Trabajo Practico: Circulos de Permutaciones Documentación

Bonino Bianca
Contreras Santiago
Remus Ezequiel

ÍNDICE

I. SONCIRCULOSIGUALES

I-A. Funciones auxiliares

I-A1. longitud:

Signatura

[Integer] $->$ Integer

- o [Integer] : Representa una lista de numeros enteros a la cual se le contaran los elementos
- o Integer : Retorna como valor la cantidad de elementos numericos de la lista

Descripción

Esta función recibe una lista como parametro, retornando la cantidad de elementos de la lista.

Como caso base tenemos una lista vacia, a cual al no tener elementos debe retornar 0. Luego, si la lista tiene elementos suma 1 al valor de la función llamada por recursión, a la cual se le pasa como parametro la cola de la lista sin el primer elemento.

I-A2. circulosIdenticos:

Signatura

Circulo $->$ Circulo $->$ Bool

- o Circulo: Primer circulo a comparar con el segundo
- o Circulo: Segundo circulo a comparar con el primero
- o Bool: La funcion devuelve *False* se no son circulos idénticos elemento a elemento y devuelve *True* si son idénticos elemento a elemento

Descripción

Esta función toma dos parametros de tipo *Circulo* compara si son completamente identicos. Si lo son retorna True si no Retorna False.

Como caso base, se toman dos Circulos vacios. Al ser ambos vacios devolvera *True*. Al pasarle dos listas evalua si las cabezas de los circulos son iguales, si se cumple llama a si misma pasandose las colas de ambos circulos. Si esto no se cumple devuelve *False*.

I-A3. circulosIgualesAux:

Signatura

Circulo $->$ Circulo $->$ Bool

- o Circulo : Primer circulo a comparar
- o Circulo : Segundo circulo a comparar
- o Bool : Devuelve True si los círculos son idénticos o iguales por rotación

Descripción

Esta función toma dos círculos y los compara para saber si son idénticos o iguales por rotación.

Como caso base, si las cabezas de los círculos son iguales, llama a la función *circulosIguales* previamente definida. Si las cabezas no son iguales se hace una llamada recursiva a la función pasando al Primer circulo y a la lista conformada por la cola de la segunda lista sumandole la cabeza al final. Es decir, si $(y : ys)$ era $[1, 2, 3, 4]$, entonces $(ys + +y)$ es $[2, 3, 4, 1]$.

I-B. Función Obligatoria

I-B1. sonCirculosIguales:

Signatura

Circulo $->$ Circulo $->$ Bool

- o Circulo :Primer Circulo a Comparar
- o Circulo :Segundo Circulo a Comparar
- o Bool : Devuelve *True* si los circulos son identicos o iguales por rotación y *False* si no lo son.

Descripción

Esta función compara dos parámetros tipo circulo con la intención de saber si cumplen con la condición de ser círculos de permutación.

Como caso base, si los circulos no tienen elementos, entonces son iguales, por lo que devuelve *True*. Si las listas no son vacias, entonces se fija si las longitudes son iguales, si lo son se fija si estas listas cumplen con la condición dada por la función *circulosIgualesAux*. Si sus longitudes no son iguales entonces devuelve *False*.

II. PERMUTACIONES

II-A. Funciones auxiliares

II-A1. cambiarAux:

Signatura

Integer $->$ Integer $->$ [Integer] $->$ [Integer]

- o Integer : Entero perteneciente a la lista a cambiar.
- o Integer : Numero por el cual se va a cambiar al elemento de la lista.
- o [Integer] : Lista de enteros a revisar.
- o [Integer] : Lista modificada.

Descripción

Esta funcion cambia al primer numero n coincidente con la cabeza de la lista y lo cambia por el numero m.

Como caso base, se toma que si la lista es vacia, al no tener elementos va a retornar una lista vacia. Luego, si tiene elementos, analiza si el primer parametro pasado coincide con el primer elemento de la lista, si es asi realiza el cambio $(m : xs)$, sino coinciden, entonces agrega la cabeza de la lista a la llamada por recursión de la función a al cual se le pasan los dos primeros elementos originales y el resto de los elementos de la lista no analizados (`cambiarAux n m xs`)

II-A2. cambiar:

Signatura

Integer $->$ Integer $->$ [[Integer]] $->$ [[Integer]]

- o Integer : Entero perteneciente a la lista a cambiar.
- o Integer : Numero por el cual se va a cambiar al elemento de la lista.
- o [[Integer]] : Lista de listas de enteros a la cual se le cambiaran los elementos
- o [[Integer]] : Lista de listas de elementos final con elementos cambiados

Descripción

Utiliza *cambiarAux* para cambiar al primer numero n que encuentra en las listas por el numero m. Solo cambia el primer n, si existen 2 n solo cambia el primero.

