ESTRUTURAS DE DADOS II

MSC. DANIELE CARVALHO OLIVEIRA

DOUTORANDA EM CIÊNCIA DA COMPUTAÇÃO - USP

MESTRE EM CIÊNCIA DA COMPUTAÇÃO – UFU

BACHAREL EM CIÊNCIA DA COMPUTAÇÃO - UFJF

PROGRAMAÇÃO DINÂMICA

3 PROGRAMAÇÃO DINÂMICA

"Dynamic programming is a fancy name for [recursion] with a table. Instead of solving subproblems recursively, solve them sequentially and store their solutions in a table."

Ian Parberry, Problems on Algorithms

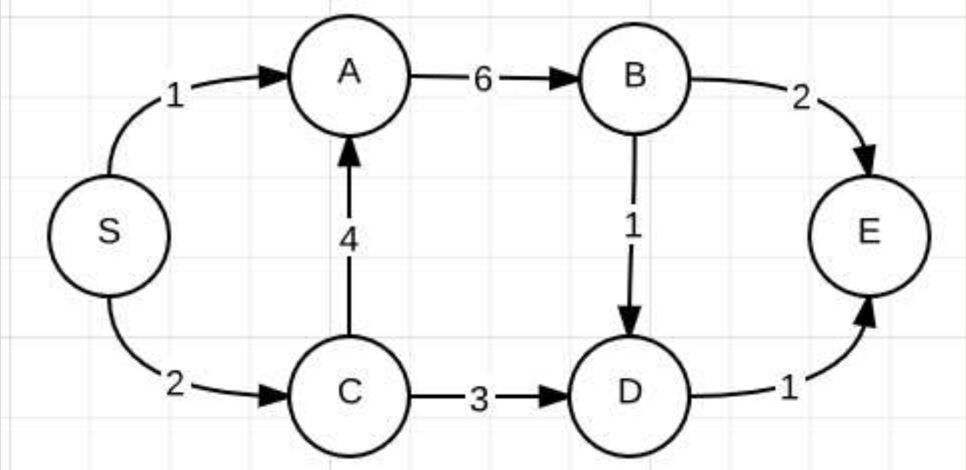
INTRODUÇÃO

- Aplicamos, em geral, a programação dinâmica a problemas de otimização

- O desenvolvimento de um algoritmo de programação dinâmica segue uma sequência de quatro etapas:
  - Caracterizar a estrutura de uma solução ótima
  - Definir recursivamente o valor de uma solução ótima
  - Calcular o valor de uma solução ótima, normalmente de baixo para cima
  - Construir uma solução ótima com as informações calculadas.

5 EXEMPLO

- Encontrar a distância entre 2 vértices em um grafo dirigido.
  - dist(u,v) = tamanho do menor caminho entre u e v.



PROGRAMAÇÃO DINÂMICA

- Problema: Coleção Ordenada de subproblemas $\{P_1, P_2, ..., P_n\}$

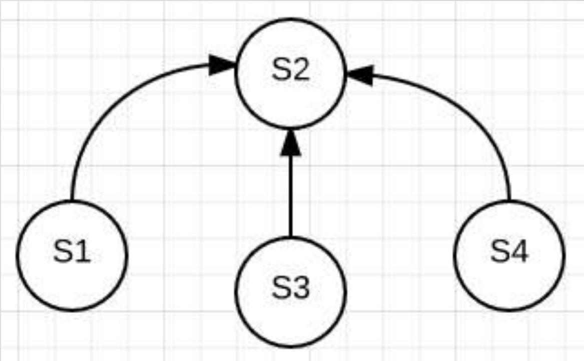
- Função ou uma relação mostrando como resolver um subproblema P_k, tendo as respostas dos subproblemas $P_1, P_2, ..., P_{k-1}$

7 QUANDO UTILIZAR PD?

- Problema é composto de subproblemas
- Existe uma ordem nos subproblemas (S*j* depende dos subproblemas S*i* | i < j)

- Todo problema resolvido por programação dinâmica pode ser reduzido a um problema de grafos acíclicos dirigidos (dag)

- Existe aresta de $S_i \rightarrow S_j$, se S_i vem antes de S_j na ordenação dos subproblemas



