Algoritmos y Estructuras de Datos III

Primer Cuatrimestre de 2015

http://www.dc.uba.ar/materias/aed3

Normas básicas de higiene y seguridad

DEPTO. DE COMPUTACION - FCEyN - UBA

Por resolución 802/98 del Consejo Directivo de la Facultad cada Departamento debe elaborar un conjunto de Normas Básicas de Seguridad para ser incluidas en las Guías de Trabajos Prácticos de todas las materias. A continuación transcribimos las normas aprobadas por el Departamento de Computación Las normas básicas de higiene y seguridad son un conjunto de medidas destinadas a proteger la salud de todos, prevenir accidentes y promover el cuidado del material de los laboratorios. Son un conjunto de prácticas de sentido común: el elemento clave es la actitud responsable y la concientización de todos: personal y alumnado.

RESPÉTELAS Y HÁGALAS RESPETAR

1. Se deberá conocer la ubicación de los elementos de seguridad en el lugar de trabajo, tales como: matafuegos, salidas de emergencia, accionamiento de alarmas, etc.

2. Observar de qué clase –A, B o C– es cada matafuego ubicado en el Departamento de Computación, y verificar qué material combustible –papel, madera, pintura, material eléctrico– se puede apagar con él. Por ejemplo, nunca usar un matafuegos clase A para apagar fuego provocado por un cortocircuito.

3. Salidas: Ante una emergencia: se podrá solicitar a un docente –JTP o profesor, u otra persona autorizada– que obtenga la llave de la puerta de planta baja. La persona autorizada que abra la puerta se hará responsable del cierre de la puerta y de la devolución de la llave.

4. Las únicas ventanas sin reja son las de los cuartos 11 y 12, dan a un balcón.

5. No se deben bloquear las rutas de escape o pasillos con equipos, mesas, máquinas u otros elementos que entorpezcan la correcta circulación.

6. No se permitirá comer, beber, ni fumar en los laboratorios. Recordar que el polvo y la tiza dañan las PC.

7. No se permitirá enchufar ni desenchufar periféricos a las CPU de los laboratorios sin la supervisión del docente o administradores de la red. No se modificará la instalación de ningún equipo.

8. No se permitirán instalaciones eléctricas precarias o provisorias. Se dará aviso inmediato a la Secretaría Técnica en caso de filtraciones o goteras que puedan afectar las instalaciones o equipos y puedan provocar incendios por cortocircuitos (interno 355).

9. Es imprescindible mantener el orden y la limpieza. Cada persona es responsable directa del lugar donde está trabajando y de todos los lugares comunes.

10. Los laboratorios contarán con un botiquín de primeros auxilios con los elementos indispensables para atender casos de emergencia.

Procedimientos ante emergencias

a. Emergencias médicas

Si ocurre una emergencia tal como: cortes o abrasiones, quemaduras o ingestión accidental de algún producto químico, tóxico o peligroso, se deberá proceder:

a) A los accidentados se les proveerán los primeros auxilios.

b) Simultáneamente se tomará contacto con el Servicio Médico (Interno 482), o al Servicio Médico de Deportes (4576-3451 / 3459)

c) Avise al Jefe de Laboratorio o autoridad del Departamento, quienes solicitarán asistencia de la Secretaría Técnica (interno 380) para que envíen personal del Departamento de Mantenimiento, Seguridad y Control o Servicios Generales según correspondan.

d) El Jefe de Departamento notificará el accidente al Servicio de Higiene y Seguridad para su evaluación e informe, donde se determinarán las causas y se elaborarán las propuestas para modificar dichas causas y evitar futuras repeticiones.

e) Centros para requerir ayuda médica:

- S.A.M.E. Teléfono 107
- Hospital Pirovano
  Av. Monroe 3555 Tel. 4542-5552 / 9279

  INTOXICACIONES
- Hospital de Niños. Dr. R. Gutiérrez
  Sánchez de Bustamante 1399. Capital Federal. Tel: 4962-6666.
- Hospital de Niños. Dr. P. de Elizalde
  Av. Montes de Oca 40 Tel. 4307-7491. Toxicología: 4300-2115

  QUEMADURAS
- Hospital de Quemados
  P. Goyena 369 Tel. 4923-4082 / 3022

  OFTALMOLOGÍA
- Hospital Santa Lucía
  San Juan 2021 Tel. 4941-7077
- Hospital Dr. P. Lagleyze
  Av. Juan B. Justo 4151 Tel. 4581-0645 / 2792

b. Incendio

- Mantenga la calma. Lo más importante es ponerse a salvo y dar aviso a los demás.
- Si hay alarma, acciónela. Si no grite para alertar al resto.
- Se dará aviso inmediatamente al Departamento de Seguridad y Control (Interno 311) informando el lugar y las características del siniestro.
- Si el fuego es pequeño y sabe utilizar un extintor, úselo. Si el fuego es de consideración, no se arriesgue y manteniendo la calma ponga en marcha el plan de evacuación.
- Si debe evacuar el sector apague los equipos eléctricos y cierre las llaves de gas y ventanas.
- Evacue la zona por la ruta asignada.
- No corra, camine rápido, cerrando a su paso la mayor cantidad de puertas. No utilice ascensores. Descienda siempre que sea posible.
- No lleve consigo objetos, pueden entorpecer su salida.
- Si pudo salir, por ninguna causa vuelva a entrar. Deje que los equipos especializados se encarguen.

Teléfonos útiles:

- BOMBEROS: Teléfono 100

- DIVISIÓN CENTRAL DE ALARMA: 4381-2222 / 4383-2222 / 4304-2222.
- CUARTEL V DE BOMBEROS DE BELGRANO: Obligado 2254 Capital Tel. 4783-2222
- BOMBEROS DE VICENTE LÓPEZ: Av. Maipú 1669 Vicente López. Tel. 4795-2222
- BOMBEROS DE SAN ISIDRO: Santa Fe 650 Martínez. Tel. 747-2222

c. Corriente eléctrica

Es imprescindible la concientización del riesgo que engendra la corriente eléctrica. Ya que si bien no es la mayor fuente de accidentes, se trata generalmente de accidentes graves, en muchos casos mortales.

Los incendios provocados por causas eléctricas son muy frecuentes. Ellos ocurren por: sobrecalentamiento de cables o equipos bajo tensión debido a sobrecarga de los conductores; sobrecalentamiento debido a fallas en termostatos o fallas en equipos de corte de temperatura; fugas debidas a fallas de aislación; autoignición debida a sobrecalentamiento de materiales inflamables ubicados demasiado cerca o dentro de equipos bajo tensión, cuando en operación normal pueden llegar a estar calientes; ignición de materiales inflamables por chispas o arco.

Shock Eléctrico

Un shock eléctrico puede causar desde un sensación de cosquilleo hasta un desagradable estímulo doloroso resultado de una pérdida total del control muscular y llegar a la muerte.

Control de los riesgos eléctricos

Los factores principales a considerar son:

- el diseño seguro de las instalaciones
- el diseño y construcción de los equipos de acuerdo a normas adecuadas
- la autorización de uso después que se ha comprobado que es seguro
- el mantenimiento correcto y reparaciones
- las modificaciones que se efectúen se realicen según normas

Las precauciones generales contra el shock eléctrico son:

- la selección del equipo apropiado y el ambiente adecuado
- las buenas prácticas de instalación
- el mantenimiento programado y regular
- el uso de acuerdo a las instrucciones del fabricante

La protección contra el shock eléctrico se consigue usando:

- equipos de maniobra con baja tensión
- la doble aislación o la construcción aislada
- las conexiones a tierra y la protección por equipos de desconexión automática
- la separación eléctrica entre las fuentes y la tierra

Frecuentemente se usan adaptadores de enchufes. Tenga siempre en cuenta que cuando se usan estos aditamentos puede desconectarse la tierra del equipo que está usando.

Información sobre la materia

Programa

1 Algoritmos. Definición de algoritmo. Modelos de computación: modelo RAM, Máquina de Turing. Complejidad, definición, complejidad en el peor caso, en el caso promedio. Algoritmos de tiempo polinomial y no polinomial. Límite inferior. Ejemplo: análisis de algoritmos de ordenamiento. Algoritmos recursivos. Análisis de la complejidad de algoritmos recursivos.

Técnicas de diseño de algoritmos: dividir y conquistar, backtracking, algoritmos golosos, programación dinámica.

2 Grafos. Definiciones básicas: adyacencia, grado de un nodo, isomorfismos, caminos, conexión, etc. Grafos bipartitos. Arboles: caracterización, árboles orientados, árbol generador. Enumeración. Grafos eulerianos y hamiltonianos. Planaridad. Coloreo. Número cromático. Matching, conjunto independiente, recubrimiento. Recubrimiento de aristas y vértices.

3 Algoritmos en grafos y aplicaciones. Representación de un grafo en la computadora: matrices de incidencia y adyacencia, listas. Algoritmos de búsqueda en grafos: BFS, DFS, A^*. Mínimo árbol generador, algoritmos de Prim y Kruskal. Arboles ordenados: códigos unívocamente descifrables. Algoritmos para detección de circuitos. Algoritmos para encontrar el camino mínimo en un grafo: Dijkstra, Ford, Dantzig. Planificación de procesos: PERT/CPM. Algoritmos heurísticos: ejemplos. Nociones de evaluación de heurísticas y de técnicas metaheurísticas. Algoritmos aproximados. Heurísticas para el problema del viajante de comercio. Algoritmos para detectar planaridad. Algoritmos para coloreo de grafos. Algoritmos para encontrar el flujo máximo en una red: Ford y Fulkerson. Matching: algoritmos para correspondencias máximas en grafos bipartitos. Otras aplicaciones.

4 Problemas NP-completos. Problemas tratables e intratables. Problemas de decisión. P y NP. Maquinas de Turing no determinísticas. Problemas NP-completos. Relación entre P y NP. Problemas de grafos NP-completos: coloreo de grafos, grafos hamiltonianos, recubrimiento mínimo de las aristas, corte máximo, etc.

Docentes

e-mail: algo3-doc@dc.uba.ar

Teórica

Min Chih Lin	oscarlin@dc.uba.ar
Irene Loiseau	irene@dc.uba.ar

Práctica

Alejandro Strejilevich de Loma	asdel@dc.uba.ar
Agustín Gutiérrez	elsantodel90@gmail.com
Javier Casal	javierjcasal@gmail.com
Gonzalo Lera Romero	gleraromero@dc.uba.ar
Leopoldo Taravilse	ltaravilse@gmail.com

Laboratorio

Diego Delle Donne	diegodd@gmail.com
Xavier Warnes	xwarnes@gmail.com
Agustín Martínez-Suñé	aemartinez@dc.uba.ar
Melanie Sclar	melaniesclar@gmail.com

Alumnos

e-mail: algo3-alu@dc.uba.ar

Régimen de aprobación

1. Se tomarán dos exámenes parciales, el primero el 16/05/2015 y el segundo el 04/07/2015, cada uno de los cuales podrá recuperarse una vez los días 13/07/2015 y 17/07/2015, respectivamente.

2. Se realizarán tres trabajos prácticos cuyas fechas de entrega serán el 10/04/2015, el 08/05/2015 y el 26/06/2015, cada uno de los cuales podrá reentregarse una vez los días 04/05/2015, 05/06/2015 y 20/07/2015, respectivamente. Para más detalles sobre los trabajos prácticos leer más adelante las instrucciones sobre el laboratorio.

3. Para aprobar los prácticos y poder dar final como alumno regular, se requiere haber aprobado los dos parciales con nota 5 (cinco) o más en cada uno, y los trabajos prácticos con 5 (cinco) o más en cada uno. La aprobación de los trabajos prácticos se firmará *únicamente* en las fechas de toma o entrega de exámenes del primer cuatrimestre de 2015. No se firmarán los prácticos después de comenzado el siguiente cuatrimestre.

4. Para aprobar la materia por promoción (es decir, sin rendir examen final) se requiere:

   a. Haber aprobado los dos parciales con nota 5 (cinco) o más en cada uno y tener 7 (siete) o más de promedio.

   b. Haber aprobado los trabajos prácticos con nota 5 (cinco) o más en cada uno y tener 7 (siete) o más de promedio.

   c. Tener los finales de las correlativas aprobados antes de la última fecha de finales del cuatrimestre.

   Las promociones se firmarán *únicamente* en las fechas de final correspondientes al primer cuatrimestre de 2015.

5. Tanto para firmar los prácticos como para la promoción la nota puede ser la obtenida en el recuperatorio.

6. Toda información referente a la materia será publicada en la cartelera de Algoritmos y Estructura de Datos III y/o en la página web y/o será enviada a la lista de alumnos de la materia por e-mail.

Pautas generales para el Laboratorio

- Se realizarán 3 trabajos prácticos en grupos de hasta 4 personas. La información relativa a cada trabajo práctico estará disponible el la página web de la materia y en los enunciados de cada TP se podrán estipular condiciones particulares que invaliden algunas de las aquí descriptas.

Instrucciones generales para los trabajos

- Normalmente los programas leerán los datos desde la entrada estándar del sistema y escribirán los resultados a la salida estándar.
- Las funciones pedidas deben estar claramente separadas del código que se utiliza para probarlas.
- Los lenguajes de programación permitidos a utilizar se establecerán en las clases de laboratorio.
- Debe intentarse minimizar la complejidad de los algoritmos.
- No es necesario programar interfaces gráficas.

Fecha de entrega

- Los trabajos deben presentarse en las fechas que figuran arriba. Cada trabajo puede recuperarse una vez. Las fechas son tentativas y los cambios serán informados a los alumnos con antelación suficiente.
- El recuperatorio de cada trabajo podrá entregarse hasta dos semana después de la devolución del mismo por parte de los docentes.

Qué hay que entregar

- El informe debe incluir los datos de los integrantes del grupo (incluyendo e-mail), un índice inicial y el enunciado del trabajo práctico.
- En general, cada ejercicio se presentará en una sección separada incluyendo: introducción al problema, resolución propuesta, pseudocódigo, detalles de implementación, respuestas a las preguntas planteadas, análisis de la complejidad de los algoritmos principales, experimentación, comentarios, conclusiones, referencias (más o menos en ese orden).
- El informe debe ser autocontenido (entenderse por sí mismo sin necesidad de conocer de antemano los problemas resueltos).
- Las hojas no deben caerse de la carpeta y debe ser posible leer el informe con comodidad.
- Además del informe escrito, deberá entregarse el mismo informe en formato digital acompañado de los códigos fuentes desarrollados e instrucciones de compilación, de ser necesarias. Estos archivos deberán enviarse a la dirección `algo3.dc@gmail.com` con el asunto *"TP X: Apellido_1, ..., Apellido_n"*, donde X es el número de TP, n es la cantidad de integrantes del grupo y *Apellido_i* es el apellido del i-ésimo integrante.
- Los programas deben estar correctamente indentados y comentados para facilitar su lectura.
- Cada ejercicio debe presentarse en un directorio distinto incluyendo:
  - archivos con los fuentes, archivo del proyecto (si lo hubiera), archivo ejecutable
  - archivos de entrada de datos (si los hubiera), archivos de salida de datos (si los hubiera)

Bibliografía

En cada práctico se mencionan las referencias que mejor se ajustan a la forma en que será dado el tema correspondiente. La cátedra tiene ejemplares de todos los libros mencionados en esta bibliografía, que pueden ser solicitadas por los alumnos a los profesores. Hasta donde llega nuestro conocimiento, los libros marcados con (a) están en la biblioteca central (Pabellón II), los marcados con (b) en la Hemeroteca del Departamento de Matemática (segundo piso del Pabellón I) y los marcados con (c) en la Infoteca del Departamento de Computación.