Como caso base, si la lista esta vacia, devuelve una lista vacia, ya que no existen elementos iguales a los

que se les vayan a pasar. Luego, si la lista tienen elementodos llama a *cambiarAux* pasandose los primeros dos parametros (n y m) y la lista que conforma a la cabeza de la lista de listas y al resultado se le agregara la llamada recursiva de la funcion pasandose la cola de la lista de listas. (`cambiarAux n m x : cambiar n m xs`)

II-A3. agregar:

Signatura

Integer $->$ [[Integer]] $->$ [[Integer]]

- o Integer : Entero a agregar a la cabeza de cada lista
- o [[Integer]] : Lista de listas a las cuales se le va a agregar el elemento del pirmer parametro
- o [[Integer]] : Lista de listas final, con los elementos agregados

Descripción

Esta funcion toma al entero n y a la lista $(x : xs)$, lo que hace es agregar en la cabeza de cada elemento de la lista de listas al elemento n.

Como caso base, si la lista esta vacia retorna una lista vacia, ya que no tiene elementos. Luego, si la lista tiene un elemento o mas entonces toma a la lista correspondiente a la cabeza de la lista y le agrega el elemento n $([n : x])$ y a este elemento lo pone conjuntamente dentro de una lista a la cual se le agregaran las llamadas recursivas de la funcion a la cual se le pasara el numero a agregar en cada lista y la cola de la lista de listas original.

II-A4. permutacionesAux:

Signatura

Integer $->$ [[Integer]] $->$ [[Integer]]

- o Integer : Elemento agregado a la lista
- o [[Integer]] : Lista a la cual se le agregaran los n elementos.
- o [[Integer]] : Lista de listas final

Descripción

Esta función agrega el elemento n primero a la lista y lo une con la llamada recursiva pasandose el numero anterior a n y la lista obtenida tras cambiar al numero anterior a n por n en la lista pasada como parametro originalmente.

Como caso base, si se le agrega ele elemento cero a la lista, esta devuelve una lista vacia, esto es porque n debe corresponderse con un número natural. Luego, si n es distinto de 0, se agrega el elemento n a la lista (xs) y se lo agrega a una lista de listas llamando por recursion a la función pasandose como parametros al numero anterior $(n-1)$ y como segundo parametro se realiza el cambio , mediante la funcion *cambiar* del elemento $n-1$ por el elemento n en la lista.

II-B. Función Obligatoria

II-B1. permutaciones:

Signatura

Integer $->$ [[Integer]]

- o Integer : Numero de orden de la lista de números
- o [[Integer]] : Lista de listas de enteros de n elementos permutados.

Descripción

Esta función crea una lista con todas las permutaciones posibles de las listas de orden n.

Realiza una llamada de permutacionesAux de n pasandose como parametro de lista a la llamada recursiva de permutaciones $(n-1)$ dando una llamada recursiva de permutacionesAux n (permutacionesAux $(n-1)$)

III. esCirculoPrimo

III-A. Funciones auxiliares

III-A1. divisoresHasta:

Signatura

Integer $->$ Integer $->$ [Integer]

- o Integer : Numero a saber la cantidad de divisores
- o Integer : Numero por el cual vas a dividir el primer numero
- o [Integer] : Lista de con los divisores del primer numero respecto del numero pasado como segundo parametro

Descripción

Esta función toma como base que cualquier numero dividido por uno tiene como único divisor al uno, por lo que en el caso de que como segundo parametro se le pase un 1 se devolvera un [1]. Si el segundo parametro es distinto de 1, entonces se fija primero si el resto de la división por el numero pasado da cero, si es así se agrega este elemento a una lista formada por este y los elementos que cumplan la llamada recurciva de la función. Si no es divisor entonces se hace una llamada recursiva pasandose como segundo parametro al numero anterior al numero original pasado (es decir $n-1$).

III-A2. esPrimo:

Signatura

Integer $->$ Bool

- o Integer : Numero el cual se quiere saber si es o no primo
- o Bool : Devuelve *True* si el numero pasado como parametro es primo.

