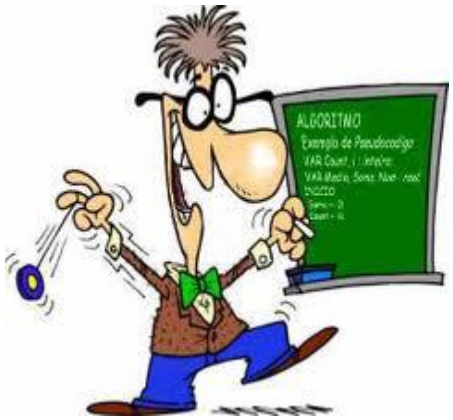
Algoritmos e Estrutura de Dados - I

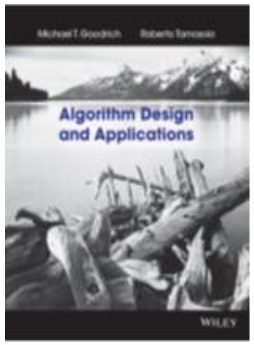
Unidade 1 - Introdução à Algoritmos

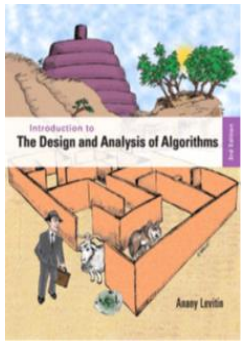


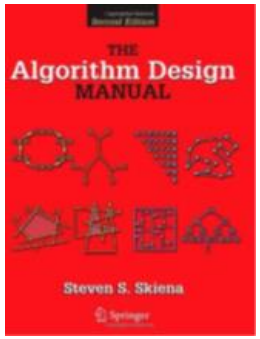
Prof. Aparecido V. de Freitas
Doutor em Engenharia
da Computação pela EPUSP
aparecidovfreitas@gmail.com

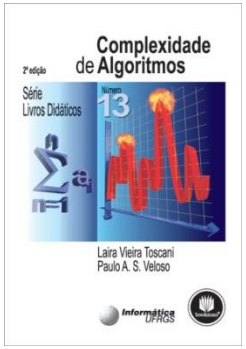
Bibliografia

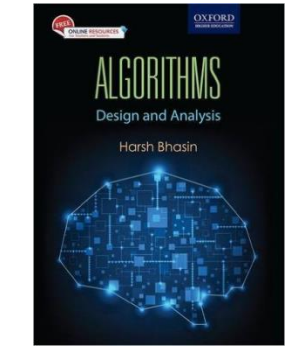
- Algorithm Design and Applications – Michael T. Goodrich, Roberto Tamassia, Wiley, 2015
- Introduction to the Design and Analysis of Algorithms – Anany Levitin, Pearson, 2012
- The Algorithm Design Manual – Steven S. Skiena, Springer, 2008
- Complexidade de Algoritmos – Série Livros Didáticos – UFRGS
- Algorithms – Design and Analysis – Harsh Bhasin – Oxford University Press – 2015
- Algoritmos – Teoria e Prática – Cormen – Segunda Edição – Editora Campus, 2002
- Data Structures using C – Oxford University Press – 2014
- Data Structures Using C – A. Tenenbaum, M. Augensem, Y. Langsam, Pearson 1995
- C By Dissection – Kelley, Pohh – Third Edition – Addison Wesley