Bibliografía básica:

[1] R. Ahuja, T. Magnanti, J. Orlin, *Network flows*, Prentice Hall, 1993. (a,c)

[2] G. Brassard, P. Bratley, *Fundamental of Algorithmics*, Prentice Hall, 1996. (c)

[3] M. Garey, D. Jonhson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. Freeman and Co., 1979. (a,c)

[4] J. Gross, J. Yellen, *Graph theory and its applications*, CRC, 1999. (c)

[5] F. Harary, *Graph theory*, Addison-Wesley, 1969 (hay una reedición de 1996). (a)

Bibliografía de consulta:

[6] A. Aho, J. Hopcroft, J. Ullman, *The design and analysis of computer algorithms*, Addison-Wesley, 1974 (hay edición en castellano). (a,c)

[7] A. Aho, J. Hopcroft, J. Ullman, *Data structures and algorithms*, Addison-Wesley, 1983 (hay edición en castellano). (a,c)

[8] A. Aho, J. Hopcroft, J. Ullman, *Foundations of Computer Science*, Computer Science Press, 1995.

[9] M. Albertson, J. Hutchinson, *Discrete Mathematics with Algorithms*, Wiley, 1988. (a,c)

[10] G. Baum, *Complejidad*, Kapelusz (I EBAI), 1986. (c)

[11] C. Berge, *The theory of graphs and applications*, Wiley, 1958. (a)

[12] C. Berge, *Graphs*, North-Holland, 1985. (b)

[13] N. Bigs, E. Lloyd, R. Wilson, *Graph theory: 1736-1936*, Oxford University Press, 1976.

[14] J. Bondy, U. Murty, *Graph theory with applications*, Macmillan Press, 1976.

[15] R. Campello, N. Maculan, *Algoritmos e heurísticas*, Editorial da Universidade Federal Fluminense, 1994. (a,c)

[16] N. Deo, *Graph theory with applications to engineering and computer science*, Prentice-Hall, 1974.

[17] S. Even, *Graph algorithms*, Computer Science Press, 1979.

[18] L. Ford, D. Fulkerson, *Flows in Networks*, Princeton University Press, 1962. (b)

[19] M. Gondran, M. Minoux, *Graphs and Algorithms*, John Wiley and Sons, 1984.

[20] E. Horowitz, S. Sahni, *Fundamentals of Computer Algorithms*, Computer Science Press, 1978.

[21] D. Knuth, *The art of computer programming*, Addison-Wesley, 1973. (a,c)

[22] Mc Hugh James, *Algorithmic Graph Theory*, Prentice-Hall International, 1990.

[23] C. Papadimitriou, *Computational Complexity*, Addison-Wesley, 1995.

[24] V. Rayward-Smith, I. Osman, C. Reeves, G. Smith, *Modern heuristic search methods*, Wiley, 1996.

[25] C. Reeves, *Modern heuristic techniques for combinatorial problems*, Blackwell, 1993.

[26] R. Sedgewick, *Algorithms in C++*, Addison-Wesley, 1998. (a,c)

[27] I. Sominsky, *Método de Inducción Matemática*, Lecciones populares de matemática, Editorial MIR, Moscú, 1985.

[28] M. Syslo, N. Deo, J. Kowaljk, *Discrete Optimization Algorithms*, Prentice Hall, 1983.

[29] M. Swamy, D. Thulasiraman, *Graphs, networks and algorithms*, John Wiley and Sons, 1981.

[30] J. Szwarcfiter, *Grafos e algoritmos computacionais*, Editora Campus, Rio de Janeiro, 1987.

[31] R. Tarjan, *Data structures and network algorithms*, Society for Industrial and Applied Mathematics, 1993. (a)

[32] A. Tucker, *Applied Combinatorics*, John Wiley and Sons, 1984. (c)

Práctica 1

Referencias para este práctico: **[9]**, [8], [27]

1.1. Probar por inducción:

a. $1 + 2 + \ldots + n = n(n+1)/2, \forall n \geq 1$

b. $1 + 3 + 5 + \ldots + (2n+1) = (n+1)^2, \forall n \geq 0$

c. $1^2 + 2^2 + \ldots + n^2 = n(n+1)(2n+1)/6, \forall n \geq 1$

d. $-1 + 2^2 - 3^2 + \ldots + (-1)^n n^2 = (-1)^n n(n+1)/2, \forall n \geq 1$

e. $(1 + 2 + 3 + \ldots + n)^2 = 1^3 + 2^3 + \ldots + n^3, \forall n \geq 1$

f. $1 \times 1! + 2 \times 2! + \ldots + n \times n! = (n+1)! - 1, \forall n \geq 1$

1.2. Encontrar una fórmula para la siguiente suma y demostrarla por inducción: $1 + 2 + 2^2 + 2^3 + 2^4 + \ldots + 2^n$.

1.3. La población de una colonia de hormigas se duplica todos los años. Si se establece una colonia inicial de 10 hormigas, ¿cuántas hormigas habrá después de n años?

1.4. Probar por inducción que para $n \geq 5$ se verifica que $2^n > n^2$.

1.5. La población de gatos en un depósito tiene la propiedad de que el número de gatos en un año es igual a la suma de la cantidad de gatos de los dos años anteriores. Si en el primer año había un solo gato, y en el segundo dos (¡suponiendo ello posible!), probar que el número de gatos en el año n es:

$$\sqrt{\frac{1}{5}} \times \left(\left(\frac{1+\sqrt{5}}{2} \right)^{n+1} - \left(\frac{1-\sqrt{5}}{2} \right)^{n+1} \right)$$

1.6. ¿Cuál es el error en la siguiente demostración?

Se quiere probar que los elementos $x_1, x_2, x_3, \ldots, x_n$, de un conjunto son iguales entre sí.

I. Paso inicial ($n = 1$): El conjunto tiene un sólo elemento x_1 que es igual a si mismo.

II. Paso inductivo: Supongamos que $x_1 = x_2 = x_3 = \ldots = x_{n-1}$. Como también vale la hipótesis inductiva para un conjunto de dos elementos, tenemos que $x_{n-1} = x_n$ y por tanto resulta que $x_1 = x_2 = x_3 = \ldots = x_{n-1} = x_n$.

1.7. ¿Cuál es el error en la siguiente demostración?

Se quiere probar que $\forall a \neq 0$ vale que $a^n = 1$.

I. Paso inicial ($n = 0$): $a^n = 1 \ \forall a$.

II. Paso inductivo: Supongamos que $a^{n-1} = 1$. Entonces $a^n = (a^{n-1} \times a^{n-1})/a^{n-2} = (1 \times 1)/1 = 1$.

Práctica 2

Referencias para este práctico: **[2]**, **[8]**, [26], [9]

2.1. ¿Qué cantidad de bits se necesitan para almacenar (en binario) un entero m?

2.2. a. Suponiendo que en una computadora dada, una operación básica de un algoritmo tarda 1 microsegundo en ejecutarse y que la complejidad del mismo es aproximadamente $10n$ (o sea un problema de tamaño n se resuelve en aproximadamente $10n$ microsegundos, donde n es el tamaño del problema), estimar el tiempo que la máquina tardará en resolver problemas donde n sea 20, 50, 100, 500 y 1000.

b. Idem a. para algoritmos de complejidades $1000n$, $\log_2 n$, $2^{n/3}$, $n^{\log_2 n}$ y 2^n.

c. ¿Cuál es el problema de mayor tamaño que puede resolverse en el tiempo que antes se resolvía un problema de tamaño $n = 1000$ en cada uno de los casos anteriores, si se dispone de una máquina 10000 veces más veloz?

2.3. a. Decidir si son verdaderas o falsas las siguientes afirmaciones y justificar:

I. n es $O(n)$
II. $3n^2 + 7n + 4$ es $O(n^2)$
III. n^i es $O(n^j)$ si $i < j$
IV. $n \log_2(n)$ es $O(n^2)$

b. Mostrar que $n!$ no es $O(r^n)$, cualquiera sea el entero positivo r. Concluir que una complejidad de $O(n!)$ es "peor"que exponencial de cualquier base.

2.4. Decir cual es la complejidad de los siguientes algortimos:

a. Dadas dos variables x, y:

`PASO 1: poner` $aux = x$
`PASO 2: poner` $x = y$
`PASO 3: poner` $y = aux$

b. Dado un vector v, su dimensión d y un elemento e:

`PASO 1: poner` $i = 1$
`PASO 2: mientras` $i < d$ `y` $e \neq v[i]$ `hacer`
`PASO 3:` $i = i + 1$
`PASO 4: si` $e = v[i]$ `imprimir` i
`PASO 5: sino imprimir ‘‘No se encontró’’`

c. Dado un número natural n:

```
función Factorial(n)
      si n = 0 devolver 1
      sino devolver (n × Factorial(n − 1))
fin función
```

d. Dado un vector v, y su dimensión d:

```
PASO 1: para i desde 1 hasta d hacer
PASO 2:        poner min = i
PASO 3:        para j desde i hasta d hacer
PASO 4:             si v[j] < v[min] poner min = j
PASO 5:        poner aux = v[i]
PASO 6:        poner v[i] = v[min]
PASO 7:        poner v[min] = aux
```

2.5. a. Mostrar que el siguiente algoritmo que lista todas las permutaciones de n elementos funciona correctamente y determinar su complejidad:

```
PASO 1: poner j = 1, construir la permutación {1} y poner S = {{1}}
PASO 2: poner j = j + 1
PASO 3: para cada permutación de j − 1 elementos {a_1, a_2, ..., a_{j−1}} en S, hacer:
PASO 4:    poner P = {a_1, a_2, ..., a_{j−1}, j}.
PASO 5:    si j < n poner S = S ∪ {P}. Si j = n imprimir P.
PASO 6:    para i = j − 1 hasta 1 hacer
PASO 7:       intercambiar las posiciones i e i + 1 de P
PASO 8:       si j < n poner S = S ∪ {P}.
PASO 9:       si j = n imprimir P.
PASO 10: si j < n ir a 2. Si j = n parar.
```

b. ¿Cuánto tardaría su computadora en correr este algoritmo para $n = 11$, $n = 15$, $n = 20$, $n = 24$?

2.6. a. Escribir en forma de algoritmo el método clásico para pasar un número de su representación binaria a su representación decimal. ¿Cuál es la complejidad de este algoritmo?

b. Analizar si los siguientes pasos corresponden o no a un algoritmo (supuestamente para determinar la representación binaria de un entero dado m).

```
PASO 1: escribir una secuencia cualquiera de 0s y 1s.
PASO 2: pasar el número anterior a decimal en la forma clásica
        (algoritmo del punto a.)
PASO 3: si el número obtenido en 2 es m parar, sino volver al paso 1.
```

c. Idem b. para:

```
PASO 1: poner k = 1.
PASO 2: listar todas las posibles combinaciones de k digitos binarios
        en orden creciente;
        transformar cada una de ellas a decimal por el método clásico
        (algoritmo del punto a.);
        si alguna de ellas es igual a m parar.
PASO 3: poner k = k + 1 e ir al paso 2.
```

d. Escribir el algoritmo clásico para pasar de la representación decimal a binaria de un número dado. Determinar su complejidad y compararla con la del algoritmo del punto c.

2.7. a. Escribir un procedimiento que dado un número entero, devuelva su factorización en números primos.

b. Escribir un algoritmo para reducir una fracción dada x/y a sus términos más chicos.

c. Determinar la complejidad de los algoritmos de los puntos a. y b.

2.8. a. Escribir el algoritmo de Euclides para calcular el máximo común divisor entre 2 números b y c en forma recursiva y no recursiva. Mostrar que su complejidad es $O(\text{mín}\{b, c\})$. ¿Puede construir un ejemplo (un peor caso) donde esta complejidad efectivamente se alcance? ¿Puede hacerlo para b y c tan grandes como se desee?

b. Analizar el siguiente algoritmo para determinar el máximo común divisor entre dos números b y c, y mostrar que su complejidad también es $O(\text{mín}\{b, c\})$.

```
PASO 1: poner g = mín{b, c}
PASO 2: mientras g > 1 hacer
PASO 3:     si b/g y c/g son enteros, informar mcd = g y parar.
PASO 4:     poner g = g − 1
PASO 5: informar mcd = 1 y parar
```

c. Las complejidades calculadas para los algoritmos de las partes a. y b. son iguales. ¿Cuál de los dos algoritmos elegiría? Justificar y comentar.

d. Probar que se puede mejorar la complejidad calculada en a. demostrando que el algoritmo de Euclides es en realidad $O(\log_2(\text{mín}\{b, c\}))$.

2.9. a. Escribir algoritmos para realizar suma, resta y multiplicación de matrices. Determinar la complejidad y expresarla en función de la dimensión de las matrices.

b. Escribir un algoritmo que dada una matriz A y un número n calcule A^n (o sea $A \times A \times \ldots \times A$, n veces). Determinar la complejidad del mismo.

c. Escribir y determinar la complejidad del algoritmo de Gauss (triangulación) para resolver un sistema de n ecuaciones lineales con n incógnitas.

d. Idem c. para el método de los determinantes (Regla de Cramer).

e. Determinar para los algoritmos de los puntos a., b., c. y d. cuál es la importancia del peor caso.

2.10. ¿Cuál es la complejidad en el caso promedio del algoritmo de búsqueda secuencial?

2.11. ¿Cuál es el límite inferior para la complejidad de un algoritmo que determine el máximo de n números?

2.12. a. Escribir un algoritmo para ordenar n elementos por el método de burbujeo y probar que su complejidad es $O(n^2)$.

b. Escribir algoritmos para ordenar n elementos por selección e inserción y calcular su complejidad.

c. ¿Cuál de los algoritmos presentados en a. y b. es mejor?

2.13. Presentamos a continuación otro algoritmo de ordenamiento llamado *Quicksort*. Sea S la secuencia que se va a ordenar.

procedimiento *Quicksort*(S)
    si long$(S) \leq 1$ **entonces devolver** S
    sino
        Partir(S, j)
        poner $S_1 \leftarrow \{S[i] : i < j\}$
        poner $S_2 \leftarrow \{S[i] : i > j\}$
        devolver *Quicksort*$(S_1) \bullet S[j] \bullet$ *Quicksort*(S_2)
fin procedimiento

```
procedimiento Partir(S,limite)
    poner i ← 1
    poner k ← long(S)
    mientras i < k hacer
        mientras i ≤ k ∧_L S[i] ≤ S[1] hacer
            poner i ← i + 1
        mientras i < k ∧_L S[k] > S[1] hacer
            poner k ← k − 1
        si i < k entonces
            Intercambiar(S, i, k)
    fin mientras
    Intercambiar(S, 1, i − 1)
    poner limite ← i − 1
fin procedimiento
```

Partir ordena S de tal forma que para todo $i < limite$, $S[i]$ es menor o igual que $S[limite]$, y para todo $i > limite$, $S[i]$ es mayor que $S[limite]$. $S[limite]$ es el primer elemento de la secuencia inicial, al que llamaremos pivote.

a. Ordenar la secuencia $20, 52, 45, 35, 15, 80, 76, 25$ usando *Quicksort*.

b. ¿Cuál es el peor caso para el algoritmo *Quicksort*? ¿Cuál es la complejidad del mismo?

c. ¿Cuál es la complejidad en el caso promedio?

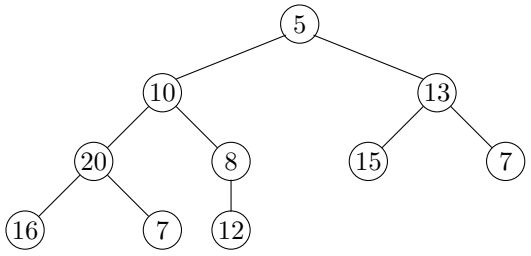
d. Si en *Partir* se toma como pivote el elemento medio de la secuencia, responder b. y c.

e. Si el pivote es la media entre el primer elemento, el ultimo y el elemento medio, responder a. y b.
(Modificando: $S_2 = \{S[i]/i \geq j\}$ y retornando
$Quicksort(S_1) \bullet Quicksort(S_2)$, pues la media puede no ser un elemento de la secuencia)

f. ¿Por qué se trata de que el pivote se acerque a la mediana?

