Ranking y unranking de árboles

Ing. Vicente Oscar Mier Vela

11 de julio del 2013

Curso propedéutico del 2013, a cargo del Dr Jose Torres-Jimenez, CINVESTAV, UNIDAD TAMAULIPAS, LABORATORIO DE TECNOLOGÍAS DE INFORMACIÓN, Parque Científico y Tecnológico TECNOTAM – Km. 5.5 carretera Cd. Victoria-Soto La Marina C.P. 87130 Cd. Victoria, Tamps. Teléfono: (834) 107 02 20 – Fax: (834) 107 02 24 y (834) 314 73 92, vinculacion@tamps.cinvestav.mx

Resumen

Ranking de árboles utilizando el algoritmo de Prüffer junto con la regla de Rufini. Unranking utilizando la operación inversa del algoritmo de Prüffer y la operación inversa de la regla de Rufini.

1. Introducción

Por medio de la aplicación del algoritmo de Prüffer [1][2] es posible asignarle un polinomio único a cada árbol posible. Posteriormente, a partir de este polinomio, es posible derivar un único número decimal, correspondiente al árbol en cuestión, obteniendo el residuo de la regla de Ruffini [3] para dicho polinomio de Prüffer.

2. Implementación en C

La implementación consiste en dos funciones: prufer_ruffini(), para obtener el número decimal único para un árbol específico, a partir de la lista de sus arcos y la cantidad de nodos; y unruffini_unpruffer(), para obtener dicha lista de arcos a partir del número de Ruffini del árbol y su número de arcos.

A continuación se muestra el código fuente de la función prufer_ruffini():

```
#include <stdio.h>
#include <stdlib.h>

#define ROWS 1024
#define COLS 2
```

```
int main() {
        int n = 11;
        int arcos[ROWS][COLS] = {
                { 1,    0 },
                { 3,    0 },
                { 9,    0 },
                { 0,    7 },
                { 7,    4 },
                { 8,    7 },
                { 10,   7 },
                { 2,    10 },
                { 8,    5 },
                { 5,    6 },
        };
        arcos[n-1][0] = arcos[n-1][1] = -2;
        prufer_ruffini(n, arcos);
        return 0;
}

int prufer_ruffini(int n, int arcos[ROWS][COLS]) {
        int i, j, min, prufer[n-2];
        int grados[n+1];
        int k;
        for(k=0;k<n-2;k++){
                for(i=0;i<n+1;i++){
                        grados[i] = 0;
                }
                printf("arco\tnodo\tnodo\n");
                for(i=0;i<n-1;i++){
                        printf( \
                                "%d\t" \
                                "%d\t%d\n", \
                                i, \
                                arcos[i][0], \
                                arcos[i][1] \
                        );
                }
                printf("--\n");
                for(i=0;i<n-1;i++){
                        for(j=0;j<n;j++){
                                if(arcos[i][0] == j || arcos[i][1] == j){
                                        grados[j]++;
                                }
                        }
                }
```

```
        printf("nodo\tgrado\n");
        for(i=0;i<n+1;i++){
                printf( \
                        "%d\t%d\n", \
                        i, \
                        grados[i] \
                );
        }
        printf("^\nEl ultimo no es un nodo. Es un basurero.\n");
        printf("--\n");
        min = n-1;
        for(i=0;i<n-1;i++){
                if(grados[i] == 1 && i < min){
                        min = i;
                }
        }
        printf( \
                "eliminar nodo %d\n", \
                min \
        );
        printf("--\n");
        grados[min] = 0;
        for(i=0;i<n-1;i++){
                if(arcos[i][0] == min || arcos[i][1] == min){
                        if(arcos[i][0] == min){
                                prufer[k] = arcos[i][1];
                        } else {
                                prufer[k] = arcos[i][0];
                        }
                        arcos[i][0] = arcos[i][1] = n;
                }
        }
        printf( \
                "prufer=%d\n", \
                prufer[k] \
        );
        printf("--\n");
}
printf("secuencia de prufer:\n");
for(i=0;i<n-2;i++){
        printf("%d ", prufer[i]);
}
printf("\n--\n");
int ruffini = prufer[0] * 11;
for(i=1;i<n-2;i++){
        ruffini *= n;
```

```
                ruffini += prufer[i];
                printf("rufi %d\n",ruffini);
        }
        printf( \
                "residuo de ruffini:\n%d\n", \
                ruffini \
        );
        return 0;
}
```

