Análise Projeto de Algoritmos

Loana Tito Nogueira

12-Março-2012

Problemas do nosso interesse

Problemas do nosso interesse

- Problemas que não admitem solução computacional não são de interesse deste curso

Problemas do nosso interesse

- Problemas que não admitem solução computacional não são de interesse deste curso

⋮

Problemas do nosso interesse

- Problemas que não admitem solução computacional não são de interesse deste curso

⋮

- Decidir quais problemas (obviamente de uma certa natureza) têm solução computacional é, por um lado, um problema muito difícil, para o qual não se conhece uma solução computacional

Problemas e seus algoritmos

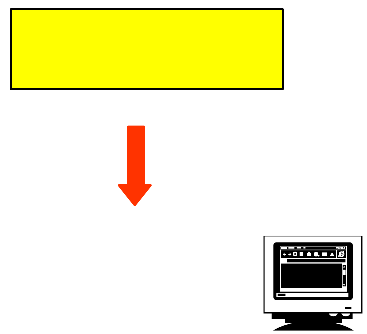
Problemas e seus algoritmos

- Neste curso, estudamos problemas que admitem uma solução computacional;

Problemas e seus algoritmos

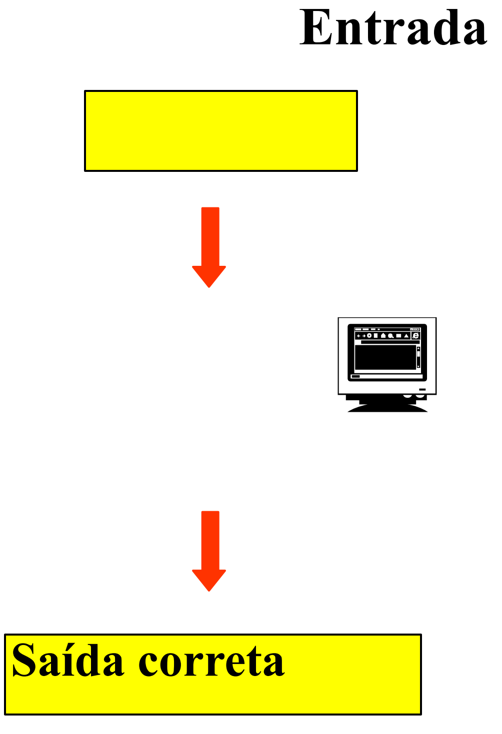
- Neste curso, estudamos problemas que admitem uma solução computacional;

Entrada



Problemas e seus algoritmos

- Neste curso, estudamos problemas que admitem uma solução computacional;



Propriedades de Algoritmos

- Pode haver vários algoritmos para o mesmo problema, que diferem entre si
  - Na forma de atacar o problema (como);
  - Na quantidade de recursos utilizados (quanto tempo ou memória); e
  - Na qualidade da resposta (exata, aproximada, ou com uma dada probabilidade de estar correta)

Propriedades de Algoritmos

- Pode haver vários algoritmos para o mesmo problema, que diferem entre si
  - Na forma de atacar o problema (como);
  - Na quantidade de recursos utilizados (quanto tempo ou memória); e
  - Na qualidade da resposta (exata, aproximada, ou com uma dada probabilidade de estar correta)

Propriedades de Algoritmos

- Pode haver vários algoritmos para o mesmo problema, que diferem entre si
  - Na forma de atacar o problema (como);
  - Na quantidade de recursos utilizados (quanto tempo ou memória); e
  - Na qualidade da resposta (exata, aproximada, ou com uma dada probabilidade de estar correta)

Propriedades de Algoritmos

- Pode haver vários algoritmos para o mesmo problema, que diferem entre si
  - Na forma de atacar o problema (como);
  - Na quantidade de recursos utilizados (quanto tempo ou memória); e
  - Na qualidade da resposta (exata, aproximada, ou com uma dada probabilidade de estar correta)

Propriedades de Algoritmos

- Estamos interessados na corretude e complexidade dos algoritmos que produzem respostas exatas. Mais especificamente, estudaremos técnicas para
- O projeto e verificação de corretude de algoritmos para um dado problema; e

Propriedades de Algoritmos

- Estamos interessados na corretude e complexidade dos algoritmos que produzem respostas exatas. Mais especificamente, estudaremos técnicas para
- O projeto e verificação de corretude de algoritmos para um dado problema; e
- Técnicas para avaliar a quantidade de recursos utilizados por um algoritmo, permitindo comparálo a outros algoritmos para o mesmo problema, independente do computador em que venha a ser implementado

Vamos resolver alguns problemas?