2.14. Un "heap" o parva es un árbol binario o casi binario (puede haber un nodo que tenga una sola hoja como sucesora) organizado de tal manera que si a cada nodo le corresponde un número, entonces el valor de un nodo es mayor o igual que el de sus hijos. Por lo tanto, la raíz del árbol es el nodo de mayor valor. Esta estructura se usa para dar otro algoritmo de ordenamiento llamado *Heapsort*.

a. Diseñar un algoritmo para convertir un árbol binario a un heap. ¿Cuál es la complejidad? Aplicar ese algoritmo para el árbol siguiente:



b. Dado un heap probar que al podar la raíz alcanza con $O(log_2(n))$ operaciones para rearmar uno nuevo.

c. A partir de esto se construye un algoritmo, *Heapsort*, para ordenar un vector de n números. Escribirlo detalladamente. (Sugerencia: los índices van de 0 a $n - 1$, donde dado i, su hijo izquierdo es $2i + 1$ y su hijo derecho es $2i + 2$)

2.15. a. Escribir un algoritmo para ordenar una secuencia por el método *Mergesort*.

b. Determinar la complejidad del algoritmo enunciado.

c. ¿En qué casos es conveniente usar este método?

Práctica 3

Referencias para este práctico: **[2]**, **[8]**, [26], [9]

3.1. Escribir una función recursiva y una no recursiva para calcular los números de Fibonacci, definidos del siguiente modo:

$$\begin{aligned} F_0 &= 0 \\ F_1 &= 1 \\ F_n &= F_{n-1} + F_{n-2},\ \forall n \geq 2 \end{aligned}$$

Determinar la complejidad de ambas. ¿Cuál resulta mejor?

3.2. El conocido problema de las torres de Hanoi fue planteado por primera vez por el matemático francés Eduard Lucas del siguiente modo:

> En el inicio de los tiempos Dios creo las torres de Brahma con 64 discos de oro puro dispuestos en agujas de diamante. Cuando la hora llegó ordenó a un grupo de padres de un Monasterio en Tibet que dieran inicio al movimiento de los discos según reglas determinadas y avisó que concluido el trabajo las torres se desmoronarían dando inicio al final de los tiempos.
> Las reglas eran las siguientes:
> - Sólo se puede mover un disco a la vez.
> - Un disco de mayor tamaño no puede descansar sobre uno más pequeño que él mismo.
> - Sólo puedes desplazar el disco que se encuentre arriba en cada aguja.

a. Escribir un algoritmo recursivo para resolver este problema.

b. Demostrar que el algoritmo presentado en a. es correcto.

c. Analizar la complejidad del algoritmo.

d. Relacionar el resultado de c. con la formulación que el matemático Lucas hizo del problema ("el final de los tiempos").

3.3. a. ***** Escribir un algoritmo recursivo para determinar, dado un conjunto de puntos en el plano cuál es el par de puntos que está más próximo entre si, de acuerdo a las siguientes ideas:

- Ordenar los puntos de acuerdo a su coordenada x. Si todos los puntos tienen la misma coordenada x, ordenar los puntos de acuerdo a su coordenada y, y para el resto del algoritmo pensar las coordenadas intercambiadas.
- Trazar una recta vertical que divida al conjunto en dos partes R_1 y R_2 de manera tal que la cantidad de puntos en cada parte sea lo más cercano posible a la mitad del total.
- Calcular las distancias mínimas en cada parte, d_1 y d_2.
- Para calcular la distancia mínima en la unión de los dos conjuntos, calcular primero $d = \text{mín}\{d_1, d_2\}$.
- Eliminar los puntos que estén a distancia mayor a d de la recta vertical que separa a R_1 y R_2.
- Ordenar los puntos en cada R_i de acuerdo al valor de la coordenada y.
- Calcular la distancia de cada uno de los puntos a sus vecinos (son a lo sumo 5). Si cualquiera de estas distancias es menor que d, actualizar el valor de d.

b. Aplicar el algoritmo anterior al siguiente conjunto de puntos
$\{(1,3{,}5), (-1{,}3, 4{,}5), (-2,-1), (2,0{,}5), (0,3), (0,2{,}1),$
$(-1{,}1,,5), (2{,}5,-3{,}4), (1,-10), (0{,}5,7), (-0{,}7,9),$
$(-3,4{,}5), (3,-1{,}2), (-2,0), (4{,}5,0{,}9), (-3{,}4,-6{,}7)\}$

c. Determinar la complejidad del algoritmo.

3.4. Hay que organizar un torneo que involucre a n competidores. Cada competidor debe jugar exactamente una vez contra cada uno de sus oponentes. Además cada competidor debe jugar un partido por día con la sola posible excepción de un día en el cual no juegue.

a. Si n es potencia de 2 dar un algoritmo que use la técnica de dividir y conquistar para armar el fixture para que el torneo termine en $n-1$ días.

b. Si $n > 1$ no es potencia de 2 dar un algoritmo para armar el fixture para que el torneo termine en $n-1$ días, si n es par o en n, si n es impar.

3.5. Sean $P_1, P_2, \ldots, P_n$ programas que se quieren almacenar en una cinta. El programa P_i requiere s_i Kb de memoria. La cinta tiene capacidad para almacenar todos los programas. Se conoce la frecuencia π_i con que se usa el programa P_i. La densidad de la cinta y la velocidad del drive son constantes. Después que un programa se carga desde la cinta la misma se rebobina hasta el principio. Si los programas se almacenan en orden $i_1, i_2, \ldots, i_n$ el tiempo promedio de carga de un programa es

$$T = c \sum_j \left(\pi_{i_j} \sum_{k \leq j} s_{i_k} \right)$$

donde la constante c depende de la densidad de grabado y la velocidad del dispositivo. Queremos construir un algoritmo goloso para determinar el orden en que se almacenan los programas que minimice T.

Analizar las siguientes estrategias de almacenamiento:

I. en orden no decreciente de los s_i

II. en orden no creciente de los π_i

III. en orden no creciente de π_i/s_i

¿Cómo se podría demostrar que la última estrategia es la mejor? (Sugerencia: analizar en cada caso qué ocurre cuando dos elementos contiguos desordenados se ordenan.)

3.6. a. Se quiere dar el vuelto a un cliente usando el mínimo número de monedas posibles. Hay monedas de 1,5,10 y 25 centavos.(Hay al menos una de cada tipo) Analizar el siguiente algoritmo goloso para resolver el problema:

```
Elegir en cada paso la moneda de mayor valor entre las disponibles.
```

Probar que si hay una solución el algoritmo goloso siempre encuentra una solución para estos valores de las monedas.

b. Mostrar ejemplos que muestren que si también hay monedas de 12 centavos, o si no hay al menos una moneda de cada tipo, puede ocurrir que el algoritmo no encuentre una solución aunque la haya.

3.7. a. Escribir un algoritmo para calcular los coeficientes binomiales usando el triángulo de Pascal, que use la técnica de programación dinámica. Explicar porque se cumple en este caso el principio de optimalidad.

b. Determinar la complejidad temporal y espacial del algoritmo de a.

c. Revisar el ejercicio 3.1 y ver que la misma idea de a. se aplica al cálculo de los números de Fibonacci.

3.8. Sea $M \in \mathbf{N}^{m \times n}$ una matriz de números naturales. Se desea obtener un camino que empiece en la casilla superior izquierda ($[1,1]$), termine en la casilla inferior derecha ($[m,n]$), y tal que minimice la suma de los valores de las casillas por las que pasa. En cada casilla $[i,j]$ hay dos movimientos posibles: ir hacia abajo (a la casilla $[i+1,j]$), o ir hacia la derecha (a la casilla $[i,j+1]$).

a. Diseñar un algoritmo eficiente basado en programación dinámica que resuelva este problema.

b. Determinar la complejidad del algoritmo propuesto (temporal y espacial).

c. Exhibir el comportamiento del algoritmo sobre la matriz que aparece a continuación.

$$\begin{bmatrix} 2 & 8 & 3 & 4 \\ 5 & 3 & 4 & 5 \\ 1 & 2 & 2 & 1 \\ 3 & 4 & 6 & 5 \end{bmatrix}$$

3.9. Sea $v = (v_1, v_2, \ldots v_n)$ un vector de números naturales, y sea $w \in \mathbf{N}$. Se desea intercalar entre los elementos de v las operaciones + (suma), $\times$ (multiplicación) y $\uparrow$ (potenciación) de tal manera que al evaluar la expresión obtenida el resultado sea w. Para evaluar la expresión se opera de izquierda a derecha ignorando la precedencia de los operadores. Por ejemplo, si $v = (3, 1, 5, 2, 1)$, y las operaciones elegidas son +, $\times$, $\uparrow$ y $\times$ (en ese orden), la expresión obtenida es $3 + 1 \times 5 \uparrow 2 \times 1$, que se evalua como $(((3 + 1) \times 5) \uparrow 2) \times 1 = 400$.

a. Diseñar un algoritmo basado en programación dinámica que dados v y w, encuentre una secuencia de operaciones como la deseada, en caso de que tal secuencia exista.

b. Determinar la complejidad del algoritmo propuesto (temporal y espacial).

3.10. Dar un algoritmo de programación dinámica para el problema de dar el vuelto con el menor número de monedas posible presentado en el ejercicio 3.6. Decir si funciona en todos los casos o sólo en algunos casos (infinito numero de monedas disponibles, para monedas de cualquier valor, etc.)

3.11. Sean u y v dos string de caracteres. Queremos transformar u en v con el menor número de operaciones posibles de alguno de los siguientes tipos:

- borrar un caracter
- agregar un caracter
- cambiar un caracter

a. Por ejemplo uno puede transformar $u = abbac$ en $v = abcbc$ en tres pasos, (borrar b del segundo lugar de u, agregar b en el penúltimo lugar, cambiar la a que está en el tercer lugar a c)
¿Es esta forma de transformar u en v óptima?

b. Escribir un algoritmo de programación dinámica que encuentre el mínimo número de operaciones necesarias para transformar u en v e informe cuales son las operaciones necesarias. ¿Cuál es la complejidad del algoritmo en función de las longitudes de u y v?

3.12. Dar ejemplos de problemas donde *no* valga el principio de optimalidad.

3.13. a. Considerar el problema clásico de las 8 reinas que consiste en ubicar 8 reinas en un tablero de ajedrez sin que ninguna amenace a las otras. ¿Cuántas posibilidades habría que considerar si se usa un algoritmo de "fuerza bruta" que analice todas las posiciones posibles?

b. Enunciar un algoritmo que use backtracking para resolver este problema. ¿Es mejor que el de a. ? ¿Es un buen algoritmo?

3.14. Escribir un algoritmo de backtracking para la siguiente versión del problema de la mochila: tenemos n tipos de objetos. Un objeto de tipo i tiene un peso positivo w_i y un valor positico v_i. Suponiendo que tenemos una cantidad no limitada de cada tipo de objetos y que la mochila puede cargar un peso máximo de W, queremos maximizar el valor total de los objetos incluídos en la mochila.

3.15. a. El problema de la suma de subconjuntos consiste en, dado un conjunto $S = \{a_1, \ldots, a_n\}$ de números naturales y otro número natural w, determinar si existe un subconjunto S' de S tal que la suma de los elementos de S' sea igual a w. Escribir un algoritmo para resolver este problema que consista en revisar todos los subconjuntos de S para determinar si alguno suma w. Este tipo de algoritmo se llama algoritmo de "fuerza bruta".

b. Analizar la complejidad del algoritmo de a.

c. ¿Se le ocurre alguna idea mejor para resolver este problema?

3.16. a. Decir cuales de los algoritmos de los prácticos 2 y 3 usan la técnica de *dividir y conquistar*. Justificar.

b. ¿Qué algoritmos de los mismos prácticos son algoritmos recursivos?

c. ¿Se puede decir que siempre es mejor usar la versión recursiva de un algoritmo? ¿Al revés?

3.17. Clasificar todos los algoritmos de este práctico en "buenos" y "malos".

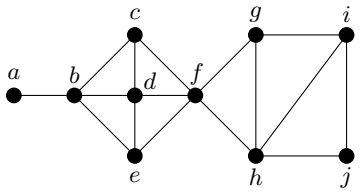
Práctica 4

Referencias para este práctico: [**4**], [14]
El objetivo de este práctico es modelar los problemas usando grafos. Algunos no podrán ser resueltos a esta altura de la materia.

4.1. Hay un conjunto de cinco personas y un conjunto de 5 trabajos para realizar. Sean las personas Carlos, Marcela, Pedro, Fernando y Andrea, y los trabajos a, b, c, d y e. Carlos está capacitado para realizar los trabajos c y d, Marcela para c, Pedro para a, b y e, Fernando para c y d, y Andrea para b y e. ¿Es posible realizar una distribución del trabajo de modo que se puedan realizar todos los trabajos simultáneamente?

4.2. Un ratón va cavando un túnel mientras recorre un cubo de queso de $30cm$ de lado. Quiere que el túnel pase por todos los subcubos de $10cm$ de lado. Si empieza por un subcubo en un vértice del cubo de queso, y se mueve hacia un subcubo que todavía no ha recorrido, ¿puede terminar en el centro del cubo?

4.3. El grafo de la figura representa una red telefónica. Los vértices representan centrales y las aristas líneas telefónicas. Se quiere estudiar la vulnerabilidad de la red ante algún defecto.



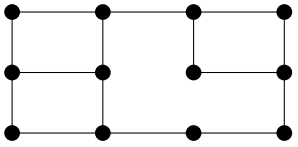
a. Determinar las líneas o centrales cuya salida de servicio impide que se realicen llamadas entre dos centrales cualesquiera.

b. Dar conjuntos minimales de líneas que mantengan conectadas todas las centrales.

c. Dar un conjunto minimal de líneas que no contenga las siguientes aristas: $(c, d), (b, c), (b, e), (d, f)$.

d. Si el número de centrales es n, ¿es cierto que en este caso con $n-1$ líneas alcanza para mantenerlas conectadas?

4.4. El grafo de la figura representa el mapa de una ciudad. Se quiere ubicar policías en las esquinas de modo que todas las cuadras estén bajo vigilancia, o sea, cada cuadra tiene que tener un policía al menos en una de las esquinas. ¿Cuál es el mínimo número de policías necesarios?



4.5. Supongamos que el siguiente programa quiere ser llevado a una máquina con n procesadores y memoria compartida, o sea que todos los procesadores tienen acceso a los mismos espacios de memoria (n suficientemente grande como para no imponer restricción de paralelismo). Supongamos también que el tiempo de ejecución de cada instrucción es de 1 microsegundo. Se sabe que la única

restricción que tenemos para que una instrucción deba ejecutarse antes que otra es que la segunda necesite algún resultado que se compute en la primera. Esto determina una relación de precedencia entre instrucciones. Luego, si dos instrucciones no dependen una de otra pueden ejecutarse al mismo tiempo.

a. Representar la relación de precedencia entre instrucciones utilizando un digrafo.

b. Haciendo una distribución de las instrucciones por los procesadores, ¿cuál será el tiempo mínimo de ejecución del programa?
$A \leftarrow 1$
$B \leftarrow 3$
$C \leftarrow 4$
$D \leftarrow A + B$
$E \leftarrow C - B$
$F \leftarrow D/C$
$G \leftarrow E * A$
$H \leftarrow F + G$

c. Si se asignan los siguientes tiempos de ejecución, ¿cómo variaría su respuesta?

asignación	$1ms$
$+$	$2ms$
$-$	$3ms$
$\times$	$4ms$
$/$	$5ms$

4.6. En un sistema operativo tenemos divididos a los procesos y a los recursos en dos conjuntos disjuntos, el de los p_i y el de los r_j respectivamente. El sistema operativo asigna recursos a procesos solo a pedido de los segundos. Cuando un proceso p_i pide un recurso r_j, queda en espera (sin ejecutar), hasta que el s.o. satisface el pedido; esto lo indicamos con un eje orientado de p_i a r_j. Cuando el s.o. asigna el recurso que requiere el proceso para ejecutar, el proceso deja de estar en espera y continúa con su ejecución; en este caso se elimina el eje de pedido y se indica que el proceso está utilizando el recurso con un eje entre ambos pero con sentido contrario al de pedido. En algún momento o bien el proceso no necesita más el recurso y lo libera –dejándolo disponible para que lo utilize otro proceso– o bien termina de ejecutar y libera todos los recursos que tiene asignados. Se entiende que cuando un proceso pide un recurso no puede estar liberando otro, pues no está ejecutando.

a. Modelar la siguiente situación:

p_1 está utilizando el recurso r_2
p_1 está utilizando el recurso r_3
p_2 pide el recurso r_1
p_3 pide el recurso r_2
p_3 pide el recurso r_1

¿Se pueden satisfacer los pedidos? Tener en cuenta que los procesos que ejecutan eventualmente liberarán los recursos que están utilizando.