EXEMPLO: SUBSEQUÊNCIA CRESCENTE MAXIMAL (LIS)

- Entrada: $S = \{a_1, \dots, a_n\}$
- Saída: Encontrar todas as LIS de S
- Exemplo: $S = \{5,2,8,6,3,6,9,7\}$

```
SSDC-MAX-PROG-DIN (A, n)
para m := 1 até n
    c[m] := 1
    para i := m−1 decrescendo até 1
        se A[i] ≤ A[m] e c[i] + 1 > c[m]
            c[m] := c[i] + 1
devolva c[1..n]
```

9 EXEMPLO: PROBLEMA DO TROCO

- Algoritmo Guloso?
  - E para moedas de 1, 10, 25 e 50, e um troco de 30? Funciona?

- Há outra maneira de resolver o problema. Por exemplo, para encontrar o troco ótimo de 30 centavos, basta encontrar a melhor solução dentre as 3 abaixo:
  - uma moeda de 1 centavo e a melhor solução para 29 centavos;
  - uma moeda de 10 centavos e a melhor solução para 20 centavos;
  - uma moeda de 25 centavos e a melhor solução para 5 centavos.
- Antes, resolvem-se todos os subproblemas menores (troco menor que 30 centavos), armazenando-se essas soluções.
- Em seguida, resolve-se o problema do troco de 30 centavos, encontrando-se a melhor solução dentre as 3 possíveis.

EXEMPLO

- Moedas de 1, 5 e 10 centavos.
- Trocos ótimos até 15 centavos.
- Suporemos que o vetor de moedas esteja ordenado, embora isso não seja necessário (nesse caso, seria encontrada outra solução também ótima).

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
QUANT	0	1	2	3	4	1	2	3	4	5	1	2	3	4	5	2
ULTIMA	0	1	1	1	1	5	5	5	5	5	10	10	10	10	10	10

```
MakeChange(moedas, troco){ // moedas: vetor de moedas disponíveis (menor é de 1 centavo)
  quant[0] = 0; // solução ótima para troco de valor 0
  ultima[0] = 0; // última moeda dessa solução
  for (cents = 1; cents <= troco; cents++) {
     quantProv = cents; // solução provisória: todas de 1 centavo
     ultProv = 1; // última moeda dessa solução
     for (j = 0; j < length(moedas); j++) {
        if (moedas[j] > cents) continue; // essa moeda não serve
        if (quant[cents - moedas[j]] + 1 <= quantProv) {
          quantProv = quant[cents - moedas[j]] + 1;
          ultProv = moedas[j];
     }}
     quant[cents] = quantProv; // solução para troco=cents
     ultima[cents] = ultProv; // última moeda dessa solução
}}
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
QUANT	0	1	2	3	4	1	2	3	4							
ULTIMA	0	1	1	1	1	5	5	5	5							

12 EXERCÍCIO

- Calcular a função de PD para Subsequência Contígua de Soma Máxima
  - Ex: S = { 5, 15, -30, 10, -5, 40, 10}

Algoritmo de Kadane

```
int maxSumSubArray(vector<int> &A)
{
    int n = A.size(); // Size of the array
    int local_max = 0;
    int global_max = INT_MIN; // -Infinity

    for (int i = 0; i < n; i++)
    {
        local_max = max(A[i], A[i] + local_max);
        if (local_max > global_max)
        {
            global_max = local_max;
        }
    }

    return global_max;
}
```

Trabalho

- Problema da Mochila 0/1
- Problema da Multiplicação de Cadeia de Matrizes
- Problema do Corte de Hastes
- Maior Subsequência Comum (LCS)
- Bellman Ford
- Problema da Soma dos Subconjuntos
- Problema do Conjunto Independente Máximo
- Escalonamento de Intervalos

Extra

- URI:
  - 1034; 1100; 1203; 1210; 1237; 1265; 1269; 1286; 1288; 1310; 1319; 1341; 1350; 1372; 1408; 1475; 1487; 1495; 1522; 1558; 1592; 1645; 1655; 1666; 1672; 1690; 1689; 1697; 1700; 1707; 1718; 1738; 1751; 1767; 1838; 1915; 1932; 1970; 1976

FIM DA AULA 19

Próxima aula:

PD: Problema de Edição e Shortest Reliable Path