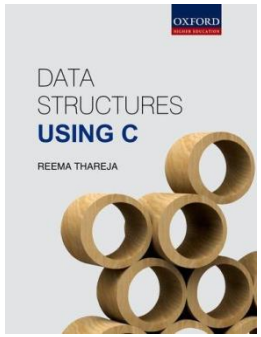


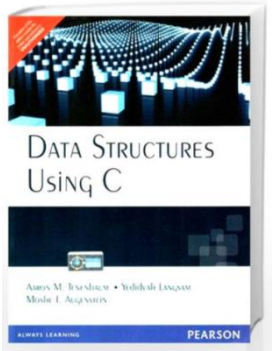


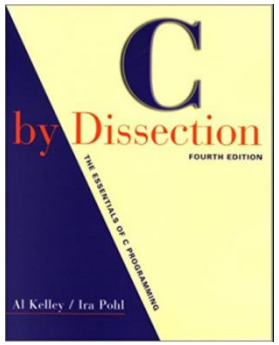




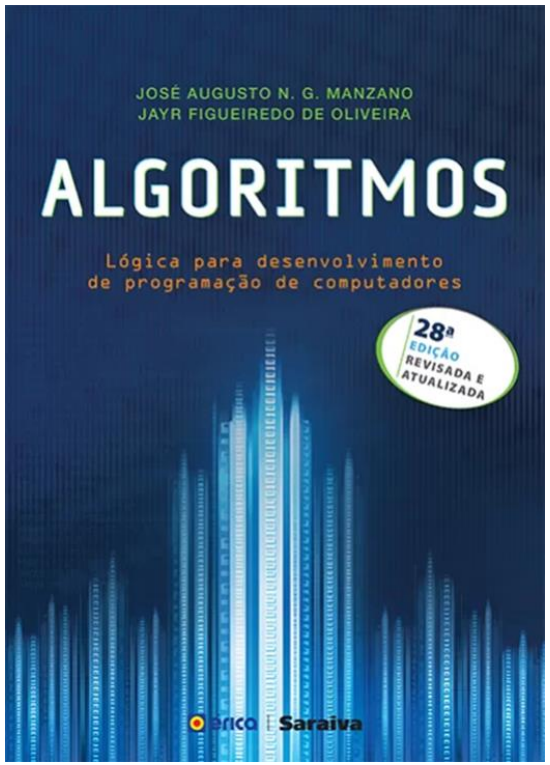


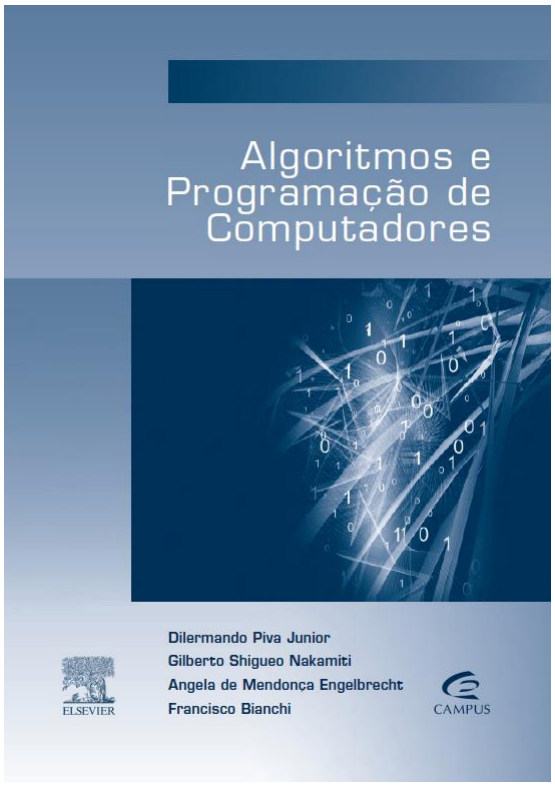


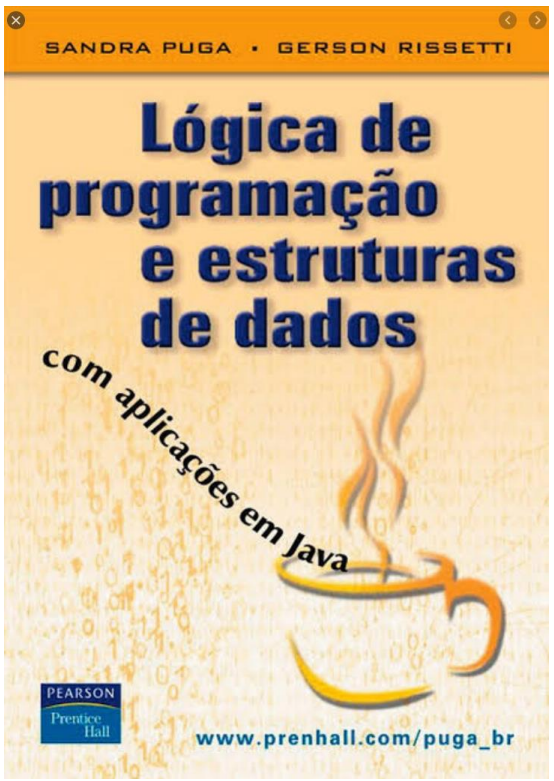




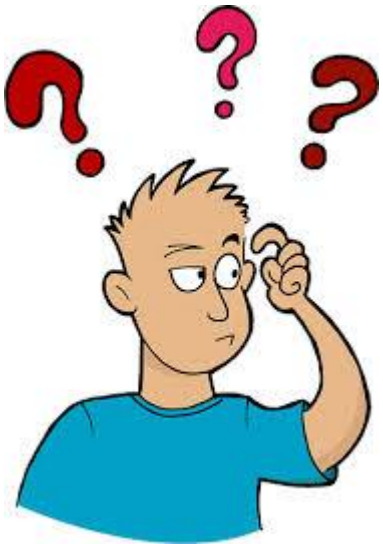
Bibliografia

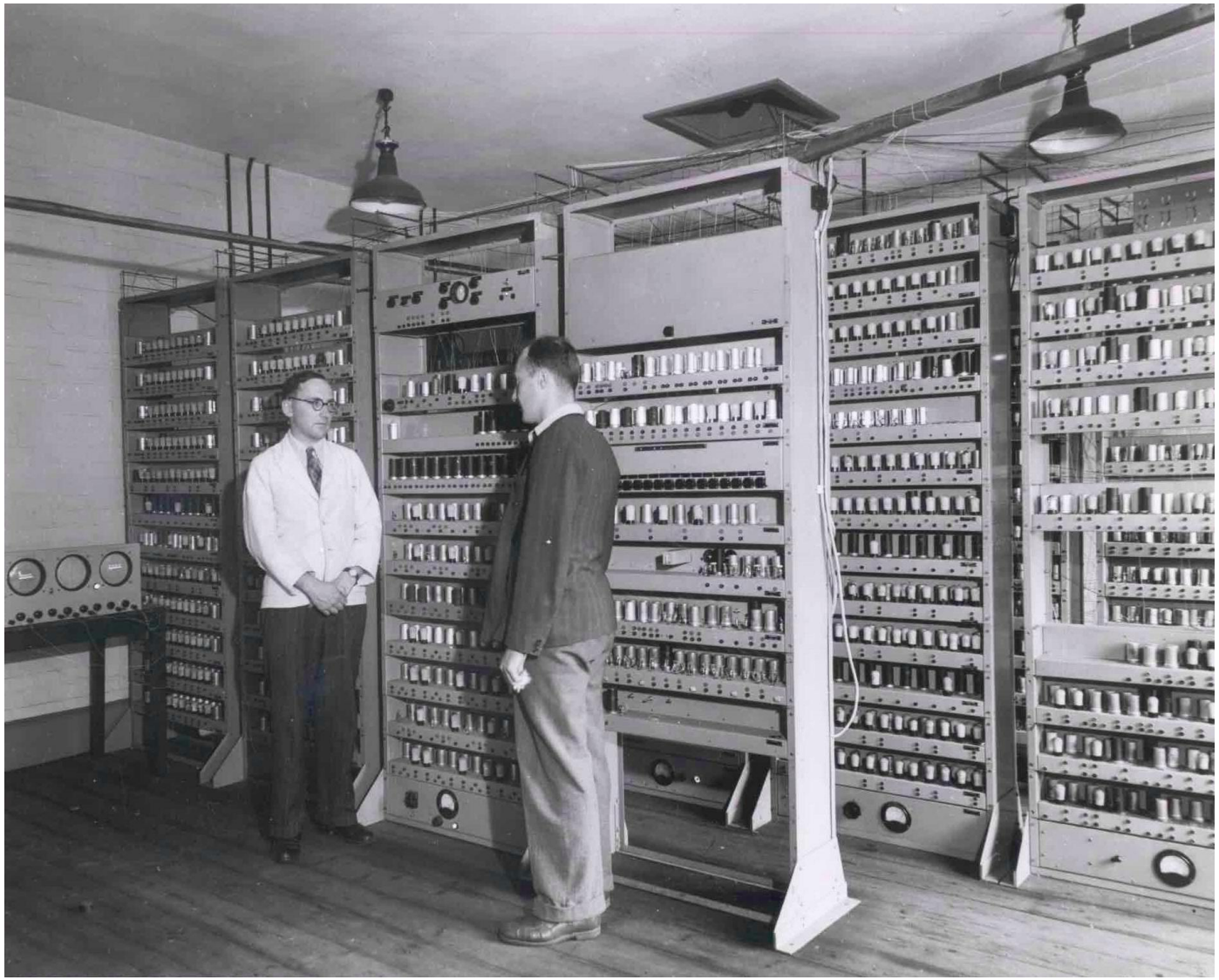


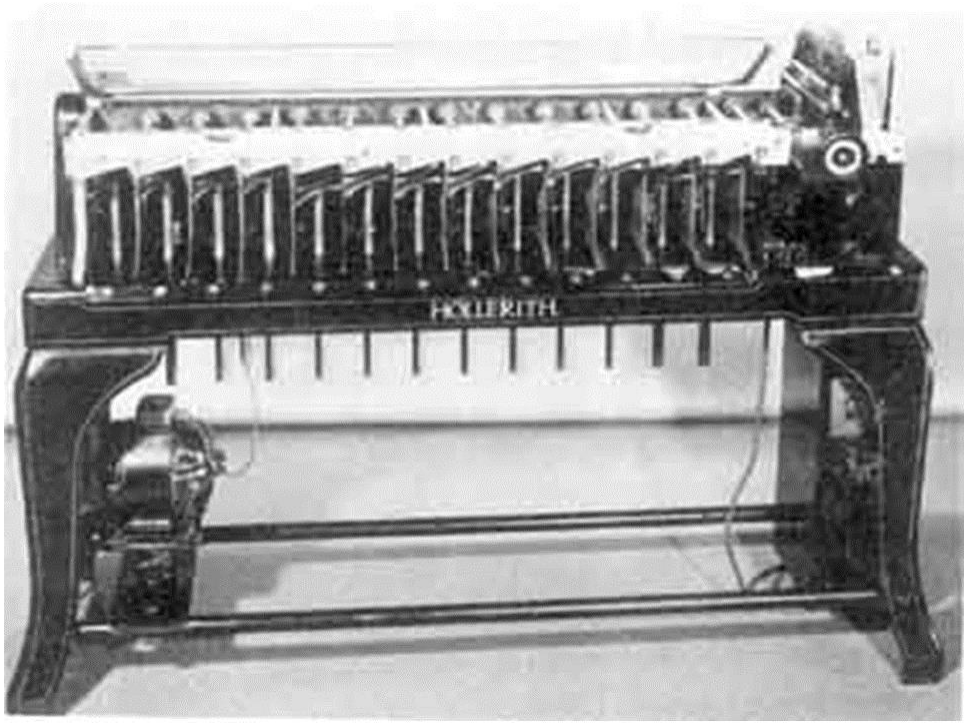


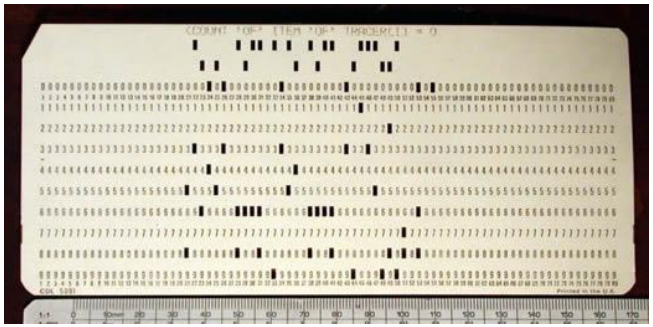


A área de Informática (TI) é recente ?






HOLLERITH



A evolução de TI afetou a vida das pessoas ?

Como seria a rotina de um banco sem TI ?

Um supermercado sem Informática ?

Carrefour

sonda Supermercados

extra hipermercados

WAL★MART SUPERCENTER

Pois é, apesar de recente…

a sociedade e o negócio das organizações cada vez mais dependem da Informática !

O Profissional de Informática tem ocupado posições de destaque nas Organizações

O Profissional de Informática é bem remunerado ?

Microsoft

ORACLE

Fonte: Revista Veja – Outubro/2009

Como ingressar no Mercado de Trabalho ?

- O Mercado basicamente é dividido em:

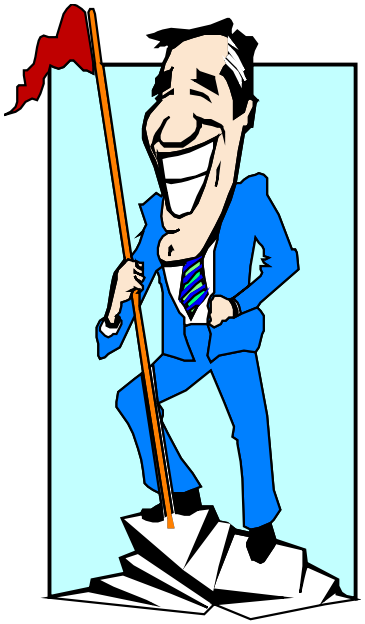
Usuários

Profissionais

Quais os cargos do Mercado ?

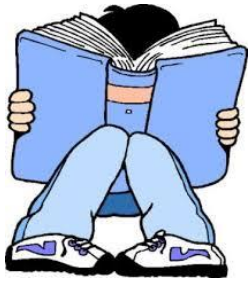
- ✓ Analistas de Sistemas
- ✓ Analistas de Negócios
- ✓ Desenvolvedores Web
- ✓ Arquitetos de Soluções
- ✓ Programadores
- ✓ Analistas de Suporte
- ✓ Analistas de Rede
- ✓ Analistas de Produção
- ✓ Gerente de Projetos
- ✓ DBA
- ✓ . ...

Como vencer profissionalmente ?



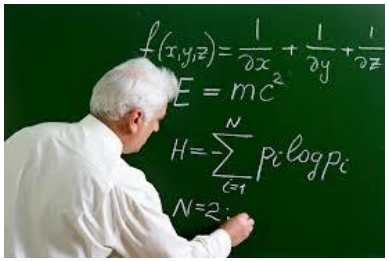
- ✓ Conhecimento
- ✓ Experiência
- ✓ Fazer a diferença
- ✓ Força de Vontade
- ✓ Iniciativa
- ✓ Ter prazer no que faz

Quais as habilidades requeridas ?



- ✓ Conhecimento Técnico
- ✓ Trabalho em Equipe
- ✓ Comunicação Escrita e Verbal
- ✓ Relacionamento Humano
- ✓ Ética

Por isso é fundamental escolher uma Universidade que ofereça estrutura , forte embasamento conceitual , programas de parceria e experiência.



A maior parte dos cargos de Informática (os mais remunerados) em geral exigem titulação em Nível Superior.

O ponto de encontro dos profissionais de informática

APinfo

imes USCS

Profissionais | Empresas | Informação | Publicidade | Fale Conosco

Filtros da Pesquisa

Limpar Pesquisa

Data: (dd/mm/aa) (opcional)

De: Até:

Palavras-chave ou código da vaga : (opcional)

programador Java

- (•) Pesquisar apenas no titulo da vaga
- ( ) Pesquisar : titulo / Descrição / cidade

- (•) Todas as palavras devem estar presentes
- ( ) Alguma das palavras deve estar presente

Filtrar

IT HELP

Tire suas dúvidas e responda dúvidas de outros profissionais de TI

VEJA O RESULTADO DA PESQUISA SALARIAL APINFO 2018

Estado

Resultado da pesquisa :

Encontradas : 83 vagas - Mostrando de 1 até 8

São Paulo - SP - 08/02/19

Analista Programador JAVA

Requisitos:
Ter atuado em sustentação de sistema
Experiência com java, utilizando conceitos:

- J2EE, EJB, Webservices
- Spring, Spring Batch e Spring MVC
- Struts, JSP, Servlets
- Javascript, jQuery
- SQL - Oracle e DB2
- Maven
- Spring

Desejável:
Conhecimento em arquitetura Cloud Google
Red Hat OpenShift
Cobol/Mainframe
C# e VB
Servidores WSAD, Websphere e Jetty
Vivência em projetos ágeis e auto gerenciáveis.

Local : Faria Lima - SP
Período: Indeterminado
Forma de contratação: CLT + Benefícios

Empresa .....: Sinqia
Código .....: 43454

Envie seu currículo

São Paulo - SP - 08/02/19

Analista Programador Java Jr

Profissionais ▾ Empresas ▾ Informação ▾ Publicidade Fale Conosco

Pesquise por vagas | Pesquise por currículos

Digite sua busca aqui... palavras chave, ou o codigo da vaga, ou do currículo

Pesquisa : Salários e mercado de TI

Pesquisa salarial.

São Paulo

Rendimento mensal em R$

Cargo	Especialidade	CLT	Terceiro
AD Adm. de Dados		6.799	8.820
Adm. de rede		4.646	6.464
Adm. De Sistemas / Aplicações		5.842	7.772
Analista DBM		5.354	6.835
Analista de Aplicações		4.136	5.308
Analista de BI		7.794	9.790
Analista de Infraestrutura		4.825	6.285
Analista de monitoramento		2.404	3.266
Analista de Negócios		7.191	9.862
Analista de processos		5.906	7.828
Analista de produção	Mainframe	5.303	6.470
Analista de produção	Unix	4.108	4.849
Analista de Projetos		6.028	7.932

Analista de Qualidade		5.173	6.880
Analista de Requisitos		7.276	9.028
Analista de segurança		5.679	7.381
Analista de sistemas	Mainframe	6.790	9.627
Analista de sistemas	C#	6.247	8.728
Analista de sistemas	Java	7.255	9.725
Analista de sistemas	PHP	4.485	5.841
Analista de sistemas	Oracle PL/SQL	6.492	8.381
Analista de Suporte	Mainframe	5.221	6.291
Analista de Suporte	Unix	4.787	5.710
Analista de Suporte	ERP	3.683	4.989
Analista de Suporte	Redes	3.305	4.300
Analista de Suporte	Windows	2.773	3.513
Analista de Suporte	Linux	2.857	3.099
Analista de Telecomunicações		3.830	5.098
Analista de testes		4.828	6.594
Analista Funcional		7.217	9.828
Analista Microinformática		2.275	2.761
Analista Programador	C#	6.137	8.432
Analista Programador	Java	6.601	8.299
Analista Programador	ASP	3.979	5.361
Analista Programador	Mainframe	4.984	6.936
Analista Programador	PHP	4.185	5.127
Analista Programador	C++	5.359	6.858
Analista Programador	ABAP	7.042	9.216
Analista Programador	Visual Basic	4.341	5.852