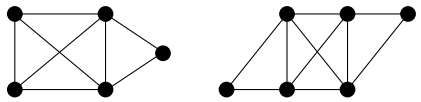
b. Analizar la siguiente situación:

p_1 está utilizando el recurso r_2
p_2 está utilizando el recurso r_3
p_3 está utilizando el recurso r_1
p_1 pide el recurso r_1
p_2 pide el recurso r_2
p_3 pide el recurso r_3

¿Se pueden satisfacer los pedidos?

c. ¿Qué tiene de particular el grafo resultante en b.?

4.7. Dados los dibujos de la figura



a. ¿Es posible realizarlos sin levantar el lápiz del papel? (Sin repetir aristas, comenzando y terminando en el mismo nodo)

b. Expresar el problema como un problema de grafos.

c. ¿Se puede particionar el grafo en circuitos simples disjuntos en las aristas?

4.8. Supongamos que un viajante de comercio debe visitar clientes en siete ciudades A, B, C, D, E, F y G. Existen las siguientes rutas con sus respectivas longitudes:

de A a B	6 Km	de B a C	7 Km	de D a E	2 Km
de A a C	4 Km	de C a D	4 Km	de C a F	3 Km
de C a G	2 Km	de E a G	4 Km	de F a G	1 Km
de A a F	6 Km	de B a G	8 Km		

¿Cuál sería el recorrido más corto para cubrir todas las ciudades y volver a la de partida? Este problema se conoce como el problema del viajante de comercio.

4.9. Supongamos que se tienen cuatro aulas y las siguientes materias con sus respectivos horarios para un mismo día:

Algebra	8 a 12 hs.
Análisis I	10 a 14 hs.
Análisis II	14 a 18 hs.
Lógica	11 a 15 hs.
Algoritmos I	12 a 16 hs.
Algoritmos II	9 a 13 hs.
Laboratorio 1	14 a 18 hs.
Laboratorio 2	14 a 18 hs.

¿Existe una forma de asignar aulas de forma que se puedan dictar todas las materias respetando los horarios?

Práctica 5

Referencias para este práctico: **[4]**, **[5]**, [12]

5.1. ¿Cuál es el mayor número de vértices que puede tener un grafo con 19 aristas y tal que sus vértices tienen grado al menos 3?

5.2. a. ¿Cuántos vértices tendrá un grafo con 12 aristas y todos sus vértices de grado 2?

b. ¿Cuántos vértices tendrá un grafo con 15 aristas, 3 vértices de grado 4 y los otros vértices de grado 3?

c. ¿Cuántos vértices tendrá un grafo con 20 aristas y todos sus vértices del mismo grado?

5.3. Probar que para un digrafo G cualquiera se verifica

$$\sum_i d_{in}(v_i) = \sum_j d_{out}(v_j)$$

5.4. Probar que en cualquier grupo de 2 o más personas, hay al menos 2 que tienen la misma cantidad de amigos dentro del grupo.

5.5. ¿Es posible que haya un grupo de 7 personas tal que cada persona conozca exactamente otras 3 personas del grupo?

5.6. Probar que un grafo es conexo si y sólo si para toda partición de V en dos subconjuntos V_1 y V_2 ($V_1 \neq \emptyset, V_2 \neq \emptyset, V_1 \cap V_2 = \emptyset, V_1 \cup V_2 = V$) hay un arco de G que une un punto de V_1 con uno de V_2.

5.7. Probar que un grafo de n vértices que tiene más de $((n-1)(n-2))/2$ aristas es conexo.

5.8. a. Probar que la unión de dos caminos simples distintos entre dos nodos p y q contiene un circuito simple.

b. ¿Qué pasa si los caminos no son simples?

5.9. Probar que en un grafo conexo cualquier par de caminos simples de longitud máxima tienen un vértice en común.

5.10. Dado un digrafo G, un vértice v es alcanzable desde un vértice u si existe un camino orientado de u a v. Un digrafo se dice fuertemente conexo si dados dos vértices cualesquiera u y v, u es alcanzable desde v.

a. Probar que si un digrafo es fuertemente conexo el grafo subyacente es conexo, pero la recíproca es falsa.

b. Orientar un grafo es darle una dirección a cada eje. Probar que un grafo conexo G es orientable de forma que se convierta en un digrafo fuertemente conexo si y sólo si cada eje de G pertenece a un circuito simple de G.

c. ¿Qué condición debe cumplir el mapa de calles de una ciudad para que se puedan hacer todas las calles de una sola mano?

5.11. a. Determinar un subgrafo completo de tamaño máximo para los grafos del ejercicio 5.17.

b. Determinar todos los subgrafos completos maximales para los grafos del ejercicio 5.17. ¿Son máximos?

5.12. Si G tiene vértices $v_1, v_2, \ldots, v_n$, se define como la secuencia de grados de G a la secuencia $d(v_1), d(v_2), \ldots, d(v_n)$. Probar que una secuencia de números enteros no negativos $d_1, d_2, \ldots, d_n$ es la secuencia de grados de un grafo (o multigrafo, o pseudografo) si y sólo si $\sum_i d_i$ es par.

5.13. La secuencia $D = (d_1, d_2, \ldots, d_n)$ se dice gráfica si hay un grafo simple con secuencia de grados D.

a. Mostrar que $(7, 6, 5, 4, 3, 3, 2)$ y $(6, 6, 5, 4, 3, 3, 1)$ no son secuencias gráficas.

b. Si D es una secuencia gráfica tal que $d_1 \leq d_2 \leq \ldots \leq d_n$ entonces:
$\sum_{i=1}^{n} d_i$ es par y
$\sum_{i=1}^{k} d_i \leq k(k-1) + \sum_{i=k+1}^{n} \text{mín}(k, d_i)$ para $1 \leq k \leq n$.
Nota: Se puede probar que esta condición es también suficiente para que una secuencia sea gráfica.

5.14. Dado un grafo $G = (V, X)$, se define el *grafo complemento* de G como $\overline{G} = (V, \overline{X})$ donde un eje $e \in \overline{X} \Leftrightarrow e \notin X$. Si G tiene n vértices, tal que exactamente $n-1$ de los mismos tienen grado impar, ¿cuántos vértices de grado impar tendrá $\overline{G}$?

5.15. ***** Sea $N(v)$ el conjunto de todos los vértices adyacentes a v y $N[v] = N(v) \cup \{v\}$. Sea R_n el conjunto de todos los grafos G con n vértices, que tienen las propiedades siguientes: para cualesquier par de vértices no contiguos u y v, o bien $N(v)$ está incluido en $N(u)$, o bien $N(u)$ está incluido en $N(v)$, y para cualesquier par de vértices contiguos u y v, o bien $N[v]$ está incluido en $N[u]$, o bien $N[u]$ está incluido en $N[v]$.

a. Demostrar que en un grafo $G \in R_n$ los vértices de un mismo grado son todos adyacentes de par en par, o bien no adyacentes de par en par.

b. Demostrar que en un grafo $G \in R_n$ existe al menos un vértice de grado $n-1$ si no hay vértices aislados.

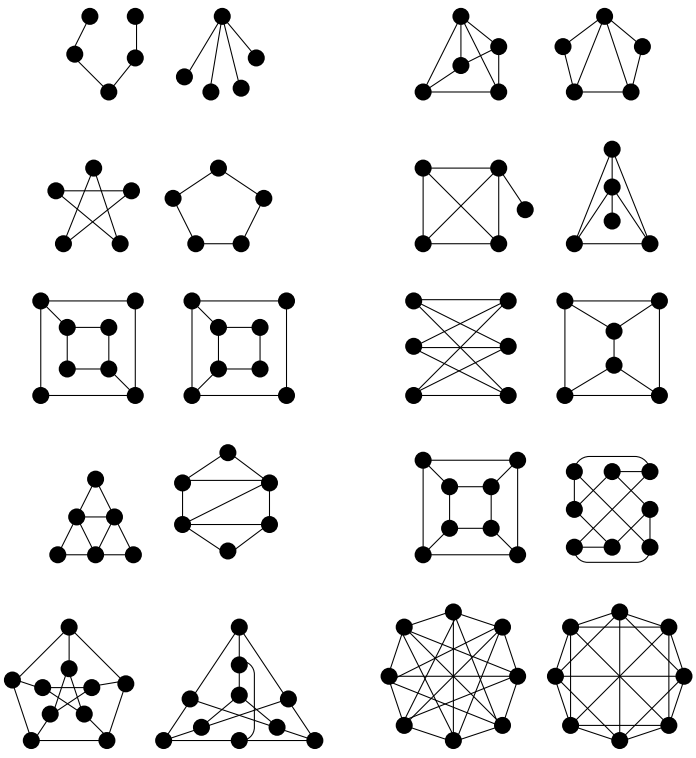
5.16. a. Dibujar todos los grafos no isomorfos de 4 vértices.

b. Dibujar todos los digrafos no isomorfos de 3 vértices.

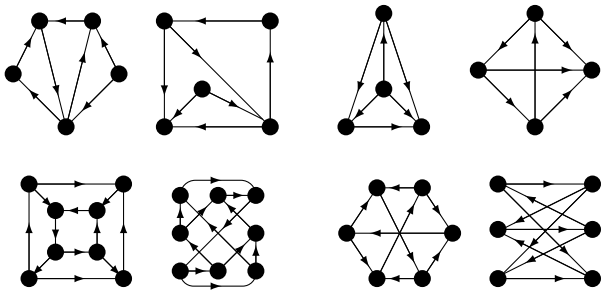
c. Dibujar todos los grafos no isomorfos de 5 vértices y 4 aristas.

d. Dibujar todos los digrafos no isomorfos de 5 vértices y 4 aristas.

5.17. ¿Cuáles de los pares de grafos de la figura son isomorfos? Justifique sus respuestas.



5.18. ¿Cuáles de los pares de digrafos de la figura son isomorfos? Justifique sus respuestas.



5.19. Sean dos grafos G y H isomorfos.

a. Mostrar que G y H tienen igual cantidad de vértices.

b. Mostrar que G y H tienen la misma cantidad de aristas.

c. Mostrar que para cada $d \geq 0$ el número de vértices de grado d es igual en los grafos G y H.

d. Mostrar que para cada l el número de ciclos simples de longitud l en los grafos G y H son iguales.

e. Mostrar que G y H tiene la misma cantidad de componentes conexas.

f. Mostrar un ejemplo de dos grafos no isomorfos que cumplan todas las condiciones anteriores.

g. Analizar conceptos similares para digrafos.

5.20. Un grafo se llama regular si todos sus vértices tienen el mismo grado.

a. Dibujar todos los grafos regulares no isomorfos de 4 vértices.

b. Dibujar todos los grafos regulares no isomorfos de 5 vértices.

5.21. Un grafo G se dice autocomplementario si G y $\overline{G}$ son isomorfos.

a. Hallar un grafo autocomplementario no trivial.

b. Mostrar que cualquier grafo autocomplementario debe tener $4k$ o $4k + 1$ vértices, para algún entero k.

5.22. Escribir un algoritmo que dados dos grafos G y H indique si son o no isomorfos. ¿Qué complejidad tiene este algortimo? ¿Es un buen algoritmo?

5.23. Dado un digrafo, se define su grafo subyacente como aquel que resulta de eliminar las direcciones a los arcos del digrafo. Si entre dos vértices había dos aristas de dirección opuesta, éstos se funden en uno solo no orientado. Dada la matriz de adyacencia de un digrafo, cómo se construye la matriz de adyacencia de su grafo subyacente?

5.24. a. Caracterizar la matriz de adyacencia de un grafo bipartito.

b. Probar que un grafo es bipartito si y sólo si para todo n impar los elementos de la diagonal de A^n son nulos (A matriz de adyacencia del grafo).

5.25. Se define la matriz R de alcance de un digrafo G como:

$$r_{ij} = \begin{cases} 1 & \text{si } v_j \text{ es alcanzable desde } v_i \\ 0 & \text{si } v_j \text{ no es alcanzable desde } v_i \end{cases}$$

¿Cómo se puede calcular R a partir de la matriz de adyacencia?

5.26. a. Escribir un algoritmo que dado un grafo G indique si es o no un grafo bipartito. Idem para digrafos.

b. Definir un procedimiento que dado un grafo G calcule $\overline{G}$. Idem para digrafos.

c. Calcular las complejidades de los algoritmos de a. y b.

5.27. a. Escribir una función que dado un grafo G verifique si cada eje del G pertenece a un circuito simple de G. Analizar distintas estructuras de datos.

b. Escribir un procedimiento que dado un grafo G, si puede lo oriente de forma que se convierte en un digrafo fuertemente conexo. Analizar distintas estructuras de datos. Utilizar la función definida en a.

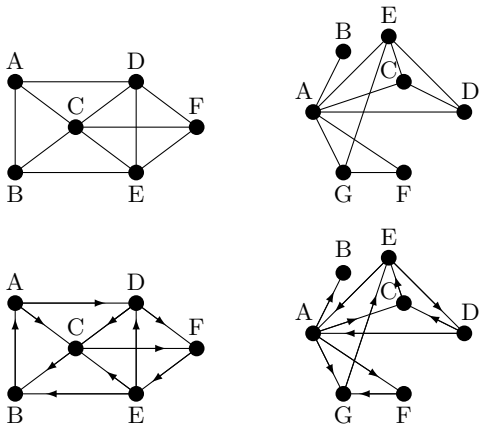
c. Analizar las complejidades de los algoritmos de a. y b.

5.28. a. Escribir un algoritmo que dado un grafo G devuelva un subgrafo completo máximo del mismo.

b. Escribir un algoritmo que dado un grafo G liste todos sus subgrafos completos maximales.

c. Analizar las complejidades de los algoritmos de a. y b.

5.29. Dados los grafos y digrafos de la figura:



a. Escribir las matrices de adyacencia e incidencia.

b. Representar mediante listas de aristas y listas de adyacencias.

c. Calcular los conjuntos de sucesores y de predecesores de los vértices de los digrafos de la figura y de los digrafos del ejercicio 5.18.

d. Dadas las siguientes matrices de adyacencia representar el correspondiente grafo o multigrafo (no orientado).

$$A = \begin{vmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \end{vmatrix} \quad A = \begin{vmatrix} 1 & 1 & 2 & 1 \\ 1 & 0 & 3 & 0 \\ 2 & 3 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{vmatrix}$$

e. Calcular el grado de los vértices de los grafos de a. y d. usando las matrices de adyacencias correspondientes.

f. Determinar todos los caminos simples de A a D y de E a B, a partir de la matriz de adyacencias de los grafos de a.

g. Representar los digrafos cuyas matrices de adyacencia son

$$A = \begin{vmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{vmatrix} \quad A = \begin{vmatrix} 1 & 1 & 2 & 1 \\ 1 & 0 & 3 & 0 \\ 2 & 3 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{vmatrix}$$

Práctica 6

Referencias para este práctico: **[4]**, **[5]**, [32]

6.1. a. ¿Cuál es el máximo número de vértices que puede tener un grafo conexo de 20 aristas?

b. Probar que si G es un árbol con un número par de ejes, entonces G tiene al menos un nodo de grado par.

c. Supongamos que el promedio de los grados de los nodos de G es exactamente 2. Si G es conexo, ¿cuántos circuitos tiene G?

