Programación Dinámica - Algoritmos Golosos

Leopoldo Taravilse[1]

[1] Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

Training Camp 2011

Contenidos

Contenidos

Sucesión de Fibonacci

Sucesión de Fibonacci

La sucesión de Fibonacci se define como $f_0 = 1$, $f_1 = 1$ y $f_{n+2} = f_n + f_{n+1}$ para todo $n \geq 0$

Sucesión de Fibonacci

Sucesión de Fibonacci

La sucesión de Fibonacci se define como $f_0 = 1$, $f_1 = 1$ y $f_{n+2} = f_n + f_{n+1}$ para todo $n \geq 0$

¿Cómo podemos computar el término 100 de la sucesión de Fibonacci?

Sucesión de Fibonacci

Sucesión de Fibonacci

La sucesión de Fibonacci se define como $f_0 = 1$, $f_1 = 1$ y $f_{n+2} = f_n + f_{n+1}$ para todo $n \geq 0$

¿Cómo podemos computar el término 100 de la sucesión de Fibonacci?
¿Cómo podemos hacerlo eficientemente?

Algoritmos recursivos

Definición

Un algoritmo se dice recursivo si calcula instancias de un problema en función de otras instancias del mismo problema hasta llegar a un caso base, que suele ser una instancia pequeña del problema, cuya respuesta generalmente está dada en el algoritmo y no es necesario calcularla.

Algoritmos recursivos

Definición

Un algoritmo se dice recursivo si calcula instancias de un problema en función de otras instancias del mismo problema hasta llegar a un caso base, que suele ser una instancia pequeña del problema, cuya respuesta generalmente está dada en el algoritmo y no es necesario calcularla.

Ejemplo

Para calcular el factorial de un número, un posible algoritmo es calcular Factorial(n) como Factorial($n-1$) $\times n$ si $n \geq 1$ o 1 si $n = 0$

Algoritmos recursivos

Definición

Un algoritmo se dice recursivo si calcula instancias de un problema en función de otras instancias del mismo problema hasta llegar a un caso base, que suele ser una instancia pequeña del problema, cuya respuesta generalmente está dada en el algoritmo y no es necesario calcularla.

Ejemplo

Para calcular el factorial de un número, un posible algoritmo es calcular Factorial(n) como Factorial($n-1$) $\times n$ si $n \geq 1$ o 1 si $n = 0$

Veamos como calcular el n-ésimo fibonacci con un algorimto recursivo

Cálculo Recursivo de Fibonacci

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

Cálculo Recursivo de Fibonacci

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

Notemos que fibo(n) llama a fibo($n-2$), pero después vuelve a llamar a fibo($n-2$) para calcular fibo($n-1$), y a medida que va decreciendo el parámetro que toma fibo son más las veces que se llama a la función fibo con ese parámetro.

Problemas de la recursión

- La función que usamos para calcular Fibonacci tiene un problema.

Problemas de la recursión

- La función que usamos para calcular Fibonacci tiene un problema.
- Llamamos muchas veces a la misma función con los mismos parámetros

Problemas de la recursión

- La función que usamos para calcular Fibonacci tiene un problema.
- Llamamos muchas veces a la misma función con los mismos parámetros
- ¿Podemos solucionar esto? ¿Podemos hacer que la función sea llamada pocas veces para cada parámetro?

Problemas de la recursión

- La función que usamos para calcular Fibonacci tiene un problema.
- Llamamos muchas veces a la misma función con los mismos parámetros
- ¿Podemos solucionar esto? ¿Podemos hacer que la función sea llamada pocas veces para cada parámetro?
- Para lograr resolver este problema, vamos a introducir el concepto de programación dinámica

Contenidos

Programación dinámica

Definición

La programación dinámica es un método para resolver problemas que tienen que llamar varias veces a las mismas funciones con los mismos parámetros que consiste en memorizar los valores de dichas funciones para calcularlas una sóla vez.

Programación dinámica

Definición

La programación dinámica es un método para resolver problemas que tienen que llamar varias veces a las mismas funciones con los mismos parámetros que consiste en memorizar los valores de dichas funciones para calcularlas una sóla vez.

¿Cómo hacemos para calcular una sóla vez una función para cada parámetro, por ejemplo, en el caso de Fibonacci?

Cálculo de Fibonacci mediante Programación Dinámica

```
int fibo[100];
int calcFibo(int n)
{
    if(fibo[n]!=-1)
        return fibo[n];
    fibo[n] = calcFibo(n-2)+calcFibo(n-1);
    return fibo[n];
}
int main()
{
    for(int i=0;i<100;i++)
        fibo[i] = -1;
    fibo[0] = 1;
    fibo[1] = 1;
    ínt fibo50 = calcFibo(50);
}
```

Ventajas de la Programación Dinámica

- La función que vimos recién que usa programación dinámica tiene una ventaja con respecto a la versión recursiva que vimos anteriormente.