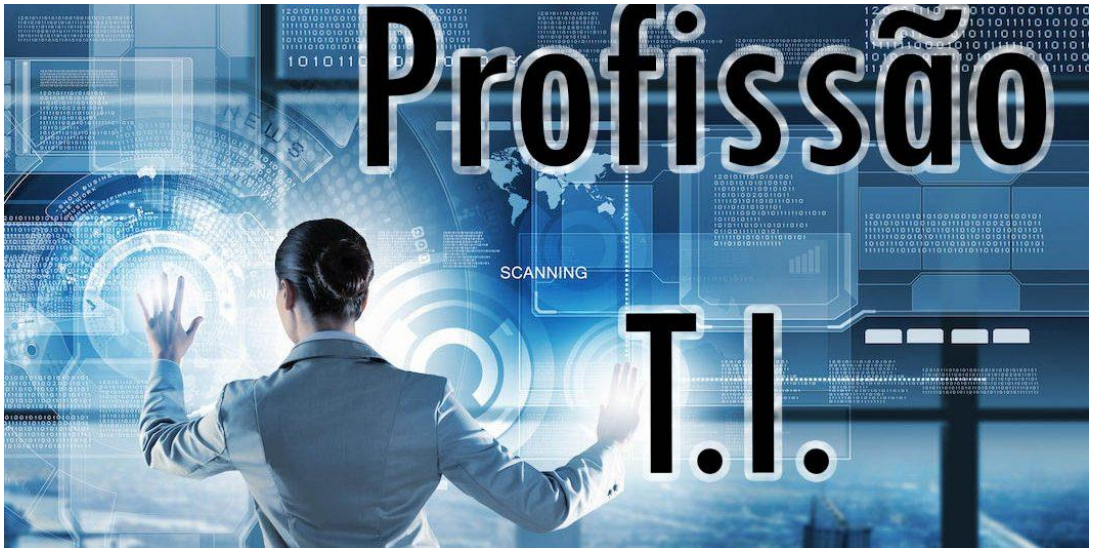
Analista Programador	Oracle PL/SQL	6.177	8.113
Analista Segurança Sistemas		6.784	8.889
Arquiteto de Sistemas		10.037	12.191
Assistente de Informática		1.799	
Auditor de Sistemas		4.272	5.868
Consultor ERP		8.892	13.475
Consultor especialista	Client/Sever	9.146	11.764
Consultor especialista	Mainframe	6.957	8.833
Coordenador de projetos		10.939	12.692
Coordenador de Suporte		6.092	7.853
Coordenador de testes		7.074	9.397
DBA Adm. Base de Dados		9.565	12.665
Engenheiro de Sistemas		7.861	10.223
Engenheiro de Telecom		6.377	7.837
Estagiário		1.318	
Gerente de Contas		8.637	9.811
Gerente de operação		9.763	10.945
Gerente de Projetos		13.304	15.560
Gerente de Sistemas		15.874	17.325
Gerente de Suporte		11.412	12.640
Instrutor		2.540	2.957
Lider de Projetos		9.789	12.782
Operador / Scheduller	Mainframe	2.354	3.026
Operador / Scheduller	Unix	2.468	2.871
Programador	C#	4.492	6.418
Programador	Java	5.320	7.890
Programador	PHP	3.554	4.498
Programador	Delphi	3.068	4.094
Programador	Visual Basic	2.371	3.061

Gerente de Suporte		11.412	12.640
Instrutor		2.540	2.957
Lider de Projetos		9.789	12.782
Operador / Scheduller	Mainframe	2.354	3.026
Operador / Scheduller	Unix	2.468	2.871
Programador	C#	4.492	6.418
Programador	Java	5.320	7.890
Programador	PHP	3.554	4.498
Programador	Delphi	3.068	4.094
Programador	Visual Basic	2.371	3.061
Programador	ABAP	5.596	8.252
Programador	Cobol	3.472	5.084
Programador	Java Script	2.756	3.658
Programador	Phyton	5.871	8.037
Programador	Visual Studio	4.651	6.436
Recrutador(a) RH para TI		3.289	4.598
Técnico de Hardware		1.854	2.252
Tecnico de Suporte		1.933	2.784
Web Designer		3.120	3.970
Webmaster		3.222	3.927

Dado – IDC*

- ✓ Atualmente há 17.000 vagas em aberto no país, por falta de mão-de-obra especializada na área de TI. (MCT)
- ✓ Haverá mais de 100.000 vagas em aberto nos próximos três anos.

* International Data Corporation



Quais são os cursos ?

- ✓ Ciência da Computação
- ✓ Sistemas de Informação
- ✓ Graduação Tecnológica
- ✓ Engenharia de Computação

Ciência da Computação

- ✓ Computação como atividade fim
- ✓ Forte embasamento tecnológico

Sistemas de Informação

- ✓ Computação como atividade meio
- ✓ Foco em na aplicação da Tecnologia em negócios

Computação na USCS

- ✓ O curso de Ciência da Computação iniciou em 1976
- ✓ Cursos/Palestras Extracurriculares
- ✓ Parcerias com Empresas de TI
- ✓ Em média 95% empregados

O que é Ciência da Computação ?

Ciência da Computação

- É uma disciplina que constrói uma sólida base científica para o estudo de diversos temas tais como:
  - projeto de computadores;
  - programação;
  - processamento de sistemas de informação;
  - determinação de soluções algorítmicas de problemas;
  - Inteligência artificial, robótica, etc…
- **Algoritmo** representa o conceito chave para a Ciência da Computação!

Questões tratadas na Ciência da Computação

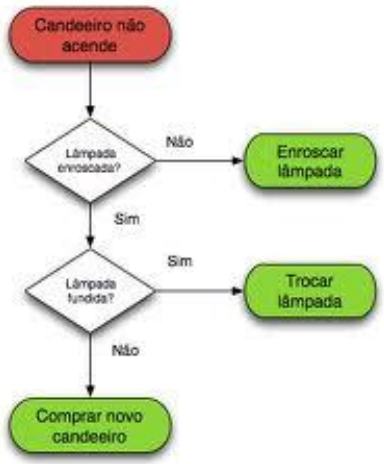
- Quais problemas podem ser resolvidos por processos algorítmicos ?
- Como analizar e comparar algoritmos ?
- Como a aplicação de algoritmos afeta a sociedade ?
- Como tornar mais eficiente os algoritmos existentes ?

Porque se estuda Algoritmos ?

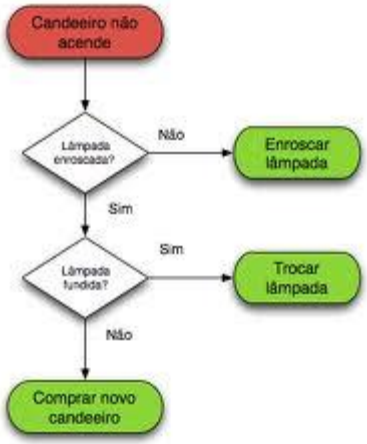
Algoritmos

- **Há razões de ordem Prática e Teórica**

> “ Algoritmos representam mais que uma área da computação. Correspondem ao núcleo da Ciência da Computação, e com toda a imparcialidade, podem ser considerados relevantes para a maioria das ciências, negócios e tecnologia”.



David Harel, 1992



Algoritmos

- Informalmente, um algoritmo é um **conjunto de passos** que define a maneira pela qual uma **tarefa** deve ser executada.

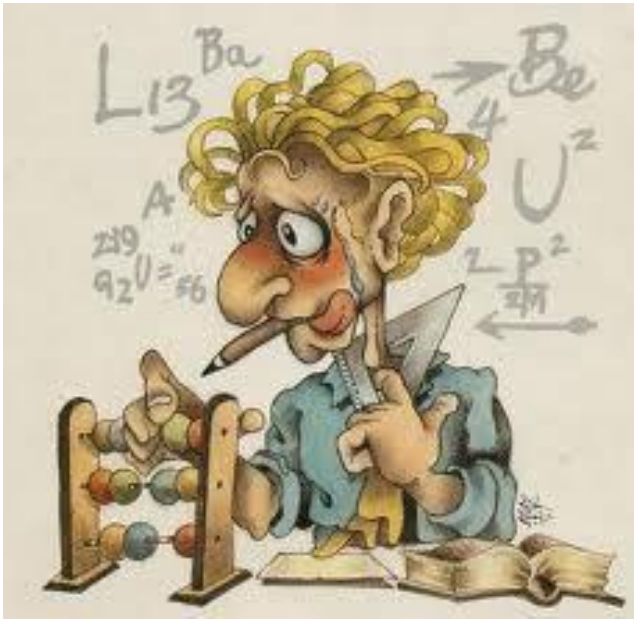
Algoritmos

- Antes do computador executar alguma tarefa, um algoritmo para executar esta tarefa deve ser descoberto e representado de uma forma que seja compatível com a máquina.
- A representação do algoritmo na máquina é denominado programa.
- O processo de desenvolvimento de programas é chamado programação.
- Programas e algoritmos são coletivamente chamados de softwares, em contraste com a máquina a qual é conhecida por hardware.

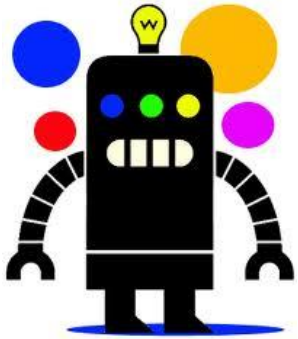
Quem começou a estudar Algoritmos ?

Algoritmos

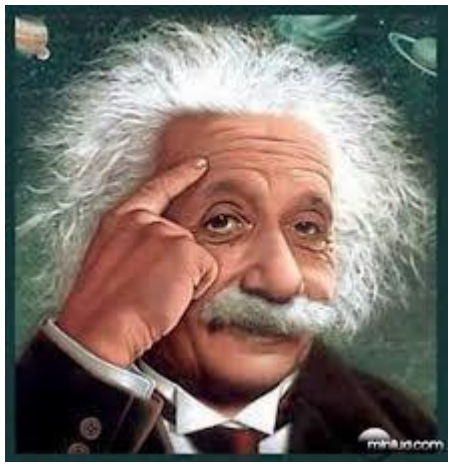
- Estudados inicialmente pelos **matemáticos**.
- Exemplo: Algoritmo de Euclides para determinar o máximo divisor comum entre dois inteiros positivos



Algoritmos



- É através da habilidade de capturar inteligência por meio de **algoritmos** que somos capazes de construir máquinas que executam tarefas úteis.
- Se nenhum algoritmo existir para resolver um problema, então a solução do mesmo está além da capacidade da máquina.
- Por esta razão, o estudo dos algoritmos representa o **core** da Ciência da Computação.



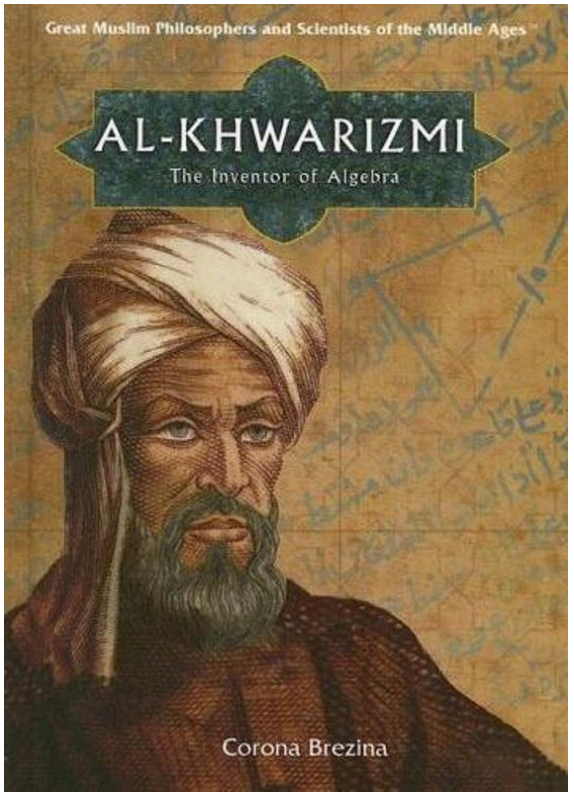
A Ciência dos Algoritmos

- Hoje, a **Ciência da Computação** é considerada a **Ciência dos Algoritmos**.
- O escopo desta Ciência é abrangente, incluindo diversas áreas tais como: Medicina, Biologia, Engenharia, Linguística, Sistemas de Informação, etc.

Qual a origem do termo ?

