Ayudantía 5

Carlos Lagos - carlos.lagosc@usm.cl

Repaso Inducción

Definición

La **inducción matemática** es un método de prueba utilizado para demostrar que una proposición es cierta para todos los números naturales. El proceso involucra dos pasos: el **paso base**, donde se verifica la proposición para un valor inicial, y el **paso inductivo**.

Repaso Inducción

Inducción débil

La **inducción débil**, o simplemente inducción matemática básica, sigue el esquema clásico del principio de inducción:

1. **Paso base**: Demostrar que la proposición es verdadera para un valor base, normalmente $n = 0$ o $n = 1$.
2. **Paso inductivo**: Suponer que la proposición es verdadera para un número $n = k$ (hipótesis inductiva) y demostrar que también es verdadera para $n = k + 1$.

Repaso Inducción

Inducción fuerte

La **inducción fuerte** es una variación del principio de inducción matemática. Se puede enunciar de la siguiente manera:

Sea $P(n)$ una afirmación que depende del parámetro entero n, y supongamos que se demuestra lo siguiente:

1. **Paso base**: $P(n_0)$ es cierta para un cierto entero n_0.
2. **Paso inductivo**: Siempre que $P(k)$ sea cierto y $P(m)$ sea cierto para cualquier entero $n_0 < m < k$, se tendrá que $P(k+1)$ es cierto.

Si ambas condiciones se cumplen, entonces la afirmación $P(n)$ será cierta para todo entero $n \geq n_0$.

Repaso Invariante de Ciclo

Definición

Un **invariante de ciclo** es una propiedad o condición que permanece verdadera antes y después de cada iteración de un ciclo en un algoritmo. Se utiliza en conjunto con la **inducción matemática** para demostrar la **correctitud** de un algoritmo, asegurando que al inicio y al final de cada repetición del ciclo, la propiedad se mantiene. Esto garantiza que, al terminar el ciclo, el algoritmo haya producido el resultado correcto.

Correctitud

Ejercicio 1

Dado un algoritmo para calcular la suma de los primeros `n` números naturales, verifica la correctitud del algoritmo.

```
int suma(int n) {
    int resultado = 0;
    int i = 1;
    while (i <= n) {
        resultado = resultado + i;
        i = i + 1;
    }
    return resultado;
}
```

Correctitud

Ejercicio 2

Demuestra que la función que calcula el factorial de un número entero positivo n es correcta.

```
int factorial(int n) {
    if (n == 0) {
        return 1;
    } else {
        return n * factorial(n - 1);
    }
}
```

Correctitud

Ejercicio 3

Demuestra que la función que calcula el n-ésimo número de Fibonacci usando recursión es correcta usando inducción matemática.

```
int fibonacci(int n) {
    if (n <= 1) {
        return n;
    } else {
        return fibonacci(n - 1) + fibonacci(n - 2);
    }
}
```

Correctitud

Ejercicio 4

Dado un algoritmo de Bubble Sort para ordenar una lista de números enteros en orden ascendente, verifica la correctitud del algoritmo usando un invariante de ciclo.

Correctitud

Código

```
void bubbleSort(vector<int>& arr) {
    int n = arr.size();
    bool swapped;
    for (int i = 0; i < n - 1; i++) {
        swapped = false;
        for (int j = 0; j < n - i - 1; j++) {
            if (arr[j] > arr[j + 1]) swap(arr[j], arr[j + 1]),swapped = true;
        }
        if (!swapped) break;
    }
}
```

Ejercicio 5

Dado un algoritmo para verificar si un número entero n es primo, verifica la correctitud del algoritmo usando un invariante de ciclo.

```
bool isPrime(int x) {
    for (int d = 2; d * d <= x; d++) {
        if (x % d == 0)
            return false;
    }
    return x >= 2;
}
```

Correctitud

Ejercicio 6

Dado un algoritmo para contar el número de formas de construir la suma n lanzando un dado una o más veces, verifica la correctitud del algoritmo usando inducción.

```
int diceCombinations(int n){
    if(!n) return 1; // Caso base: si n es 0, hay una forma de hacer la suma.
    int sum = 0; // Inicializa la suma de combinaciones.
    if(n < 6){
        for(int i = 1; i <= n; i++)  sum += diceCombinations(n - i);
    } else {
        for(int i = 1; i <= 6; i++) sum += diceCombinations(n - i);
    }
    return sum; // Retornar el total de combinaciones.
}
```

Correctitud

Ejercicio 7

Dado un algoritmo que calcula el número mínimo de pasos necesarios para reducir un número entero n a cero, restando uno de sus dígitos en cada paso, verifica la correctitud del algoritmo usando inducción.

Correctitud

Código

```
int potencia(int a, int b) {
    int resultado = 1;
    for (int i = 0; i < b; i++) resultado *= a;
    return resultado;
}
int D(int n, int i) { return (n / potencia(10, i - 1)) % 10; }
int removingDigits(int n) {
    if (n == 0) return 0;
    int minimo = INT_MAX,i = 1;
    while (n / potencia(10, i - 1) > 0) {
        int digito = D(n, i);
        if (digito != 0)  minimo = min(minimo, removingDigit(n - digito) + 1);
        i += 1;
    }
    return minimo;
}
```

FIN