UDESC

Complexidade de Algoritmos

Prof. Diego Buchinger
diego.buchinger@outlook.com
diego.buchinger@udesc.br

Prof. Cristiano Damiani Vasconcellos
cristiano.vasconcellos@udesc.br

UDESC

Abordagens para Resolução de Problemas

UDESC

Abordagens para Resolução de Problemas

Muitos problemas podem ser resolvidos de diversas maneiras diferentes.

Contudo, dependendo da abordagem escolhida, uma solução pode ser melhor ou pior do que outra em questões de tempo ou uso de memória.

UDESC

Abordagens para Resolução de Problemas

Podemos classificar os principais métodos de resolução de problemas em relação a abordagem utilizada:

- Indução Matemática – Fórmula
- Divisão e Conquista (*divide and conquer*)
- Algoritmos Gulosos (*greedy*)
- Algoritmos de Tentativa e Erro (*backtracking*)
- Programação dinâmica (*dynamic programming*)
- Algoritmos de aproximação (*approximation*)

Indução Matemática – Fórmula

Abordagem mais simples possível.

O problema pode ser reduzido a uma fórmula ou a um conjunto de fórmulas matemáticas.

Exemplo: descobrir as raízes de uma equação de segundo grau.

UDESC

Algoritmos de Divisão-e-Conquista
(*Divide and Conquer*)

Já estudamos esta abordagem.

Desmembrar o problema original em vários subproblemas semelhantes (menores), resolver os subproblemas (executando o mesmo processo recursivamente) e combinar as soluções.

Exemplos: ordenar um vetor de elementos (*Merge-Sort*, *Quick-Sort*) e multiplicação entre inteiros grandes

UDESC

Algoritmos Gulosos
(*Greedy Algorithms*)

Uma técnica para resolver problemas de otimização: minimizar ou maximizar um determinado valor.

Um algoritmo guloso sempre faz a escolha que parece ser a melhor em um determinado momento, esperando que essa melhor escolha local leve ao melhor resultado do problema como um todo.

Uma técnica simples, mas que na maioria das vezes não leva ao resultado ótimo.

UDESC

Algoritmos Gulosos
(*Greedy Algorithms*)

Alguns problemas que podem ser resolvidos com algoritmos gulosos:

- Problema do melhor troco - canônico
- Problema da Árvore Geradora Mínima
- Escalonamento de tarefas
- Codificação de Huffman
- Problema da Mochila Fracionada

Problema do Melhor Troco - canônico

Considere que o sistema econômico de um local utiliza unidades monetárias de valor: M_0, M_1, M_2, ... M_n;

Qual é o menor número de unidades monetárias que representam um determinado valor V?

Exemplo: Moedas disponíveis: 1, 5, 10, 25, 50, 100

Valor do troco: 3.82

Qual a menor quantidade de moedas a serem entregues e quais são elas?

Problema do Melhor Troco - canônico

Para sistemas monetários **canônicos** como os usualmente utilizados, podemos adotar a abordagem de escolher o maior número possível das maiores moedas:

Exemplo: Moedas disponíveis: 1, 5, 10, 25, 50, 100

Valor do troco: $ 3.82

382 / 100 = 3 e sobram $ 82	3 moedas de 100
82 / 50 = 1 e sobram 32	1 moeda de 50
32 / 25 = 1 e sobram 7	1 moeda de 25
7 / 5 = 1 e sobram 2	1 moeda de 5
2 / 1 = 2 e sobram 0	2 moedas de 1

Problema do Melhor Troco - canônico

CUIDADO! Para sistemas monetários **não canônicos** a abordagem apresentada pode não resultar no melhor troco possível:

