Programação Dinâmica

henriqueramosqs
henriqueramos.qs

O que é programação dinâmica ?

O que é programação dinâmica ?

É uma técnica de resolução de problemas onde estes são divididos em subproblemas e as respostas destes são armazenadas, para evitar recalculos

O que é programação dinâmica ?

É uma técnica de resolução de problemas onde estes são divididos em subproblemas e as respostas destes são armazenadas, para evitar recalculos

(É um nome chique para recursão com tabelinha)
~Ian Parberry

Típicos problemas de Programação Dinâmica envolvem:

- Maximizar / minimizar funções
- Realizar contagens
- Dizer se algo é possível ou não de ser feito

Exemplo mais comum

Faça um programa que receba Q *queries (1<=Q<=1e5)*, cada uma contento um número N(1<=N<=1e6). Para cada *query*, calcule F(n), onde n é o n-ésimo número da sequência de Fibonacci.

- F(0) = 1
- F(1) = 1
- F(n) = F(n-1)+F(n-2), para n>=2

Solução trivial

Para cada query, calcular recursivamente F(x)

Solução trivial

Para cada query, calcular recursivamente F(x)

Complexidade: O(q*n) -> ~16 minutos

Solução esperada

Amazenar os valores de F() previamente numa tabela dp, e para cada query, imprimir dp[n]

Complexidade: O(q+n) -> ~0.011 segundos

Problema da mochila (o classico dos clássicos)

D - Knapsack 1

AtCoder is a programming contest site for anyone from beginners to experts. We hold weekly programming contests online.

AtCoder

Solução trivial

Para cada subconjunto possível, checar se ele atende os requisitos

Complexidade: O(n*2**n) -> 4*1e6 anos

Será que existe alguma solução gulosa para o problema?

Tentativa 1) Selecionar os itens com maior valor que não excedem o peso limite

N=3, W = 10
Itens disponíveis (w_i, v_i): (6, 30), (4, 20), (3, 50)

Será que existe alguma solução gulosa para o problema?

Tentativa 1) Selecionar os itens com maior valor que não excedem o peso limite

N=3, W = 10
Itens disponíveis (w_i, v_i): (6, 30), (4, 20), (3, 50)

Será que existe alguma solução gulosa para o problema?

Tentativa 1) Selecionar os itens com maior valor que não excedem o peso limite

N=3, W = 10
Itens disponíveis (w_i, v_i): (6, 30), (4, 20), (3, 50)

Ans = 50

Será que existe alguma solução gulosa para o problema?

Tentativa 1) Selecionar os itens com maior valor que não excedem o peso limite

N=3, W = 10
Itens disponíveis (w_i, v_i): (6, 30), (4, 20), (3, 50)

Ans = 70

Será que existe alguma solução gulosa para o problema?

Tentativa 2) Selecionar os itens com menor peso primeiro

N=4, W = 8
Itens disponíveis (w_i, v_i): (1, 1), (2, 2), (3, 10), (4, 7)

Será que existe alguma solução gulosa para o problema?

Tentativa 2) Selecionar os itens com menor peso primeiro

N=4, W = 8
Itens disponíveis (w_i, v_i): (1, 1), (2, 2), (3, 10), (4, 7)

Será que existe alguma solução gulosa para o problema?

Tentativa 2) Selecionar os itens com menor peso primeiro

N=4, W = 8
Itens disponíveis (w_i, v_i): (1, 1), (2, 2), (3, 10), (4, 7)

Será que existe alguma solução gulosa para o problema?

Tentativa 2) Selecionar os itens com menor peso primeiro

N=4, W = 8
Itens disponíveis (w_i, v_i): (1, 1), (2, 2), (3, 10), (4, 7)

Ans = 13

Será que existe alguma solução gulosa para o problema?

Tentativa 2) Selecionar os itens com menor peso primeiro

N=4, W = 8
Itens disponíveis (w_i, v_i): (1, 1), (2, 2), (3, 10), (4, 7)

Ans = 17

Será que existe alguma solução gulosa para o problema?

Tentativa 3) Selecionar os com melhor razão valor/peso primeiro

N=3, W=50
Itens disponíveis (w_i, v_i): (10,60), (20,100),(30,120)

Será que existe alguma solução gulosa para o problema?

Tentativa 3) Selecionar os com melhor razão valor/peso primeiro

N=3, W =50
Itens disponíveis (w_i, v_i): (10,60), (20,100),(30,120)

Será que existe alguma solução gulosa para o problema?

Tentativa 3) Selecionar os com melhor razão valor/peso primeiro

N=3, W =50
Itens disponíveis (w_i, v_i): (10,60), (20,100),(30,120)

Ans = 160

Será que existe alguma solução gulosa para o problema?

