Escriba el código en python de una implementación eficiente del algoritmo de Kruskal usando `union_find`.
  - c) Calcule el tiempo  $O(\bullet)$  requerido por su implementación argumentando la validez de su respuesta de manera precisa.