Noções de Análise de Algoritmos

Projeto e Análise de Algoritmo — QXD0041

Prof. Fabio Dias

fabiodias@ufc.br

Universidade Federal do Ceará

2º semestre/2024

Objetivo

- Estimar o tempo de execução de um algoritmo de forma analítica.
- Estimar a pior entrada que pode ser dada a um algoritmo (aquela que demorará mais tempo para ser executada).

Problemas computacionais

Um problema computacional é uma relação entre um conjunto de instâncias e um conjunto de soluções:

- uma instância é um conjunto de valores conhecidos
- uma solução é um conjunto de valores a computar
- cada instância corresponde a uma ou mais soluções
- Problema de decisão, problema de busca, problema de otimização, etc.

Exemplo de problema: teste de primalidade

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Problema: determinar se um dado número é primo

- instâncias: números inteiros
- soluções: sim ou não

Exemplo:

- Instância: 9411461
- Solução: sim

Exemplo:

- Instância: 8411461
- Solução: não

Exemplo de problema: ordenação

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Problema: ordenar os elementos de um vetor

- instâncias: conjunto de vetores de inteiros
- soluções: conjunto de vetores de inteiros em ordem crescente

Exemplo:

- Instância:

1										n
33	55	33	44	33	22	11	99	22	55	77

- Solução:

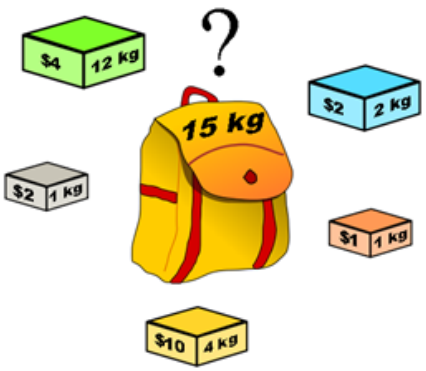
1										n
11	22	22	33	33	33	44	55	55	77	99

Exemplo de problema: Problema da Mochila

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Problema:

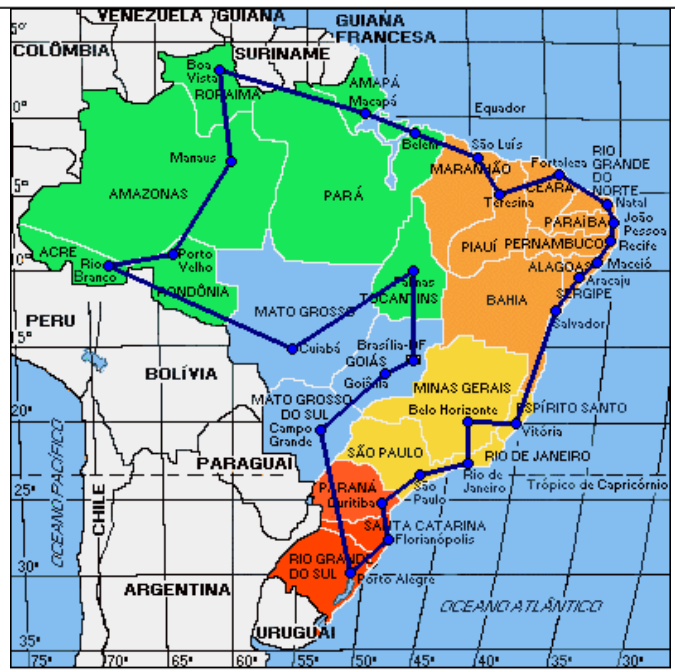
- Um ladrão está assaltando uma casa que tem n objetos que lhe interessam.
- Cada objeto tem um valor e um peso associado.
- Sabendo que a sua mochila poderá levar no máximo um peso K, o ladrão pretende determinar quais os objetos que ele deve roubar, para maximizar o lucro do assalto.
- Aplicações em Investimento de capital, problema de corte e empacotamento, etc.



Exemplo de problema: Problema do Caixeiro Viajante

Problema:

- Suponha que você quer conhecer todas as cidades do Ceará a partir de uma cidade inicial, percorrer todas as outras cidades exatamente uma única vez e voltar à cidade inicial, fazendo o trajeto mais curto.
- Aplicações no Problema de Roteamento, etc.



Tamanho da Entrada

- Ordenação: Tamanho do vetor.
- Problema da Mochila: Quantidade objetos.
- Problema do Caixeiro Viajante: Quantidade de cidades (vértices) e a quantidade de ligações entre as cidades (arestas). Tamanho do Grafo.