Descripción

Esta función cuenta la cantidad de divisores de la lista realizada por la funcion *divisoresHasta*. Si esta lista solo tiene 2 elementos, entonces devolvera *True*.

III-A3. circuloPrimoAux:

Signatura

Integer $->$ Circulo $->$ Bool

- o Integer : Un numero cualquiera
- o Circulo : Un circulo cualquiera
- o Bool : Si es un circulo primo devuelve True.

Descripción

Esta funcion toma como caso base el de un entero cualquiera y un circulo de orden 1, en tal caso se fija si la suma de ambos elementos es primo.

Si el circulo pasado como segundo parametro tiene un orden mayor a dos, entonces se fija si la cabeza del circulo mas la cabeza de la cola del circulo son primos, si es verdadero llama a la recurcion pasando el elementro original como primer parametro y la cola del circulo como segundo parametro. Si este paso es falso, entonces devuelve *False*

III-B. Función Obligatoria

III-B1. esCirculoPrimo:

Signatura

Circulo $->$ Bool

- Circulo : Circulo al cual se lo somete a las condiciones de ser un circulo primo.
- Bool : Devuelve *True* si el circulo es primo.

Descripción

Lo unico que hace esta funcion es llamar a la funcion circuloPrimoAux pasandole como parametros la cabeza del circulo y el circulo completo.

IV. estaRepetidoPrimero

IV-A. Funciones auxiliares

IV-A1. estaRepetidoPrimeroAux:

Signatura

Circulo $->$ [Circulo] $->$ Bool

- Circulo : Circulo el cual se quiere saber si esta en una lista de Circulos
- [Circulo] : Lista de Circulos sobre el cual se quiere saber si el primer circulo esta repetido
- Bool : Devuelve *True* si el primer circulo esta incluido en la lista de circulos

Descripción

Como caso base le llegan un circulo y una lista con solo un elemento yse someten a las condiciones de la funcion *sonCirculosIguales*.

Luego, si entra un circulo y una lista de circulos con mas de un elemento, se divide en dos casos, nos fijamos primero si el circulo y la cabeza del circulo son iguales, si lo son quiere decir que el circulo esta dentro de la lista, por lo que devuelve *True*, sino, como la lista es extensa se hace una llamada recursiva de la función pasandose como parametro el circulo el cual se quiere saber si esta en la lista y la cola de la lista de circulos.

IV-B. Función Obligatoria

IV-B1. estaRepetidoPrimero:

Signatura

[Circulo] $->$ Bool

- [Circulo] : Lista de circulos
- Bool : Devuelve *True* si el primer circulo esta incluido mas de una vez en la lista.

Descripción

Esta funcion se fija si el primer elemento de la lista de circulos esta repetido dentro de dicha lista.

Para esto, toma como caso base que, si la lista tiene solo un elemento la función devolvera *False*, ya que existe un unico elemento en la lista. Luego, si la lista tiene mas de un elemento llama a la función *estaRepetidoPrimeroAux* pasandole como primer parametro a la cabeza del circulo (primer cierculo) y como segundo parametro a la lista de circulos que quedan dentro de la lista. (Ver funcionamiento de *estaRepetidoPrimeroAux*).

V. listaCirculosPrimos

V-A. Funciones auxiliares

V-A1. listaCircAux:

Signatura

[Circulo] -¿[Circulo]

- [Circulo] : Lista de circulos sometidas a condiciones
- [Circulo] : Retorna una lista de circulos primos

Descripción

Esta función se fija dentro de una lista de circulos pasada como parametro, que circulos cumplen con la condicion de ser circulos primos.

Como caso base toma que si el circulo esta vacio, entonces no habra circulos primomos, por ende devuelve la lista vacia. Luego, se llama a las funciones *estaRepetidoPrimer*, a la cual se le pasa la lista completa y *esCirculoPrimo*, a la cual se le pasa solo la cabeza, si se cumple una o la otra entonces hace un llamado recursivo de la función pasandole la cola de la lista de circulos. Si no se cumple ninguna, entonces se le agrega la cabeza de la lista al llamado recursivo de la funcion pasandose la cola de la lista como parametro.

V-B. Función Obligatoria

V-B1. listaCirculosPrimos:

Signatura

Integer $->$ [Circulo]

- Integer : Numero de elementos que deben tener los cirulos
- [Circulos] : Retorna una lista. Vacia si no tiene circulos primos, con elementos si los tiene.