Y aquí, el código fuente de la función unruffini_unpruffer():

```
#include <stdio.h>
#include <stdlib.h>

int main() {
        unruffini_unprufer(11, 196083774);
        return 0;
}

int unruffini_unprufer(int n, int ruffini){
        int i, j, k, min, mindex, prufer[n-2], arcos[n-1][2], aux[n];
        printf("residuo de ruffini:\n%d\n--\n",ruffini);
        for(i=n-3;i>0;i--){
                prufer[i] = ruffini%n;
                ruffini -= ruffini%n;
                ruffini /= n;
                printf("rufi %d\n",ruffini);
        }
        prufer[0] = ruffini;
        printf("secuencia de prufer:\n");
        for(i=0;i<n-2;i++){
                printf( \
                        "%d ", \
                        prufer[i] \
                );
        }
        printf("\n--\n");
        k = 0;
        for(i=0;i<n;i++){
                for(j=0;j<n-2;j++){
                        if(prufer[j] == i){
                                break;
                        }
                }
                if(j == n-2){
                        aux[k++] = i;
```

```
        }
}
for(i=k;i<n;i++){
        aux[i] = n;
}
printf( \
        "lista auxiliar:\n" \
);
for(i=0;i<n;i++){
        printf("%d ",aux[i]);
}
printf("\n--\n");
for(j=0;j<n-2;j++){
        min=n;
        mindex=0;
        for(i=0;i<n;i++){
                if(aux[i]<min){
                        min=aux[i];
                        mindex=i;
                }
        }
        arcos[j][0] = min;
        arcos[j][1] = prufer[j];
        for(i=j+1;i<n-2;i++){
                if(prufer[i] == prufer[j]){
                        aux[mindex]=n;
                        break;
                }
        }
        if(i == n-2){
                aux[mindex]=prufer[j];
        }
}
int c = 0;
for(i=0;i<n;i++){
        if(aux[i]<n){
                arcos[n-2][c++]=aux[i];
        }
}
printf("--\n");
printf("arco\tnodo\tnodo\n");
for(i=0;i<n-1;i++){
        printf( \
                "%d\t%d\t%d\n", \
                i, \
                arcos[i][0], \
```

```
                        arcos[i][1] \
                );
        }
        return 0;
}
```

