Desafios de Programação
Programação Dinâmica

Wladimir Araújo Tavares [1]

[1]Universidade Federal do Ceará - Campus de Quixadá

23 de abril de 2018

1 Programação Dinâmica

Problema da atribuição

- Problema: Calcular o número de diferentes atribuições de n diferentes tópicos para n diferentes estudantes tal que todos estudantes recebem exatamente um tópico que ele gosta.
- Entrada: número de pessoas e matriz de preferência
- Saída: número de atribuições possíveis.

Entrada

- Entrada

  11
  1 0 0 1 0 0 0 0 0 1 1
  1 1 1 1 1 0 1 0 1 0 0
  1 0 0 1 0 0 1 1 0 1 0
  1 0 1 1 1 0 1 1 0 1 1
  0 1 1 1 0 1 0 0 1 1 1
  1 1 1 0 0 1 0 0 0 0 0
  0 0 0 0 1 0 1 0 0 0 1
  1 0 1 1 0 0 0 0 0 0 1
  0 0 1 0 1 1 0 0 0 1 1
  1 1 1 0 0 0 1 0 1 0 1
  1 0 0 0 1 1 1 1 0 0 0

- Saída

  7588

Estrutura Recursiva

- $dp(i, mask)$, onde $0 \leq i \leq n-1$ representa um estudante e $mask$ uma máscara de bits cujo i-ésimo bit representa se a i-ésima tarefa foi atribuída ou não.
- $dp(i, mask) = \sum_{j \in mask \wedge gosta[i][j]} dp(i+1, mask^{(1 << j)})$

Algoritmo Recursivo

```
int assign(int k, int mask ){
   if( mask == 0) return 1;
   if( dp[mask] != -1) return dp[mask];
   long long int contador = 0LL;
   for(int j = 0; j < n; j++){
            if( (mask & (1<<j) ) != 0 ){
                     if( cost[k][j] == 1){
                              contador += assign(k+1, mask^(1<<j) );
                     }
            }
   }
   dp[mask] = contador;
   return dp[mask];
}
```

Algoritmo Bottom-up

```
dp.resize(1<<n);
dp[0] = 1;
for(int mask = 1; mask < (1 <<n); mask++){
  int x = __builtin_popcount(mask) - 1;
  dp[mask] = 0;
  for(int j = 0; j < n; j++){
    if( cost[x][j] && ( mask & (1<<j) )!= 0){
          dp[mask] += dp[mask^(1<<j) ];
    }
  }
}
cout << dp[(1<<n)-1] << endl;
```

Caminho Hamiltoniando

- Problema: Um caminho hamiltoniano é um caminho que percorre todos os vértices de um grafo não repetindo nenhum vértice. Encontre um caminho hamiltoniano com o menor custo, onde o custo do caminho é dado pelo somatório dos pesos das arestas.
- Entrada: Um grafo.
- Saída: peso do caminho com o menor custo.

Estrutura Recursiva

- $D[mask][v]$ menor custo de caminho hamiltoniano que visita todos os vértices na máscara de bits mask e terminando em v. Se o i-ésimo bit representa se o vértice i está presente ou não.
- $D[mask][v] = min_{j \in mask} D[mask \text{ xor } (1 << j)][j] + custo[j][v]$
- $D[\emptyset][v] = 0$
- Solução $= min_{j \in \{0 \ldots n-1\}} D[((1 << n) - 1) \text{ xor } 1 << j][j]$

Estrutura Recursiva

```
int tsp(int mask, int v){

    if( mask == 0 ) return 0;

    if( dp[mask][v] != -1) return dp[mask][v];


    int min = INF;
    for(int i = 0; i < n; i++)
    {
        if( (mask & (1<<i)) != 0 ){
            int q = tsp(mask ^ (1<<i), i) + dist[i][v];
            if( q < min ) min = q;
        }

    }
    dp[mask][v] = min;

    return dp[mask][v];
}
```

Problema Naruto

A Akatsuki está planejando atacar a Vila para capturar Naruto. O Chefe da Aldeia sabe sobre o plano e o número de membros, N, da Akatsuki que vão atacar. Então, o chefe planejou uma emboscada. O Chefe da Aldeia selecionou uma equipe de N membros (um para cada membro da Akatsuki). Cada membro pode atacar exatamente um membro da Akatsuki e um membro da Akatsuki NÃO é atacado por mais de um membro. Na emboscada haverá N lutas (um de cada N membros e N membros da Akatsuki). Você é o chefe da aldeia. Você conhece as posições (coordenadas x e coordenadas y) de todos os membros da Akatsuki e membros do seus guerreiros. Sua tarefa é designar cada guerreiro para exatamente um membro da equipe da Akatsuki de tal forma que a soma da distância entre eles seja mínima. Distância entre dois pontos e será igual a $|x_1 - x_2| + |y_1 - y_2|$.