Desafios de Programação
Programação Dinâmica

Wladimir Araújo Tavares [1]

[1]Universidade Federal do Ceará - Campus de Quixadá

30 de março de 2017

O que é programação dinâmica?

- Wikipédia: Método para resolver problemas complexos dividindo-os em subproblemas mais simples.

Passos para resolver problemas de PD

- Definir os subproblemas
- Escrever a relação de recorrência que relaciona os subproblemas
- Reconhecer e resolver os casos bases

Coin Change

- Problema: Dado n, encontre o número de diferentes maneiras de escrever n como soma de 1, 3 e 4
- Exemplo: para n=5, a resposta é 6

$$
\begin{aligned}
5 &= 1+1+1+1+1 \\
&= 1+1+3 \\
&= 1+3+1 \\
&= 3+1+1 \\
&= 1+4 \\
&= 4+1
\end{aligned}
$$

Coin Change

- Definir subproblemas
  - Seja D_n o número de maneira de escrever n como a soma de 1, 3 e 4
- Encontre a recorrência
  - Considere um possível solução $n = x_1 + \ldots + x_m$
  - Se $x_m = 1$ então a soma do resto dos termos é n-1.
  - Assim, todas as somas $(n = x_1 + \ldots + x_m)$ terminadas com $x_m = 1$ é igual a D_{n-1}

Coin Change

- Encontre a recorrência
  - $D_n = D_{n-1} + D_{n-3} + D_{n-4}$
- Resolver os casos bases
  - $D_0 = 1$
  - $D_1 = D_2 = 1$
  - $D_3 = 2$

Implementação

```
D[0] = D[1] = D[2] = 1;
D[3] = 2;
for(int i = 4; i <= n; i++){
 D[i] = D[i-1] + D[i-3] + D[i-4];
}
```

LIS

- Problema: Dado uma sequência $a[0 \ldots n-1]$, encontre a maior subsequência crescente de a.
- Exemplo: a[] = {2,5,3,8,4,6}.

Subsequência crescente de a
2
2,5
2,5,8
2,5,6
2,3,8
2,3,4
2,3,4,6
5
5,8
5,6
⋮

LIS

- Definir subproblemas
  - Seja C_i o tamanho da maior subsequência crescente de $a[0 \ldots i]$ que contém a_i como último elemento.
- Encontre a recorrência
  - $C_i = max\{C_j + 1 | a_j < a_i, 0 \leq j \leq i\}$
- Resolva os casos bases
  - $C_0 = 1$

Implementação

```
C[0] = 1;
for(int i = 1; i < n; i++){
    C[i]=1;
    for(int j = 0; j < i; j++){
        if( a[j] < a[i])
            C[i] = C[i] > C[j]+1 ? C[i] : C[j]+1;
    }
}
ans = 0;
for(int i = 0; i < n; i++) ans = ans > C[i] ? ans : C[i];
```

Implementação

```
set <int> st;
set <int>::iterator it;

st.clear();
for(int i = 0; i < n; i++){
     st.insert(a[i]);
     it = st.find(a[i]);
     it++;
     if( it != st.end() )
          st.erase(it);
}
ans = st.size();
```

LCS

- Problema: Dado duas strings x e y, encontre o tamanho da maior subsequência comum (LCS)
- Exemplo:
  - x: A**BC**BD**AB**
  - y: **B**D**CAB**C
  - BCAB é a maior subsequência encontrada e o seu tamanho é 4

LCS

- Defina os subproblemas
  - Seja D_{ij} o comprimento da LCS de $x[1 \ldots i]$ e $y[1 \ldots j]$
- Encontre a recorrência
  - Se $x_i = y_j$, então o caractere está na LCS
    - ★ $D_{ij} = D_{i-1,j-1} + 1$
  - Caso contrário, x_i ou y_j não contribuem para o LCS, então ele pode ser removido
    - ★ $D_{ij} = max(D_{i-1,j}, D_{i,j-1})$
- Resolva os casos bases:
  - $D_{i0} = D_{0j} = 0$

Implementação

```
int D[1001][1001];

n = strlen(x);
m = strlen(y);

for(int i = 0; i <= n; i++) D[i][0] = 0;
for(int j = 0; j <= m; j++) D[0][j] = 0;

for(int i = 1; i <= n; i++)
    for(int j = 1; j <= m; j++)
        if( x[i-1] == y[j-1] )
            D[i][j] = D[i-1][j-1] + 1;
        else
            D[i][j] = max(D[i-1][j], D[i][j-1]);
```

LCS

	ε	B	D	C	A	B	C
ε	0	0	0	0	0	0	0
A	0	0	0	0	1	1	1
B	0	1	1	1	1	2	2
C	0	1	1	2	2	2	3
B	0	1	1	2	2	3	3
D	0	1	2	2	2	3	3
A	0	1	2	2	3	3	3
B	0	1	2	2	3	4	4

Reduzindo os requisitos de memória

```
int D[2][1001];
int ans;
n = strlen(x);
m = strlen(y);

for(int j = 0; j <= m; j++) D[0][j] = 0;

for(int i = 1; i <= n; i++){
    int ii = i&1;
    D[ii][0] = 0;
    for(int j = 1; j <= m; j++){
        if( x[i-1] == y[j-1] )
            D[ii][j] = D[1-ii][j-1] + 1;
        else
            D[ii][j] = max(D[1-ii][j], D[ii][j-1]);
    }
}
ans = D[n&1][m];
```

Subset sum

- Problema: Dado um conjunto de n números a_i cuja a soma é L e $K \leq L$. Existe um subconjunto de números a_i cuja soma é K?
- Exemplo: a[] = 1,3,4,7

K	RESPOSTA	PROVA
8	SIM	{1,7}
9	NÃO	
10	SIM	{7,3}

Subset Sum

- Defina os subproblemas
  - Seja $M_{ij} = \begin{cases} 1, & \text{se existe um subconjunto } a[1 \ldots i] \text{ cuja soma é j} \\ 0, & \text{caso contrário} \end{cases}$
- Encontre a recorrência
  - Seja $M_{ij} = M_{i-1,j} || M_{i-1,j-a[i]}$
- Resolva os casos bases:
  - $M_{i0} = 1, 0 \leq i \leq n$
  - $M_{0j} = 0, 1 \leq j \leq K$

Implementação

```
for(int i=0; i <= n; i++) m[i][0] = 1;
for(int j=1; j <= K; j++) m[0][j] = 0;

for(int i = 1; i <= n; i++){
    for(int j = 1; j <= K; j++){
        if( j < a[i-1])
            m[i][j] = m[i-1][j];
        else
            m[i][j] = m[i-1][j] || m[i-1][j-a[i-1]];
    }
}

ans = M[n][K];
```

Implementação

```
for(int i=0; i <= n; i++) M[i][0] = 1;
for(int j=1; j <= K; j++) M[0][j] = 0;

for(int i = 1; i <= n; i++){
    for(int j = 1; j <= K; j++){
        if( j < a[i-1])
            M[i][j] = M[i-1][j];
        else
            M[i][j] = M[i-1][j] || M[i-1][j-a[i-1]];
    }
}

ans = M[n][K];
```

Tabela de subproblemas

	0	1	2	3	4	5	6	7	8	9
0	1	0	0	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0	0	0
2	1	1	0	1	1	0	0	0	0	0
3	1	1	0	1	1	1	0	1	1	0
4	1	1	0	1	1	1	0	1	1	0