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

Auxiliar 5 - Análisis Amortizado y Algoritmos sobre strings

Profesor: Gonzalo Navarro
Auxiliares: Dustin Cobas
Bernardo Subercaseaux

P1. **Union & Find**

Analizaremos un par de variantes de la estructura Union-Find, donde partimos con n elementos separados y realizamos $m \geq n$ operaciones Find (note que las operaciones Union son a lo más $n-1$).

- Modifique el análisis realizado en clases para mostrar que el costo de las operaciones es $O(n \log n + m)$.
- Demuestre que, si se hacen todos los Union antes de todos los Find, el costo total es $O(m)$.

P2. **Next Smaller Value**

Dado un arreglo $A[1..n]$, se quiere construir otro arreglo $NSV[1..n]$, llamado "next smaller value". Concretamente, $NSV[i] = \min\{j > i, A[j] < A[i]\}$, suponiendo $A[n+1] = -\infty$. Diseñee un algoritmo de tiempo $O(n)$ para construir NSV y use técnicas de análisis amortizado para demostrar que su complejidad es $O(n)$.

Nota: No hicimos P3 ni P4 en la auxiliar ya que no habían visto estructuras sobre string en clases todavía

P3. **Trie-hard**

Diseñe un algoritmo basado en un Trie que, dada una lista de palabras (diccionario) L, y una serie de nuevas palabras $p_1, \ldots, p_n$, y un entero k cuente por cada p_i cuántas palabras $l \in L$ difieren de p_i en a lo más k caracteres.

P4. **Mínima Rotación Cíclica**

Dada una palabra p, diseñe un algoritmo que encuentre la rotación de p que sea lexicográficamente menor.

Por ejemplo, las rotaciones de la palabra *aloha* son *aloha*, *lohaa*, *ohaal*, *haalo*, *aaloh*

1 Soluciones

P1. Union & Find

- Union cuesta tiempo constante según el análisis de clase. Es decir, en total cuestan O(n). El costo de las operaciones Find lo vamos a dividir en 2, una parte se la cobraremos a los nodos, y otra parte a la operación misma. Le cobramos a cada nodo por el que Find pasa, excepto la raíz y. sus hijos, cuyo costo se lo cobramos a la operación misma.

  Cada vez que le cobramos a un nodo, significa que ese nodo tiene una nueva raíz, ya que cuando un nodo mantiene su raíz, después de hacerle find se cuelga directamente de esta, y dado que es hijo de su raíz, no le cobramos. Cada vez que a un nodo se le cambia la raíz, su subárbol se está al menos duplicando, lo que solo puede pasar $O(\log n)$ veces. Dado que hay n nodos, en total les cobramos $O(n \log n)$.

  Para finalizar, el costo que paga la operación misma es constante, ya que solo paga por 1 raíz y sus 2 hijos. Dado que hay m de estas operaciones, cuesta $O(m)$ en total, de donde concluimos.
- Mismo esquema de contabilidad que antes, pero notemos que ahora la raíz de un nodo no puede cambiar, ya que no hay más Union, por lo tanto los nodos solo pagan su primer find ($O(n)$ ya que hay n nodos), para luego quedar colgados a su raíz, y por tanto la operación misma paga su costo ($O(m)$ en total ya que hay m operaciones).

P2. Next Smaller Value

Se inicializa un stack S, que contendrá los elementos previos cuyo NSV aún no se conoce.

```
1 for i ∈ [1..n] do
2 |   while A[i] < A[S.top()] do
3 |   |   NSV[S.pop()] ← i
4 |   end
5 |   S.push(i)
6 end
```

Esto es lineal por el mismo argumento del multipop visto en clase.

P3. Trie-hard

Insertamos todas las palabras de L en un Trie. Luego por cada palabra p_i usaremos un algoritmo similar al de búsqueda para contar cuántas palabras hay que difieran de p_i.

P4. Mínima Rotación Cíclica

Consideramos $p' = p + p$

Notemos que un sufijo de p' de largo mayor o igual a $|p|$ se corresponde con rotación cíclica, ya que incluye necesariamente un sufijo de p seguido por p, y por lo tanto si consideramos los primeros $|p|$ caracteres, tenemos una rotación cíclica.

Por ejemplo, si consideramos para *alohaaloha* el sufijo *ohaaloha*, que tiene más de los 5 caracteres de aloha, lo podemos descomponer en $oha + aloha$, notemos que *oha* aparece repetido, y si descartamos ocurrencia en la segunda mitad, tenemos $oha + al$ que justamente es una rotación de *aloha*, ya que tiene un sufijo seguido por un prefijo y sus largos suman 5.

Armamos entonces el SuffixArray S para p'. El primer valor en S que está entre 0 y $|p| - 1$ es la respuesta. Nos convencemos con un ejemplo.

$p = aloha$ $p' = alohaaloha$.

Los sufijos de p' son:

$\{alohaaloha, lohaaloha, ohaaloha, haaloha, aaloha, aloha, loha, oha, ha, a\}$

Al ordenarlos obtenemos (sus índices):

$\{9, 4, 5, 0, 8, 3, 6, 2, 7\}$

Como podemos notar, los sufijos cuyo índice va después de $|p| - 1$ corresponden a sufijos de la segunda ocurrencia de p en p' y por lo tanto no representan una rotación válida