Universidad de Chile
Departamento de Ciencias de la Computación

CC4102 **Diseño y Análisis de Algoritmos** **Otoño 2020**

Jérémy Barbay

Tarea 4: Randomización y Hashing

(Fecha de entrega: [2020-06-24 Wed])

Problema 1 (3 pts)
Considere la definición del problema de *Intersección elemental*: dado un elemento x, un parámetro entero k y k arreglos ordenados $(A_i)_{i\in[1..k]}$ con n_i elementos, donde $n_1 \geq n_2 \geq \ldots \geq n_k$, decida si x pertenece a la intersección $\cap_{i\in[1..k]}A_i$ de estos arreglos. Si $x \in \cap_{i\in[1..k]}A_i$, se retorna el valor `true` seguido por las posiciones de x en cada arreglo. Si $x \notin \cap_{i\in[1..k]}A_i$, se retorna el valor `false` seguido por el índice del arreglo que no contiene x, y el rango de inserción de x en este arreglo. Considere el siguiente algoritmo aleatorizado para el problema de intersección elemental:

- Inicialización: parámetros x, k, $(A_i)_{i\in[1..k]}, (n_i)_{i\in[1..k]}$; k variables enteras p_i a 0, para cada $i \in [1..k]$; la variable L de tipo *conjunto* a $\{1, \ldots, k\}$; la variable entera i (cuyos valores serán elegidos uniformemente al azar en L); la variable booleana `found` a `false`.
- Mientras que !`found` y $L \neq \emptyset$:
  - Asigna a i un valor elegido uniformemente al azar en L.
  - $L \leftarrow L\backslash\{i\}$.
  - Encuentra el rango de insercion p_i de x en A_i usando búsqueda binaria.
  - Si $A_i[p_i] \neq x$: Asigna a `found` el valor `true`.
- Si $A_i[p_i] = x$ (x está en la intersección): Retorna el valor `true`, seguido por el vector $(p_i)_{i\in[1..k]}$.
- De otra manera (x no está en la intersección): Retorna el valor `false`, seguido por el par (i, p_i).

Dado un x, denotamos $\texttt{rank}_i$ el rango de inserción de x en A_i, para cada i, y r la cantidad de arreglos que no contienen x. Para poder expresar con precisión las preguntas, primero definiremos algunos conceptos. Sea A un algoritmo, ya sea aleatorizado o determinista, y x un input cualquiera del algoritmo. Definimos $c(A, x)$ como el tiempo total que toma ejecutar A sobre x. Note que si A es un algoritmo determinista, entonces $c(A, x)$ es un valor entero determinado, mientras que si A es un algoritmo aleatorizado, entonces $c(A, x)$ es en realidad una variable aleatoria. Dado un problema P definimos:

- El conjunto de algoritmos deterministas que resuelven P, al cuál llamaremos D, y el conjunto de algoritmos aleatorizados que resuelven P, al cuál llamaremos R.
- El conjunto de inputs posibles de P, al cuál llamaremos X.

- El rendimiento determinístico en el peor caso de P como

$$\min_{A \in D} \max_{x \in X} c(A, x)$$

  Es decir, es lo que tarda el mejor algoritmo en el peor de sus inputs. Note que esto coincide con la idea de cota inferior ajustada que veíamos al principio del curso.

- El rendimiento en el peor caso para un algoritmo aleatorizado A como

$$\max_{x \in X} \mathbb{E}[c(A, x)]$$

  Es decir, el valor esperado del rendimiento de A en su peor caso.