Descripción

Esta función recive un entero que representa el orden de los circulos que conformaran a la lista. Estos circulos seran todas las permutaciones posibles de orden n, dentro de este se evalua cuales son primos y se los guarda en una lista que sera el retorno de la función.

Basicamente, lo que hace esta funcion es, tomar el numero entero y pasarcelo a la función *listaCircAux* a la cual se le pasara como parametro a la lista de permutaciones de orden n.

VI. contarCirculosPrimos

VI-A. Funciones auxiliares

VI-A1. longitud1:

Signatura

[Circulo] $->$ Integer

- o [Circulo] : Lista de circulos
- o Integer : Retorna la cantidad de elementos

Descripción

Esta funcion cuenta la cantidad de circulos dentro de una lista de circulos. Su funcionamiento es similar a la funcion *longCirculos*.

Toma como caso base que si la lista no tiene elementos, devuelve 0. Luego, si tiene elementos se le suma uno a la llamada recursiva passandole solo la cola de la lista de circulos.

VI-B. Función Obligatoria

VI-B1. contarCirculosPrimos:

Signatura

Integer $->$ Integer

- o Integer : Numero del orden de la lista de circulos a analizar
- o Integer : Devuelve la cantidad de circulos primos de orden n posibles

Descripción

Esta función cuenta la cantidad de circulos primos de orden n.

Para esto se le pasa a la función el orden de los circulos y lo que hace es contar mediante la función *longitud1* el resultado de pasarle como parametro el orden a la funcion *listaCIrculosPrimos*.

VII. Optativo

VII-A. Funciones auxiliares

VII-A1. ultimoElemento:

Signatura

Circulo $->$ Integer

- o Circulo : Circulo del cual se quiere conocer el ultimo elemento de la lista
- o Integer : Entero que representa al ultimo elemento del circulo.

Descripción

Esta funcion toma un tipo circulo como parametro y retorna el ultimo elemento de la lista.

Como caso base, toma a un circulo con un solo elemento, devolviendose asi ese unico elemento presente en el circulo. Luego, si el circulo tiene mas de un elemento, entonces se hace una llamada recursiva pasandose la cola del circulo pasado como parametro.

VII-A2. circuloInverso:

Signatura

Circulo $->$ Circulo

- o Circulo : Circulo el cual se quiere invertir
- o Circulo : Circulo inverso al pasado como parámetro

Descripción

Esta función toma como parametro un Circulo y devuelve su circulo inverso.

Como caso base se toma a ia lista vacia, la cual al no tener elementpo devuelve dicha lista vacia. Luego, si la lista tiene elementos, entonces se hace una llamada recursiva a la funcion pasandose como parametro la cola de la lista y a eso se le sumara la cabeza de la cola.

VII-A3. circulosEspejadosAux:

Signatura

Integer $->$ Circulo $->$ Circulo $->$ Bool

- o Integer : Representa al ultimo numero del circulo pasado como parametro.
- o Circulo : Primer circulo a ser comparado
- o Circulo : Segundo circulo a ser comparado
- o Bool : Devuelve *True* si son espejados.

Descripción

Esta funcion se fija si los circulos son espejados por simetria respecto del entero que le pases como primer parametro o s. En particular, el entero pasado como parametro, si son circulos espejados, entonces deberia corresponderse con el ultimo elemento del segundo circulo.

Como caso base si la cabeza del primer circulo coincide con el numero pasado como parametro (que representa al ultimo numero de la lista), si es asi se fija si un circulo es el inverso del otro. Si el entero no coincide con la cabeza de la primer lista, entonces hace una llamada recursiva, pasando como segundo parametro al circulo formado por anteponer la cola del circulo a la cabeza, es decir, si el circulo original era $[a, b, c, d]$ el parametro pasado como recursividad sera $[b, c, d, a]$.

VII-A4. sonCirculosEspejados:

Signatura

Circulo $->$ Circulo $->$ Bool

- o Circulo : Primer circulo a ser comparado
- o Circulo : Segundo circulo a ser comparado
- o Bool : Devuelve *True* si son circulos de permutacion espejados.

Descripción

Esta función compara dos circulos con el proposito de saber si son simetricos o rotaciones uno del otro.

Primero se fija si las longitudes de los circulos son iguales. Si lo son, entonces utiliza la funcion *circulosEspejadosAux* pasando como entero al ultimo elemento del segundo circulo y al primer circulo como segundo parametro y el segundo circulo como tercer parametro. Si no son de igual longitud, entonces devuelve *False*.

