Mezcla aleatoria

Teoría de Algoritmos I (75.29 / 95.06)

Ing. Víctor Daniel Podberezski
vpodberezski@fi.uba.ar

Enunciado

Sea

Set A de n elementos

Queremos

Generar un listado de A ordenado aleatoriamente

Usos

Mezclar un conjunto de elementos se utiliza en

- Juegos de azar
- Reproducción de música aleatoria
- Modelos estadísticos
- Simulaciones
- Pruebas de complejidad algorítmica

Método 1: Permutación por ordenamiento

Para cada i elemento en A

Generaremos un numero p_i aleatorio como su clave

Utilizando p_i para cada elemento a_i como “clave”

Ordenaremos A

Podemos elegir

Cualquier algoritmo de ordenamiento como caja negra para resolverlo

Pseudocódigo

```
Sea A[1…n] conjunto a ordenar
Sea P[1…n] vector numérico // vector de prioridades

Desde j=1 a n
    P[j] = random_value(1…x)

Ordenar A utilizando P como clave

Retornar A
```

Problemas del método

La generación de la clave de ordenamiento (valor entre 1 y x)

De cada elemento en A

Puede perder uniformidad aleatoria

Si existen claves duplicadas

El ordenamiento

De claves repetidas depende el método de ordenamiento (puede ser estable o no)

Y es un proceso determinístico

Problemas del método (cont.)

Para disminuir la posibilidad de claves repetidas

Podemos establecer el valor X >> n (por ejemplo n^5)

Aun así la posibilidad persistirá

Se puede agregar un registro de claves utilizadas

y volver a seleccionar si surge una ya utilizada

Si evitamos claves repetidas

Podemos utilizar cualquier algoritmo de ordenamiento como caja negra

Análisis de complejidad

El proceso de generacion de claves (con x>>n)

- Es O(n) en tiempo
- En O(n) en espacio

El proceso de ordenamiento

- Es O(nlogn)
- Es O(n) en espacio (depende del método elegido, podría ser O(1))

La complejidad total

- Temporal: O(nlogn)
- Espacial: O(n)

Análisis de uniformidad

Llamamos

E_i al evento que el elemento A[i] obtiene la i-esima menor clave

Suponemos

Que todo evento E_i ocurre (corresponde a una permutación posible)

Y son independientes entre ellos

Entonces

$$Pr(E_1 \cap E_2 \cap \ldots \cap E_n) = \overbrace{Pr(E_1)}^{1/n} * \overbrace{Pr(E_2 / E_1)}^{1/(n-1)} * \ldots * \overbrace{Pr(E_n / E_{n-1} \cap E_{n-2} \cap \ldots \cap E1)}^{1}$$

$$Pr(E_1 \cap E_2 \cap \ldots \cap E_n) = 1 / n!$$

Si probamos cualquier otra permutación

Veremos que su probabilidad también es 1 / n!

Por lo tanto este método genera una permutación aleatoria uniforme

Método 2: Algoritmo de Fisher-Yates

Algoritmo de mezcla

Propuesto por Ronald A. Fisher y Frank Yates

En libro "Statistical tables for biological, agricultural and medical research" (1948)

También conocido como

"barajado del sombrero"

Tiene gran cantidad de variantes

(analizaremos 1 variante)

Fisher Yates: Descripción "popular"

Se introducen todos los números

en un sombrero,

se agita el contenido(se mezclan)

Se van sacando de a uno

Y se listan en el mismo orden en que se sacan

hasta que no quede ninguno.

El resultado es el conjunto mezclado.

Descripción algoritmica

Para cada elemento A[i]

- Generamos un valor x al azar entre i y n
- Intercambiamos A[i] con A[x]

```
Sea A[1…n] conjunto a ordenar

Desde i=1 a n
    intercambiar A[i] con A[random_value(i…n)]

Retornar A
```

Análisis de uniformidad

En la primera posicion A[1]

Puede quedar cualquier elemento de A con posibilidad 1/n

En la segunda posición A[2]

Puede quedar cualquier elemento de A menos el que quedo en el primer lugar con posibilidad 1/(n-1)

En la posición A[x]

Puede quedar cualquier elemento menos los x-1 que salieron antes: 1/(n-x+1)

Cualquier permutación

Tiene probabilidad también es 1 / n!

Por lo tanto este método genera una permutación aleatoria uniforme

Ejemplo

Si

$A = \{a_1,a_2,a_3,a_4\}$

i	Rand	Intercambio
1	[1 a 4] → 2	$\{a_1,a_2,a_3,a_4\} \rightarrow \{a_2,a_1,a_3,a_4\}$
2	[2 a 4] → 3	$\{a_2,a_1,a_3,a_4\} \rightarrow \{a_2,a_3,a_1,a_4\}$
3	[3 a 4] → 3	$\{a_2,a_3,a_1,a_4\} \rightarrow \{a_2,a_3,a_1,a_4\}$
4	-	$\{a_2,a_3,a_1,a_4\}$

Un pequeño cambio… un significativo cambio

Es un error común

Cambiar el rango de intercambio en cada iteración

Pasar de

```
Desde i=1 a n
    intercambiar A[i] con A[random_value(i…n)]
```

A:

```
Desde i=1 a n
    intercambiar A[i] con A[random_value(1…n)]
```

Pero tiene resultados que invalidan la uniformidad aleatoria

Un pequeño cambio... (cont.)

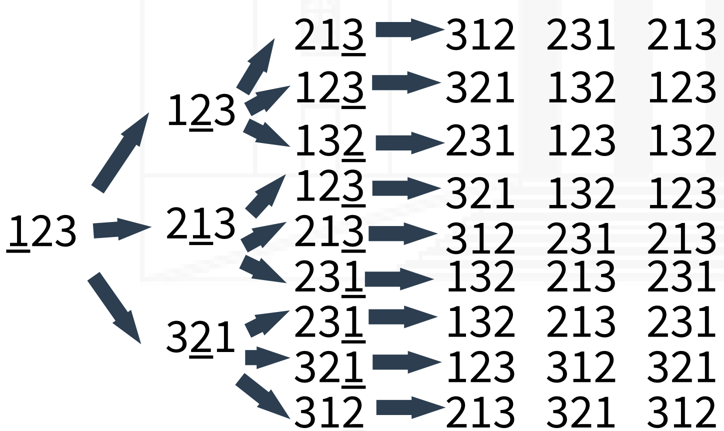
Si tenemos

$A = \{a_1,a_2,a_3\}$

Existen 6 ordenamiento posibles

$\{a_1,a_2,a_3\}$, $\{a_1,a_3,a_2\}$, $\{a_3,a_2,a_1\}$, $\{a_3,a_1,a_2\}$, $\{a_2,a_1,a_3\}$, $\{a_2,a_3,a_1\}$

Pero utilizando el algoritmo modificado



No queda uniformemente aleatorio!

Perm	#
$\{a_1,a_2,a_3\}$	4
$\{a_1,a_3,a_2\}$	5
$\{a_3,a_2,a_1\}$	4
$\{a_3,a_1,a_2\}$	4
$\{a_2,a_1,a_3\}$	5
$\{a_2,a_3,a_1\}$	5

Presentación realizada en Junio de 2020