Vamos resolver alguns problemas?

- **Busca de um elemento em um vetor:**

Vamos resolver alguns problemas?

- **Busca de um elemento em um vetor:**
  - **Problema:** Dado um vetor *v* de *n* inteiros e um inteiro *x*, determinar se *x* está no vetor

Vamos resolver alguns problemas?

- **Busca de um elemento em um vetor:**
  - **Problema:** Dado um vetor v de n inteiros e um inteiro x, determinar se x está no vetor
  - **Algoritmo:** Procure em todas as posições do vetor até encontrar i tal que $v[i]=x$.

Vamos resolver alguns problemas?

- **Busca de um elemento em um vetor:**
  - **Problema:** Dado um vetor v de n inteiros e um inteiro x, determinar se x está no vetor
  - **Algoritmo:** Procure em todas as posições do vetor até encontrar i tal que $v[i]=x$.
  - Este problema é "fácil", tem um algoritmo simples e de complexidade proporcional ao número de elementos do vetor.

- **Ordenação dos elementos de um conjunto:**

Vamos resolver alguns problemas?

- **Ordenação dos elementos de um conjunto:**
  - **Problema:** Ordenar *n* valores comparáveis

Vamos resolver alguns problemas?

- **Ordenação dos elementos de um conjunto:**
  - **Problema:** Ordenar *n* valores comparáveis
  - **Algoritmo:** Selecionar sucessivamente o menor elemento e retorná-lo, retirando-o do conjunto

Vamos resolver alguns problemas?

- **Ordenação dos elementos de um conjunto:**
  - **Problema:** Ordenar *n* valores comparáveis
  - **Algoritmo:** Selecionar sucessivamente o menor elemento e retorná-lo, retirando-o do conjunto
  - Este problema é "fácil", tem um algoritmo simples e de complexidade proporcional ao quadrado do número de elementos *n* do vetor

Vamos resolver alguns problemas?

- **Ordenação dos elementos de um conjunto:**
  - **Problema:** Ordenar *n* valores comparáveis
  - **Algoritmo:** Selecionar sucessivamente o menor elemento e retorná-lo, retirando-o do conjunto
  - Este problema é "fácil", tem um algoritmo simples e de complexidade proporcional ao quadrado do número de elementos *n* do vetor
    - **É o melhor que podemos fazer?**

Vamos resolver alguns problemas?

- **Atribuição de professores às disciplinas:**

Vamos resolver alguns problemas?

- **Atribuição de professores às disciplinas:**

- **Problema:** Dados um conjunto de *n* professores, um conjunto de *n* disciplinas e as listas de disciplinas que cada professor pode ministrar, determinar se existe uma atribuição de disciplinas a professores de que forma que cada professor ministre exatamente uma disciplina

Vamos resolver alguns problemas?

- **Atribuição de professores às disciplinas:**

- **Problema:** Dados um conjunto de *n* professores, um conjunto de *n* disciplinas e as listas de disciplinas que cada professor pode ministrar, determinar se existe uma atribuição de disciplinas a professores de que forma que cada professor ministre exatamente uma disciplina

- **Algoritmo:** Tentar atribuir professores às disciplinas de todas as formas possíveis até encontrar uma que resolva o problema.

Vamos resolver alguns problemas?

- **Atribuição de professores às disciplinas:**

- **Problema:** Dados um conjunto de n professores, um conjunto de n disciplinas e as listas de disciplinas que cada professor pode ministrar, determinar se existe uma atribuição de disciplinas a professores de que forma que cada professor ministre exatamente uma disciplina

- **Algoritmo:** Tentar atribuir professores às disciplinas de todas as formas possíveis até encontrar uma que resolva o problema.

  - **Simples, mas com alta complexidade!!!!**

Vamos resolver alguns problemas?

- **Problema do Caixeiro Viajante:**

Vamos resolver alguns problemas?

- **Problema do Caixeiro Viajante:**

- **Problema:** Dadas *n* cidades e as distâncias entre elas, determinar a sequência em que as cidades (todas) devem ser visitadas de forma que a distância total percorrida seja mínima

Vamos resolver alguns problemas?