- El rendimiento aleatorizado en el peor caso de P como

$$\min_{A \in R} \max_{x \in X} \mathbb{E}[c(A, x)]$$

a. ¿Cuál es el rendimiento determinístico en el peor caso del problema de intersección elemental, sobre todas las instancias con valores $\texttt{rank}_i$ y r fijos? Puede presentarlo en notación asintótica. (*Hint:* Note que si $r = 0$ el problema es trivial dado que los $\texttt{rank}_i$ están fijos. ¿Qué es necesario hacer para resolver el problema cuando $r > 0$?) Justifique.

b. ¿Cuál es el rendimiento esperado de este algoritmo aleatorizado[1] en el peor de sus inputs? Responda con una expresión en notación asintótica para cuando r no está fijo, y pruebe que cuando r está fijo la respuesta es $\mathcal{O}\left(\frac{1}{r}\sum_{i \in [1..k]}(\lg n_i)\right)$.

c. De y justifique la mejor cota inferior que pueda para el rendimiento aleatorizado en el peor caso del problema de intersección elemental. Considere tanto el caso en que r está fijo como el caso en que no. ¿Son estas cotas ajustadas con respecto a las obtenidas en b)? (*Hint:* Define una distribución de instancias dificiles para deducir una cota inferior sobre la complejidad de cualquier algoritmo randomizado via el teorema de Yao-von Neuman.)

[1] Se puede ignorar el rendimiento de la fase de inicialización.

Problema 3 (3 pts)

a. EN GRUPO: **PIENSEN** Cuáles son las ventajas e inconvenientes de las siguientes estrategias de manejo de colisiones para `Closed Hashing`?

   (a) si $h(x)$ esta ocupado, prueba $h(x) + i$ para $i \in \{1, \dots\}$ hasta encontrar una posición libre;

   (b) si $h(x)$ esta ocupado, prueba $h(x) + ip$ para $i \in \{1, \dots\}$ hasta encontrar una posición libre, para p y n sin factor común;

   (c) si $h(x)$ esta ocupado, prueba $h(x)+ih'(x)$ para $i \in \{1, \dots\}$ hasta encontrar una posición libre.

b. INDIVIDUALMENTE: **PROGRAME** tales estrategias (una para cada alumno), y para cada una identifica

   (a) uno de sus mejores caso de uso, y

   (b) uno de sus peores caso de uso.

c. EN GRUPO: **JUNTEN** los seis casos de uso producidos previamente.

d. INDIVIDUALMENTE: **MIDA**

   - Ejecuta cada una de las 3 estrategias sobre los seis casos de usos (18 casos en totales), midiendo
     - la cantidad de comparaciones ejecutadas, y
     - el tiempo de ejecución.

e. EN GRUPO **ANALICEN**

   - Para cada uno de los seis casos de uso deben presentar
     - un gráfico comparando la cantidad de comparaciones ejecutadas por cada una de las 3 estrategias de manejo de colisión, y
     - un gráfico comparando el tiempo de ejecución de las 3 estrategias de manejo de colisión.
   - Analicen los resultados y escriban el reporte.
   - ¿Los resultados obtenidos por los diferentes miembros de su grupo fueron consistentes? Discuta y ejemplifique comparando un par de gráficos.

Entrega de la Tarea

Se espera que se *implementen* los algoritmos, *realicen* los experimentos correspondientes y se entregue un *informe* que indique claramente los siguientes puntos:

a. Las *hipótesis* escogidas antes de realizar los experimentos.

b. El *diseño experimental*, incluyendo la idea general de la implementación de los algoritmos, la generación de las instancias y las medidas de rendimiento utilizadas.

c. La *presentación de los resultados* en forma de una descripción textual, tablas y/o gráficos.

d. El *análisis e interpretación* de los resultados.

Instrucciones

- Puede utilizar Python o Java. Para el informe se recomienda utilizar LaTeX.
- Siga buenas prácticas (*good coding practices*) en sus implementaciones.
- Escriba un informe claro y conciso (largo sugerido: 5 páginas). Las ponderaciones del informe y la implementación en su nota final son las mismas.
- La entrega será a través de U-Cursos y deberá incluir el informe junto con el código fuente de la implementación, y todas las indicaciones necesarias para su ejecución en un archivo `README.md`.
- Se permiten atrasos sin descuento por un máximo de 4 dias.