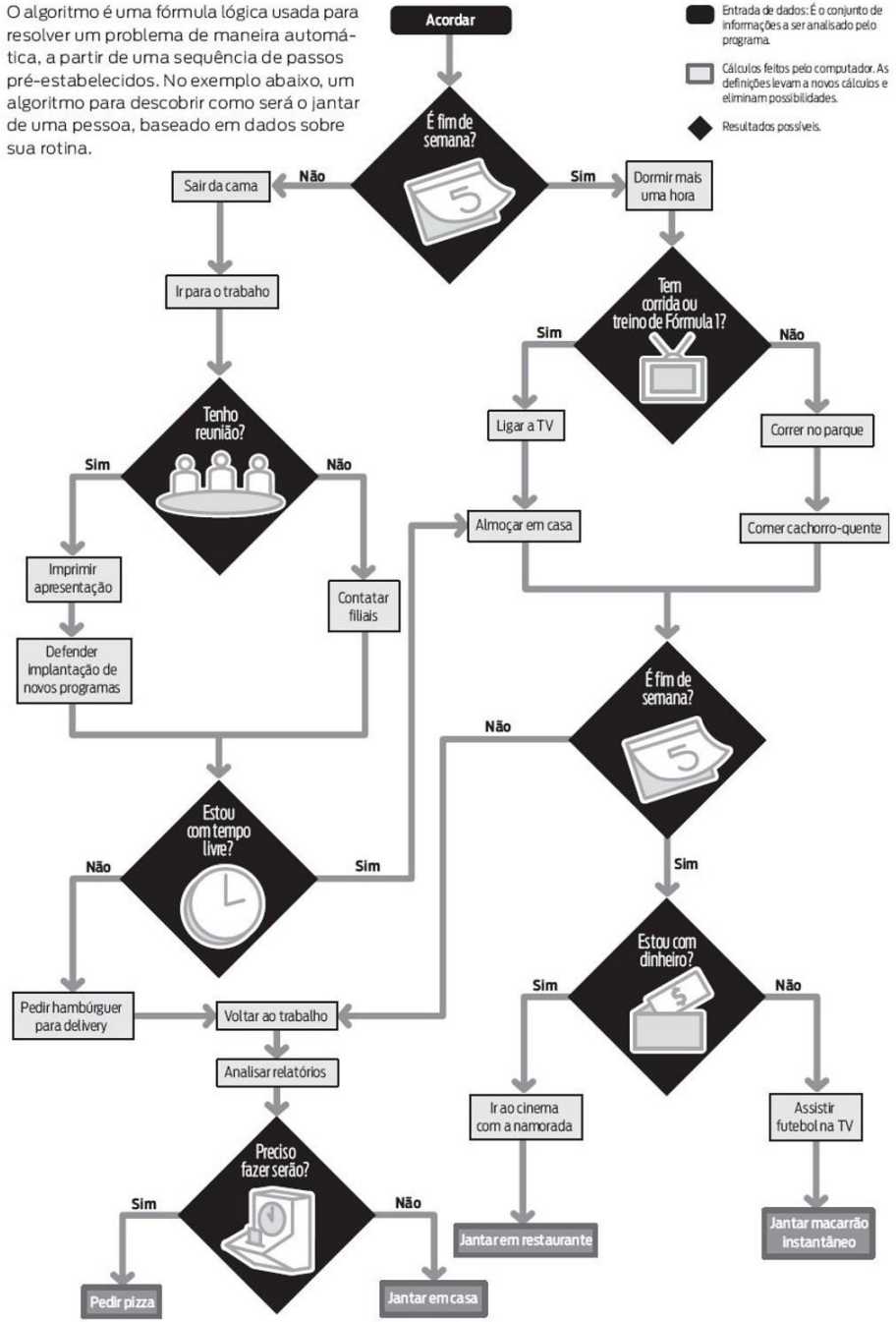
- A palavra algoritmo tem origem no sobrenome, Al-Khwarizmi, do matemático persa do século IX Mohamed ben Musa.

- http://pt.wikipedia.org/wiki/Algoritmo



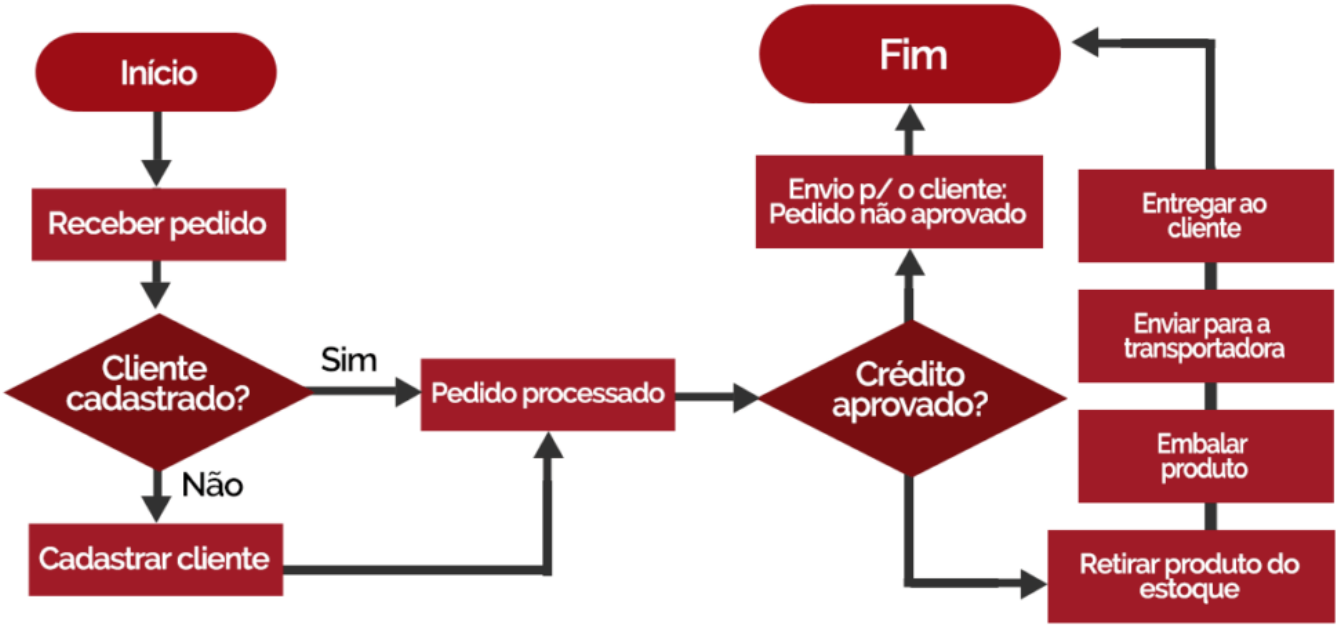
Algoritmo

O algoritmo é uma fórmula lógica usada para resolver um problema de maneira automática, a partir de uma sequência de passos pré-estabelecidos. No exemplo abaixo, um algoritmo para descobrir como será o jantar de uma pessoa, baseado em dados sobre sua rotina.



Algoritmo

- Um algoritmo é uma sequência não-ambígua de instruções para resolver um problema, i. e, para obter uma saída requerida a partir de qualquer entrada legítima em uma quantidade finita de tempo.

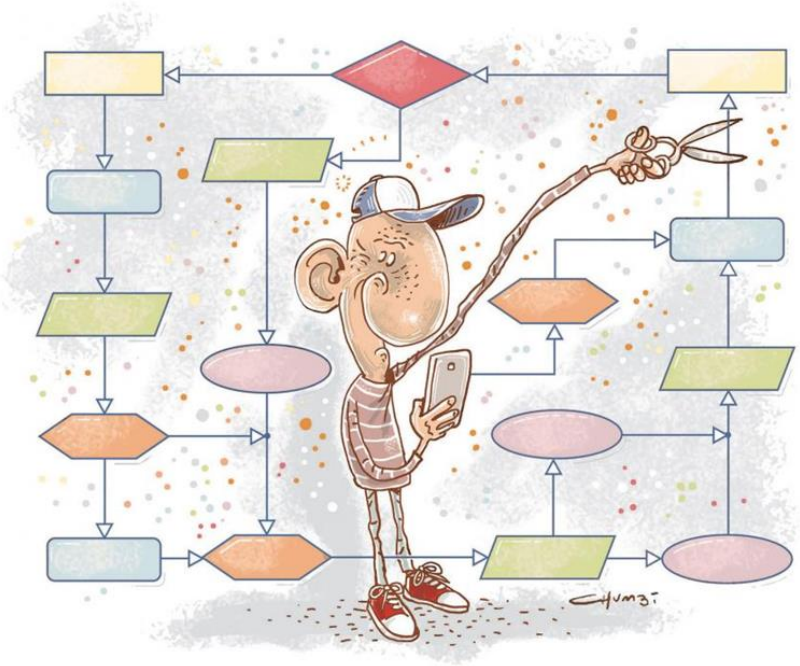


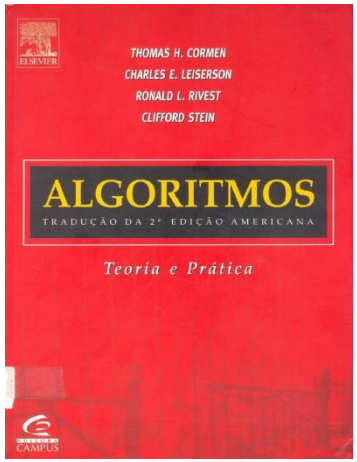
Algoritmo (Cormen)

- Um *algoritmo* é qualquer procedimento computacional bem definido que toma algum valor ou conjunto de valores como entrada e produz algum valor ou conjunto de valores como saída.

- Portanto, um algoritmo é uma sequência de passos computacionais que transformam a entrada na saída.

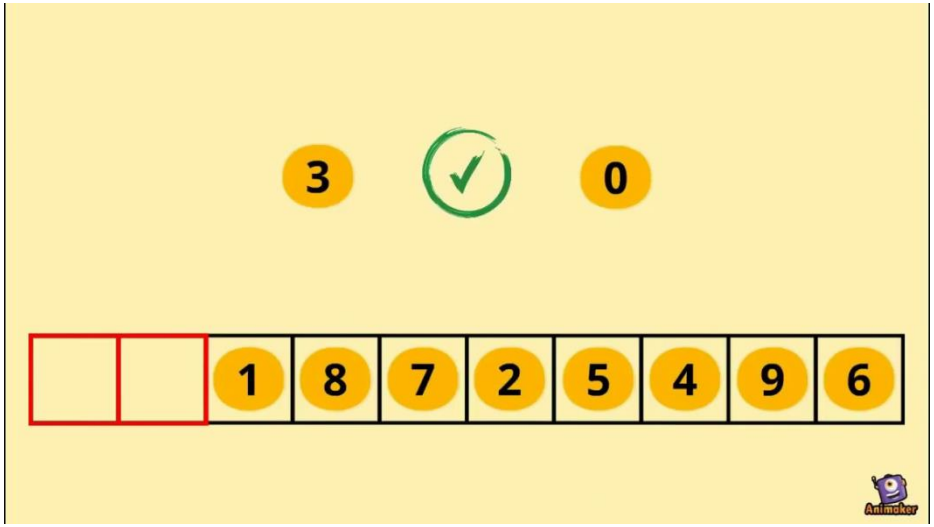
- Ferramenta para resolver um problema computacional bem especificado.