Exemplo Trivial: Moedas disponíveis: 1, 3, 4

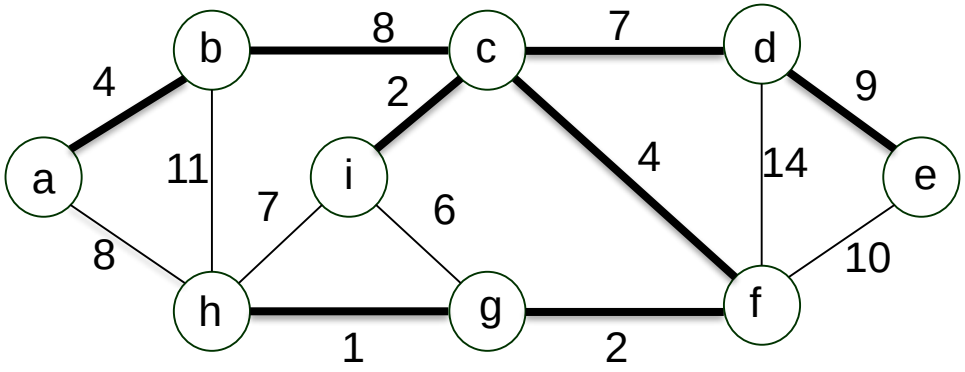
Valor do troco: $ 0.06

Qual é o número mínimo de moedas para o troco?

Problema da Árvore Geradora Mínima

Dado um grafo com seus respectivos vértices e arestas, determinar qual a árvore que contém todos os vértices deste grafo com o menor custo possível.

Escolhemos sempre a melhor aresta dos nós já inclusos que não formem um circuito (Prim)



Escalonamento de Tarefas

Dada uma lista de tarefas a serem executadas com um horário de início e um horário de término, determinar qual a quantidade máxima de atividades que podem ser executadas

Exemplo: Um auditório só pode ser utilizado para um evento por vez. Em um dia com muitos eventos, deseja-se determinar qual é o maior número de eventos que podem ser realizados no auditório, e quais são eles (OBS: pode haver mais de uma solução).

Evento	1	2	3	4	5	6	7	8	9	10	11
Inicio	3	8	5	1	6	12	0	8	5	2	3
término	5	11	7	4	10	14	6	12	9	13	8

Escalonamento de Tarefas

Solução gulosa: ordenamos os eventos pelo horário de término (em ordem crescente) e sempre que possível pegamos o evento com menor horário de término.

Evento	4	1	7	3	11	9	5	2	8	10	6
Inicio	1	3	0	5	3	5	6	8	8	2	12
término	4	5	6	7	8	9	10	11	12	13	14

Codificação de Huffman

Um método de compressão de dados. Esse método usa a frequência com que cada símbolo ocorre, em uma coleção de dados, para determinar um código de tamanho variável para o símbolo.

Caractere	Frequência	Código (fixo)	Código (variável)
a	45	000	0
b	13	001	101
c	12	010	100
d	16	011	111
e	9	100	1101
f	5	101	1100

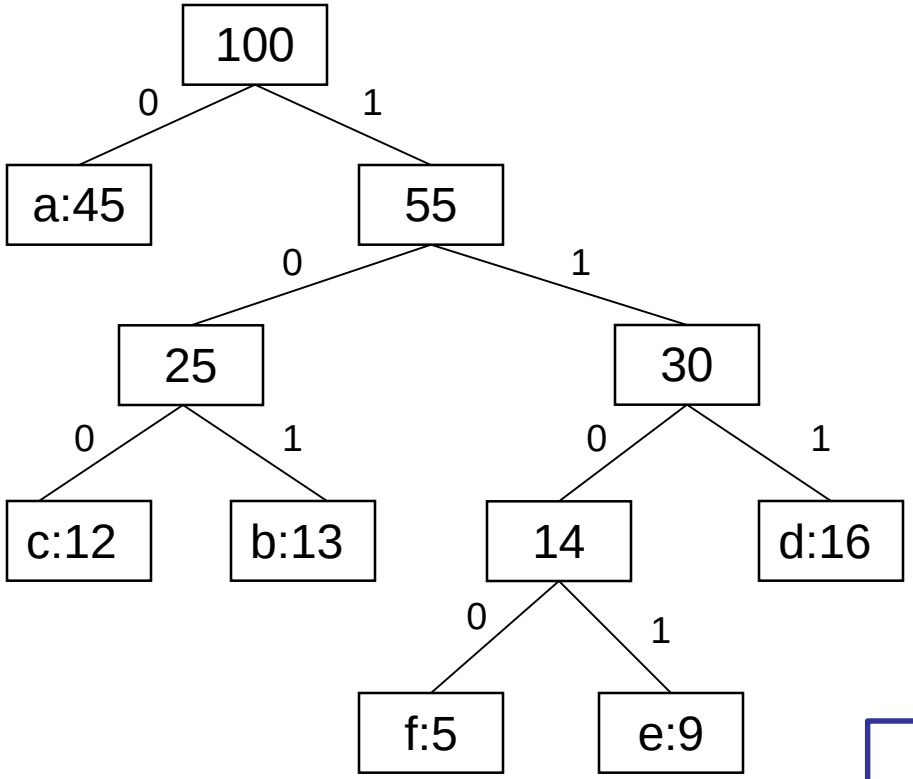
Exemplo:

... faca ...

... 101000010000 ...

... 110001000 ...

Codificação de Huffman



Exemplo:

... faca ...

... 101000010000 ...

... 110001000 ...

Codificamos usando a notação:
Filhos a esquerda: 0
Filhos a direita: 1
Até chegar a letra desejada

Codificação de Huffman

Huffman(C)

$n \leftarrow |C|$

$Q \leftarrow C$

para $i \leftarrow 1$ **até** n - 1

$z \leftarrow$ AlocaNo()

$x \leftarrow z.esq \leftarrow$ Extrair-Minimo(Q)

$y \leftarrow z.dir \leftarrow$ Extrair-Minimo(Q)

$z.valor = x.valor + y.valor$

Inserir(Q, z)

retorne Q

Testando o algoritmo:
Como ficaria a codificação para a string:

"testando o teste"

Construir a árvore e o mapa de codificação

Importante:

Note que para decodificar uma string, é necessário conhecer a árvore que foi usada para gerá-la!
Ou seja, espaço utilizado: nova string (menor) + árvore

Problema da Mochila Fracionada

Dark Version: Um ladrão está assaltando uma loja e possui uma mochila que pode carregar até **P** kg sem arrebentar. Sabendo os pesos e valores dos itens, qual é o maior valor possível que o ladrão conseguirá roubar?

Na versão da mochila fracionada os elementos podem ser pegos em fração (apenas parte do objeto, com proporção linear entre peso e valor)

Exemplo: Mochila com capacidade 9 kg

Objetos	1	2	3	4	5	6
Peso	1	3	2	5	7	9
Valor	3	2	4	7	9	12

Problema da Mochila Fracionada

Solução Gulosa: Ordenamos o vetor de itens em relação a proporção: [valor / peso]

Objetos	1	3	4	6	5	2
Peso	1	2	5	9	7	3
Valor	3	4	7	12	9	2
Proporção	3	2	1.4	1.33	1.29	0.67

UDESC

Tentativa e Erro
(*Backtracking*)

Outra técnica para resolver problemas de otimização

Testa todas as possíveis soluções até encontrar a(s) melhor(es) solução(ões) para um problema, utilizando as ideias de busca em profundidade ou busca em largura

Pode utilizar validações para diminuir (de maneira não muito significativa) o escopo de busca [ex: desconsiderar soluções parciais piores do que uma solução final já encontrada]

Geralmente não adequada para instâncias muito grandes

UDESC

Tentativa e Erro
(*Backtracking*)

Alguns problemas clássicos que são resolvidos (também) com *backtracking* (instâncias pequenas):

- Problema do passeio do cavalo
- Puzzle 3x3
- Sokoban
- Caixeiro Viajante

Passeio do Cavalo

Partindo de uma posição inicial, qual é o menor número de movimentos que um cavalo deve realizar para chegar a qualquer uma das casas de um tabuleiro?