Tentativa 3) Selecionar os com melhor razão valor/peso primeiro

N=3, W =50
Itens disponíveis (w_i, v_i): (10,60), (20,100),(30,120)

Ans = 220

Vamos tentar outra abordagem 🤔

Solução esperada

dp[i][j] = melhor soma de valores com os i primeiros itens que não exceda j

Solução esperada

dp[i][j] = melhor soma de valores com os i primeiros itens que não exceda j

dp[i][j] = max(dp[i-1][j], dp[i][j-1], v[i] + dp[i-1][j-v[i]])

Solução esperada

dp[i][j] = melhor soma de valores com os i primeiros itens que não exceda j

dp[i][j] = max(dp[i-1][j], dp[i][j-1], v[i] + dp[i-1][j-v[i]])

Casos de borda: i =0, v[i]<j

Solução esperada

dp[i][j] = melhor soma de valores com os i primeiros itens que não exceda j

dp[i][j] = max(dp[i-1][j], dp[i][j-1], v[i] + dp[i-1][j-v[i]])

Casos de borda: i =0, v[i]<j

Complexidade: O(W*n)

Problema da mochila II: uma pequena adaptação

AtCoder

E - Knapsack 2

AtCoder is a programming contest site for anyone from beginners to experts. We hold weekly programming contests online.

AtCoder /

Problema da mochila II: uma pequena adaptação

Solução

dp[i][j] = menor soma de pesos para conseguir valor >=j com os i primeiros itens

dp[i][j] = max(dp[i-1][j], dp[i][j+1], v[i] + dp[i-1][j-v[i]])

Casos de borda: i =0, v[i]<j

Por que essas abordagens funcionaram?

(Ou melhor, o que é necessário para uma abordagem qualquer com DP funcionar)

Por que essas abordagens funcionaram?

(Ou melhor, o que é necessário para uma abordagem qualquer com DP funcionar)

- Sobreposição dos subproblemas
- Estrutura ótima dos subproblemas

Problema de Kadane

Dado um vetor, encontre a maior soma de um subvetor
https://cses.fi/problemset/task/1143

Problema de Kadane

Dado um vetor, encontre a maior soma de um subvetor
https://cses.fi/problemset/task/1143

Solução

Para toda posição, qual a maior soma de um vetor terminando ali?
dp[i]=max(v[i],v[i]+dp[i-1],0);

Problema da moeda

Dado uma lista de moedas, contar de quantas formas você consegue atingir a soma x? https://cses.fi/problemset/task/1635

Problema da moeda

Dado uma lista de moedas, contar de quantas formas você consegue atingir a soma x? https://cses.fi/problemset/task/1635

Solução

dp[i][j] = de quantas formas consigo fazer a soma j com os i primeiros numeros
dp[i][j]=dp[i-1][j]+dp[i-1][j-v[i]]

O que pode ser desafiador numa solução com Dp?

- Definir os estados da tabela
- Encontrar alguma propriedade que limite a quantidade de transições
- Otimizar o espaço
- Combinar a abordagem com o uso de alguma estrutura de dados

Bitmasks: um auxílio MUITO Válido

32 -> 100000   17 -> 10001   23 ->010111

Como podemos interpretar um número binário?

Bitmasks: um auxílio MUITO Válido

32 -> 100000 17 -> 10001 23 ->010111

Como podemos interpretar um número binário?

- Elementos de um conjunto (ex: 010101 possui elementos 0,2,4)

Bitmasks: um auxílio MUITO Válido

32 -> 100000 17 -> 10001 23 ->010111

Como podemos interpretar um número binário?

- Elementos de um conjunto (ex: 010101 possui elementos 0,2,4)
- Permissões (ex, 1110 = possuo efetuar as transições 3, 2, e 1)

Bitmasks: um auxílio MUITO Válido

32 -> 100000 17 -> 10001 23 ->010111

Como podemos interpretar um número binário?

- Elementos de um conjunto (ex: 010101 possui elementos 0,2,4)
- Permissões (ex, 1110 = possuo efetuar as transições 3, 2, e 1)
- Ligado/desligado

Bitmasks: alguma operações básicas

AND	OR	XOR
0101101	0101101	0101101
1111101	1111101	1111101
?	?	?

Bitmasks: alguma operações básicas

AND	OR	XOR
0101101	0101101	0101101
1111101	1111101	1111101
0101101	1111101	101000

Bitmasks: alguma operações básicas

0101101>>2 =?

0101101<<2 =?

Bitmasks: alguma operações básicas

0101101>>2 =001011

0101101<<2 =10110100

Bitmasks: alguma operações básicas

- Ver se um bit está ligado:
- Setar o i-ésimo bit para 1:
- Setar o i-ésimo bit para 0:
- Mudar o i-ésimo bit para 0(0 vira 1, 1 vira 0):
- Achar o bit menos significativo:
- Achar o bit mais significativo:

Bitmasks: alguma operações básicas

- Ver se um o i-ésimo bit está ligado: `((1<<i)&x)!=0`
- Setar o i-ésimo bit para 1: `x|=(1<<i)`
- Setar o i-ésimo bit para 0: `x&= ~(1<<i)`
- Mudar o i-ésimo bit para 0(0 vira 1, 1 vira 0): `x^=(1ll<<i)`
- Achar o bit menos significativo: `x&(~x +1)`
- Achar o bit mais significativo: `rep(bit,0,LOG)if((1<<bit)>x)return bit-1`

Bitmasks: problemas

- Counting Tilings
- Square Subsets

Bitmasks: soma por subsets

E se eu tiver uma dp onde dp[mascara] faz transições para todas suas submascaras

Ex: dp[01010] = dp[01000]+ dp[00000] +dp[00010]

Bitmasks: soma por subsets

Abordagem 1

```
for(int msk = 0;msk<(1ll<<N);msk++){
    for(int submsk = 0;submsk<(1ll<<N);submsk++){
        if((msk&submsk)==msk){
            // é submask
        }
    }
}
```

Complexidade?