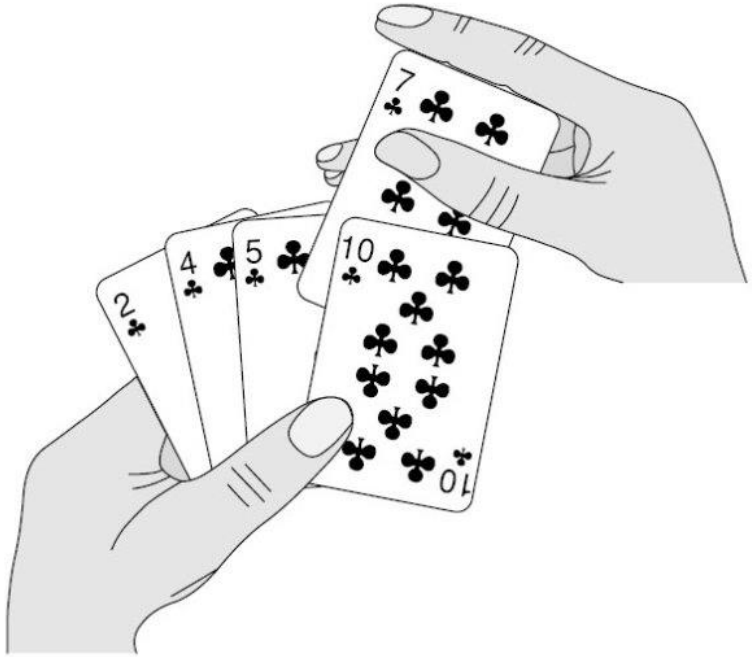
Exemplo - Sorting

- Considere o problema de ordenar uma sequência de números.
- Este problema surge com frequência na prática e oferece um solo fértil para o desenvolvimento de muitas técnicas de projeto e ferramentas de análise.

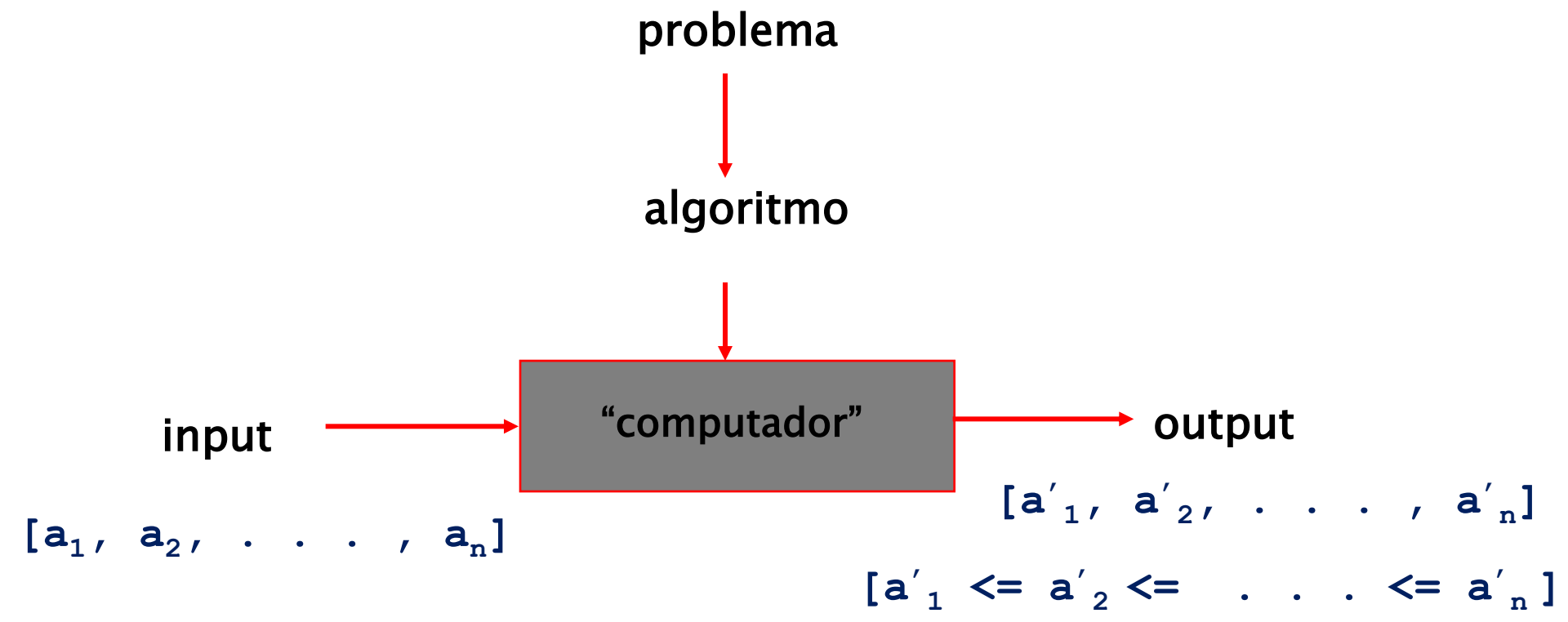


Como especificar o Sorting ?

- Entrada: uma sequência de n números : [a1, a2, . . . , an].
- Saída: uma permutação (reordenação): [a'1, a'2, . . . , a'n] da sequência de entrada, tal que [a'1 <= a'2 <= . . . <= a'n].

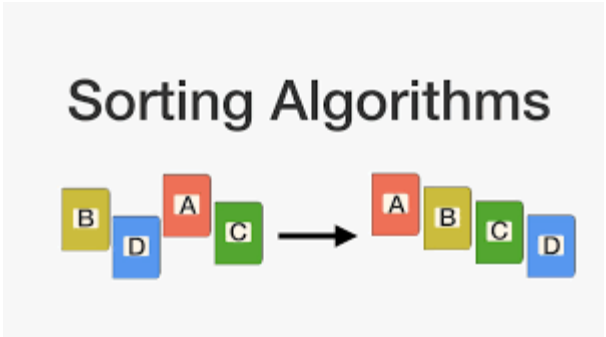


Sorting

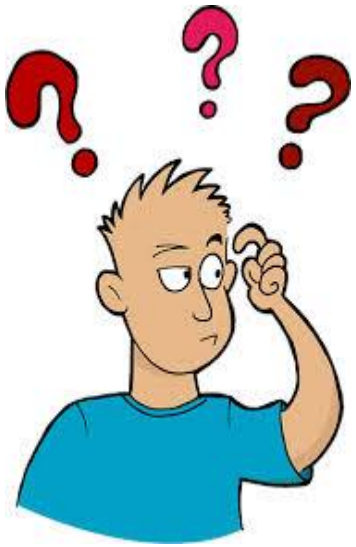

problema
algoritmo
input
"computador"
output
[a1, a2, . . . , an]
[a'1, a'2, . . . , a'n]
[a'1 <= a'2 <= . . . <= a'n]

Sorting - Observações

- A ordenação é uma operação fundamental em Ciência da Computação;
- Muitos programas a utilizam como uma etapa intermediária;
- Como resultado, um grande número de bons algoritmos de ordenação têm sido desenvolvidos;
- Exemplos de Algoritmos de sorting: *Bubble*, *Quick*, *Selection*, *Insertion*, *Merge*, *Heap.*



Que tipo de problemas são resolvidos por algoritmos ?

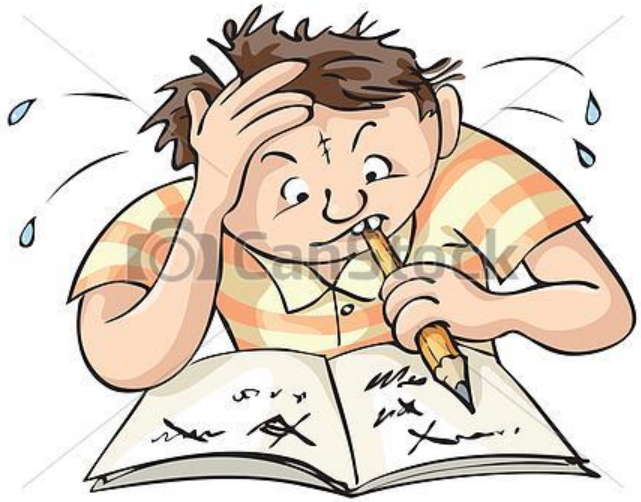


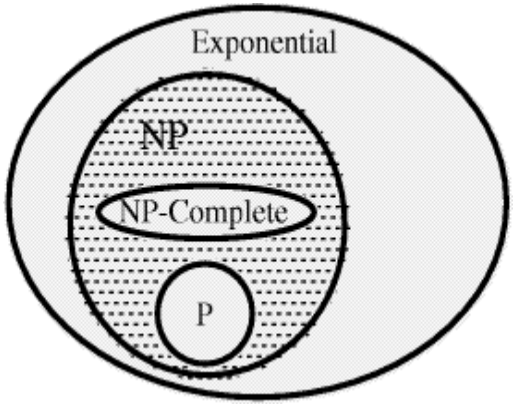
Problemas Computacionais

- Ordenação (Sorting)
- Busca (Searching)
- Processamento de Strings
- Problemas geométricos
- Problemas numéricos
- Problemas combinatórios
- Caixeiro Viajante
- Jogos
- Torres de Hanoi
- Otimização

Problemas difíceis

- A medida habitual da eficiência de um algoritmo é a velocidade, isto é, quanto tempo um algoritmo demora para produzir seu resultado;
- Porém existem alguns problemas para os quais não se conhece uma solução eficiente;
- Existe um subconjunto destes problemas que são denominados **NP-completos**.

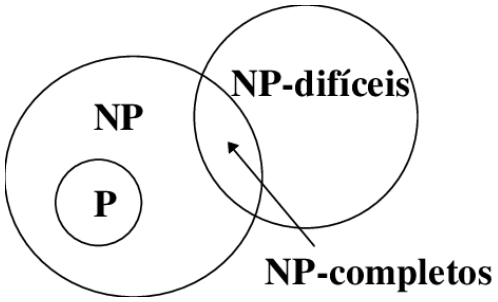




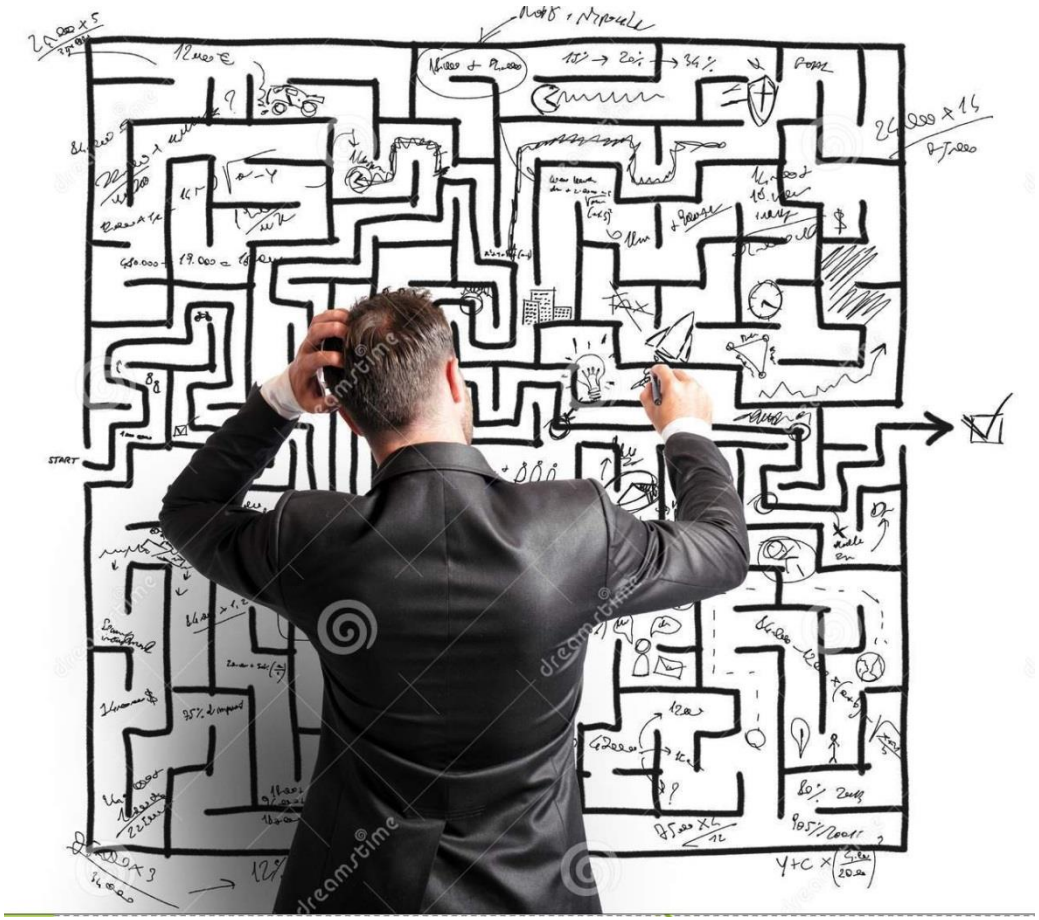
O que os problemas NP-completos têm de especial ?