LEMBRETE: o cavalo (*knight*) anda apenas em movimentos 'L'

Exemplo: tab. 5x5 c->2,1

- Busca em Profundidade...
- Busca em Largura...

UDESC

Tentativa e Erro
(*Backtracking*)

Solução usando Busca em Profundidade:

[+] usa menos memória (apenas tabuleiro)

[–] repete muitos movimentos (mais lento)

Solução usando Busca em Largura:

[+] não repete movimentos (mais rápido)

[–] usa mais memória (para salvar as posições)

em alguns casos se torna inviável

Puzzle 3x3

Montar a imagem original reorganizando as peças, podendo movimentar as peças usando a única lacuna.

Como resolver computacionalmente?

Puzzle 3x3

Como resolver computacionalmente?

- Abstrair que os movimentos se baseiam na lacuna: ela pode ir para cima, baixo, direita e esquerda;
- Representar o ‘puzzle’ como uma estrutura de dados;
- Armazenar a quantidade de movimentos (mínima) necessárias para alcançar uma dada posição.
- Descobrir o mínimo para chegar à posição final.

Puzzle 3x3

Exemplo:

Peças são representadas por números entre 1 e 8

Lacuna é representada pelo número 0

1	**0**	**3**
4	2	5
7	8	6

Representação decimal/string: 103 425 786

Transições: 013 425 786 / 130 425 786 / 123 405 786

Sokoban

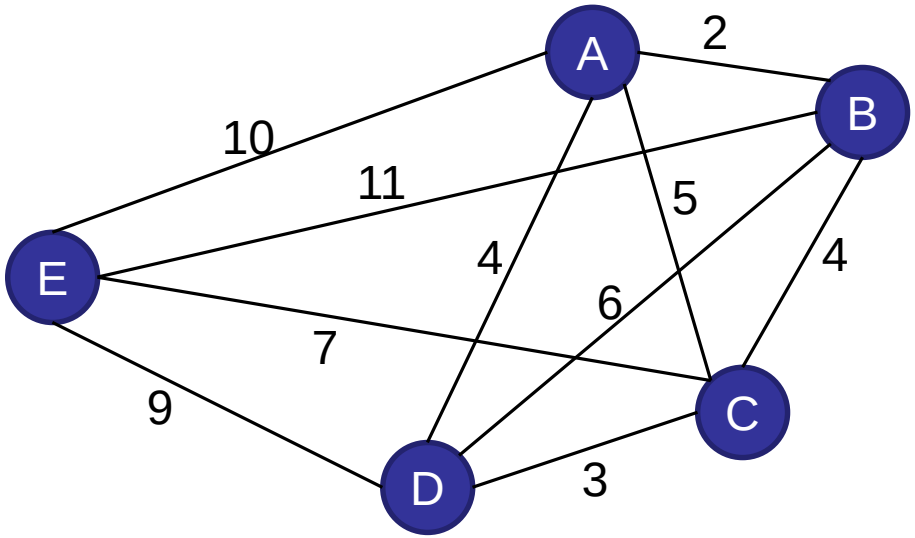
Levar as caixas para pontos objetivos, em qualquer ordem, com o menor número de movimentos possível.

Quais valores representam um estado neste problema?

Problema do Caixeiro Viajante

Um vendedor deseja visitar *n* cidades e retornar a cidade de origem. Dado um grafo não orientado completo com *n* vértices, onde existe um custo *c*(*i*, *j*) (associado a cada aresta) para viajar da cidade *i* a cidade *j*.

Qual o trajeto com custo mínimo?



UDESC

Exercícios

Escreva como seria codificada a sequência de caracteres: “Abracadabra pe de cabra” utilizando a codificação de Huffman. Mostre também a árvore criada no processo de codificação.

(LEMBRETE: os espaços em branco também são caracteres)

Escreva um programa que informa qual o número mínimo de jogadas para resolver o seguinte “puzzle3x3”

1	**0**	**3**
4	2	5
7	8	6

[**Linguagens**: C/C++/C#/JAVA/Python/Haskell]