Bitmasks: soma por subsets

Abordagem 1

```
for(int msk = 0;msk<(1ll<<N);msk++){
    for(int submsk = 0;submsk<(1ll<<N);submsk++){
        if((msk&submsk)==msk){
            // é submask
        }
    }
}
```

Complexidade? O(4^N)

Bitmasks: soma por subsets

Abordagem 2

```
for(int msk = 0;msk<(1ll<<N);msk++){
    for(int submsk = msk;submsk>0; submsk=msk&(submsk-1)){

    }
}
```

Complexidade?

Bitmasks: soma por subsets

Abordagem 2

```
for(int msk = 0;msk<(1ll<<N);msk++){
    for(int submsk = msk;submsk>0; submsk=msk&(submsk-1)){

    }
}
```

Complexidade? O(3^N)

Bitmasks: soma por subsets

Solução ótima: sum over subsets

Seja dp[x][i] =contribuição de todos os números que são submascaras de x mas que só diferem nos primeiros i bits?

Ex: dp[**101**1010][3] = {**101**0000, **101**0010, **101**100, **101**1010}

Bitmasks: soma por subsets

Solução ótima: sum over subsets

Como são as transições?

Se o i-ésimo bit estiver desligado: dp[msk][i]=dp[msk][i-1]

Bitmasks: soma por subsets

Solução ótima: sum over subsets

Como são as transições?

Se o i-ésimo bit estiver desligado: dp[msk][i]=dp[msk][i-1]

Se o i-ésimo bit estiver ligado: dp[msk][i] -> dp[msk^(i<<i)][i-1], dp[msk][i-1]

Bitmasks: soma por subsets

Solução ótima: sum over subsets

Como são as transições?

Se o i-ésimo bit estiver desligado: dp[msk][i]=dp[msk][i-1]

Se o i-ésimo bit estiver ligado: dp[msk][i] -> dp[msk^(i<<i)][i-1], dp[msk][i-1]

Complexidade?

Bitmasks: soma por subsets

Solução ótima: sum over subsets

Como são as transições?

Se o i-ésimo bit estiver desligado: dp[msk][i]=dp[msk][i-1]

Se o i-ésimo bit estiver ligado: dp[msk][i] -> dp[msk^(i<<i)][i-1], dp[msk][i-1]

Complexidade? O(N*2^N)

Bitmasks: soma por subsets

Exemplo

Problem - E

Codeforces. Programming competitions and contests, programming community

Codeforces

Otimizações de memória

Redução de Estados

Dado uma lista de números, de quantas formas consigo separálos em dois grupos onde sum(a)%sum(b)==0?
1<=n<=100
1<=a[i]<=100

Redução de Estados

Dado uma lista de números, de quantas formas consigo
separálos em dois grupos onde sum(a)%sum(b)==0?
1<=n<=100
1<=a[i]<=100

dp[i][sum_a][sum_b]= dp[i+1][sum_a+ v[i]][sum_b] +dp[i+1][sum_a][sum_b+ v[i]]

Redução de Estados

Dado uma lista de números, de quantas formas consigo
separálos em dois grupos onde sum(a)%sum(b)==0?
1<=n<=100
1<=a[i]<=100

dp[i][sum_a][sum_b]= dp[i+1][sum_a+ v[i]][sum_b] +dp[i+1][sum_a][sum_b+ v[i]]

Eu preciso mesmo armazenar as duas somas?

Redução de Estados

Dado uma lista de números, de quantas formas consigo
separálos em dois grupos onde sum(a)%sum(b)==0?
1<=n<=100
1<=a[i]<=100

dp[i][sum_a][sum_b]= dp[i+1][sum_a+ v[i]][sum_b] +dp[i+1][sum_a][sum_b+ v[i]]

Eu preciso mesmo armazenar as duas somas?
NÃO! Dado sum_a e o índice, eu sei quanto há em sum_b

Redução de Estados

Dado uma lista de números, de quantas formas consigo
separálos em dois grupos onde sum(a)%sum(b)==0?
1<=n<=100
1<=a[i]<=100

dp[i][sum_a]= dp[i][sum_a] + dp[i][sum_a+ v[i]]

Redução de Estados

Nem sempre eu preciso armazenar todos os estados, às vezes alguns são inferidos por outros

Redução de Estados - Par ou ímpar

https://cses.fi/problemset/task/2229

Redução de Estados - Par ou ímpar

https://cses.fi/problemset/task/2229

Às vezes, uma solução passa em complexidade, mas não em memória. Se dp[i] consumir apenas de dp[i-1], podemos transformar d[n] em dp[2] (0= índice par, 1 = índice impar)

Por hoje é isso😊

Link do contest