Problemas NP-completos

- Embora ainda não tenha sido encontrado um algoritmo eficiente para problemas ***NP-completos***, ninguém jamais provou que não é possível existir um algoritmo para esse fim;
- O conjunto de problemas ***NP-completos*** têm a propriedade notável de que, se existe um algoritmo eficiente para qualquer um deles, então existem algoritmos eficientes para todos;
- Vários problemas ***NP-completos*** são semelhantes, mas não são idênticos, a problemas para os quais conhecemos algoritmos eficientes;
- Exemplo: problema do caixeiro viajante;
- Sob certas hipóteses, há algoritmos eficientes que fornecem aproximações razoáveis.



Algoritmos devem produzir saídas corretas!



Corretismo

- Algoritmos corretos usualmente são acompanhados por uma prova formal, o qual explica o porquê o algoritmo gera saídas corretas para todas as instâncias do problema;
- Sem a prova matemática, as vezes, podemos nos enganar a respeito do corretismo de um algoritmo;
- A intuição, muitas vezes, pode nos levar a resultados errôneos.

Corretismo

- Para qualquer algoritmo, devemos provar que ele sempre retorna a saída desejada , para todas as instâncias legais do problema;

- Algoritmos corretos não são sempre óbvios para a maioria dos problemas de otimização.

Corretismo

- Um algoritmo é dito **correto** se, para cada instância de entrada ele pára com a saída correta;
- Dizemos que um algoritmo **correto** resolve o problema computacional dado;
- Um algoritmo **incorreto** pode não parar em algumas instâncias de entrada, ou então pode parar com outra resposta que não a desejada.

Algoritmos incorretos devem ser descartados ?

Algoritmos incorretos

- Ao contrário do que se poderia esperar, às vezes os algoritmos incorretos podem ser úteis, se sua taxa de erros puder ser controlada.
- Exemplo: algoritmos para se localizar grandes números primos.

Corretismo - Ilustração

- Suponha um robot equipado com uma ferramenta de solda para soldar pontos de um placa de circuito.
- O robot deve executar o trabalho de solda em determinados pontos de contato.
- O robot recebe uma quantidade de pontos de contato, devendo visitar o primeiro, o segundo, etc. .. até finalizar o trabalho...

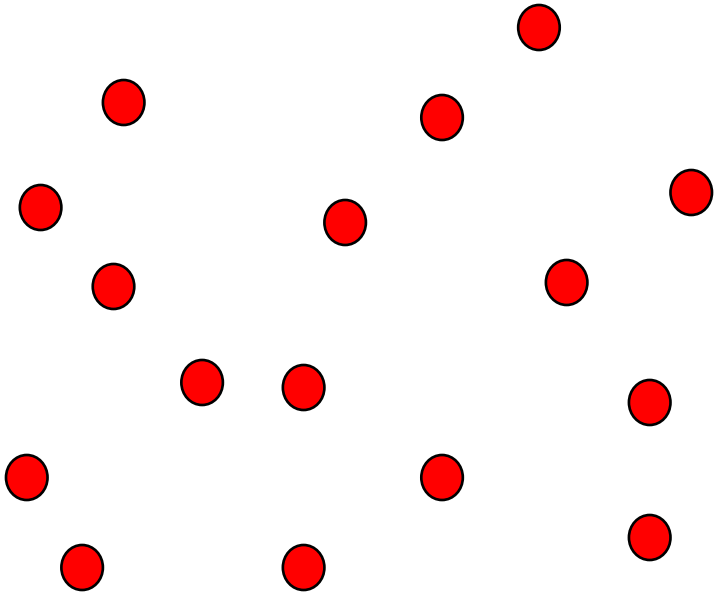
Corretismo - Ilustração

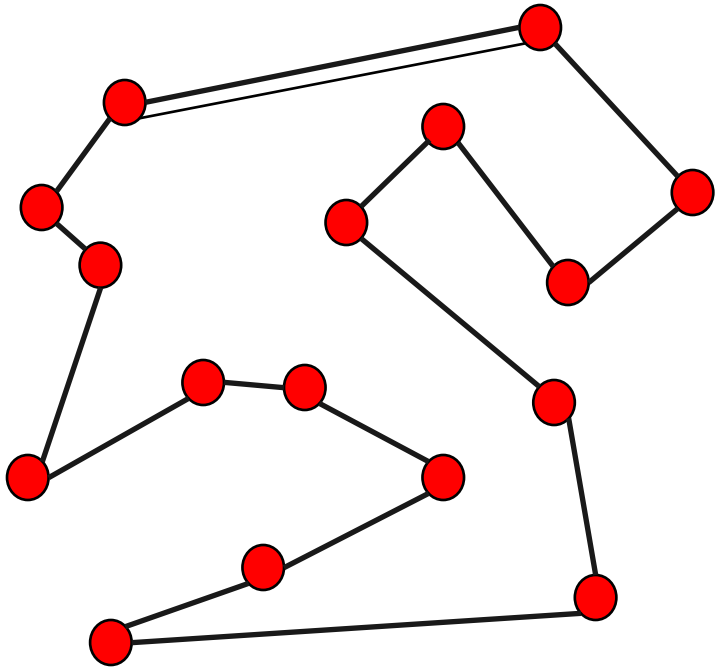
- Robots são **caros** !
- Assim, queremos minimizar o tempo que o braço do robot leva para soldar os pontos de contato...
- Assumimos que o braço do robot se move com velocidade constante.
- Assim, o tempo para processar a placa é proporcional à distância percorrida pelo braço do robot...

Qual é o problema ?

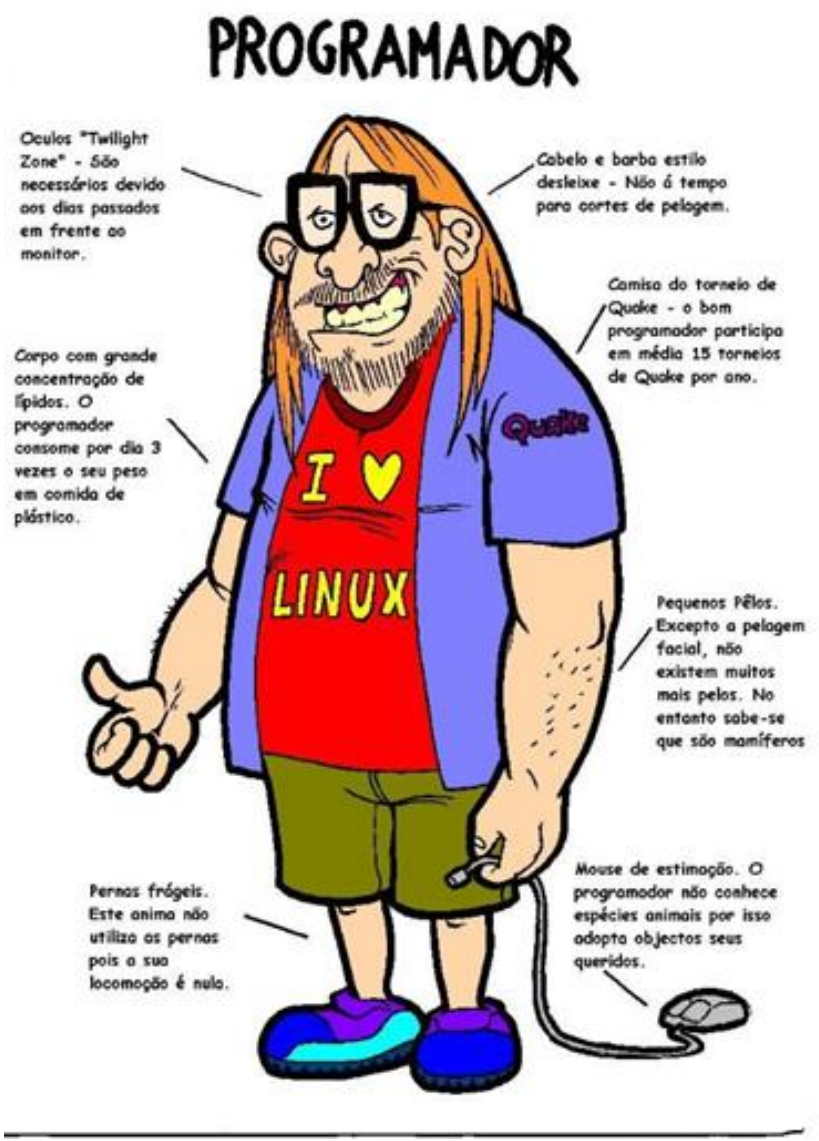
- **INPUT**: Um conjunto S de pontos num plano.
- **OUTPUT**: O caminho mais curto para visitar todos os pontos do ciclo.

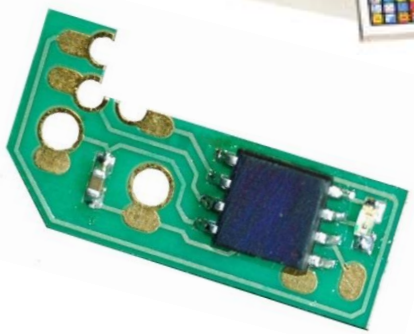
Encontrar o ciclo mais curto para visitar todos os pontos de solda



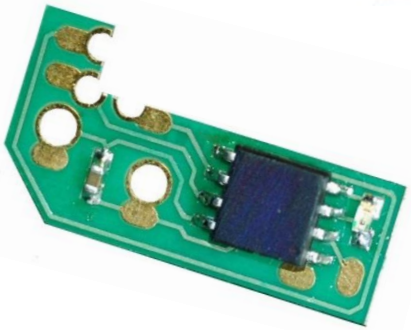


Um programador foi contratado para programar o braço do Robot



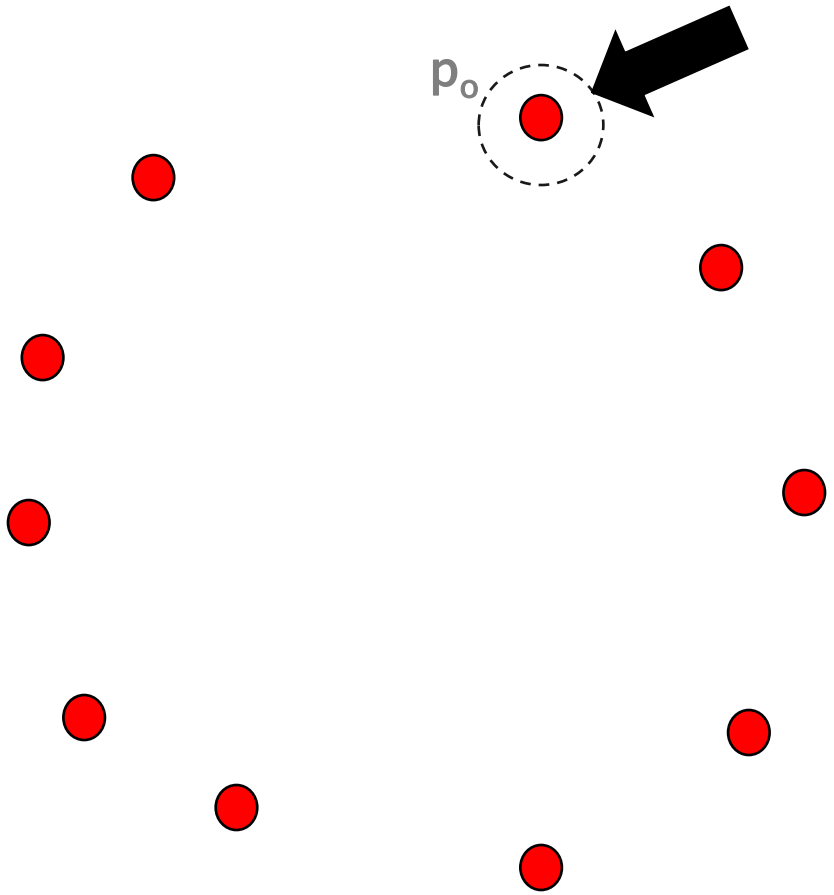


Pense um pouco (5 minutos) em algum algoritmo para programar o braço do Robot ...