6.2. Es cierto que:

a. Si un grafo no tiene puentes entonces todo nodo tiene grado par?

b. Un grafo conexo con n ejes tiene exactamente un circuito?

c. Existe un grafo conexo con $n+1$ ejes que tiene exactamente dos circuitos?

(tanto en b. como en c., n es la cantidad de nodos del grafo)

6.3. a. Probar que todo árbol no trivial es un grafo bipartito.

b. ¿Hay árboles bipartitos completos?

6.4. Dibujar todos los árboles no isomorfos de 4 vértices, 5 vértices y 6 vértices.

6.5. Probar que todo árbol con dos o más vértices tiene al menos 2 hojas.

6.6. Se dice que un grafo es un bosque si no tiene ciclos. Probar que G es un bosque si y solo si al sacar cualquier eje de G aumenta el número de componentes conexas.

6.7. Probar que el grafo complemento de un árbol es conexo o tiene un vértice aislado y el resto forma un subgrafo completo.

6.8. ***** Probar que la secuencia gráfica $(d_1, d_2, \ldots, d_n), d_i \in \mathbf{N}$, es la secuencia de un árbol si y solo si $\sum_i d_i = 2(n-1)$ y es conexo.

6.9. a. ¿Puede haber un árbol binario con un número par de vértices? Justificar.

b. ¿Cuál es el máximo número de vértices que puede tener un árbol m-ario de altura h?

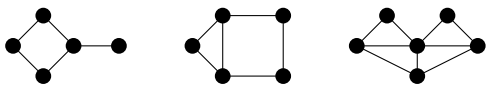
c. Probar que un árbol m-ario de altura h tiene a lo sumo m^h hojas.

d. Probar que un árbol m-ario con l hojas tiene altura $h \geq \lceil \log_m l \rceil$. Si el árbol es balanceado, probar que $h = \lceil \log_m l \rceil$.

e. Se define el nivel de un vértice v en un árbol como su distancia a la raíz, o sea, como la longitud del único camino de la raíz al vértice. Mostrar que en un árbol binario con l hojas, la suma de los niveles de las mismas es mayor igual a $\lceil l \times \log_2 l \rceil$, y el nivel promedio es mayor o igual a $\log_2 l$.

Nota: $\lceil x \rceil = \text{mín}\{z \in \mathbf{Z} : z \geq x\}$.

6.10. Dibujar todos los árboles generadores de los siguientes grafos:

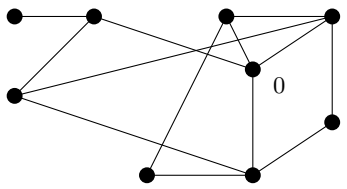


6.11. a. Probar que si un grafo conexo tiene un único árbol generador entonces es un árbol.

b. ¿Es posible reconstruir un grafo si se conocen todos sus árboles generadores? ¿Cómo?

c. * Dar un método para determinar el número de árboles generadores de un grafo sin listarlos explícitamente.

6.12. a. Usar *BFS* para numerar el grafo de la figura a partir del nodo marcado con 0.



b. Idem a. para *DFS*.

6.13. Escribir un algoritmo recursivo y otro iterativo que implementen *BFS*. ¿Qué técnicas utilizan estos algoritmos?

6.14. a. Probar que con *DFS* se alcanzan todos los vértices de un grafo conexo.

b. Idem a. para *BFS*.

6.15. a. Escribir un programa para contruir un árbol generador dado por *DFS* dada la matriz de adyacencia de un grafo.

b. Idem para *BFS*.

6.16. Probar que la altura de cualquier árbol generador determinado por *DFS* a partir de una raíz dada r, es mayor o igual que la altura de un árbol generador construído por *BFS* a partir de la misma raíz r.

6.17. a. ¿Cómo podría reescribirse alguno de los algoritmos anteriores para usarlo para determinar si un grafo es o no conexo?

b. Enunciar un algoritmo para determinar el número de componentes conexas de un grafo.

6.18. a. Determinar la complejidad del algoritmo de *Prim*.

b. Escribir un programa para el algoritmo de *Prim*.

c. ¿Qué técnica utiliza este algoritmo?

6.19. Dado el algoritmo de *Kruskal* para determinar un árbol generador mínimo de un grafo conexo $G = (V, X)$ y una función $l(e) : X \to \mathbf{R}$:

`PASO 1: poner` $i = 1, T = \{e\}$`, con` e `de longitud mínima`

`PASO 2: mientras` $i < n - 1$ `hacer:`

`PASO 3:` `elegir` e `tal que` $l(e)$ `sea mínima entre los que no forman circuito con los arcos que ya están en` T

`PASO 4:` `poner` $T = T \cup \{e\}$

`PASO 5:` `poner` $i = i + 1$

a. Probar que el algoritmo de *Kruskal* construye un árbol generador mínimo.

b. Determinar la complejidad del algoritmo de *Kruskal*.

c. Escribir un programa para el algoritmo de *Kruskal* utilizando matrices de adyacencia y listas de adyacencias como estructuras de datos.

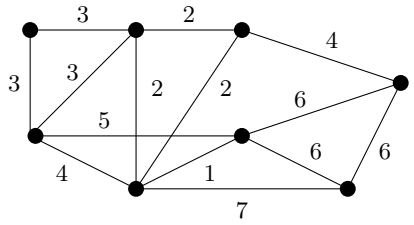
d. ¿Qué técnica utiliza este algoritmo?

6.20. Vialidad Nacional quiere construir, de la forma más económica posible, caminos que vinculen 5 ciudades (aunque para ir de una a otra haya que pasar por una tercera). Los costos de los tramos entre cada par de ciudades están dados en la tabla. Decir que tramos deberían construirse.

	B	C	D	E
A	5	10	80	90
B		70	60	50
C			8	20
D				10

6.21. a. Enunciar un algoritmo que determine un árbol generador máximo de un grafo dado.

b. Aplicar el algoritmo al grafo de la figura.



6.22. Probar que si todos los arcos de un grafo G tienen distinta longitud entonces G tiene un único árbol generador mínimo.

6.23. Sea T un árbol generador mínimo de un grafo G determinado por el algoritmo de *Kruskal*.

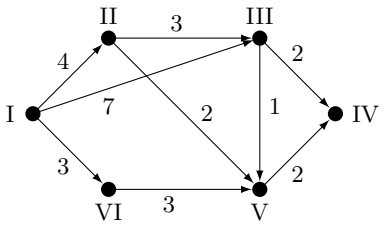
a. Probar que T contiene todos los arcos de longitud mínima salvo que los mismos incluyan un circuito.

b. Realizar la misma demostración del punto anterior pero sin asumir que el árbol fue generado por un algoritmo en particular.

Práctica 7

Referencias para este práctico: **[19]**, **[4]**, [1]

7.1. a. Usar el algoritmo de *Dijkstra* para calcular el camino más corto desde el vértice I a todos los demás en el siguiente grafo:



b. Suponiendo que todos los arcos tuvieran la misma longitud repetir a. usando *BFS*.

c. Suponiendo que después de resolver a. se descubriera que falta un arco en el grafo o que hay que modificar alguna longitud, ¿es necesario repetir el algoritmo completo o se pueden aprovechar los resultados obtenidos?

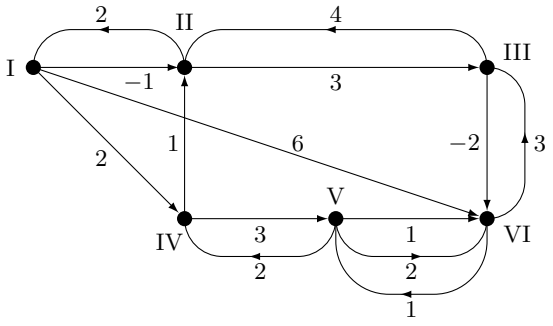
7.2. a. Dar un ejemplo que muestre porque no se puede aplicar el algoritmo de *Dijkstra* cuando existen arcos de longitud negativa. ¿Se puede construir un ejemplo en el cual *Dijkstra* funcione correctamente a pesar de tener arcos de longitud negativa?

b. ¿Cuál es el trabajo computacional necesario para calcular los caminos de longitud mínima entre todos los pares de vértices de un grafo usando *Dijkstra*?

c. ¿Se puede considerar a *Dijkstra* como un algoritmo goloso?

7.3. Reescribir el algoritmo de *Dijkstra* para el caso en que los nodos del grafo estén organizado en un heap de acuerdo a los valores de π. Mostrar que en este caso el trabajo computacional que requiere el algoritmo es $O(m \log_2 n)$. ¿En qué casos conviene este tipo de estructuras?

7.4. a. Usar el método de *Ford* para determinar el camino mínimo entre el vértice I y los demás en el siguiente grafo:



b. Calcular el camino más corto entre todos los pares de vértices del grafo de la parte a. usando el método de *Floyd*.

c. Idem b. usando el método de *Dantzig*. Comparar con b.

d. Idem 7.1c para *Ford*, *Floyd* y *Dantzig*.

7.5. a. ¿Cómo se puede modificar el algoritmo de *Ford* para usarlo para detectar circuitos de longitud negativa? ¿Qué pasa si no todos los vértices son alcanzables desde el vértice 1?

b. Mostrar que el algoritmo de *Ford* requiere en el peor caso $O(mn)$ comparaciones.

7.6. a. ¿Cómo influye en el algoritmo de *Floyd* la manera en qué se hayan numerado los vértices del grafo? ¿Y en *Dantzig*?

b. Estimar el número de operaciones necesarias para ejecutar el algoritmo de *Floyd*.

c. idem b. para *Dantzig*.

d. ¿Cómo puede usarse el algoritmo de *Floyd* para detectar la existencia de circuitos de longitud negativa? ¿Y el de *Dantzig*?

e. ¿Se pueden aplicar los algoritmos de *Floyd* y *Dantzig* a grafos no orientados?

7.7. Explicar porqué el algoritmo de *Floyd* es un algoritmo que usa la técnica de programación dinámica.

7.8. a. Escribir un programa para el algoritmo de *Dijkstra*. Utilizar las estructuras vistas para representar grafos (además de matriz de adyacencia) para implementarlo.

b. idem a. para *Ford*, *Floyd* y *Dantzig*.

7.9. ***** Si se quiere determinar el camino más corto de el nodo s al nodo t, usando *Dijkstra*, se puede parar cuando se ha rotulado en forma definitivia el nodo t. Veremos aquí una modificación del algoritmo que hace que hace que t sea rotulado más rápidamente en la práctica.

Sea $h(i) \geq 0$ una cota inferior de la longitud del camino más corto de un nodo i al nodo t, que satisface $h(i) \leq h(j) + c_{ij}$ para todo eje (i, j). Por ejemplo si los nodos son puntos en el plano se puede tomar h como $h(i) = \sqrt{(x_i - x_t)^2 + (y_i - y_t)^2}$, es decir la distancia euclideana entre i y t. En este caso $h(i)$ es una cota inferior a la longitud del camino más corto entre i y t.

a. Sea $c_{ijh} = c_{ij} + h(j) - h(i)$ para todo (i, j). Mostrar que con estas nuevas longitudes los caminos mínimos entre dos nodos no cambian.

b. Si aplicamos *Dijkstra* con c_{ijh} como longitudes de los arcos, ¿por qué esta modificación (llamada algoritmo A^*) mejora el comportamiento empírico del algoritmo?

c. ¿Qué pasa si la función h no cumple la condición de que $h(i) \leq h(j) + c_{ij}$? ¿Puede usarse el algoritmo A^*?

7.10. Beba tiene un departamento de vacaciones en Punta del Este, que quiere alquilar en el período que va del 1/12/95 al 31/3/96, por periodos cortos. Ha recibido numerosas ofertas, en las que figuran el día que la persona ingresa al departamento (después de las 15hs), el día que se va (antes de las 12hs) y el monto ofrecido del alquiler. Modelar el problema de elegir las ofertas que maximicen el beneficio de Beba como un problema de camino mínimo.

7.11. La siguiente tabla muestra algunos posibles horarios de guardia para los choferes de una compañía de ómnibus. Se quiere asegurar, al menor costo posible, que al menos un chofer esté de guardia en el período de 9 a 17hs. Modelar y resolver este problema como un problema de camino mínimo.

Horario	9-13	9-11	12-15	12-17	14-17	12-16	16-17
Costo	30	18	21	38	20	22	9

7.12. Modelar el problema de la mochila, para el caso donde se pueden llevar varios elementos iguales, como un problema de camino mínimo.

7.13. ***** Una compañía de construcciones necesita las siguientes cantidades de personal calificado para una tarea especial en los meses de marzo a agosto:

Mes	Marzo	Abril	Mayo	Junio	Julio	Agosto
Personal	4	6	7	4	6	2

El personal debe ser contratado por mes. Si en febrero había 3 personas y en setiembre deben quedar 3 también, el problema que se quiere resolver es, determinar cuántos trabajadores deben contratarse o despedirse en cada mes, de modo de minimizar el costo, sabiendo que:

- Emplear un trabajador nuevo cuesta 100$ por trabajador y mandarlo a otra obra 160$.
- No se pueden contratar más de 3 trabajadores por mes, y no se pueden trasladar más de un tercio de los trabajadores de la obra al final de cada mes.
- El costo de tener un obrero de más es de 200$ por persona y el de tener uno menos también, porque se deben pagar horas extra. No se pueden pagar más del 25 % de horas extra, sobre el horario regular.

Formular este problema como un problema de camino mínimo y resolverlo usando programación dinámica.

7.14. Volvamos al problema del práctico 3 de dar el cambio con monedas. En su forma más general podemos decir que el problema del cambio de dinero consiste en determinar si es posible cambiar un número p dado en monedas de denominaciones conocidas $a_1, \ldots, a_k$. Por ejemplo, si $k = 3, a_1 = 3, a_2 = 5, a_3 = 7$, podremos cambiar los números 8, 54, etc., pero no el número 4.

En general, este problema consiste en responder si $p = \sum_{i=1}^{k} a_i x_i$, para $x_i \geq 0, i = 1, \ldots, k$.

Modelar el problema como un problema de camino mínimo.

a. Describir un método para identificar todos los números en un rango dado $[l, u]$ que pueden ser cambiados.

b. Describir un método para decidir si un número p puede ser cambiado, y luego identificar las denominaciones de las monedas, tal que sean la menor cantidad posible.

7.15. Dada la red *PERT* de un proyecto se calcula por el método *CPM*:

- t_i = fecha temprana en la cual una etapa o evento i puede ser completado.
- l_i = fecha tardía o última en la cual puede completarse una etapa i sin atrasar todo el proyecto.

a. Deducir a partir de t_i y l_i fórmulas para calcular para cualquier actividad $i \to j$ la fecha más temprana y tardía (tc_{ij} y lc_{ij}) de comienzo de la misma y (tf_{ij} y lf_{ij}) de terminación.

b. Se define el margen libre de la actividad $i \to j$ como $t_j - t_i - d_{ij}$, y el margen independiente como $t_j - l_i - d_{ij}$. Interpretar el significado de estos márgenes.