Algoritmos

Um algoritmo é uma sequência de instruções que

- recebe uma instância de um problema computacional
- devolve uma solução correspondente à instância recebida

Observações:

- a instância recebida é chamada de entrada
- a solução devolvida é chamada de saída
- toda instrução deve ser bem definida

Descrição de algoritmos

Podemos escrever um algoritmo de várias maneiras:

- em uma linguagem de programação, como C, Pascal, Java, Python...
- em português, ou outra língua natural
- em pseudocódigo, como no livro de CLRS

Usaremos apenas as duas últimas opções!

Análise de Algoritmos

- A análise de algoritmos é a área que estuda como estimar **teoricamente** os recursos que um algoritmo precisará a fim de resolver um problema computacional.

- Ao se analisar um algoritmo, estamos geralmente preocupados com duas medidas:
  - tempo de execução (ou tempo de processamento)
  - espaço de memória utilizado pelo algoritmo

- Neste momento, nos interessa apenas estudar o tempo de execução, mas a análise feita aqui se estende também à análise do espaço de memória.

Tempo de processamento

Como comparar dois algoritmos que resolvem um certo problema em termos da sua eficiência?

Tempo de processamento

Como comparar dois algoritmos que resolvem um certo problema em termos da sua eficiência?

- Medir o tempo (em microssengundos, etc.) gasto por um algoritmo.
  - Em certos contextos, não é uma boa opção. **Por quê?**

Tempo de processamento

Como comparar dois algoritmos que resolvem um certo problema em termos da sua eficiência?

- Medir o tempo (em microssengundos, etc.) gasto por um algoritmo.
  - Em certos contextos, não é uma boa opção. **Por quê?**
  - Depende do compilador
    - Pode preferir algumas construções ou otimizar melhor.

Tempo de processamento

Como comparar dois algoritmos que resolvem um certo problema em termos da sua eficiência?

- Medir o tempo (em microssengundos, etc.) gasto por um algoritmo.
  - Em certos contextos, não é uma boa opção. **Por quê?**
  - Depende do compilador
    - Pode preferir algumas construções ou otimizar melhor.
  - Depende do hardware
    - GPU vs. CPU, desktop vc. smartphone.

Tempo de processamento

Como comparar dois algoritmos que resolvem um certo problema em termos da sua eficiência?

- Medir o tempo (em microssengundos, etc.) gasto por um algoritmo.
  - Em certos contextos, não é uma boa opção. **Por quê?**
  - Depende do compilador
    - Pode preferir algumas construções ou otimizar melhor.
  - Depende do hardware
    - GPU vs. CPU, desktop vc. smartphone.
  - Depende da linguagem de programação e habilidade do programador

Complexidade do algoritmo

Queremos analisar algoritmos:

- Independentemente do computador ou implementação
- Em função do valor de n (tamanho da entrada)
- Expressá-la como uma função matemática, digamos, $T(n)$
- A análise também supõe uma entrada n grande

Para isso, considera-se que um algoritmo é subdividido, em uma quantidade finita de **passos**.

- Um passo é uma instrução indivisível e de tempo constante, ou seja, independente de condições de entrada e processamento.
  - Exemplo: soma, multiplicação, atribuição, comparação.
- A quantidade de passos necessários ao cumprimento de um algoritmo é denominada complexidade do algoritmo.

Medindo a complexidade de algoritmos

Soma de um vetor e consumo de tempo


```
1 int soma(int v[], int n) {
2   int soma = 0;
3   for (int i = 0; i < n; i++)
4     soma = soma + v[i];
5   return soma;
6 }
```

Soma de um vetor e consumo de tempo


```
1 int soma(int v[], int n) {
2   int soma = 0;
3   for (int i = 0; i < n; i++)
4     soma = soma + v[i];
5   return soma;
6 }
```

Consumo de tempo:

- Linha 2: tempo c_2 (declaração de variável e atribuição)
- Linha 3: tempo c_3 (atribuições, acessos, comparação e incremento)
  - Essa linha é executada $n + 1$ vezes
- Linha 4: tempo c_4 (soma e atribuição)
  - Essa linha é executada n vezes
- Linha 5: tempo c_5 (`return`)

O tempo de execução é igual a

$$c_2 + c_3 \cdot (n + 1) + c_4 \cdot n + c_5$$

Soma de um vetor e consumo de tempo

Cada c_i não depende de n, depende apenas do computador

Sejam $a := \max c_i$.

Se $n \geq 1$, temos que o tempo de execução é menor ou igual a

$$\begin{aligned} c_2 + c_3 \cdot (n+1) + c_4 \cdot n + c_5 &= c_2 + c_3 + c_5 + (c_3 + c_4) \cdot n \\ &\leq 3a + 2a \cdot n \\ T(n) &= 2n + 3 \end{aligned}$$

Isto é, o crescimento do tempo é linear em n

Soma de matrizes

Sejam A e B duas matrizes quadradas de ordem n que devem ser somadas.

```
1 void soma_matrizes (int **A, int **B, int **C, int n){
2   for (int i = 0; i < n; i++)
3     for (int j = 0; j < n; j++)
4       C[i][j] = A[i][j] + B[i][j];
5 }
```

- Qual o tempo de execução desse algoritmo?

