Pontificia Universidad Católica de Chile
Departamento de Ciencia de la Computación
IIC1253 - Matemáticas Discretas

Tarea 6

19 de junio de 2025

1º semestre 2025 - Profesores P. Bahamondes - D. Bustamante - P. Barceló

Requisitos

- La tarea es individual. Los casos de copia serán sancionados con la reprobación del curso con nota 1,1.
- **Entrega**: Hasta las 23:59 del viernes 27 de junio a través del buzón habilitado en el sitio del curso (Canvas).
  - Esta tarea debe ser hecha completamente en LaTeX. Tareas hechas a mano o en otro procesador de texto **no serán corregidas**.
  - Debe usar el template LaTeX publicado en la página del curso.
  - Cada solución de cada problema debe comenzar en una nueva hoja. ***Hint:*** Utilice `\newpage`
  - Los archivos que debe entregar son el archivo `PDF` correspondiente a su solución, junto con un `zip`, conteniendo el archivo `tex` que compila su tarea. Si su código hace referencia a otros archivos, debe incluirlos también.
- El no cumplimiento de alguna de las reglas se penalizará con un descuento de 0.5 en la nota final (acumulables).
- No se aceptarán tareas atrasadas (salvo que utilice su cupón `#problemaexcepcional`).
- Si tiene alguna duda, el foro de Github (issues) es el lugar oficial para realizarla.

Pregunta 1

Sea $G = (V, E)$ un grafo simple que no contiene un circuito hamiltoniano (es decir, un ciclo que pasa por cada vértice exactamente una vez). Comience a agregar arcos arbitrariamente a E hasta llegar a un grafo $H = (V, E')$ que contiene un circuito hamiltoniano (tal H debe existir porque un grafo completo siempre contiene un circuito hamiltoniano). Sea (a, b) el último arco que agregamos en este proceso. Por lo tanto, si sacamos el arco (a, b) de E' obtenemos un grafo $H' = (V, E \setminus \{(a, b)\}$ que no contiene ningún circuito hamiltoniano.

Considere un circuito hamiltoniano en H. Tal circuito debe pasar por (a, b). Podemos asumir entonces, sin pérdida de generalidad, que tal circuito hamiltoniano es de la forma:

$$(v_1 = a) \leftrightarrow (v_2 = b) \leftrightarrow v_3 \leftrightarrow \ldots \leftrightarrow v_n \leftrightarrow (v_1 = a),$$

donde $\leftrightarrow$ denota que vértices consecutivos son adyacentes.

1. Demuestre que si b es adyacente a algún v_i en H', para $3 \leq i \leq n$, entonces a no puede ser adyacente a v_{i-1} en H'.
2. En base a lo anterior, demuestre que $\delta_{H'}(a) + \delta_{H'}(b) < n$, donde $\delta_{H'}$ denota el grado de un nodo en H'.
3. Concluya que si $\delta_G(u) + \delta_G(v) \geq n$ para todo par u, v de vértices no adyacentes en G con $u \neq v$, entonces G tiene circuito hamiltoniano (donde, como antes, δ_G denota el grado de un vértice en G).

Pregunta 2

Cuando modelamos con lógica, convertimos una instancia de un problema y en una fórmula, de modo que la instancia es una solución del problema si y sólo si la fórmula es satisfactible. A esto se le llama **reducción**. En la práctica, requerimos reducciones que puedan realizarse en tiempo polinomiales con respecto al tamaño de la instancia del problema original.

Consideremos el siguiente algoritmo, que permite reducir el problema de k-coloración de grafos (es decir, decidir si un grafo puede colorearse con k colores sin que vértices adyacentes compartan color) al problema de satisfactibilidad booleana (SAT).

Algorithm 1: Reducción de k-Coloración a SAT

```
Function GraphKColoringToSAT(G = (V, E), k):
    φ ← true;
    foreach v ∈ V do
        C ← false;
        for i = 1 to k do
            C ← C ∨ c_{v,i};
        φ ← φ ∧ C
        for i = 1 to k − 1 do
            for j = i + 1 to k do
                φ ← φ ∧ (¬c_{v,i} ∨ ¬c_{v,j});
    foreach (u, v) ∈ E do
        for i = 1 to k do
            φ ← φ ∧ (¬c_{u,i} ∨ ¬c_{v,i});
    return φ
```

1. Explique brevemente por qué la fórmula construida modela la k-coloración, es decir, que la fórmula modela que (a) a cada nodo se le asigna un único color y (b) dos nodos adyacentes no pueden tener el mismo color.

2. Para efectos del tiempo del algoritmo, consideraremos como operación representativa la asignación ($\leftarrow$). Demuestre que el algoritmo puede ejecutarse en tiempo polinomial.

   Suponga que tiene un algoritmo `SAT-Solver` que, dado como input una fórmula en lógica proposicional, puede resolver el problema de satisfactibilidad booleana en tiempo $\mathcal{O}(f(n))$ donde n es el tamaño de la fórmula.

3. Construya un algoritmo que utilice a `SAT-Solver` y a `GraphKColoringToSAT` como subrutinas para determinar si un grafo es k-coloreable.

4. Estudie la complejidad de su algoritmo con notación $\mathcal{O}$.