Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

Ayudantía Examen

IIC2133 - Estructuras de datos y algoritmos

Segundo semestre, 2017

1. Demostración

a. **P3-I2/2017-2**
En clases estudiamos dos algoritmos para encontrar árboles de cobertura (o extensión) de costo mínimo. Considera el siguiente algoritmo: Partimos con el grafo original y vamos eliminando aristas en orden decreciente de costos. En particular, cuando evaluamos una arista, a partir de la de mayor costo, la eliminamos si al hacerlo no desconectamos el grafo que tenemos hasta este momento. Demuestra que este algoritmo es correcto.

b. **P3-I3/2017-1**
Sea $G = (V, E)$ un grafo no dirigido y $T \subseteq E$ el único árbol de cobertura de costo mínimo (MST) de G. Suponga ahora que G' se construye agregando nodos a G y aristas que conectan estos nuevos nodos con nodos de G. Finalmente, sea T' un MST de G'.

i) Demuestre que no necesariamente $T \subseteq T'$.

ii) Dé una condición necesaria y suficiente para garantizar que $T \subseteq T'$. Demuéstrela.

2. Programación Dinámica

a. **Multiplicación de Matrices**
Imagine que vuelve a álgebra lineal y tiene que multiplicar una serie de matrices $M_1, M_2, ..., M_n$. Considere que se cumple que M_i es de $A \times B$ y M_{i+1} es de $B \times C$ para todo $i = 1...n-1$ (es decir, cada matriz se puede multiplicar con su vecina), y que al multiplicar dos matrices M_i y M_{i+1}, se deben hacer $A * B * C$ multiplicaciones de escalares.
Para terminar lo más rápido posible la tarea, usted se pregunta en qué orden debe elegir las matrices para que se hagan la menor cantidad de operaciones posibles.
Ejemplo: $M_1 \times M_2 \times M_3$ donde M_1 es de 15×3, M_2 es de 3×20 y M_3 es de 20×4.

- Si multiplica así: $(M_1 \times M_2) \times M_3$, el costo total es $15 * 3 * 20 + 15 * 20 * 4 = 2100$
- Si multiplica así: $M_1 \times (M_2 \times M_3)$, el costo total es $3 * 20 * 4 + 15 * 3 * 4 = 420$

Haga un algoritmo de programación dinámica que determine cuál es el menor costo con que se puede multiplicar las matrices dadas.

b. **Dados**
Se tienen n dados, cada uno con m caras (numeradas de 1 a m). Se quiere encontrar la cantidad de combinaciones distintas que suman un número X.
Por ejemplo: 3 dados, cada uno con 6 caras, ¿de cuántas maneras distintas se puede sumar 10?

3. Flujo

a. **Elecciones**
Para las elecciones es muy importante no saturar las calles ni los medios de transporte, por lo que lo ideal es asignar a la gente a un local de votación que le quede cerca ya que sino deberán tomar una micro o ir en auto. El problema es que los locales mismos tampoco pueden saturarse: tienen una capacidad limitada determinada por la cantidad de gente que se espera atender simultáneamente.

a) Teniendo para cada persona una lista de los locales de votación en los que está dispuesto a votar, y las capacidades de cada local de votación, propón un algoritmo que encuentre una asignación que permita ir a votar a la mayor cantidad posible de personas a un local escogido por ellos. (Cada persona solo puede ser asignada a un solo local).

b) Considera ahora que las personas tienen sus preferencias de locales rankeadas, es decir, asignan un puntaje a cada uno de los locales que prefieren. ¿Cómo se puede modificar el algoritmo anterior para este caso?

c) Por último, considera que el SERVEL prefiere ciertas personas para ciertos locales, lo que significa que para cada local las personas tiene cierto ranking (por su parte, el ranking de las personas hacia los locales se mantiene) ¿Cómo se puede modificar el algoritmo anterior para este caso?

4. Otros

a. **Laberinto**
Para construir un laberinto se hace de la siguiente forma: El laberinto parte con una grilla cuadrada de $N \times N$. Entre cada par de posiciones aledañas existe un muro. La forma de unir sectores es botar paredes. El laberinto es correcto si todas las posiciones son alcanzables entre ellas a través de una única ruta. Además, un laberinto es no trivial si se forma de manera aleatoria (no sigue un patrón que lo haga trivial). Describa un algoritmo que permita construir un laberinto bien formado y no trivial.