3. Ejemplo de ejecución

La ejecicución de la función prufer_ruffini(), como se muestra en su código fuente, produce la siguiente salida:

```
arco    nodo    nodo
0       1       0
1       3       0
2       9       0
3       0       7
4       7       4
5       8       7
6       10      7
7       2       10
8       8       5
9       5       6
--
nodo    grado
0       4
1       1
2       1
3       1
4       1
5       2
6       1
7       4
8       2
9       1
10      2
11      0
^
El ultimo no es un nodo. Es un basurero.
--
eliminar nodo 1
--
prufer=0
--
arco    nodo    nodo
0       11      11
```

```
1       3       0
2       9       0
3       0       7
4       7       4
5       8       7
6       10      7
7       2       10
8       8       5
9       5       6
--
nodo    grado
0       3
1       0
2       1
3       1
4       1
5       2
6       1
7       4
8       2
9       1
10      2
11      0
^
El ultimo no es un nodo. Es un basurero.
--
eliminar nodo 2
--
prufer=10
--
arco    nodo    nodo
0       11      11
1       3       0
2       9       0
3       0       7
4       7       4
5       8       7
6       10      7
7       11      11
8       8       5
9       5       6
--
nodo    grado
0       3
1       0
2       0
```

```
3       1
4       1
5       2
6       1
7       4
8       2
9       1
10      1
11      0
^
El ultimo no es un nodo. Es un basurero.
--
eliminar nodo 3
--
prufer=0
--
arco    nodo    nodo
0       11      11
1       11      11
2       9       0
3       0       7
4       7       4
5       8       7
6       10      7
7       11      11
8       8       5
9       5       6
--
nodo    grado
0       2
1       0
2       0
3       0
4       1
5       2
6       1
7       4
8       2
9       1
10      1
11      0
^
El ultimo no es un nodo. Es un basurero.
--
eliminar nodo 4
--
```

```
prufer=7
--
arco    nodo    nodo
0       11      11
1       11      11
2       9       0
3       0       7
4       11      11
5       8       7
6       10      7
7       11      11
8       8       5
9       5       6
--
nodo    grado
0       2
1       0
2       0
3       0
4       0
5       2
6       1
7       3
8       2
9       1
10      1
11      0
^
El ultimo no es un nodo. Es un basurero.
--
eliminar nodo 6
--
prufer=5
--
arco    nodo    nodo
0       11      11
1       11      11
2       9       0
3       0       7
4       11      11
5       8       7
6       10      7
7       11      11
8       8       5
9       11      11
--
```

```
nodo    grado
0       2
1       0
2       0
3       0
4       0
5       1
6       0
7       3
8       2
9       1
10      1
11      0
^
El ultimo no es un nodo. Es un basurero.
--
eliminar nodo 5
--
prufer=8
--
arco    nodo    nodo
0       11      11
1       11      11
2       9       0
3       0       7
4       11      11
5       8       7
6       10      7
7       11      11
8       11      11
9       11      11
--
nodo    grado
0       2
1       0
2       0
3       0
4       0
5       0
6       0
7       3
8       1
9       1
10      1
11      0
^
```

```
El ultimo no es un nodo. Es un basurero.
--
eliminar nodo 8
--
prufer=7
--
arco    nodo    nodo
0       11      11
1       11      11
2       9       0
3       0       7
4       11      11
5       11      11
6       10      7
7       11      11
8       11      11
9       11      11
--
nodo    grado
0       2
1       0
2       0
3       0
4       0
5       0
6       0
7       2
8       0
9       1
10      1
11      0
^
El ultimo no es un nodo. Es un basurero.
--
eliminar nodo 9
--
prufer=0
--
arco    nodo    nodo
0       11      11
1       11      11
2       11      11
3       0       7
4       11      11
5       11      11
6       10      7
```

```
7       11      11
8       11      11
9       11      11
--
nodo    grado
0       1
1       0
2       0
3       0
4       0
5       0
6       0
7       2
8       0
9       0
10      1
11      0
^
El ultimo no es un nodo. Es un basurero.
--
eliminar nodo 0
--
prufer=7
--
secuencia de prufer:
0 10 0 7 5 8 7 0 7
--
rufi 10
rufi 110
rufi 1217
rufi 13392
rufi 147320
rufi 1620527
rufi 17825797
rufi 196083774
residuo de ruffini:
196083774
```

Mientras que la ejecución de unruffini_unpruffer(), como se muestra en su respectivo código, produce:

```
residuo de ruffini:
196083774
--
rufi 17825797
rufi 1620527
rufi 147320
```

```
rufi 13392
rufi 1217
rufi 110
rufi 10
rufi 0
secuencia de prufer:
0 10 0 7 5 8 7 0 7
--
lista auxiliar:
1 2 3 4 6 9 11 11 11 11 11
--
--
arco     nodo     nodo
0        1        0
1        2        10
2        3        0
3        4        7
4        6        5
5        5        8
6        8        7
7        9        0
8        0        7
9        10       7
```

Referencias

[1] http://www.proofwiki.org/wiki/Labeled_Tree_from_Pr %C3 %BCfer_Sequence

[2] http://en.wikipedia.org/wiki/Pr %C3 %BCfer_sequence

[3] http://en.wikipedia.org/wiki/Ruffini %27s_rule