VII-A5. sonCirculosIgualesOEspejados:

Signatura

Circulo $->$ Circulo $->$ Bool

- o Circulo : Primer circulo a ser comparado
- o Circulo : Segundo circulo a ser comparado
- o Bool : Devuelve *True* si son circulos de espejados o iguales.

Descripción

Esta funcion lo que hace es tomar dos listas y evaluar

mediante las funciones *sonCirculosIguales* y *sonCirculosEspejados*. Esto lo hace fijandose si se cumple una u otra condicion mediante el comparador logico *or*.

VII-A6. repetidoPrimeroEspejadosAux:

Signatura

Circulo $->$ [Circulo] $->$ Bool

- o Circulo : Circulo a evaluar
- o [Circulo] : Lista de circulos
- o Bool : Devuelve *True* si el circulo del primer parametro esta dentro de la lista.

Descripción

Esta funcion toma al circulo del primer parametro y evalua si pertenece a la lista de circulos, con la posibilidad de que este espejado dentro de la lista de circulos.

Como caso base tenemos a una lista de circulos con un solo elemento, en ese caso evalua a los circulos mediante la funcion *sonCirculosIgualesOEspejados*. Luego, si la lista de circulos tiene mas de un elemento, entonces primero evalua si el primer elemento de la lista coincide con el circulo pasado, si es asi devuelve *True*, sino toma un paso recursivo pasando el circulo a saber si esta repetido y la cola de la lista de circulos.

VII-A7. estaRepetidoPrimeroEspejados:

Signatura

[Circulo] $->$ Bool

- o [Circulo] : Lista de Circulos
- o Bool : Devuelve True si el primer elemento del circulo esta repetido dentro de la lista de circulos

Descripción

Esta funcion revisa si el primer elemento de la lista de circulos esta repetido, siendo posible que pueda encontrarse como un circulo espejado.

Como caso base toma que si la lista de circulos tiene solo un elemento, entonces al haber solo un elemento no existira ninguno con el cual comparar por lo que retorna *False*. Luego, si la lista de circulos tiene mas de un elemento se llama a la funcion *repetidoPrimeroEspejadosAux* pasandose como parametro el primer circulo de la lista como primer parametro y como segundo parametro la cola restante de la lista de circulos.

VII-A8. lisCircPrimosEspejadosAux:

Signatura

[Circulo] $->$ [Circulo]

- o [Circulo] : Lista a evaluar
- o [Circulo] : Retorna la lista con solo los elementos primos de la lista pasada como parámetro.

Descripción

Esta funcion se fija dentro de los elementos de la lista de circulos cuelaes estan repetidos o no son primos y los descarta, dejando solo la lista de circulos primos de la lista.

Como caso base, si la lista esta vacia, entonces devuelve una lista vacia. Luego, si la lista no esta vacia evalua mediante las funciones *estaRepetidoPrimeroEspejados* si el elemento esta repetido en alguna de sus formas y con la funcion *esCirculoPrimo* negada, ya que lo comparamos con un *or*, si se cumple esta condicion entonces hace recursividad llaamandose a si misma pasandose la cola de la lista de Circulos. Si no se cumple dicha condición, entonces a al elemento le agrega la recurcion sobre la cola de la lista.

VII-A9. listaCirculosPrimosEspejados:

Signatura

Integer $->$ [Circulo]

- o Integer : Numero de orden de los circulos de la lista
- o [Circulo] : Retorna una lista solo con los circulos Primos espejados de orden del Integer pasado.

Descripción

Esta función toma un entero como parametro que refleja el orden de los circulos de la lista a tener en cuenta, compara si alguno es primo espejado y devuelve la lista solo con dichos elementos.

Al integer pasado lo pasa a la funcion *lisCircPrimosEspejadosAux* a la cual le pasa como parametro la lista conformada por las permutaciones de orden n.

VII-B. Función Optativa

VII-B1. contarCirculosPrimos:

Signatura

Integer $->$ Integer

- o Integer : Orden de los elementos de la lista
- o Integer : Cantidad de elementos primos espejados de orden n

Descripción

Esta funcion cuenta la cantidad de elementos de orden n que son primos espejados.

Esta función utiliza la funcion *longitud1* pasandole como lista parametro la lista resultante de pasarle el orden n a la función *listaCirculosPRimosEspejados*.