

Análise de Algoritmos

Introdução à Projeto e análise de algoritmos

Projeto e análise de algoritmos



- Introdução aos Algoritmos Gulosos
- Técnica para projeto de algoritmos eficientes
- Permite atender requisitos de performance em casos de problemas complexos

Adequada em situações específicas



Técnica voltada para:

- Problemas de otimização
- Trabalha com a melhor escolha em cada ponto do processo, para assim compor uma solução ótima



Técnica voltada para:

- Problemas de otimização
- Trabalha com a melhor escolha em cada ponto do processo, para assim compor uma solução ótima

Visão geral de funcionamento:

Quando subproblemas podem ocorrer de forma repetida em diversas etapas



Técnica voltada para:

- Problemas de otimização
- Trabalha com a melhor escolha em cada ponto do processo, para assim compor uma solução ótima

Visão geral de funcionamento:

- Quando subproblemas podem ocorrer de forma repetida em diversas etapas
- Trata cada situação de forma a realizar a melhor escolha naquele momento



Técnica voltada para:

- Problemas de otimização
- Trabalha com a melhor escolha em cada ponto do processo, para assim compor uma solução ótima

Visão geral de funcionamento:

- Quando subproblemas podem ocorrer de forma repetida em diversas etapas
- Trata cada situação de forma a realizar a melhor escolha naquele momento
- Considera que a soma das melhores escolhas parciais leva à melhor escolha global



Comparação com programação dinâmica:

- Ambas tratam de um problema geral a partir de uma subdivisão em subproblemas menores
- O resultado final depende da integração dos resultados parciais



Comparação com programação dinâmica:

- Ambas tratam de um problema geral a partir de uma subdivisão em subproblemas menores
- O resultado final depende da integração dos resultados parciais

Diferença básica:

- Algoritmos gulosos usam abordagens mais simples, para escolhas locais consideradas ótimas
- Possibilidade de redução dos aspectos tratados em cada interação



Voltados para problemas de otimização

Problemas de otimização

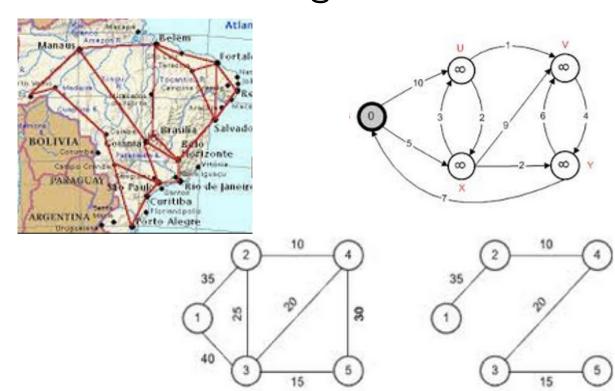
- Podem ter diversas soluções possíveis
- Valores ótimos dependem de contextualização
- Existem parâmetros de aceitação
- Existem várias soluções ótimas



Exemplos de problemas tratados:

- Seleção de tarefas
- Árvore geradora mínima
- Caminhos mais curtos em grafos

ALC: UNKNOWN	the state of the s	M	_	Charles No. 1 1 1 1 1	,000,000	- bear
ei, li Tjorn						
and the same of						
1914						340.00 + 340 + 340 +
19,45		100	194		OCHS:	
INV ex.	- 8	166	194		IOCHER:	990 110 17067
4046		196	E14		ECYCN.	2003/04/05
-Chees	- N.	196	C14	There is a year throught		Street (Antiles India
JAPAN				\$ KOOLSHIP GRAND TOO		James Sal Common St. Berlin
A19.0		18.		CONTRACTOR SANCTON		P11 944
19099		146	. (19		0大河州.	
1 Is Northead		100	143		60000	
1000		113	.00			
A3990	19			Children Control (19)		part (a female in 1 de
3486		196	-:194	2 months and part (4)	THE R.	Description .
	- 8	- 1		Additional Law Market		Mary Name and Street
-5494A	. 146	- 16	1114		sorula M	
interpt.	. 14	- 18	- 613			
753.00			124			
-049/TE	- 1			SAME THE PROPERTY.		FM 919
500.04	B	714	F18	STREET LOCK KIND SPAN		SHAMMER B
-00559am	- K	196	138	3 became property (1996)	E0038	STATE OF STATE
Tron.	- B	(%)	- 01	The a major should be	000028	The ALCOHOLDS
larie .		76	-101	Characteristic explication		profit for limit born
(444)			112	CALLOCAN LY VIOLATIVA	01.02	
A00144	- 16			This proposite (Obto	ECOD.	Michigan
ATM	- B	735	110	STORE AND PERSONS ASSESSED.	the resident	Australia barries
dia.		- 05	129	196	ALC: UNKNOWN	
7617	. 6	/8		\$ 35000 per hand/5000	DC1086	(A) to become
PER	. K	- 125	C14	CARDOLINESK, SYNC		PERMIT
40 M			1112	3.940390694065455		347710
(jef ut.	N	791	313	3.600.644mcNuSSES		34436
hill the same	. 15		CIE			Northead
for et	- 1	170	192	2 0000 mm (445 (50)		Spinisters.
(0,75a)			197	350 HOPE NO. 15 (15)		2550X 6655 THE
14/5.01	- 15	146	198	3 NO AND RESERVED TO SERVE		THE BOY ATTOM
printer.	- 14	-31%	137	None Action As Publisher		me"Chille Shine
u/s0 ps	. 15	536	138	- Shrannamh-650		and School street
specified.		- 595	-53E	Sommer to Str.		the Senior type byje
Pat		-175	109	SMITTH THE STATE OF THE STATE O		14
MANUAL.	.75	192	.716	_35000000000000000000000000000000000000		
FOM	. 8	195	F18.	3.8990 ALPAN SEC.		H.ist
igen.	3	196	518	Committee to [4]		las.
Oyes	- 5	- / / 794	323	Shrannening 80		304
loforn.		- 10	192	35000000000000000000000000000000000000		
Norther Al	15	- 18	199	S beakers restrict		PROMEMICS.
West,	- 15	AF96	CHE.			Me
APRIL MARK	- B	- 495	102	Shranasan (MSS)		Fyr intra tribes
rear in		- 75	-101	Sharping NOW		District Sea
NH.	. 16		- 111	25000 to 100000		3106/07/10006
pa .	N	79.	- 191	2 900 NOW CENT		3,5690,00917005
hali	- 4	186	144	The base belower of the best		Employ Substitution



Algoritmos Gulosos – estrutura geral



```
função ALGORITMOGULOSO(C: conjunto)
    S \leftarrow \emptyset
    enquanto C \neq \emptyset e não solução(S) faça
        x \leftarrow \text{seleciona } C
        C \leftarrow C \{x\}
        se é viável S \cup \{x\} então
             S \leftarrow S \cup \{x\}
        fim se
    fim enquanto
    se solução(S) então
        retorne S
    senão
        retorne "Não existe solução!"
    fim se
fim função
```

ightharpoonup C é o conjunto de candidatos ightharpoonup S é o conjunto que irá conter a solução



Etapas gerais:

Determinar a subestrutura ótima do problema



- Determinar a subestrutura ótima do problema
- Descrever o processo recursivo de solução



- Determinar a subestrutura ótima do problema
- > Descrever o processo recursivo de solução
- Analisar o resultado da escolha gulosa (deve ser somente um subproblema)



- Determinar a subestrutura ótima do problema
- > Descrever o processo recursivo de solução
- Analisar o resultado da escolha gulosa (deve ser somente um subproblema)
- Avaliar se a escolha gulosa é sempre correta e segura



- Determinar a subestrutura ótima do problema
- > Descrever o processo recursivo de solução
- Analisar o resultado da escolha gulosa (deve ser somente um subproblema)
- Avaliar se a escolha gulosa é sempre correta e segura
- Desenvolver um algoritmo recursivo



- > Determinar a subestrutura ótima do problema
- > Descrever o processo recursivo de solução
- Analisar o resultado da escolha gulosa (deve ser somente um subproblema)
- Avaliar se a escolha gulosa é sempre correta e segura
- Desenvolver um algoritmo recursivo
- Transformar o algoritmo de recursivo em iterativo



Abordagem geral

- Expressar o problema de otimização de forma que:
 - Seja definida uma única escolha
 - Seja identificado um único subproblema para resolver



