COMPLEJIDAD COMPUTACIONAL

ALGORITMOS NO-DETERMINISTAS

MARCO SILVA HUERTA

Semestre: 2024-2

1. Ruta más corta

1.1. Dar su forma canónica

Problema de decisión: Dada una gráfica no dirigida $G = (V, E)$, donde V es el conjunto de vértices y E es el conjunto de aristas, y dados dos vértices u y v, y un entero positivo k, ¿existe una trayectoria entre u y v cuyo peso total sea menor que k?

1.2. Diseñar un algoritmo no-determinístico polinomial

- Partimos que trabajaremos con:
  - Una grafica
  - Aristas
  - Vértices
  - Dos vértices preseleccionados
  - un entero
- Vamos a comenzar con que distancias entre vértices son desconocidas.
- Creamos un arreglo para guardar esas distancias (distancias)
- Esas distancias se van a inicializar con valores infinitos
- Excepto para la distancia del vertice de inicio y final la cual será 0.
- Usaremos una cola de prioridad para explorar todo los vértices.
- El vertice de inicio será el primero en agregarse a esta cola con distancia 0 (elegido aleatoriamente)
- Para poder explorar toda la gráfica vamos a seleccionar los vértices adyacentes al vértice que tengamos evaluando en la cola
- Una vez alcanzado vamos a actualizar las distancias a sus vecinos
- La actualización debe ser si es que encontramos un camino más corto
- Una vez que terminamos de revisar toda la grafica verificamos si la distancia entre los vértices de inicio y final es menor a la distancia k
- Si lo es, hemos encontrado la ruta

- Si no, no hay una ruta menor.
- Nuestro proceso termina cuando todos los vértices fueron explorados o cuando la cola de prioridad está vacía.

Ahora, en cada iteración del bucle principal, obtenemos aleatoriamente un vértice adyacente para explorar. Esto introduce no determinismo en la selección de vértices adyacentes y cumple con la primera fase del algoritmo no determinístico.

2. 3-SAT

2.1. Dar su forma canónica

Problema de decisión: Dada una fórmula booleana en forma normal conjuntiva donde cada cláusula tiene exactamente tres literales (una instancia de 3-SAT), ¿existe una asignación de verdad que satisface la fórmula?

2.2. Diseñar un algoritmo no-determinístico polinomial

- Fase de adivinación:

  Adivina la asignación de verdad: Para cada variable en la fórmula, adivina si es verdadera o falsa. Esta es la parte no determinística del algoritmo. No sabemos cuál es la correcta, al azar adivinamos. Dado que son dos variables esto podría verse como que estamos lanzando una moneda.

- Fase de verificación:

  Verifica la asignación de verdad: Ahora, con la asignación de verdad adivinada, verificamos si satisface todas las cláusulas de la fórmula. Para hacer esto, recorremos cada cláusula y verificamos si al menos una variable es verdadera bajo la asignación donde se adivina.

  - Si encontramos una cláusula que no es verdadera bajo la asignación adivinada, entonces sabemos que la asignación adivinada no satisface la fórmula, y el algoritmo se detiene y devuelve un mensaje de *no encontrado*
  - Si todas las cláusulas son verdaderas bajo la asignación adivinada, entonces la asignación adivinada satisface la fórmula, y el algoritmo se detiene y devuelve *si encontrado*

Referencias

[1] C. M. Andreas y S. Guido, *Introduction to Machine Learning with Python*, 1th. O'Reilly, 2016.

[2] S. Russell y P. Norvig, *Inteligencia Artificial Un Enfoque moderno*, 2nd. Pearson Prentice Hall, 2016.