- **Problema do Caixeiro Viajante:**

- **Problema:** Dadas *n* cidades e as distâncias entre elas, determinar a sequência em que as cidades (todas) devem ser visitadas de forma que a distância total percorrida seja mínima

- **Algoritmo:** Calcular a distância total de cada percurso e escolher o menor.

Vamos resolver alguns problemas?

- **Problema do Caixeiro Viajante:**

- **Problema:** Dadas *n* cidades e as distâncias entre elas, determinar a sequência em que as cidades (todas) devem ser visitadas de forma que a distância total percorrida seja mínima

- **Algoritmo:** Calcular a distância total de cada percurso e escolher o menor.

  - **Algoritmo simples, com complexidade O(n!)**

Vamos resolver alguns problemas?

- **Problema do Caixeiro Viajante:**

- **Problema:** Dadas *n* cidades e as distâncias entre elas, determinar a sequência em que as cidades (todas) devem ser visitadas de forma que a distância total percorrida seja mínima

- **Algoritmo:** Calcular a distância total de cada percurso e escolher o menor.

  - **Algoritmo simples, com complexidade O(n!)**
    - **Não se conhece algoritmo polinomial para este problema**
      - **NP-completo (difícil)**

Vamos resolver alguns problemas?

- **Problema de Parada:**

Vamos resolver alguns problemas?

- **Problema de Parada:**

- **Problema:** Dado um programa qualquer *P* e uma entrada *E* do programa, determinar se o programa *P* pára quando alimentado com a entrada *E*.

Vamos resolver alguns problemas?

- **Problema de Parada:**

- **Problema:** Dado um programa qualquer *P* e uma entrada *E* do programa, determinar se o programa *P* pára quando alimentado com a entrada *E*.

- **Algoritmo: ??????????**

Vamos resolver alguns problemas?

- **Problema de Parada:**

- **Problema:** Dado um programa qualquer *P* e uma entrada *E* do programa, determinar se o programa *P* pára quando alimentado com a entrada *E*.

- **Algoritmo: ??????????**
- **Este é um problema indecidível, é impossível demonstrar matematicamente que não existe uma solução computacional para ele (dentro do nosso conceito do que seja um computador)**

Objetivos

- Discutir conceitos básicos sobre análise e complexidade de algoritmos como parte do aprendizado de longo prazo para resolução e classificação de problemas computacionais
- Trataremos de problemas que possuem algoritmos polinomiais
- Mostraremos alguns problemas para os quais não se conhecem algoritmos polinomiais
- Ao final do curso, os alunos devem ser capazes de projetar algoritmos, provar sua corretude e analisar sua complexidade.

Técnicas de Projeto de Algoritmos

- Indução Matemática
- Divisão e Conquista
- Programação Dinâmica
- Algoritmo Guloso
- Busca Exaustiva (Backtracking)

Classes de Problemas

Noções Preliminares

Noções Preliminares

- **O que é um algoritmo correto?**

Noções Preliminares

- **O que é um algoritmo correto?**
  - Aquele que pára para toda instância do problema, retornando uma solução correta

Noções Preliminares

- **O que é um algoritmo correto?**
  - Aquele que pára para toda instância do problema, retornando uma solução correta
- **O que é analisar um algoritmo?**

Noções Preliminares

- **O que é um algoritmo correto?**
  - Aquele que pára para toda instância do problema, retornando uma solução correta
- **O que é analisar um algoritmo?**
  - Predizer a quantidade de recursos utilizados (memória, tempo de execução, número de processadores, ...)
  - Na maioria dos casos estaremos interessados em avaliar o tempo de execução gasto pelo algoritmo
  - Contaremos o número de operações efetuadas

Noções Preliminares

- **O que é um algoritmo correto?**
  - Aquele que pára para toda instância do problema, retornando uma solução correta
- **O que é analisar um algoritmo?**
  - Predizer a quantidade de recursos utilizados (memória, tempo de execução, número de processadores, ...)
  - Na maioria dos casos estaremos interessados em avaliar o tempo de execução gasto pelo algoritmo
  - Contaremos o número de operações efetuadas

Noções Preliminares

- **O que é um algoritmo correto?**
  - Aquele que pára para toda instância do problema, retornando uma solução correta
- **O que é analisar um algoritmo?**
  - Predizer a quantidade de recursos utilizados (memória, tempo de execução, número de processadores, ...)
  - Na maioria dos casos estaremos interessados em avaliar o tempo de execução gasto pelo algoritmo
  - Contaremos o número de operações efetuadas

Noções Preliminares

- Como representar um algoritmo?