Ventajas de la Programación Dinámica

- La función que vimos recién que usa programación dinámica tiene una ventaja con respecto a la versión recursiva que vimos anteriormente.
- Llama menos veces a cada función

Ventajas de la Programación Dinámica

- La función que vimos recién que usa programación dinámica tiene una ventaja con respecto a la versión recursiva que vimos anteriormente.
- Llama menos veces a cada función
- Para calcular calcFibo(n-1) necesita calcular calcFibo(n-2), pero ya lo calculamos antes, por lo que no es necesario volver a llamar a calcFibo(n-3) y calcFibo(n-4)

Ventajas de la Programación Dinámica

- La función que vimos recién que usa programación dinámica tiene una ventaja con respecto a la versión recursiva que vimos anteriormente.
- Llama menos veces a cada función
- Para calcular calcFibo(n-1) necesita calcular calcFibo(n-2), pero ya lo calculamos antes, por lo que no es necesario volver a llamar a calcFibo(n-3) y calcFibo(n-4)
- Así podemos calcular calcFibo(50) mucho más rápido ya que este algoritmo es lineal mientras que el anterior era exponencial.

Números combinatorios

Ejemplo

Otro ejemplo de un problema que puede ser resuelto mediante programación dinámica es el de los números combinatorios

Números combinatorios

Ejemplo

Otro ejemplo de un problema que puede ser resuelto mediante programación dinámica es el de los números combinatorios

Cómo lo calculamos

El combinatorio $\binom{n}{k}$ se puede calcular como $\frac{n!}{k!(n-k)!}$ pero generalmente como es un número muy grande, se suele tener que calcular módulo P para algún entero P que suele ser un primo bastante grande, y dividir módulo P involucra hacer muchas cuentas y no tan sencillas, por lo que lo más eficiente es calcular $\binom{n}{k}$ como $\binom{n-1}{k-1} + \binom{n-1}{k}$ salvo que k sea 0 o n en cuyo caso es 1.

Calculo recursivo del número combinatorio

Algoritmo recursivo

```
int comb(int n, int k)
{
    if(k==0||k==n)
        return 1;
    else
        return comb(n-1,k-1)+comb(n-1,k);
}
```

Calculo recursivo del número combinatorio

Algoritmo recursivo

```
int comb(int n, int k)
{
    if(k==0||k==n)
        return 1;
    else
        return comb(n-1,k-1)+comb(n-1,k);
}
```

- Este algoritmo tiene un problema. ¿Cuál es el problema?

Calculo recursivo del número combinatorio

Algoritmo recursivo

```
int comb(int n, int k)
{
    if(k==0||k==n)
        return 1;
    else
        return comb(n-1,k-1)+comb(n-1,k);
}
```

- Este algoritmo tiene un problema. ¿Cuál es el problema?
- Calcula muchas veces el mismo número combinatorio. ¿Cómo arreglamos esto?

Número combinatorio calculado con programación dinámica

Algoritmo con Programación Dinámica

```
int comb[100][100];
int calcComb(int n, int k)
{
    if(comb[n][k]!=-1)
       return comb[n][k];
    if(k==0||k==n)
        comb[n][k] = 1;
    else
        comb[n][k] = calcComb(n-1,k-1)+calcComb(n-1,k);
    return comb[n][k];
}
```

Número combinatorio calculado con programación dinámica

Algoritmo con Programación Dinámica

```
int comb[100][100];
int calcComb(int n, int k)
{
    if(comb[n][k]!=-1)
       return comb[n][k];
    if(k==0||k==n)
        comb[n][k] = 1;
    else
        comb[n][k] = calcComb(n-1,k-1)+calcComb(n-1,k);
    return comb[n][k];
}
```

Este algoritmo asume que comb está inicializado en -1 en todas sus posiciones.

Problemas

Algunos problemas para practicar programación dinámica.

- http://goo.gl/A9fs3
- https://www.spoj.pl/problems/BORW/
- http://goo.gl/qP6p8

Contenidos

Subconjuntos

Subconjuntos de un conjunto

Es muy común que aparezcan problemas en los que hay que iterar sobre los subconjuntos de un conjunto y que para calcular una función sobre un subconjunto haya que calcularla previamente sobre sus subconjuntos, para esto utilizamos programación dinámica.

Subconjuntos

Subconjuntos de un conjunto

Es muy común que aparezcan problemas en los que hay que iterar sobre los subconjuntos de un conjunto y que para calcular una función sobre un subconjunto haya que calcularla previamente sobre sus subconjuntos, para esto utilizamos programación dinámica.