7.16. Se quieren planificar las actividades necesarias para la fabricación de una máquina grande que se va fabricando por partes según se describe en el siguiente cuadro:

nombre	*descripción*	*duración*	*predecesores*
A	buscar elementos para parte 1	5	
B	buscar elementos para parte 2	3	
C	buscar elementos para parte 3	10	
D	armar parte 1	7	A
E	armar parte 2	10	B
F	armar parte 4	5	D,E
G	armar parte 3	9	B,C
H	armado final	4	F,G
I	inspección final y prueba	2	H

a. Armar el grafo *PERT* (actividades en los arcos) correspondiente a este problema.

b. Construir una tabla donde para cada actividades aparezcan los valores tc_{ij}, lc_{ij}, tf_{ij} y lf_{ij} y los márgenes total, libre e independiente. Indicar qué actividades están en el camino crítico. ¿Cuál es el tiempo mínimo necesario para completar la fabricación?

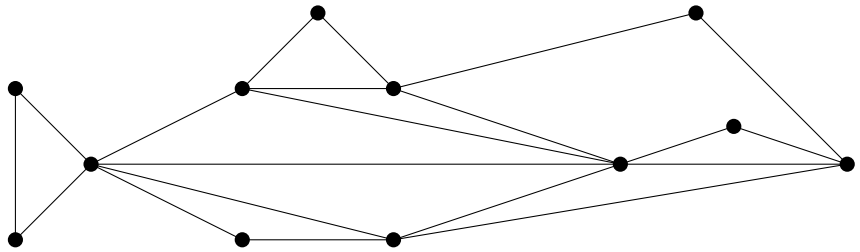
c. Construir el grafo de actividades en los nodos y repetir b. a partir del mismo.

Práctica 8

Referencias para este práctico: **[5]**, **[4]**, [12]

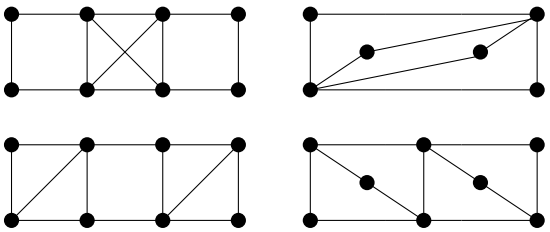
8.1. a. Encontrar un circuito euleriano en el grafo de la figura.

b. Encontrar una partición en circuitos simples de las aristas del grafo de la figura.



8.2. Analizar nuevamente el ejercicio 4.7.

8.3. Determinar si los grafos de la figura tienen circuitos eulerianos. Sacar conclusiones.



8.4. Probar que un grafo G conexo tiene un circuito euleriano si y sólo si el conjunto de aristas de G se puede particionar en circuitos simples de modo que cada eje pertenezca a uno y sólo uno de dichos circuitos.

8.5. Llamemos $w(G)$ al número de componentes conexas de G. Mostrar que si G es conexo y todos sus vértices tienen grado par, entonces para cualquier vértice v se cumple que $w(G - v) \leq \frac{d(v)}{2}$.

8.6. Dar condiciones bajo las cuales un grafo tiene un camino euleriano pero no un circuito euleriano. Demostrar.

8.7. Un grafo se dice arbitrariamente atravesable desde un vértice v_0 si con el siguiente procedimiento siempre se construye un circuito euleriano:

```
salir de v_0 por cualquier eje incidente
mientras sea posible hacer:
     al llegar a un vértice u, salir por un eje no utilizado
fin
```

a. Probar que un grafo que tiene un circuito euleriano es arbitrariamente atravesable desde un vértice v_0 si y sólo si todo ciclo simple contiene a v_0.

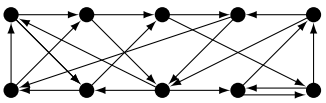
b. Probar que si G es arbitrariamente atravesable desde v_0 entonces v_0 tiene grado máximo.

8.8. a. ¿Para qué valores de n, K_n tiene circuito euleriano?

b. ¿Hay algún K_n que tenga camino euleriano pero no circuito?

8.9. a. Probar que un digrafo conexo tiene un circuito euleriano orientado si y sólo si $d_{in}(v) = d_{out}(v)$ para todo vértice v del digrafo.

b. Encontrar un circuito euleriano orientado en el digrafo de la figura.



c. Analizar el concepto de camino euleriano orientado para un digrafo.

8.10. Probar que todo digrafo conexo que tiene un circuito euleriano orientado es fuertemente conexo. ¿Es cierta la recíproca?

8.11. Dada la demostración del teorema de *Euler*:

a. Escribir una función que determine si todos los vértices son de grado par.

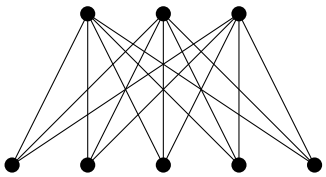
b. Escribir un programa que dado un grafo, verifique si cumple con las condiciones del teorema de *Euler* (utilizar la función definida en a.) y genere un circuito.

c. Analizar estructuras de datos alternativas para implementar.

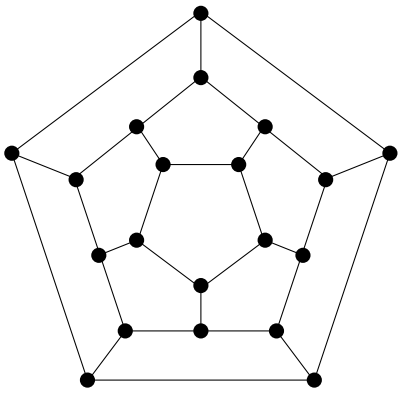
d. Calcular la complejidad de a. y b.

e. Mostrar que se puede realizar una implementación de b. con complejidad $O(m)$.

8.12. Decir por qué el grafo de la figura no tiene camino ni circuito hamiltoniano.



8.13. El siguiente es el grafo original en el cual Hamilton basó su juego (ver [13]). Encontrar un circuito hamiltoniano en el mismo. Una versión del juego original consistía en que uno de los jugadores elegía un camino con 5 vértices y el otro debía extender el camino a un circuito hamiltoniano. ¿Hay algún camino simple de 5 vértices que no pueda ser extendido a un circuito hamiltoniano?



8.14. Un digrafo G se dice completamente conexo si cada par de vértices está conectado con exactamente un eje orientado en una de las dos posibles direcciones. Probar que si un digrafo G es completamente conexo entonces tiene un camino hamiltoniano orientado.

8.15. Probar que un grafo bipartito con un número impar de vértices no contiene un circuito hamiltoniano.

8.16. Si G es un grafo con $n \geq 4$ y d (grado mínimo) $\geq n-2$ entonces G tiene un circuito hamiltoniano.

8.17. Sea G un grafo y u y v dos nodos no adyacentes tal que $d(u) + d(v) \geq n$. Probar que G es hamiltoniano sii $G+$ el eje uv es hamiltoniano.

8.18. ***** Si 15 profesores cenaron juntos en una mesa circular durante una conferencia, y si cada noche cada profesor se sentó entre dos profesores distintos, ¿cuántos días duró como máximo la conferencia?

8.19. Mostrar directamente sin utilizar ningún teorema ya demostrado en clase que cualquier grafo simple de 6 vértices, todos de grado 3, tiene un circuito hamiltoniano.
Sugerencia: Pensar cómo es el complemento del grafo.

8.20. Analizar la siguiente modificación al algoritmo DFS para determinar un circuito hamiltoniano en un grafo:

Inicializar
Entrar un grafo G con vértices $\{1, 2, \ldots, n\}$
poner $path(1) \leftarrow 1$, $path(i) \leftarrow 0$ para $i = 2, \ldots, n+1$
$j \leftarrow 2$
$forward \leftarrow$ **true**
mientras $j < n+1$ **hacer**
  si $forward =$ **true entonces**
    Build
  sino
    $path(j) \leftarrow 0$
    $j \leftarrow j - 1$
    si $j \neq 1$ **entonces**
      Build
    sino
      devolver que no hay circuito hamiltoniano y parar
    fin si
  fin si
  si $j = n+1$ **entonces**
    si $(1, path(n)) \in X$ **entonces**
      $path(j) \leftarrow 1$
    sino
      $forward \leftarrow$ **false**
      $j \leftarrow j - 1$
    fin si
  fin si
fin mientras
devolver path y parar.

procedimiento *Build*
Encontrar el menor $k > path(j)$ tal que $k \neq path(i)$ para $i < j$ y $(path(j-1), k) \in X$.
si no existe tal k **entonces**
  $forward \leftarrow$ **false**
sino
  $path(j) \leftarrow k$
  $j \leftarrow j + 1$
  $forward \leftarrow true$

fin si

a. Mostrar que el conjunto de aristas que brinda el algoritmo principal cuando devuelve $path()$ es un circuito hamiltoniano.

b. Mostrar que si el algoritmo no encuentra un circuito hamiltoniano entonces G no es hamiltoniano.

c. Determinar la complejidad de este algoritmo. ¿Cuál era la complejidad de DFS? ¿Qué técnica se ha usado en este modificación de DFS? Comentar.

d. Encontrar una familia de grafos de n vértices tal que el algoritmo cree más de $(n-3)!$ caminos ninguno de los cuales pueda extenderse a un circuito hamiltoniano.

8.21. Basándose en la demostración del teorema que dice que dado un grafo G si para todo vértice v se verifica que $d(v) \geq \frac{n}{2}$ entonces el grafo tiene circuito hamiltoniano.

a. Escribir una función que dado un grafo G determine si en él se cumplen las hipótesis del teorema.

b. Escribir un programa que dado un grafo G, verifique si se cumplen las condiciones del teorema y si es así, que genere un circuito hamiltoniano en G.

c. Analizar estructuras de datos alternativas para implementar.

d. Calcular la complejidad de los algoritmos de a. y b.

e. Decir en que punto falla el algoritmo cuando se lo aplica a un grafo que no cumple las hipótesis del teorema. Dar ejemplos.

8.22. Dado un grafo $G = (V, X)$ donde a cada eje $e \in X$ se le ha asignado una longitud $l(e)$. El problema conocido como problema del viajante de comercio consiste en determinar el circuito hamiltoniano de longitud total mínima. Suponiendo que G sea un grafo completo construir un algoritmo que genere todos los posibles circuitos hamiltonianos de G y determine uno de longitud mínima. Analizar la complejidad de dicho algoritmo.

8.23. Analizar el siguiente algoritmo para resolver el problema del viajante en forma aproximada:

PASO 1: Dar un grafo completo G con longitudes asignadas a las aristas.

PASO 2: Determinar un árbol generador mínimo T de G.

PASO 3: Construir un grafo orientado $D(T)$ reemplazando cada eje de T por dos aristas orientadas en direcciones opuestas.

PASO 4: Usar el algoritmo de *Euler* modificado para grafos orientados para determinar un circuito Euleriano en $D(T)$. Suponer que el circuito queda formado por las aristas
$\{(x_1, y_1), (x_2, y_2), \ldots, (x_{2n-2}, y_{2n-2})\}$

PASO 5: Poner $C \leftarrow \{x_1\}$ y marcar x_1.

PASO 6: Para $i \leftarrow 1, 2n-3$ hacer

PASO 7: Si y_i no está marcado, marcarlo y poner $C \leftarrow C \cup \{y_i\}$.

PASO 8: Poner $C \leftarrow C \cup \{x_1\}$.

PASO 9: Informar C y parar.

a. Mostrar un ejemplo en que el resultado del algoritmo anterior no sea un circuito hamiltoniano de longitud mínima.

b. Analizar la complejidad del algoritmo.

c. Probar que si $l(C)$ es la longitud del circuito determinado por el algoritmo y $l(H)$ es la longitud de un circuito hamiltoniano de longitud mínima y además en el grafo se cumple que vale la desigualdad triangular para las distancias, entonces $l(C) \leq 2\ l(H)$. Comentar este resultado.

8.24. Analizar el siguiente algoritmo goloso (método del vecino más cercano) para resolver el problema del viajante de comercio en un grafo completo. Elegir v_1 un vértice cualquiera.. Poner $C_1 = \{v_1\}$. Mientras C_j no contenga todos los vértices de V, elegir v_{j+1} como el vértice más cercano a v_j y poner $C_{j+1} = C_j \cup \{v_{j+1}\}$. Agregar el eje entre v_n y v_1.

a. Mostrar que este no es un algoritmo exacto para resolver el porblema del viajante de comercio. Determinar su complejidad.

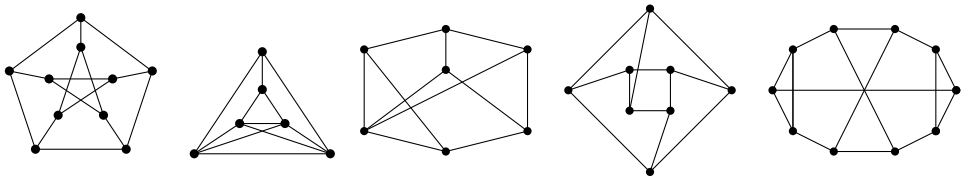
b. ¿Es una buena heurística?

c. Escribir un nuevo algoritmo para el mismo problema, que elija en primer lugar el nodo más lejano a v_1, y después vaya eligiendo e insertando en cada paso el nodo que aún no está en el camino ya construido C_j y que está más cerca de algún nodo de C_j. ¿Es este algoritmo exacto? ¿Es bueno? ¿Es mejor que al anterior? ¿Peor?

Práctica 9

Referencias para este práctico: **[4]**, **[22]**, **[28]**, [17], [9], [5]

9.1. Decir cuáles de los siguientes grafos son planares. Justificar.



9.2. Probar que si T es un árbol, T es planar.

9.3. a. Modificar la fórmula de *Euler* para que sirva para un grafo con k componentes conexas.

b. Mostrar que la fórmula $m \leq 3n - 6$ es válida también para grafos planares no conexos.

9.4. Probar que un grafo planar donde todo vértice tiene grado mayor o igual a 3, se verifica que $m \leq 3r - 6$.

9.5. a. Probar que todo grafo planar tiene por lo menos un vértice de grado menor o igual a 5.

b. Probar que todo grafo planar con menos de 12 vértices tiene un vértice de grado a lo sumo 4.

c. Probar que si G tiene 11 o más vértices, entonces G o su complemento son no planares.

9.6. ¿Es cierto que todo grafo no planar es homeomorfo a K_5 o $K_{3,3}$?

9.7. Probar que si G es un grafo planar con circuitos, todos ellos de longitud mayor o igual a k, la desigualdad $m \leq 3n - 6$ puede ser mejorada a $m \leq \dfrac{(n-2) \times k}{k-2}$.

9.8. Supongamos que se dibujan l cuerdas de un círculo que se cortan en p puntos interiores (se supone que en cada intersección se cortan solamente 2 líneas). ¿Cuántas regiones se forman de este modo en el interior del círculo?.

9.9. ***** Un grafo se llama platónico si es conexo, planar, todos sus vértices tienen el mismo grado d_1 y todas sus regiones tienen el mismo número d_2 de aristas frontera (con $d_1 \geq 3$ y $d_2 \geq 3$). Probar que si G es un grafo platónico:

a. $m = \dfrac{d_1 \times n}{2}$ y $r = \dfrac{d_1}{d_2} \times n$

b. $n \times (2d_1 + 2d_2 - d_1 \times d_2) = 4d_2$

c. como n y $4d_2$ son enteros positivos, resulta que: $2d_1 + 2d_2 - d_1 \times d_2 > 0$. Usar esto para probar que $(d_1 - 2)(d_2 - 2) < 4$.

d. Encontrar los 5 valores de d_1 y d_2 que verifican esta última desigualdad y dibujar los grafos plátonicos correspondientes.

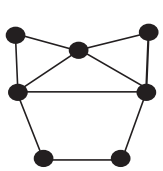
9.10. a. Aplicar el algoritmo para decidir si un grafo es planar de *Demoucron*, *Malgrange* y *Pertuiset*, a los grafos del ejercicio 9.1.

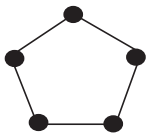
b. Escribir detalladamente el algoritmo de *Demoucron*, y mencionar las estructuras que usaría para implementarlo. Determinar su complejidad. Usar la relacion entre aristas y vértices que tiene un grafo planar.