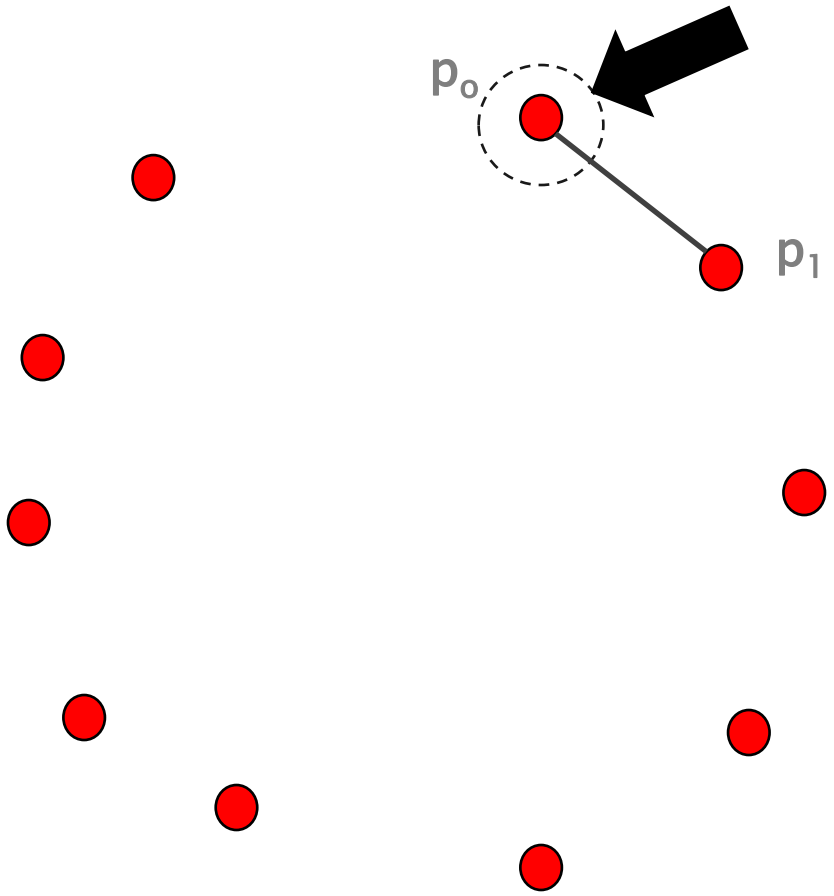
Heurística Nearest-Neighbor

- Inicie em algum ponto p_o e prossiga até o seu vizinho mais próximo p_1.



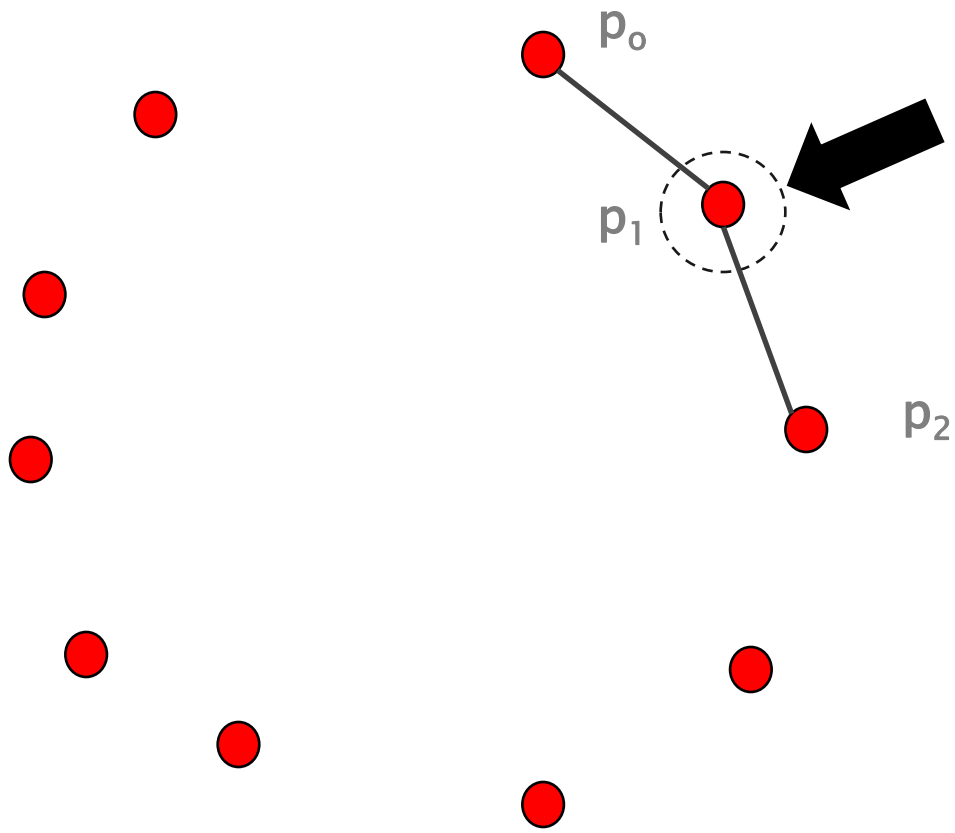
Heurística Nearest-Neighbor

- Inicie em algum ponto p_o e prossiga até o seu vizinho mais próximo p_1.



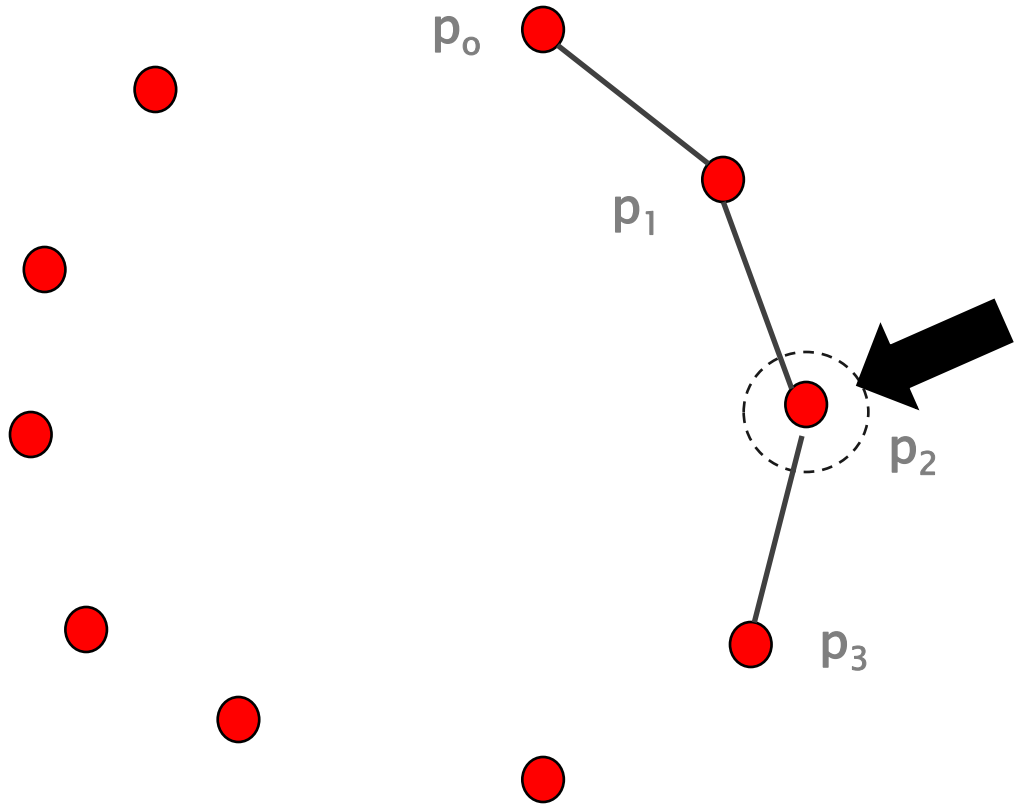
Heurística Nearest-Neighbor

- A partir de p_1, prossiga até seu vizinho mais próximo não-visitado, excluindo p_0 como um candidato.



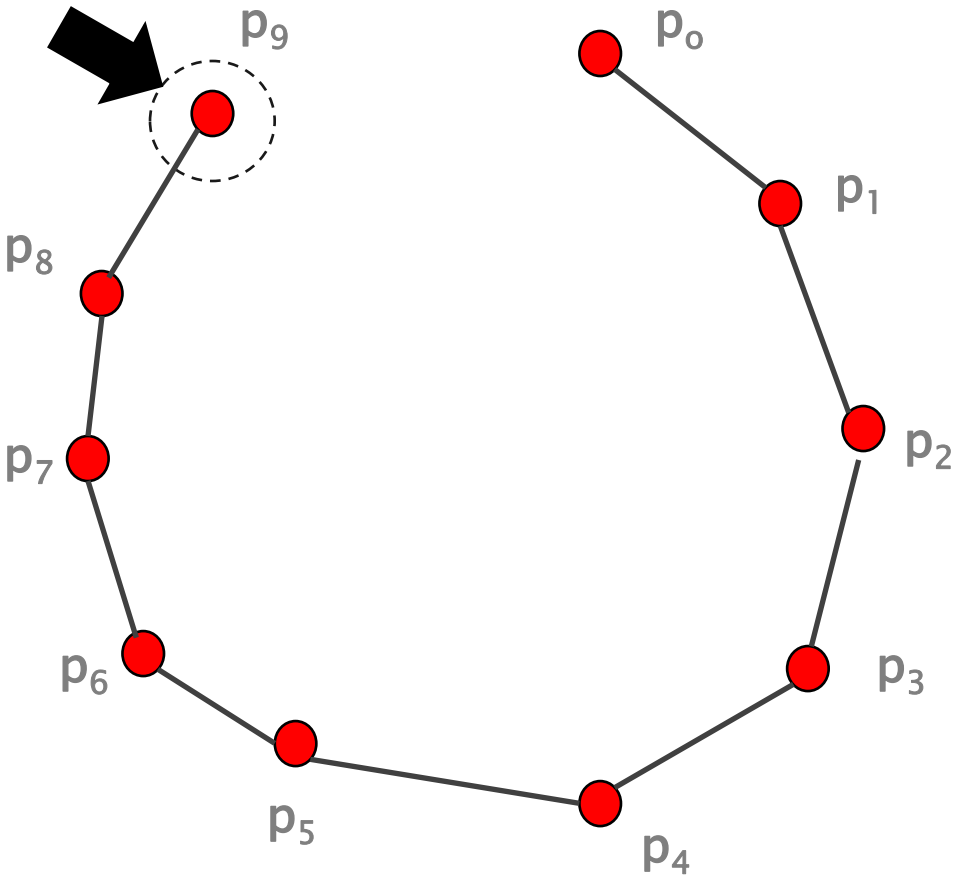
Heurística Nearest-Neighbor

- Repita este processo até passar por todos os pontos não-visitados.



