Universidad de Chile
Departamento de Ciencias de la Computación
CC4102 - Diseño y Análisis de Algoritmos

Auxiliar 11 - "Algoritmos Probabilistas"

Profesor: Gonzalo Navarro
Auxiliar: ~~Manuel~~ Ariel Cáceres Reyes

9 de Agosto del 2018

P1. Polinomios

Dados 3 polinomios en una variable p, q de grado n y r de grado $2n$. Muestre un algoritmo probabilístico que tome tiempo $\mathcal{O}(n)$ en determinar si $pq = r$.

P2. Las Vegas vs Monte Carlo

a) Muestre que si se tiene un algoritmo de tipo Las Vegas de tiempo esperado T y una constante $c > 0$. Se puede construir, para el problema, un algoritmo Monte Carlo de tiempo cT y probabilidad de error a lo más $1/c$.

b) ¿Siempre se puede transformar un algoritmo Monte Carlo en un algoritmo de tipo Las Vegas? ¿Que restricción adicional sobre el problema podemos colocar para que sea cierto?

P3. Corte Mínimo de un Grafo

Un corte o *cut* de un grafo no dirigido conexo $G = (V, E)$, con $|V| = n$ y $|E| = m$, es un subconjunto E' de E tal que el grafo sin estas aristas $(V, E \setminus E')$, tiene dos o más componentes conexas. El corte mínimo del grafo o *min-cut* corresponde al corte del grafo que tiene menor cantidad de aristas.

Para resolver este problema utilizaremos el siguiente algoritmo:

```
1 (V', E') = (V, E)
2 for i ← 1 ... n − 2 do
3 |   Elegimos una arista e ∈ E' uniformemente al azar y la contraemos del grafo (V', E')
  |   actualizando V' y E'.
4 end
5 return E'
```

a) Muestre que el algoritmo encuentra un *cut* del grafo.

b) Muestre que la probabilidad que el algoritmo encuentre el *min-cut* es al menos $\frac{2}{n(n-1)}$.

c) Muestre como reducir la probabilidad de error (no encontrar el *min-cut*) a $\frac{1}{n^2}$.

P1. Elegimos uniformemente $x \in \{0, \ldots, 4n+1\}$ y luego respondemos `true` si $p(x) \cdot q(x) = r(x)$ y `false` en otro caso.

- En el caso de que la igualdad sea correcta, el algoritmo responde `true` de manera correcta.
- Sin embargo, cuando la igualdad es falsa ($pq \neq r$) el algoritmo podría responder erróneamente `true` en caso que $p(x) \cdot q(x) = r(x)$ o equivalentemente cuando el polinomio $g = pq - r$ se hace cero, i.e. cuando x es cero de g.
  Veamos que g es de grado a lo más $2n+1$ por lo que tiene a lo más $2n+1$ raíces, y luego la probabilidad que el algoritmo se equivoque es:

$$\mathbb{P}(x \; es \; raiz \; de \; g) \leq \frac{2n+1}{4n+2} = \frac{1}{2}$$

P2. Preliminares :

- Desigualdad de Markov: Si X es una variable aleatoria no negativa con valor esperado positivo entonces

$$\forall c > 0, \; \mathbb{P}(X \geq c\mathbb{E}(X)) \leq \frac{1}{c}$$

- Algoritmo Montecarlo: Algoritmo probabilístico que termina en un tiempo determinado, pero tiene una probabilidad de error distinta de 0.
- Algoritmo Las Vegas: Algoritmo probabilístico con probabilidad de error 0, pero que no asegura el término de su ejecución, solo asegura un valor esperado finito del tiempo que demora.

a) Si tenemos un algoritmo A "Las Vegas" que resuelve el problema en tiempo esperado $T > 0$, creamos el siguiente algoritmo "Montecarlo":
   - Correr A por cT tiempo, si ha terminado, respondemos lo mismo que A, si no respondemos "`null`".

   Definamos la variable aleatoria X como el tiempo que demora el algoritmo A. Por Desigualdad de Markov sabemos que $\mathbb{P}(X \geq cT) \leq \frac{1}{c}$, pero $\mathbb{P}(X \geq cT)$ es la probabilidad de que A se demore más de cT y por lo tanto una cota superior a la probabilidad de que nuestro algoritmo "Montecarlo" se equivoque.

b) La condición para hacer la transformación es contar con un procedimiento eficiente que verifique el `output` dado por el algoritmo "Montecarlo" sea correcto, pues si lo tenemos podemos hacer un algoritmo "Las Vegas" que corra el "Montecarlo" hasta que este último entregue una respuesta correcta.

P3. a) Supongamos que los nodos en V' (final) son v' y u' y sean $v, u \in E$, nodos que pasaron a ser parte de v', u' en el algoritmo respectivamente. E' (final) es un corte del grafo pues en

$(V, E \setminus E')$ no existe un camino de $v \to u$ o dicho de otro modo, todo camino de $v \to u$ pasa por E' (lo cual es cierto pues v' y u' corresponde a una "partición" de V, y E' corresponde a las aristas entre ambas partes, por lo que todo camino que vaya de una parte a otra debe tomar una de estas aristas).

b) Supongamos que un *min-cut* es C, de k aristas. Observemos lo siguiente:

1. Para que el algoritmo responda C se debe cumplir que en cada iteración la arista escogida no esté en C.
2. El grado de los nodos del grafo debe ser al menos k. Y por lo tanto el grafo tiene al menos $nk/2$ aristas.

Definamos entonces E_i como el evento que en la i-ésima iteración del algoritmo escojamos una arista que no está en C. La probabilidad que nos interesa calcular entonces es $\mathbb{P}(E_1 \cap E_2 \cap \ldots E_{n-2}) = \mathbb{P}(E_1)\mathbb{P}(E_2|E_1)\ldots\mathbb{P}(E_{n-2}|E_1 \cap E_2 \cap \ldots E_{n-3})$.
Veamos que en general $\mathbb{P}(E_i|E_1 \cap E_2 \cap \ldots E_{i-1})$ corresponde a la probabilidad que no se escoja una arista de C en el paso i dado que en ninguno de los pasos anteriores esto ha sucedido. Dado que no se ha escogido ninguna arista de C para contraer tenemos un grafo de $n - i + 1$ nodos cuyo *min-cut* es de tamaño $\geq k$ (si hubiese un corte más chico C no sería mínimo) y por la observación 1. el grafo debe tener al menos $(n - i + 1)k/2$ aristas así la probabilidad de escoger una arista de C es $\leq 2/(n - i + 1)$ y por lo tanto la de no hacerlo $\geq 1 - 2/(n - i + 1) = \frac{n-i-1}{n-i+1}$. Luego, la probabilidad que el algoritmo encuentre C es mayor o igual a :

$$\prod_{i=1}^{n-2} \frac{n-i-1}{n-i+1} = \frac{2}{n(n-1)}$$

c) Si repetimos el algoritmo $n(n-1)\ln n$ veces de manera independiente y retornamos el menor de los cortes encontrados como *min-cut*. En este caso la probabilidad que ninguno de las ejecuciones encuentre un *min-cut* es menor o igual a:

$$\left(1 - \frac{2}{n(n-1)}\right)^{n(n-1)\ln n} \leq e^{-2\ln n} = \frac{1}{n^2}$$