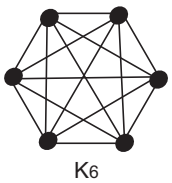
Práctica 10

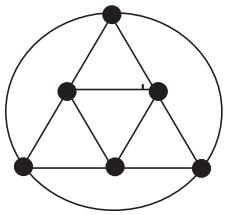
Referencias para este práctico: **[4]**, **[22]**, **[28]**, [17], [9], [5]

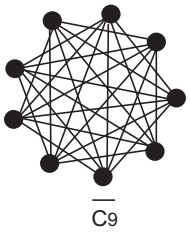
10.1. *a*) Calcular el número cromático de los siguientes grafos:

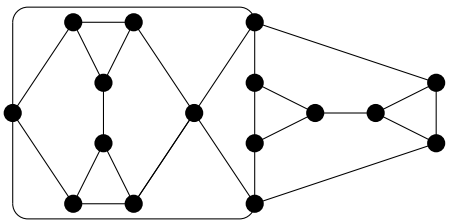












10.2. Se tiene una lista de tareas a desarrollar. Existen tareas que utilizan el mismo recurso, y cada tarea demora una hora. Dado que los recursos son limitados (se dispone de una unidad por recurso), es imposible que dos tareas que comparten algún recurso, se desarrollen al mismo tiempo. Determinar cuántas horas son necesarias como mínimo para cumplir con todas las tareas. Modelar el problema como un problema de coloreo.

10.3. Decidir si las afirmaciones siguientes son V o F. Justificar:

a) Si un grafo G contiene al grafo completo K_s como subgrafo inducido entonces $\chi(G) \geq s$.

b) Si todos los subgrafos completos de G tienen tamaño menor o igual que s, entonces $\chi(G) \leq s$.

c) El problema de encontrar el número cromático de un grafo G es equivalente al problema de encontrar el subgrafo completo máximo de G.

d) Si $\chi(G) \leq s$ entonces el grado de todo vértice de G es menor o igual que $s - 1$.

10.4. Se va a realizar un congreso en el cual se dictarán varios cursos. Se tiene la lista de inscriptos y en base a ella se quiere armar el cronograma de horarios de tal manera que todos los participantes puedan asistir a los cursos que se anotaron, es decir, no se superpongan los horarios de cursos que serán asistidos por una misma persona. Modelar este problema como un problema de coloreo.

10.5. Se quieren usar torres existentes para transmisión de ondas para celular. El problema es que dos torres que están muy cerca pueden interferirse, si se utiliza la misma frecuencia para ambas. ¿Cuál es el menor número de frecuencias diferentes que hace falta usar en este caso? Modelar este problema como un problema de coloreo y resolver para la instancia dada por la matriz de distancias entre las torres y la distancia mínima para usar la misma frecuencia es 50 km..

	B	C	D	E	F	G	H
A	170	140	28	189	25	186	67
B		100	46	70	26	2	214
C			0	56	110	34	16
D				18	30	90	13
E					134	48	53
F						14	55
G							50

10.6. Un grafo G es color crítico si sacando un vértice cualquiera disminuye el número cromático de G.

a) Para cada grafo del ejercicio 10.1, decidir si, al sacar cualquier vértice de los grafos , disminuye el número cromático.

b) Probar que todo grafo k-cromático ($k \geq 2$) contiene un subgrafo k-cromático color crítico.

10.7. Probar que si G es k-cromático y color crítico entonces:

a) G es conexo.

b) Todo vértice de G tiene grado mayor o igual a $k-1$.

c) G no tiene un vértice tal que al sacarlo, G quede un grafo disconexo.

d) G no contiene un conjunto de corte que sea un subgrafo completo de G.

10.8. El grafo junta de los grafos G y H, denotado $H+G$, se define agregando un eje entre cada vértice de H y cada vértice de G. Probar que $\chi(G+H) = \chi(G) + \chi(H)$. Concluir que el grafo junta de dos grafos color críticos es color crítico.

10.9. Probar que para cualquier conjunto independiente S de un grafo color crítico G se cumple que $\chi(G-S) = \chi(G) - 1$.

10.10. Sea G un grafo

a) Probar que G es bipartito si y sólo si $\chi(G) = 2$ (suponiendo que tiene al menos una arista).

b) Enunciar un algoritmo para determinar si un grafo es 2-coloreable.

c) Determinar la complejidad del algoritmo dado en *b*).

10.11. Sea G un grafo de n vértices, $C_{max}(G)$ el tamaño del subgrafo completo máximo y I_{max} el tamaño del conjunto independiente máximo. Probar que $C_{max}(G) + I_{max} \leq \chi(G) + \chi(\overline{G})$.

10.12. Probar que si G es un grafo de n vértices y $\overline{G}$ es su complemento, entonces:

a) $\chi(G) + \chi(\overline{G}) \leq n+1$

b) * $n \leq \chi(G) \times \chi(\overline{G}) \leq \left(\frac{n+1}{2}\right)^2$

c) * $\chi(G) + \chi(\overline{G}) \geq 2\sqrt{n}$

10.13. Sea G un grafo regular de n vértices. Probar que si $\chi(G) + \chi(\overline{G}) = n+1$ entonces G es uno de los siguientes grafos:

a) el grafo formado por n vértices aislados

b) K_n

c) un circuito simple de longitud impar.

10.14. Analizar el siguiente algoritmo secuencial para colorear un grafo:

1. Dado un grafo $G = (V, X)$ con $V = \{v_1, v_2, \ldots, v_n\}$, poner $f(v_1) = 1$.

2. Para $i = 2, n$ poner $f(v_i) = \text{mín}\{k/k \geq 1$ y $f(v_j) \neq k$ para $(1 \leq j < i)$ y v_j adyacente a $v_i\}$

3. Parar.

a) Determinar la complejidad del algoritmo de a.

b) Decidir si este algoritmo siempre calcula el número cromático de un grafo. ¿ De qué depende que esto suceda ?

c) Dar ejemplos donde el algoritmo no encuentre número cromático, si existe.

d) Demostrar que para todo grafo G existe un orden de rotulado de los vértices para el cuál el algortimo de a. determina el número cromático de G.

e) Deducir que si se corre el algoritmo de a. para todas las posibles permutaciones de los rótulos de G se obtiene un algoritmo exacto para determinar el número cromático de un grafo G. ¿Cuál es la complejidad de este algoritmo?.

10.15. *****

a) Probar que si $v_1, \ldots v_n$ es un ordenamiento de los vértices de V, $\chi(G) \leq \max_i\{\text{mín}\{d_i + 1, i\}\}$.

b) ¿Hay algún caso en que valga la igualdad en la fórmula de a.?

c) Probar que $US = \text{máx}\,\text{mín}\{i, d(v_i) + 1\}$ es una cota superior del número de colores usado por el algoritmo secuencial con ese ordenamiento. ¿Es esta una buena cota? ¿Hay algún caso en que la heurística use realmente este número de colores?

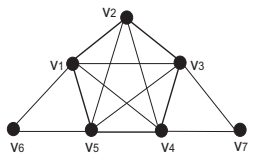
d) Supongamos que en el algoritmo secuencial se ordenan previamente los vértices de V de modo que quede $d(v_1) \geq d(v_2) \geq \ldots \geq d(v_n)$. Llamamos a este algoritmo, algoritmo secuencial "largest first" SLF y llamamos ULF a la cota superior del número de colores usados por el algoritmo en este caso. Mostrar que se verifica que ULF $\leq$ US.

e) ¿Se puede concluir de b. que el algoritmo SLF es siempre mejor que el algoritmo secuencial con otros ordenamientos de los vértices? ¿Se puede obtener una buena cota para la diferencia entre el número de colores usado por ULF y el número cromático del grafo?

10.16. *a*) Escribir un algoritmo exácto para determinar el número cromático usando la técnica de backtracking

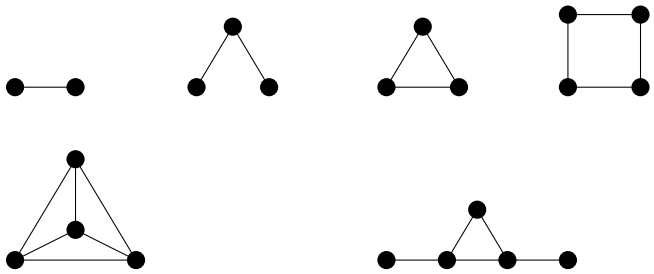
b) Probar que el algoritmo dado en a. es correcto y determinar la complejidad.

10.17. Hallar el número cromático de G. De cuántas formas distintas se puede colorear G si se disponen de 19 colores?



10.18. El polinomio cromático $P_G(k)$ de un grafo rotulado G es un polinomio en k, tal que para cada k, $P_G(k)$ es la cantidad de maneras distintas de colorear el grafo con k colores.

a) Calcular el polinomio cromático de cada uno de los siguientes grafos:



K_n

b) ¿Cómo se puede determinar el número cromático de un grafo a partir de su polinomio cromático?

c) Dar una cota inferior para el grado del polinomio cromático de un grafo.

10.19. *a*) Escribir un algoritmo para calcular el polinomio cromático usando un árbol donde los "hijos" de un nodo se construyen poniendo un eje entre dos nodos no adyacentes (si usan distinto color) o "pegándolos" (si usan el mismo color).

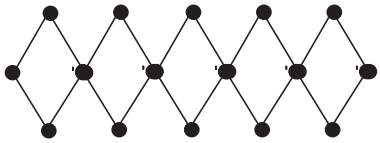
b) Explicar por qué la función $P_G(k)$ es efectivamente un polinomio en k.

c) Determinar la complejidad del algoritmo de a.

10.20. Hallar el polinomio cromático de:

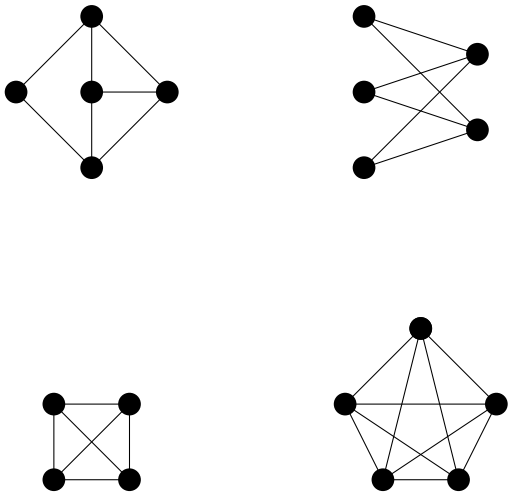
a) Los ciclos de n nodos

b) Los grafos G_n, donde G_n es un grafo compuesto de n ciclos de 4 vértices dispuestos en fila unidos por un vértice.



Gn

10.21. Calcular el índice cromático de los siguientes grafos:



10.22. (Teorema de Vizing) Sea $\delta(G)$ el grado máximo de G. Dar ejemplos de grafos, donde el índice cromático sea igual a $\chi'(G) = \delta(G)$ y grafos donde $\chi'(G) = \delta(G) + 1$.

10.23. Supongamos que 11 equipos deben jugar entre si. Suponiendo que un equipo no puede jugar más de un partido por día, cuantos días son necesarios? Modelar este problema cono un problema de coloreo de aristas y como un problema de coloreo de vértices.

10.24. Dado un grafo G, se define el grafo linea $L(G)$ como el grafo que tiene un vértice por cada arista de G, y dos vértices de $L(G)$ son adyacentes si y solo si, las aristas correspondientes a esos vértices se intersecan en G. Probar que $\chi'(G) = \chi(L(G))$.

10.25. Sea $m(G)$ el tamaño máximo de un matching en G.

a) Demostrar que $\chi'(G)\, m(G) \geq \mid E(G) \mid$.

b) Usando el ejercicio anterior, hallar una cota inferior para el número cromático de un grafo linea $L(G)$.

Práctica 11

Referencias para este práctico: **[1]**, **[4]**, [17], [19]

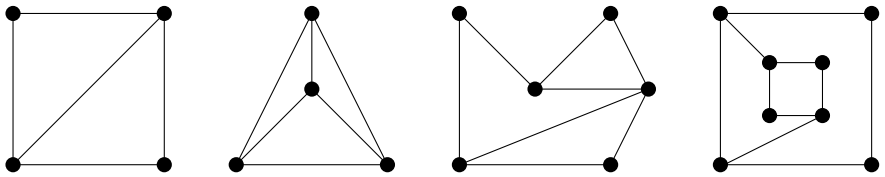
11.1. Para cada grafo de la figura, encontrar:

a. las correspondencias maximales y cuáles de ellas son máximas.

b. los recubrimientos minimales de los vértices y cuáles son mínimos.

c. los conjuntos independientes maximales y cuáles son máximos.

d. los recubrimientos minimales de las aristas y cuáles son mínimos.



11.2. Decidir si es verdadera o falsa cada una de las siguientes afirmaciones. Si es verdadera probarla y si es falso encontrar un contraejemplo.

a. Todo grafo tiene al menos una correspondencia (matching).

b. Todo grafo tiene al menos un recubrimiento de los vértices.

c. Todo grafo tiene al menos un conjunto independiente.

d. Todo grafo tiene al menos un recubrimiento de las aristas.

e. En todo grafo, cualquier recubrimiento de vértices tiene un número de aristas mayor o igual que el de cualquier correspondencia.

f. En todo grafo, cualquier recubrimiento de aristas tiene una cantidad mayor o igual de vértices que cualquier conjunto independiente.

g. Cualquier recubrimiento de vértices contiene una correspondencia máxima.

11.3. Probar que:

a. Un grafo tiene un recubrimiento de los vértices si y sólo si no tiene vértices aislados.

b. Si G tiene n ($n > 1$) vértices, cualquier recubrimiento de los vértices de G tiene al menos $\lceil n/2 \rceil$ aristas.

c. Todo recubrimiento contiene un recubrimiento minimal.

d. Un recubrimiento minimal no contiene circuitos.

e. Cuál es el máximo número de aristas que puede tener un recubrimiento minimal de los vértices?

f. Un recubrimiento de los vértices de un grafo es minimal si y sólo si no contiene caminos ni circuitos simples de longitud mayor o igual a 3.

11.4. Probar que si G es un grafo sin vértices aislados, se verifica que $\alpha \leq \beta$ donde α es el cardinal del conjunto independiente máximo y β es el cardinal del recubrimiento de vértices mínimo.

11.5. Probar que en un grafo bipartito G, $m \leq \alpha\beta$, donde m es la cantidad de aristas, α es el cardinal del conjunto independiente máximo de G y β es el cardinal del recubrimiento mínimo de aristas de G.

11.6. Revisar los ejercicios de modelización del práctico 4. ¿Cuáles se pueden modelizar como problemas de matching, recubrimiento o de conjunto independiente? ¿Cómo se los puede resolver?

11.7. Analizar el siguiente algoritmo para determinar un conjunto independiente en un grafo G:

`PASO 1:` Poner $I \leftarrow \emptyset$

`PASO 2:` Mientras $V \neq \emptyset$ hacer

`PASO 3:` Encontrar un vértice x tal que $d(x, G)$ sea mínimo.

`PASO 4:` Poner $I \leftarrow I \cup \{x\}$

`PASO 5:` Poner $V \leftarrow V \backslash (\{x\} \cup \{y : y \text{ adyacente a } x\})$.

`PASO 6:` Poner $G \leftarrow (V, X(V))$.

`PASO 7:` Informar I

a. Probar que el algoritmo anterior encuentra un conjunto independiente en G.

b. Determinar la complejidad del mismo. ¿Qué técnica utiliza?

c. Mostrar que el algoritmo no necesariamente encuentra un conjunto independiente máximo.

d. Dado que no se conocen algoritmos de complejidad polinomial para el problema de determinar un conjunto independiente maximo en un grafo, se podría usar este algoritmo como algoritmo aproximado para resolver el problema? ¿Sería un "buen" algoritmo aproximado?