Abordagem geral

- Expressar o problema de otimização de forma que:
 - Seja definida uma única escolha
 - Seja identificado um único subproblema para resolver
- Mostrar que existe uma escolha ótima e que a escolha gulosa é sempre segura



Abordagem geral

- Expressar o problema de otimização de forma que:
 - Seja definida uma única escolha
 - Seja identificado um único subproblema para resolver
- Mostrar que existe uma escolha ótima e que a escolha gulosa é sempre segura
- Demonstrar a subestutura ótima:
 - A solução ótima para o problema, combinada com a escolha gulosa leva à solução global ótima



Propriedades da escolha gulosa (1)

Montar uma solução global ótima fazendo escolhas locais ótimas



Propriedades da escolha gulosa (1)

- Montar uma solução global ótima fazendo escolhas locais ótimas
- A cada momento, considera a escolha a fazer, sem analisar os demais subproblemas



Propriedades da escolha gulosa (1)

- Montar uma solução global ótima fazendo escolhas locais ótimas
- A cada momento, considera a escolha a fazer, sem analisar os demais subproblemas
- Ocorre de cima para baixo, reduzindo cada instância do problema a um problema menor



Propriedades da escolha gulosa (1)

- Montar uma solução global ótima fazendo escolhas locais ótimas
- A cada momento, considera a escolha a fazer, sem analisar os demais subproblemas
- Ocorre de cima para baixo, reduzindo cada instância do problema a um problema menor
- Diferente de programação dinâmica: a seleção da melhor opção não depende das demais seleções; não age de baixo para cima.



Propriedades da escolha gulosa (2)

- Aplicável em problemas com subestrutura ótima:
 - Uma solução ótima para o problema é também a solução ótima para os subproblemas



Propriedades da escolha gulosa (2)

- Aplicável em problemas com subestrutura ótima:
 - Uma solução ótima para o problema é também a solução ótima para os subproblemas
- Avaliar se a escolha gulosa em um subproblema, em conjunto com a escolha anterior, leva à solução ótima



Exemplo simples – troco de moedas

Visão geral: como identificar o menor número de moedas para dar como troco para um determinado valor?



Exemplo simples – troco de moedas

Visão geral: como identificar o menor número de moedas para dar como troco para um determinado valor?

Versão intuitiva: repetição de um ciclo com a escolha da maior moeda possível, a cada etapa.



Exemplo simples – troco de moedas

Visão geral: como identificar o menor número de moedas para dar como troco para um determinado valor?

Versão intuitiva: repetição de um ciclo com a escolha da maior moeda possível, a cada etapa.

Ex.: R\$ 2,89 [100, 50, 25, 10, 1]



Exemplo simples – troco de moedas

Visão geral: como identificar o menor número de moedas para dar como troco para um determinado valor?

Versão intuitiva: repetição de um ciclo com a escolha da maior moeda possível, a cada etapa.

Ex.: R\$ 2,89 [100, 50, 25, 10, 1] \rightarrow 2x1,00 + 3x0,25 + 1x0,10 + 4*0,01

[10]



Exemplo simples – troco de moedas

Visão geral: como identificar o menor número de moedas para dar como troco para um determinado valor?

Versão intuitiva: repetição de um ciclo com a escolha da maior moeda possível, a cada etapa.

Ex.: R\$ 2,89 [100, 50, 25, 10, 1]

- \rightarrow 2x1,00 + 3x0,25 + 1x0,10 + 4*0,01
- \rightarrow 2x1,00 + 1x0,50 + 1x0,25 + 1*0,10+ 4*0,01 [9]

fim função



Exemplo simples – troco de moedas

```
função Troco(n)
    const C \leftarrow \{100, 25, 10, 5, 1\}

    ▷ C é o conjunto de moedas

    S \leftarrow \emptyset

⊳ S é o conjunto que irá conter a solução

                                                                                   \triangleright s é a soma dos itens em S
    s \leftarrow 0
    enquanto s \neq n faça
         x \leftarrow 0 maior item em C tal que s + x \le n
         se este item não existe então
             retorne "Não foi encontrada uma solução!"
         fim se
         S \leftarrow A \cup \{\text{uma moeda de valor x}\}\
         s \leftarrow s + x
    fim enquanto
    retorne S
```