Ejemplo

Un conjunto se dice de suma prima si la suma de sus elementos es un número primo. Definimos la primalidad de un conjunto A como el máximo k tal que existen $A_1, A_2, \ldots, A_k = A$ tales que son todos de suma prima con $A_i \subset A_{i+1}$, o 0 si A no es de suma prima. Dados n números enteros positivos distintos ($1 \leq n \leq 10$) calcular la primalidad del conjunto compuesto por esos n números

Máscaras de bits

- Podemos iterar sobre los subconjuntos de un conjunto usando vectores que tengan los elementos del conjunto y crear vectores con todos los subconjuntos, pero esto es muy caro en tiempo y memoria.

Máscaras de bits

- Podemos iterar sobre los subconjuntos de un conjunto usando vectores que tengan los elementos del conjunto y crear vectores con todos los subconjuntos, pero esto es muy caro en tiempo y memoria.
- Un subconjunto de un conjunto se caracteriza por tener (1) o no tener (0) a cada elemento del conjunto.

Máscaras de bits

- Podemos iterar sobre los subconjuntos de un conjunto usando vectores que tengan los elementos del conjunto y crear vectores con todos los subconjuntos, pero esto es muy caro en tiempo y memoria.
- Un subconjunto de un conjunto se caracteriza por tener (1) o no tener (0) a cada elemento del conjunto.
- Si tenemos un conjunto de 10 elementos, sus subconjuntos pueden ser representados como números entre 0 y 1023 ($2^{10} - 1$). Para cada número, si el *i*-ésimo bit en su representación binaria es un 1 lo interpretamos como que el *i*-ésimo número del conjunto está en el subconjunto representado por ese número.

Máscaras de bits

- Cuando un número representa un subconjunto de un conjunto según los ceros o unos de su representación binaria se dice que este número es una máscara de bits

Máscaras de bits

- Cuando un número representa un subconjunto de un conjunto según los ceros o unos de su representación binaria se dice que este número es una máscara de bits
- Para ver si un número *a* representa un subconjunto del subconjunto que representa un número *b* tenemos que chequear que bit a bit si hay un 1 en *a* hay entonces un 1 en *b*

Máscaras de bits

- Cuando un número representa un subconjunto de un conjunto según los ceros o unos de su representación binaria se dice que este número es una máscara de bits
- Para ver si un número *a* representa un subconjunto del subconjunto que representa un número *b* tenemos que chequear que bit a bit si hay un 1 en *a* hay entonces un 1 en *b*
- Esto se puede chequear viendo que *a* OR *b* sea igual a *b* donde OR representa al OR bit a bit

Máscaras de bits

```
int mask[1024];
int conjunto[10];
bool esDeSumaPrima(int n)
{
    int t = 0;
    for(int i=0;i<10;i++)
    if((n&(1<<i)!=0)
        t += conjunto[i];
    if(esPrimo(t))
        return true;
    return false;
    //Estamos asumiendo que
    //tenemos una función
    //que nos dice si un
    //número es primo
}
```

```
int primalidad(int n)
{
    if(!esDeSumaPrima(n))
        return 0;
    if(mask[n]!=-1)
        return mask[n];
    mask[n] = 1;
    for(int i=0;i<n;i++)
    if((i|n)==n)
        mask[n] = max(mask[
            n],primalidad(i
            )+1);
    return mask[n];
}
```

Máscaras de bits

- Estamos asumiendo que en conjunto aparecen cargados los números del conjunto y que mask empieza inicializado en -1.

Máscaras de bits

- Estamos asumiendo que en conjunto aparecen cargados los números del conjunto y que mask empieza inicializado en -1.
- Cuando iteramos sobre los subconjunto del conjunto representado por n iteramos sobre los $i < n$. ¿Porqué hacemos esto?

Máscaras de bits

- Estamos asumiendo que en conjunto aparecen cargados los números del conjunto y que mask empieza inicializado en -1.
- Cuando iteramos sobre los subconjunto del conjunto representado por n iteramos sobre los $i < n$. ¿Porqué hacemos esto?
- Si i representa un subconjunto propio del que representa n entonces $i < n$, si iteramos sobre todo i en el rango de subconjuntos entonces consideramos al conjunto como un subconjunto de si mismo y la solución sería incorrecta. ¿Se dan cuenta porqué?

Máscaras de bits

- Estamos asumiendo que en conjunto aparecen cargados los números del conjunto y que mask empieza inicializado en -1.
- Cuando iteramos sobre los subconjunto del conjunto representado por n iteramos sobre los $i < n$. ¿Porqué hacemos esto?
- Si i representa un subconjunto propio del que representa n entonces $i < n$, si iteramos sobre todo i en el rango de subconjuntos entonces consideramos al conjunto como un subconjunto de si mismo y la solución sería incorrecta. ¿Se dan cuenta porqué?
- Si $i > n$ entonces hay al menos un bit de i que está en 1 que en n está en 0, por lo que i no es un subconjunto de n.

Problemas

- http://goo.gl/iKtIH