Programação Gulosa em JavaScript

Marcos Vinicius Silva

Dezembro, 2023

1 Introdução

Este relatório descreve a implementação de um algoritmo baseado em programação gulosa em JavaScript para resolver um problema de otimização. O código foi projetado para encontrar o maior prêmio possível após remover dígitos de um número de N dígitos.

2 Código Implementado

2.1 Implementação em JavaScript

Aqui está a implementação em JavaScript do algoritmo.

2.2 Explicação do Código

2.2.1 Função `encontrarMaiorPremio`

Esta função recebe três parâmetros: `N` (número total de dígitos), `D` (número de dígitos a serem removidos) e `numero` (o número original). O objetivo é encontrar o maior prêmio possível mantendo o maior número de dígitos significativos.

```
function encontrarMaiorPremio(N, D, numero) {
  const pilha = [];

  for (let digito of numero) {
    // Verifica se há dígitos para remover e se o último dígito na pilha é menor que o a
    while (D > 0 && pilha.length > 0 && parseInt(pilha[pilha.length - 1]) < parseInt(dig
      pilha.pop(); // Remove o último dígito da pilha
      D--; // Decrementa o número de dígitos a serem removidos
    }
    if (pilha.length < N - D) {
      pilha.push(digito); // Adiciona o dígito atual à pilha
    }
  }
```

```
  return pilha.join(''); // Retorna o maior prêmio possível
}
```

O algoritmo itera pelo número original, mantendo uma pilha dos dígitos significativos. Se um dígito maior for encontrado na iteração, os dígitos menores são removidos da pilha, até que não seja mais necessário remover ou até o número de dígitos removidos atingir o limite. Isso garante que os dígitos mais altos sejam mantidos para formar o maior prêmio possível.

2.2.2 Função `processarCasos`

Essa função gerencia a entrada e saída do programa, lendo os valores de `N` e `D`, e chamando a função `encontrarMaiorPremio` para calcular o maior prêmio possível.

```
function processarCasos() {
  // Aguarda a entrada dos valores N e D
  rl.on('line', (linha) => {
    const [N, D] = linha.split(' ').map(Number);

    if (N === 0 && D === 0) {
      rl.close();
    } else {
      // Aguarda o número escrito pelo apresentador
      rl.once('line', (numero) => {
        const maiorPremio = encontrarMaiorPremio(N, D, numero.trim());
        console.log(maiorPremio); // Imprime o maior prêmio possível
        processarCasos(); // Chama recursivamente para processar mais casos
      });
    }
  });
}

processarCasos();
```

Essa função utiliza a interface `readline` do Node.js para ler os valores de `N` e `D` e, em seguida, aguarda a entrada do número escrito pelo apresentador. Após calcular o maior prêmio possível com base nos dados fornecidos, imprime o resultado e continua a processar mais casos até que a condição de saída seja atingida (`N` e `D` sejam ambos zero).

2.3 Programação Gulosa: Aplicação no Código

A estratégia gulosa é aplicada na função `encontrarMaiorPremio`, onde durante a iteração pelos dígitos do número original, a lógica gulosa é utilizada para manter os dígitos mais altos possíveis, descartando os menores sempre que necessário. Isso assegura que o maior prêmio possível seja calculado, mantendo a eficiência da

3 Conclusão

O algoritmo demonstra a aplicação da estratégia gulosa para resolver um problema específico de otimização, encontrando o maior prêmio possível após a remoção de dígitos de um número. Embora a estratégia gulosa nem sempre garanta a solução ótima, neste contexto específico, foi eficaz para atingir o objetivo proposto.