Análise Amortizada

Estrutura de Dados Avançada — QXD0015

Prof. Atílio Gomes Luiz
gomes.atilio@ufc.br

Universidade Federal do Ceará
Campus Quixadá

2º semestre/2025

Análise Amortizada

- Na análise amortizada estamos preocupados com a **média** ao longo de uma **sequência de operações**.
  - Algumas operações podem ser dispendiosas, mas outras podem ser mais rápidas e, no no final, eles se igualam.
  - Normalmente aplicado à análise de uma estrutura de dados.

- Serve para analisar uma sequência de operações ou iterações onde o pior caso individual não reflete o pior caso da sequência.

Análise Amortizada

- É diferente da **análise regular do pior caso**, onde o pior custo de uma operação é analisado.
  - Pode haver correlações entre as operações e o pior caso por operação pode ser demasiado pessimista porque a única maneira de ter uma operação cara pode ser ter muitas operações baratas antes.

Análise Amortizada

- É diferente da **análise regular do pior caso**, onde o pior custo de uma operação é analisado.
  - Pode haver correlações entre as operações e o pior caso por operação pode ser demasiado pessimista porque a única maneira de ter uma operação cara pode ser ter muitas operações baratas antes.

- É diferente da **análise de caso médio** e não há probabilidade ou experança envolvida.
  - A análise amortizada ainda nos dá uma visão do **pior cenário possível**.

Análise Amortizada

Definições

Custo Amortizado

O custo amortizado por operação para uma sequência de n operações é o custo total das operações dividido por n.

Método agregado

Examina o custo total das operações e toma a média.

Exemplo: Vetor Dinâmico

Vetor dinâmico

- Imagine que temos a estrutura de dados **Vetor Dinâmico** implementada como um array de inteiros.
  - itens que chegam são armazenados ao final do array: push_back()
  - quantidade de elementos que vão chegar é desconhecida

Vetor dinâmico

- Imagine que temos a estrutura de dados **Vetor Dinâmico** implementada como um array de inteiros.
  - itens que chegam são armazenados ao final do array: push_back()
  - quantidade de elementos que vão chegar é desconhecida

O que acontece quando um item chega e o array está cheio?

Vetor dinâmico

- Imagine que temos a estrutura de dados **Vetor Dinâmico** implementada como um array de inteiros.
  - itens que chegam são armazenados ao final do array: push_back()
  - quantidade de elementos que vão chegar é desconhecida

O que acontece quando um item chega e o array está cheio?

- Precisamos criar um array novo e maior, copiar os elementos existentes, e continuar a partir daí.
- Esta execução da operação `push_back()` é dispendiosa.

Vetor dinâmico

Qual seria uma boa estratégia para redimensionar o array?

Estratégia

Cada vez que o array encher, dobramos seu tamanho

Algoritmo: Inserção ao final do vetor


```
push_back(T, x)
 1  if T.capacity == 0 then
 2       allocate T.arr with T.size == 0 and T.capacity == 1
 3  # If table is full, double it
 4  if T.size == T.capacity then
 5       aux ← new array of capacity 2 * T.capacity
 6       Copy all items from T.arr para aux
 7       T.arr ← aux
 8  # Insert x into the array
 9  T.arr[T.size] ← x
10  T.size = T.size + 1
```

Vetor dinâmico

Primeiro, vamos estabelecer um modelo de custos:

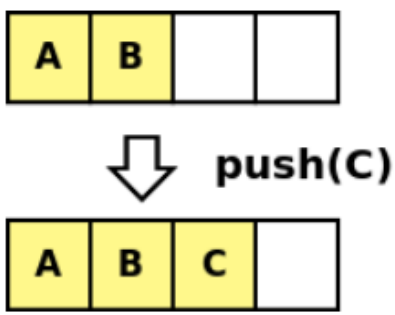
- **Inserir** um elemento em um slot de um array existente custa 1
- **Alocar** um novo array custa 0 (zero)

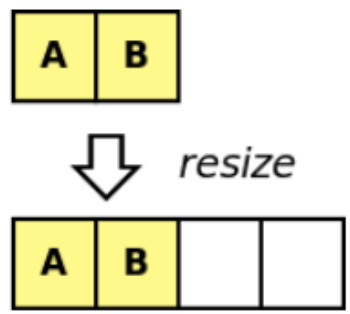
Vetor dinâmico

Primeiro, vamos estabelecer um modelo de custos:

- **Inserir** um elemento em um slot de um array existente custa 1
- **Alocar** um novo array custa 0 (zero)

Isso significa que um push_back sem redimensionamento custa 1, e que o redimensionamento custa n, com n elementos sendo copiados para o novo array.

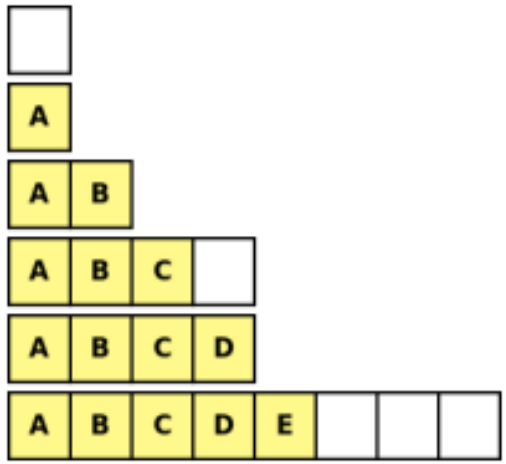




Método agregado: Vetor dinâmico

- No **método agregado**, inicialmente calculamos o custo total $T(n)$ de uma sequência de n operações `push_back` realizadas sobre o vetor.
  - O método agregado rastreia precisamente quais operações serão as mais custosas.
- O custo amortizado por operação é então $T(n)/n$.
- Este custo amortizado aplica-se a cada operação, mesmo quando existem operações diferentes na sequência.

Método agregado: Vetor dinâmico



- O custo das inserções nos slots é apenas n
- O custo dos redimensionamentos é $1 + 2 + 4 + \cdots + 2^i$ para algum $2^i < n$
- Esta soma é menor que $2n$
- **Custo Total** de n de `pushes` = custo(inserções) + custo(resizes) $< 3n$
- A **média** de n operações é < 3, ou seja, nosso custo amortizado, é $O(1)$.

Método Agregado

Considerações

- Usando a **análise amortizada (método agregado)**, mostramos que duplicar o tamanho do array é uma boa estratégia com tempo amortizado constante (por operação).

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

FIM