Programação Dinâmica: de conceitos básicos a técnicas avançadas

Tiago Reis

8 de agosto de 2017

Introdução

- Técnica desenvolvida por Richard Bellman (1953).
- Paradigma envolvido em diversos tipos de problemas.
- Onipresente em competições de programação.

Introdução

- Ideia: resolução de um problema a partir da solução de subproblemas.
- Se os subproblemas se sobrepõem, podemos armazenar essas soluções e reutilizá-las: *memoização*.
- Em outras palavras, podemos construir uma solução recursiva em que a mesma chamada da função é feita várias vezes, mas computada apenas uma.

Exemplo 1

- Temos várias crianças em uma fila, cada uma segurando uma quantidade de balões.
- A primeira e a segunda criança da fila seguram um balão cada.
- Da terceira em diante, cada uma segura um número de balões igual à soma dos números de balões das duas crianças imediatamente à frente dela.
- Quantos balões a n-ésima criança tem?

Exemplo 1

```
int balloons(int pos) {
  if (pos == 1 || pos == 2) return 1;

  int ans = balloons(pos - 1) + balloons(pos - 2);
  return ans;
}
```

Exemplo 1

```
int memo[MAXN] = {-1, -1, ..., -1};

int balloons(int pos) {
  if (pos == 1 || pos == 2) return 1;
  if (memo[pos] != -1) return memo[pos];

  int ans = balloons(pos - 1) + balloons(pos - 2);
  return memo[pos] = ans;
}
```

Exemplo 2

- Dado um vetor de inteiros, devemos escolher um subconjunto desses inteiros de forma que não haja dois elementos consecutivos e que a soma deles seja máxima.
- Para o vetor $v = (8, 4, 2, 7, 1, -3, 3)$, por exemplo, devemos escolher os números 8, 7 e 3, que somam 18.
- Uma vez que você decide incluir ou não um dos elementos do vetor no subconjunto escolhido, o problema se reduz a uma instância menor do problema original.

Exemplo 3

- Um noivo precisa adquirir seu traje de casamento usando no máximo M dinheiros ($1 \leq M \leq 200$).
- Há C opções de vestuário, cada uma com K modelos, e ele deve escolher um modelo para cada tipo de vestuário. Restrições: $1 \leq C, K \leq 20$.
- Qual é o conjunto mais caro que ele pode obter sem estourar o orçamento?
- Uma estratégia gulosa funcionaria? Busca completa é viável? Temos que recorrer a programação dinâmica?

Exemplo 4

- Em um armário de gavetas, cada gaveta pode ser trancada individualmente.
- Uma gaveta é considerada segura se está trancada e é a gaveta mais de cima do armário ou se está trancada e a gaveta imediatamente acima dela também está.
- Em um armário de N gavetas, de quantas formas podemos tornar K delas seguras?

Exemplo 5: usando bitmasks para representar o estado

- Problema do caixeiro viajante: TSP.
- Devemos encontrar o ciclo de menor custo que passe exatamente uma vez por todos os vértices.
- Como armazenar o estado? Como saber quais vértices já visitamos? Com bits!

Exemplo 5: usando bitmasks para representar o estado

```
int tsp(int pos, int bitmask) { //bitmask marks visited nodes
  if (bitmask == (1 << (n + 1)) - 1)
    return dist[pos][0]; //return trip to close the loop
  if (memo[pos][bitmask] != -1)
    return memo[pos][bitmask];

  int ans = 2000000000;
  for (int nxt = 0; nxt <= n; nxt++) // O(n) here
    // if nxt is not visited
    if (nxt != pos && !(bitmask & (1 << nxt)))
      ans = min(ans, dist[pos][nxt] +
                tsp(nxt, bitmask | (1 << nxt)));
  return memo[pos][bitmask] = ans;
}
```

Exemplo 6

- Contar o número de soluções de uma equação diofantina.

$$x_1 + x_2 + \cdots + x_k = N$$

- Solução via programação dinâmica e via análise combinatória.
- Quais as vantagens e desvantagens de cada método?

Exemplo 7

- Contar o número de árvores binárias não rotuladas com n vértices.
- E se as árvores forem rotuladas?
- A resposta é uma sequência famosa em combinatória. Os primeiros números são: 1, 1, 2, 5, 14, 42, 132, 429, ...

Exemplo 7

- Números de Catalan.
- Número de expressões válidas com n pares de parênteses.
- Número de caminhos monotônicos em um reticulado $n \times n$ do canto inferior esquerdo ao canto superior direito, sem passar acima da diagonal.
- Número de triangulações de um polígono de $n + 2$ lados.

Exemplo 8: "divide and conquer"

- Há uma prisão com L celas em uma linha, cada uma com um criminoso de periculosidade C_i.
- Há G guardas, cada um deve ser responsável por um segmento contíguo de prisioneiros. O risco de que o prisioneiro i escape é C_i vezes o número de pessoas que o guarda responsável por i está guardando.
- Minimize o risco total.
- Custo de particionar os l primeiros prisioneiros em g grupos é

$$f(g, l) = \min_{0 \leq k \leq l} \{f(g-1, k) + cost(k+1, l)\}.$$

Exemplo 8: "divide and conquer"

```
void fill(int g, int l1, int l2, int p1, int p2) {
    // calculating P[g][l] and F[g][l] for l1 <= l <= l2,
    // knowing that p1 <= P[g][l] <= p2

    if (l1 > l2) return;

    int lm = (l1 + l2) / 2;
    P[g][lm] = -1;
    F[g][lm] = INF;
    for (int k = p1; k <= p2; k++) {
        ll new_cost = F[g-1][k] + cost(k+1,lm);
        if (F[g][lm] > new_cost) {
            F[g][lm] = new_cost;
            P[g][lm] = k;
        }
    }

    fill(g, l1, lm-1, p1, P[g][lm]);
    fill(g, lm+1, l2, P[g][lm], p2);
}
```

Exemplo 9: "convex hull trick"

- Relação recursiva do tipo

$$dp[i] = \min_{j<i}\{dp[j] + m[j] \cdot x[i]\}$$

- Qual a forma de interpretar essa expressão geometricamente?

Outros tipos de problemas

- PD em árvore: maximizar soma de valores em nós sem que um nó e seu pai sejam escolhidos ao mesmo tempo.
- PD sobre dígitos: contar quantos números menores que um limite U têm soma dos dígitos igual a um determinado número.

Outros tipos de problemas

- PD em árvore: maximizar soma de valores em nós sem que um nó e seu pai sejam escolhidos ao mesmo tempo.
- PD sobre dígitos: contar quantos números menores que um limite U têm soma dos dígitos igual a um determinado número.
- Muitos outros. Treine!