Noções Preliminares

- **Como representar um algoritmo?**
  - Pseudo-Código (abstrato, independe de implementação)

- **Como verificar a corretude de um algoritmo?**
  - Demonstração formal

Técnicas de Demonstração

- A **demonstração direta** de uma implicação p ⇒ q é uma sequência de passos lógicos (implicações):

$$p \Rightarrow p_1 \Rightarrow p_2 \Rightarrow \ldots \Rightarrow p_n = q,$$

Que resultam, por transitividade, na implicação desejada.

Cada passo da demonstração é um axioma ou teorema provado previamente

Exemplo: Provar que $\sum_{i=1}^{k} (2i-1) = k^2$

- Dica: Utilizar as propriedades de somatório

Demonstração por Contrapositiva

- A **contrapositiva** de $p \Rightarrow q$

Demonstração por Contrapositiva

- A **contrapositiva** de $p \Rightarrow q$ é $\sim q \Rightarrow \sim p$

Demonstração por Contrapositiva

- A **contrapositiva** de $p \Rightarrow q$ é $\sim q \Rightarrow \sim p$
- A contrapositiva é equivalente à implicação original.

A veracidade de $\sim q \Rightarrow \sim p$ implica a veracidade de $p \Rightarrow q$, e vice-versa.

Demonstração por Contrapositiva

- A **contrapositiva** de $p \Rightarrow q$ é $\sim q \Rightarrow \sim p$
- A contrapositiva é equivalente à implicação original.

A veracidade de $\sim q \Rightarrow \sim p$ implica a veracidade de $p \Rightarrow q$, e vice-versa.

- A técnica é útil quando é mais fácil demonstrar a contrapositiva do que a implicação original
- Para demonstrarmos a contrapositiva de uma implicação, podemos utilizar qualquer técnica de demonstração

Exemplo: Prove que se $2|3m$ então $2|m$

Demonstração por contradição

- A Demonstração por **contradição** envolve supor absurdamente que a afirmação a ser demonstrada é falsa e obter, através de implicações válidas, uma conclusão contraditória.
- A contradição obtida implica que a hipótese absurdo é falsa e, portanto, a afirmação é de fato verdadeira.
- No caso de uma implicação $p \Rightarrow q$, equivalente a $\sim p \vee q$, a negação é $p \wedge \sim q$

Exemplo: Seja A um conjunto, prove que $\varnothing \subseteq A$, qualquer que seja A

Exemplo: Seja A um conjunto, prove que $\emptyset \subseteq A$, qualquer que seja A

- Por contradição e utilizando a definição de subconjunto

Exemplo2: Prove que o maior inteiro que divide ambos n e n+1 é 1

Demonstração por Casos

- Na demonstração por casos, particionamos o universo de possibilidades em um conjunto finito de casos e demonstramos a veracidade da implicação para cada caso

Demonstração por Casos

- Na demonstração por casos, particionamos o universo de possibilidades em um conjunto finito de casos e demonstramos a veracidade da implicação para cada caso

- Para demonstrar cada caso individual, qualquer técnica de demonstração pode ser utilizada

Exemplo: Mostre que a paridade de dois inteiros x e y de mesma paridade é sempre par

Indução Matemática

- Na **Demonstração por Indução**, queremos demonstrar a validade de P(n), uma propriedade P com um parâmetro natural n associado, para todo valor de n
- Há um número infinito de casos a serem considerados, um para cada valor de n. Demonstramos os infinitos casos de uma só vez:
  - Base da Indução: Demonstramos P(1)
  - Hipótese de Indução: Supomos que P(n) é verdadeiro
  - Passo de Indução: Provamos que P(n+1) é verdadeiro, a partir da hipótese de indução

Exemplo: Prove, por indução, $\sum_{i=1}^{k} (2i-1) = k^2$

Exercícios:

- 1- Demonstre que para todo natural x e n, $x^n - 1$ é divisível por x- 1
- Mostre que $\Sigma_{i=1}^{n}$ $3+5i = 2,5n^2 + 5.5n$
- Mostre que $\Sigma_{i=1}^{n}$ $i = n(n+1)/2$
- Prove que todo número pode ser escrito como a soma de diferentes potências de 2.