Técnicas para Programação Competitiva

Paradigmas de Resolução de Problemas

Samuel da Silva Feitosa

Aula 7

Introdução

- Nesta aula vamos discutir 3 paradigmas de resolução de problemas que são usados comumente em competições de programação.
  - Complete Search, Divisão e Conquista e Algoritmos Gulosos.
- Todos são importantes ferramentas para os competidores.
  - Apesar da maioria das soluções de força bruta não serem suficientes para as competições, são importantes para obter mais compreensão do problema.

Complete Search

Força Bruta

Complete Search

- Também conhecido como força bruta ou recursive backtracking.
  - É um método para resolver um problema por percorrer inteiramente o espaço de busca para obter a solução desejada.
  - Durante a busca é possível não explorar algumas partes do espaço de busca, uma vez que elas forem determinadas como soluções inválidas.

Força Bruta

- Pode ser usado para resolver quase todos os problemas.
  - A ideia é gerar todas as soluções possíveis para o problema usando força bruta e então selecionar a melhor solução ou contar o número de soluções, dependendo do problema.
  - Geralmente não é suficiente para solucionar os problemas de programação competitiva em tempo aceito pelos juízes automatizados.

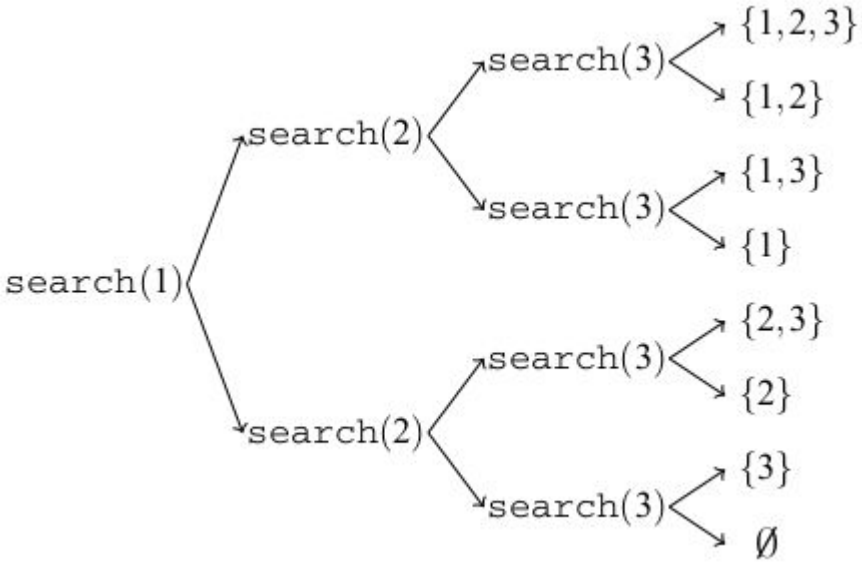
Backtracking

- Um algoritmo de backtracking inicia com uma solução vazia e estende a solução passo a passo.
  - A busca é executada **recursivamente** em todas as maneiras possíveis de uma solução ser construída.
- Alguns exemplos:
  - Geração de subconjuntos, geração de permutações, etc.

Geração de Subconjuntos

- Recursivamente gera todos os subconjuntos de até um determinado valor.

```
void search(int k) {
      if (k == n+1) {
            // process subset
      } else {
            // include k in the subset
            subset.push_back(k);
            search(k+1);
            subset.pop_back();
            // don't include k in the subset
            search(k+1);
      }
}
```



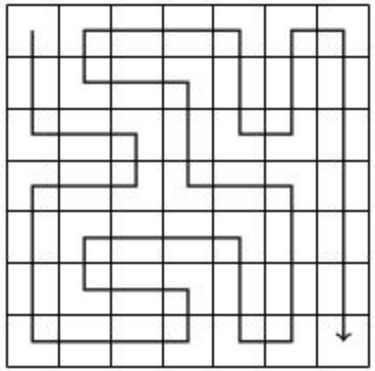
Geração de Permutações

- A biblioteca padrão do C++ possui uma função com esta finalidade.

```
for (int i = 1; i <= n; i++) {
    permutation.push_back(i);
}
do {
    // process permutation
} while (next_permutation(permutation.begin(),
                          permutation.end()));
```

Caminhos em um Grid

- Calcular o número de caminhos em um Grid *n* x *n* partindo do topo esquerdo até a extremidade inferior direita.
  - Cada quadro pode ser visitado uma única vez.
  - Em um grid de 7 x 7, existem 111712 caminhos.



Divisão e Conquista

Divisão e Conquista

- É um paradigma de resolução de problemas no qual um problema se torna mais simples a partir da divisão dele em partes menores, para então combinar cada parte.
- Normalmente executa os seguintes passos:
  - Divide o problema original em subproblemas, usualmente pela metade.
  - Encontra as subsoluções para cada subproblema, as quais devem ser mais simples.
  - Se necessário, combina as subsoluções para obter uma solução para o problema principal.
- Exemplos:
  - Vários algoritmos de ordenação, busca binária, etc.

Greedy Problems

Algoritmos Gulosos

Algoritmos Gulosos

- Constrói uma solução sempre fazendo a escolha que parece melhor em dado momento.
  - Nunca volta atrás em uma escolha, construindo diretamente uma solução a partir das decisões locais.
  - Por isso, geralmente são implementados de forma muito eficiente.
- A dificuldade neste paradigma é encontrar uma estratégia que sempre produza uma solução ótima para o problema.
  - As escolhas ótimas locais também devem ser escolhas ótimas globais.
- Exemplos:
  - **Problema da moeda**, Problemas de agendamento, Tarefas e prazos, etc.

Problema da Moeda

- Dado um conjunto de moedas ou cédulas, a tarefa é formar uma quantidade de dinheiro *n* usando as mesmas.
  - Considerando os valores:
    - {1, 2, 5, 10, 20, 50, 100, 200}
    - e *n* = 520, é preciso pelo menos 4 moedas.
    - 200 + 200 + 100 + 20
- O algoritmo guloso sempre seleciona a moeda ou cédula com o maior valor possível.
  - Este algoritmo sempre funciona com esse conjunto de moedas/cédulas.
  - Porém, nem sempre funciona para outros conjuntos de moedas/cédulas.

Considerações Finais

- Nesta aula vimos 3 técnicas para resolução de problemas de programação competitiva.
- Apesar de serem simples, são úteis para resolver diversos problemas.
  - Outras técnicas mais elaboradas serão apresentadas nas próximas aulas.