Outras Situações

- Exemplo: busca sequencial.

```
1 int busca(int v[], int n, int x) {
2   for (int i = 0; i < n; i++)
3     if (v[i] == x)
4       return i;
5   return -1;
6 }
```

Outras Situações

- Exemplo: busca sequencial.

```
1 int busca(int v[], int n, int x) {
2   for (int i = 0; i < n; i++)
3     if (v[i] == x)
4       return i;
5   return -1;
6 }
```

- Nestes casos, a análise de complexidade consiste em avaliar o algoritmo em situações extremas.

- Diferentemente dos algoritmos anteriores, na maior parte dos algoritmos conhecidos a função de complexidade $T(n)$ não é geral, ou seja, pode mudar conforme características da entrada.

Complexidade de melhor e pior caso

- Na análise da complexidade, definimos três casos de entrada para um algoritmo, como forma de mensurar o custo do algoritmo de resolver determinado problema diante de diferentes entradas.

- Temos três casos a considerar, Melhor Caso, Pior Caso e Caso Médio.

Complexidade do melhor caso

Melhor Caso

- O menor tempo de execução (analítico) considerando qualquer entrada de tamanho n.
- Um limite inferior para o tempo de execução.
- Procure deduzir qual configuração que a entrada do problema deve estar para que o algoritmo execute esse menor tempo.

Complexidade do pior caso

Pior Caso

- O maior tempo de execução (analítico) considerando qualquer entrada de tamanho n.
- Um limite superior para o tempo de execução.
- Procure deduzir qual configuração que a entrada do problema deve estar para que o algoritmo execute esse maior tempo.

Complexidade do caso médio

Caso Médio

- É a média dos tempos de execução (analítico) considerando qualquer entrada de tamanho n.
- Seria algo como executasse o algoritmo com diversas instâncias de mesmo tamanho de entrada e calculássemos a média do tempo de execução.
- Calcula-se esse caso médio usando probabilidade.

Pior caso, caso médio e melhor caso

Em geral, queremos analisar o pior caso do algoritmo.

- A análise do melhor caso pode ser interesse, mas é rara.
- A análise do caso médio é mais difícil
  - É uma análise probabilística
  - Precisamos fazer suposições sobre os dados de entrada

Busca Sequencial


```
1 int busca(int v[], int n, int x) {
2   for (int i = 0; i < n; i++)
3     if (v[i] == x)
4       return i;
5   return -1;
6 }
```

Busca Sequencial

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

```
1 int busca(int v[], int n, int x) {
2   for (int i = 0; i < n; i++)
3     if (v[i] == x)
4       return i;
5   return -1;
6 }
```

- Melhor caso: $T(n) = 3$
- Pior caso: $T(n) = 2n + 2$
- A análise do caso médio é mais difícil.

Busca Sequencial - Caso Médio


```
1 int busca(int v[], int n, int x) {
2   for (int i = 0; i < n; i++)
3     if (v[i] == x)
4       return i;
5   return -1;
6 }
```

- Qual a função de complexidade dado que a chave se encontra no índice i?
- $T(n) = 2i + 1$.
- Seja $0 \leq p \leq 1$ a probabilidade da chave está no vetor.
- Supondo que a probabilidade da chave está em qualquer posição é a mesma, então, p/n é a probabilidade da chave está no índice i.

Busca Sequencial - Caso Médio

$$T(n) = (1 - p)(2n + 2) + \sum_{i=1}^{n} \frac{p}{n}(2i + 1)$$

$$= 2n - 2np - 2p + 2 + \frac{p}{n} \sum_{i=1}^{n}(2i + 1)$$

$$= 2n - 2np - 2p + 2 + \frac{p}{n} \sum_{i=1}^{n} 2i + \frac{p}{n} n$$

$$= 2n - 2np - p + 2 + \frac{2p}{n} \sum_{i=1}^{n} i$$

$$= 2n - 2np - p + 2 + \frac{2p}{n} \frac{(n+1)n}{2}$$

$$= 2n - np + 2$$

Melhor e Pior Casos de Alguns Problemas

Problema	Melhor Caso	Pior Caso
Busca linear	Elemento na primeira posição	Elemento não contido no vetor
Problema da Mochila	Todos os objetos juntos cabem na mochila	Todas as combinações de objetos cabem na mochila
Caixeiro Viajante	Não existe.	Sempre.