11.8. Analizar el siguiente algorimo para colorear un grafo:

`PASO 1:` $U \leftarrow V$, $k \leftarrow 0$

`PASO 2:` Mientras $U \neq \emptyset$ hacer

`PASO 3:` Poner $k \leftarrow k + 1$

`PASO 4:` determinar un conjunto independiente máximo W en el subgrafo de G generado por U.

`PASO 5:` Asignar a los nodos de W el color k.

`PASO 6:` Poner $U \leftarrow U \backslash W$.

`PASO 7:` Informar k, que es el número de colores con que se ha coloreado el grafo.

a. Probar que el algoritmo produce un coloreo válido del grafo. ¿Calcula el número cromático de G?

b. ¿Cuál es la complejidad del algoritmo?

c. Si el problema de encontrar un conjunto independiente máximo pudiera ser resuelto en forma polinomial, ¿esto implicaría, a partir del algoritmo anterior que lo mismo ocurriría para el problema de coloreo de grafos?

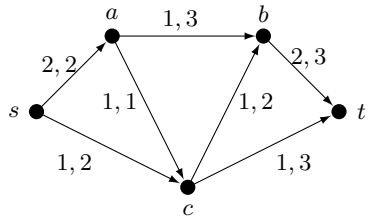
d. Probar que reemplazando el `PASO 4`, por el algoritmo del ejercicio 11.7, se obtiene una heurística para el problema de determinar el numero cromático de un grafo, de complejidad polinomial. ¿Es una "buena" heurística?

11.9. Dado un grafo bipartito $G = (V, X)$ con $V = V_1 \cup V_2$, o sea todos las aristas del grafo sean la forma (v, w), con $v \in V_1$ y $w \in V_2$, se dice que hay una correspondencia completa C de los vértices de V_1 en los vértices de V_2 si existe una correspondencia C tal que todo vértice de V_1 es incidente a un arco de C.

a. Dar ejemplos de grafos bipartitos que tengan y no tengan una correspondencia completa de V_1 en V_2.

b. Probar que G tiene una correspondencia completa de V_1 en V_2 si y sólo si todo subconjunto de r vértices de V_1 es adyacente a r o más vértices de V_2, para todo r positivo.

11.10. Dada la red de la figura donde en cada arco figura la capacidad y el valor de un flujo dado:

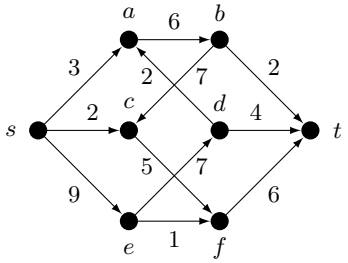


a. Determinar todos los posibles caminos de aumento.

b. Construir a partir del flujo dado, el flujo máximo de s a t.

c. Encontrar un corte mínimo que separe a s de t. ¿Qué ocurre con el valor del flujo calculado en b. en los arcos del corte mínimo?

11.11. Encontrar el flujo máximo que puede ir de s a t en la siguiente red:



11.12. a. Construir una red de 4 nodos en la que el método de *Ford y Fulkerson* necesita F iteraciones para obtener el flujo de valor máximo (partiendo de un flujo inicial con valor 0).

b. ¿Cuántas comparaciones se necesitan dado el número de aristas de una red para cada iteración del método de *Ford y Fulkerson*?

11.13. ***** Mostrar que el algoritmo de *Edmonds y Karp* (modificación del algoritmo original de *Ford y Fulkerson*) tiene complejidad $O(nm^2)$.

11.14. Decir si son correctas o no las siguientes afirmaciones. Justificar.

a. Si todos los arcos de una red tienen distintas capacidades existe un único corte de capacidad mínima.

b. Si todos los arcos en una red tienen distintas capacidades existe un único conjunto de arcos por el cual puede pasar un flujo de valor máximo.

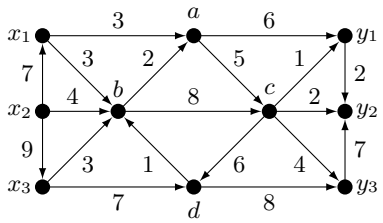
11.15. ¿Cómo se puede hacer para encontrar el flujo mínimo en una red con fuente s y sumidero t, donde existen cotas inferiores para el flujo que debe pasar por cada arcos?

11.16. ¿Cómo se puede modificar el método de *Ford y Fulkerson*, cuando se usan redes no orientadas para evitar que se pueda tener simultáneamente $f(v_i \to v_j) > 0$ y $f(v_j \to v_i) > 0$?

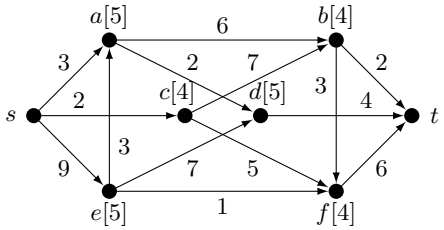
11.17. Un agente de viaje debe arreglar el viaje de 10 turistas de Buenos Aires a Viena en una fecha dada sabiendo que hay 7 lugares libre de Buenos Aires a Río, 4 de Río a París, 8 de parís a Viena, 2 de Buenos Aires a Viena, 5 de Buenos Aires a Madrid, 3 de Madrid a París y 2 de Madrid a Viena. ¿ Puede organizar el viaje? ¿ Cómo?

11.18. Supongamos que hay 5 comisiones formadas, de las cuales cada una desea enviar un representante distinto a una reunión. Los miembros de A son a, b y c, los miembros de B son b y c, los miembros de C son a, b y d, los de D son d, e, f y los de E son e y f. Transformar el problema en un problema de flujo máximo y resolverlo.

11.19. En la siguiente red x_1, x_2, x_3 son las fuentes de algún elemento. Se dispone de 5 unidades en x_1, 10 en x_2 y 5 en x_3. Se requieren 5 unidades en y_1, 10 en y_2, y 5 en y_3 y se desea saber si se pueden enviar a través de la red. (sugerencia: agregar una fuente s y un sumidero t artificiales a la red).



11.20. En la siguiente red además de las capacidades de los arcos, los vértices distintos de s y t tienen una cota superior de flujo que pueden pasar a través de ellos. Encontrar el flujo máximo en este caso. (Sugerencia: reemplazar cada vértice por dos vértices nuevos).



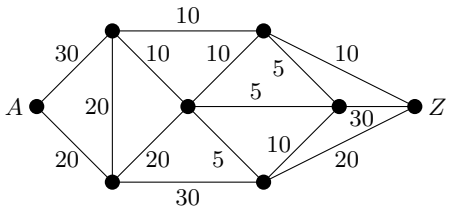
11.21. a. Sea G un grafo conexo no orientado. Mostrar que existen k caminos que no tienen aristas en común entre a y b si y sólo si cualquier corte que separe a de b tiene al menos k arcos.

b. Sea G un grafo conexo no orientado. Mostrar que existen k caminos sin vértices en común si y sólo si cualquier conjunto de vértices que 'desconecta' a de b tiene al menos k vértices.

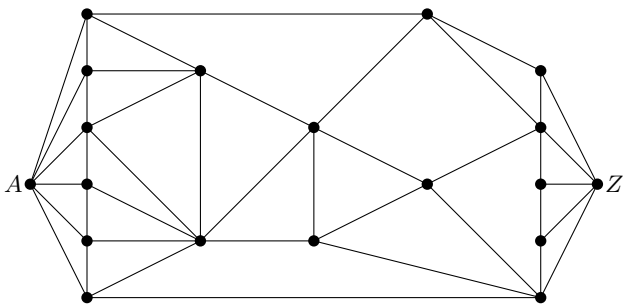
11.22. a. ¿Cómo se puede calcular el número de caminos disjuntos en las aristas que se pueden trazar entre a y b, para a y b vértices dados?

b. idem a. para camino sin vértices en común.

11.23. La red no orientada que se presenta a continuación representa la red de cables troncales de un sistema telefónico. La capacidad de un eje es el número máximo de llamadas que una línea puede soportar. Se quiere determinar cuál es el número máximo de llamadas que se pueden hacer a través de la red de la localidad A a la localidad Z.



11.24. a. Se quieren mandar mensajeros de A a Z en el grafo que se muestra a continuación. Como ciertos tramos de camino pueden estar bloqueados se requiere que cada mensajero use distintas aristas, es decir que los caminos que sigan pueden tener vértices pero no aristas en común.



b. ¿Qué pasa si hasta tres mensajeros pueden usar cada eje?

Práctica 12

Referencias para este práctico: **[3]**, [30], [23]

12.1. Mostrar que los siguientes problemas pertenecen a P:

a. Ordenar una lista de n números.

b. Chequear si un grafo es conexo.

c. Chequear si un grafo tiene un ciclo.

12.2. Mostrar que la relación $\leq_p$ es transitiva ($P_1 \leq_p P_2$ y $P_2 \leq_p P_3 \Rightarrow P_1 \leq_p P_3$).

12.3. Considerar el conjunto de todos los grafos que contienen a K_4 como subgrafo inducido. ¿Pertenece a P el problema de determinar si un grafo es de esta clase? ¿Pertenece a *NP*?

12.4. Diseñar algoritmos no determinísticos polinomiales para los problemas que aparecen a continuación. Describir los algoritmos utilizando pseudocódigo ampliado con la instrucción *Guess()*. Calcular la complejidad de los algoritmos propuestos.

a. Determinar si un grafo es conexo.

b. Determinar si dos grafos son isomorfos.

c. Determinar si un grafo tiene un circuito hamiltoniano.

12.5. Demostrar que si un problema $\Pi \in NP$, entonces Π puede ser resuelto en tiempo determinístico $O(2^{p(n)})$.

12.6. Decir si las siguientes afirmaciones son verdaderas, falsas o no se sabe:

a. $P \subseteq NP$

b. $P \subseteq co\text{-}NP$

c. $P = NP \cap co\text{-}NP$

d. $P = NP \Rightarrow co\text{-}NP = NP$

e. $co\text{-}NP = NP \Leftrightarrow \text{SAT} \in co\text{-}NP$

f. $co\text{-}NP \subseteq NP \Rightarrow NP = co\text{-}NP$

g. $co\text{-}NP \subseteq NP \Rightarrow P = NP$

12.7. ¿Es cierto que si dos problemas X e Y son *NP-completos* entonces $X \leq_p Y$, y también $Y \leq_p X$? Justificar.

12.8. Sean X e Y dos problemas de decisión tales que $X \leq_p Y$. ¿Qué se puede inferir?

a. Si X está en P entonces Y está en P.

b. Si Y está en P entonces X está en P.

c. Si Y es *NP-completo* entonces X también.

d. Si X es *NP-completo* entonces Y también.

e. Si Y es *NP-completo* y X está en *NP* entonces X es *NP-completo*.

f. Si X es *NP-completo* e Y está en *NP* entonces Y es *NP-completo*.

g. X e Y no pueden ser ambos *NP-completos*.

12.9. Decir si las siguientes afirmaciones son verdaderas o falsas:

a. Si $P = NP$ entonces todo problema *NP-completo* es polinomial.

b. Si $P = NP$ entonces todo problema *NP-hard* es polinomial.

c. Las clases *NP-completo* y *co-NP-completo* son disjuntas si y solamente si $P \neq NP$ (un problema de decisión es *co-NP-completo* cuando pertenece a *co-NP* y todo otro problema perteneciente a *co-NP* es polinomialmente reducible a él).

12.10. ¿Qué se puede inferir del hecho de que el problema del viajante de comercio (TSP) es *NP-completo*, suponiendo $P \neq NP$?:

a. No existe un algoritmo que resuelva instancias arbitrarias de TSP.

b. No existe un algoritmo que eficientemente resuelva instancias arbitrarias de TSP.

c. Existe un algoritmo que eficientemente resuelve instancias arbitrarias de TSP, pero nadie lo ha encontrado.

d. TSP no está en P.

e. Todos los algoritmos que resuelven TSP corren un tiempo polinomial para algunas entradas.

f. Todos los algoritmos que resuelven TSP corren un tiempo exponencial para todas las entradas.

12.11. Considere el problema de isomorfismo en grafos (¿es el grafo G isomorfo al grafo H?). ¿Qué puede decir de este problema? ¿Esto implica $P \neq NP$?

12.12. ¿Qué se puede inferir del hecho de que *Isomorfismo* está en *NP* pero no se sabe si es *NP-completo*, suponiendo $P \neq NP$?:

a. Existe un algoritmo que resuelve instancias arbitrarias de *Isomorfismo*.

b. Existe un algoritmo que resuelve eficientemente instancias arbitrarias de *Isomorfismo*.

c. Si encontramos un algoritmo polinomial que resuelva *Isomorfismo*, podemos usarlo como una "caja negra" para resolver *TSP*.

12.13. a. Sea un grafo $G = (V, X)$ y $W \subseteq V$. Demostrar la equivalencia de las siguientes afirmaciones:
- $V - W$ es un recubrimiento de arcos de G.
- W es un conjunto independiente de G.
- W es un clique de G^c (el grafo complemento de G).

b. A partir del punto anterior, encontrar reducciones polinomiales entre los problemas Mínimo Recubrimiento de Arcos, Máximo Conjunto Independiente y Máximo Clique.

c. ¿A qué clase de complejidad pertenecen estos 3 problemas? ¿Por qué?

12.14. Construir una reducción polinomial de *Clique* a *SAT* (sugerencia: considerar variables booleanas representando la matriz de adyacencia de un grafo, junto con $k \times n$ variables v_{ij}, donde n es la cantidad de vértices del grafo y k es el tamaño del clique. v_{ij} sería True sii el i-ésimo elemento del clique es el j-ésimo vértice del grafo).

12.15. Encontrar una reducción polinomial del problema *Circuito Hamiltoniano* al problema del viajante de comercio (*TSP*) (sugerencia: construir un grafo completo con pesos en los arcos tales que el menor circuito hamiltoniano corresponda a un circuito hamiltoniano del grafo original). Sabiendo que el primer problema es *NP-completo*, ¿qué se puede decir del segundo?

12.16. ¿Cuál de los siguientes dos problemas está en P y cuál es *NP-completo*? Justificar.

a. Entrada: Un grafo conexo con n nodos, dos nodos u y v de G, y un entero k. Pregunta: ¿Existe un camino simple de u a v de longitud k?

b. Entrada: Un grafo conexo con n nodos, y dos nodos u y v de G. Pregunta: ¿Existe un camino simple de u a v de longitud 8?

12.17. Utilizando la NP-completitud de otro problema, demostrar que el problema *Camino mínimo condicionado* es *NP-completo*:
Entrada: un grafo $G = (V, X)$ con un peso positivo asociado a cada eje, un par de vértices $s, t \in V$, un subconjunto de nodos $V' \subseteq V$ y un entero $k > 0$.
Pregunta: ¿Existe un camino entre s y t en G que pasa por todos los vértices de V' y de peso total menor o igual que k?

12.18. Mostrar que el problema de decidir si una fórmula booleana tiene al menos dos diferentes asignaciones de variables que la hacen verdadera, es
NP-completo.

12.19. ¿Se podría utilizar una máquina con alto paralelismo para resolver un problema *NP-completo* en tiempo polinomial, suponiendo $P \neq NP$?

12.20. Discutir algunas de las alternativas posibles cuando se debe resolver en la práctica un problema que se sabe que es *NP-completo*.

12.21. ¿Cuál sería la importancia de que alguien...

a. encuentre un algoritmo polinomial que resuelve un problema *NP-completo*?

b. pruebe que un problema en *NP* es intratable?

12.22. a. ¿Cuál es la diferencia entre un algoritmo polinomial y uno pseudo-polinomial?

b. Dar un ejemplo de un algoritmo pseudo-polinomial. Indicar en qué circunstancias este algoritmo presenta dificultades.