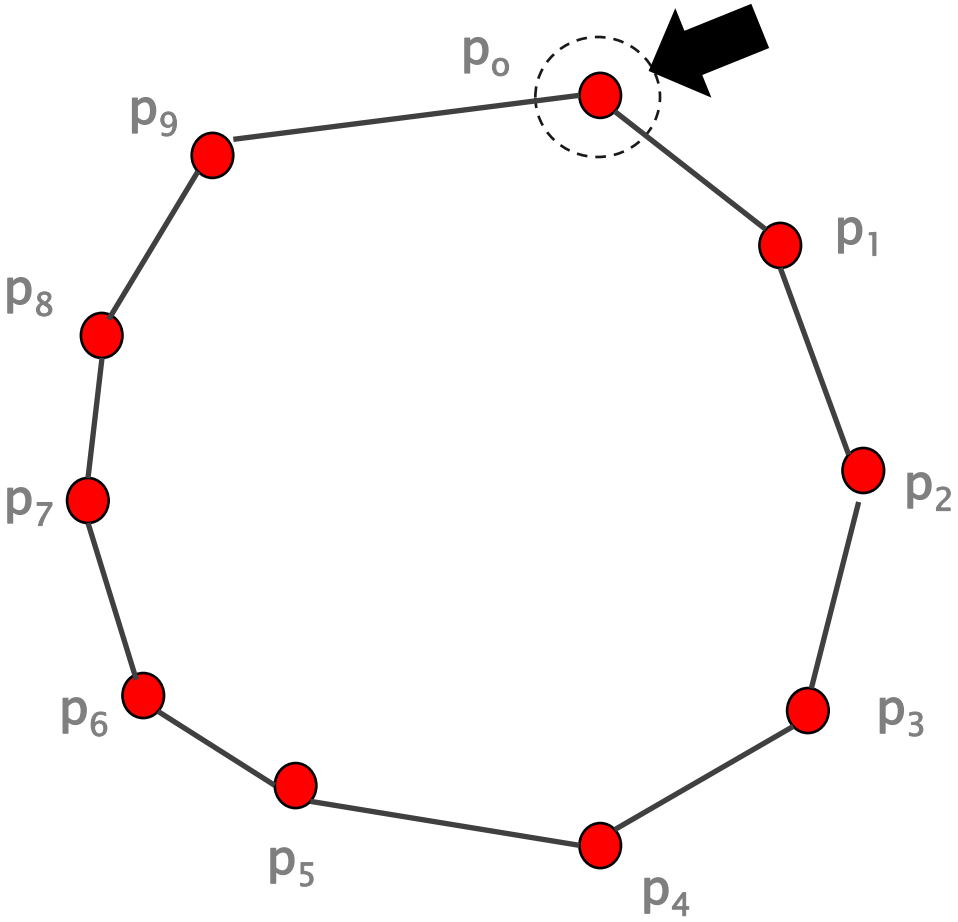
Heurística Nearest-Neighbor

- Repita este processo até passar por todos os pontos não-visitados.



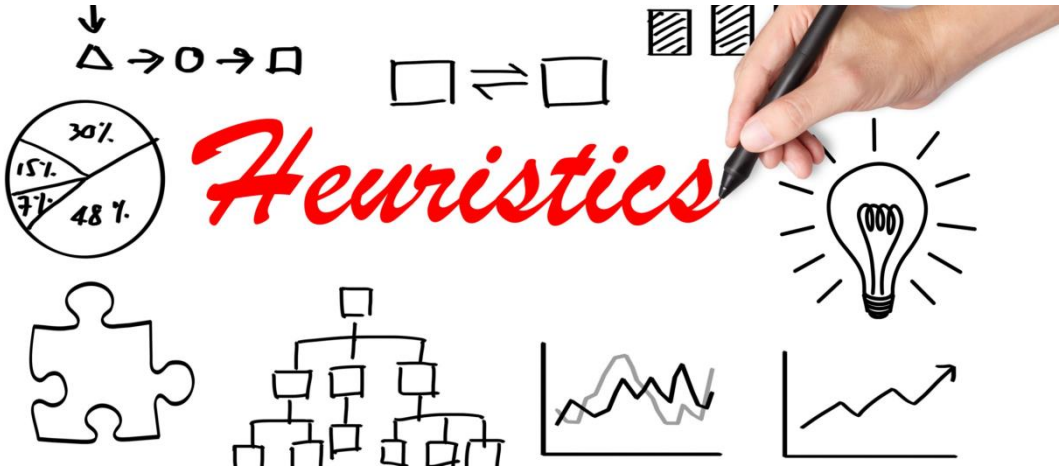
Heurística Nearest-Neighbor

- Ao final retorne a p_o, fechando o ciclo.



Heurística Nearest-Neighbor

- Inicie em algum ponto $\mathbf{p_o}$ e prossiga até o seu vizinho mais próximo p_1.
- A partir de $\mathbf{p_1}$, prossiga até seu vizinho mais próximo não-visitado, excluindo $\mathbf{p_0}$ como um candidato.
- Repita este processo até passar por todos os pontos não-visitados.
- Ao final retorne a $\mathbf{p_o}$, fechando o ciclo.



Heurística Nearest-Neighbor

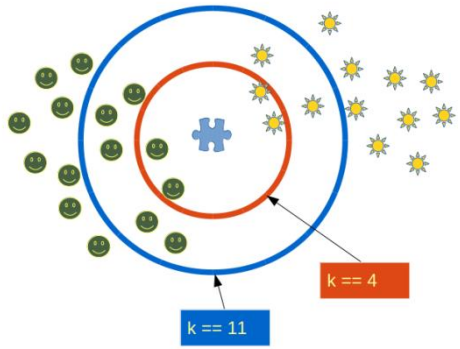
Pick and visit an initial point p_0

$p = p_0$
$i = 0$

While there are still unvisited points
$i = i + 1$
Let p_i be the closest unvisited point to p_{i-1}
Visit p_i

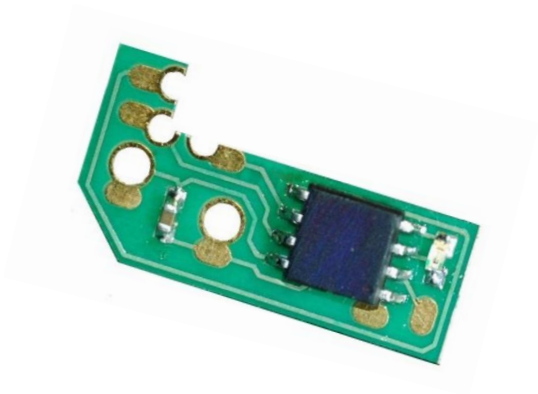
Return to p_0 from p_i

[Steven Skiena]



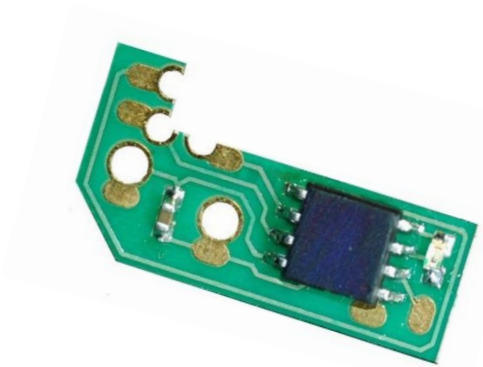
NearestNeighborTSP(P)

O que você achou do algoritmo ?



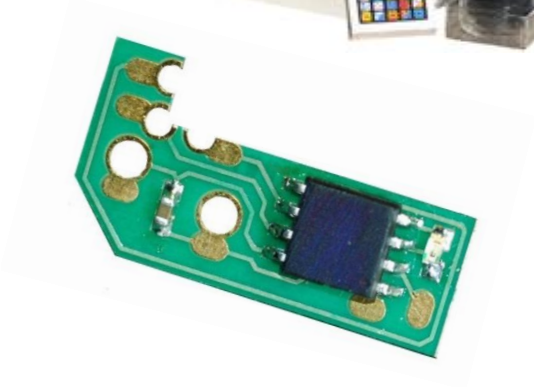
NearestNeighborTSP(P)

- ✓ Simples de se entender;
- ✓ Fácil de ser implementado;
- ✓ Faz sentido visitar pontos mais próximos se queremos minimizar o tempo total do ciclo;
- ✓ Trabalha perfeitamente na figura anterior.



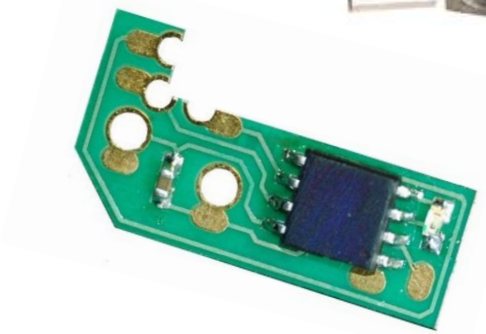
NearestNeighborTSP(P)

No entanto, o algoritmo nem sempre retorna a melhor rota!!!

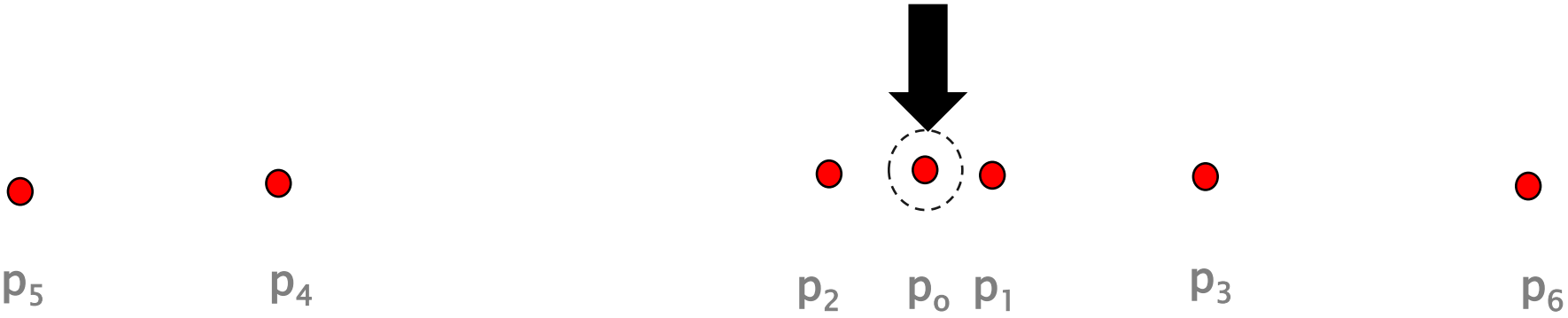


NearestNeighborTSP(P)

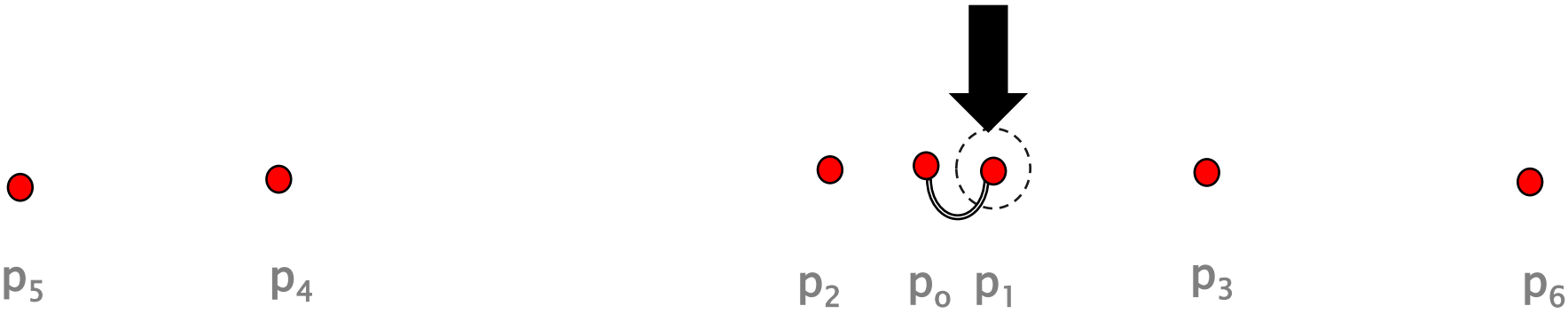
Vamos considerar outra instância do problema ...



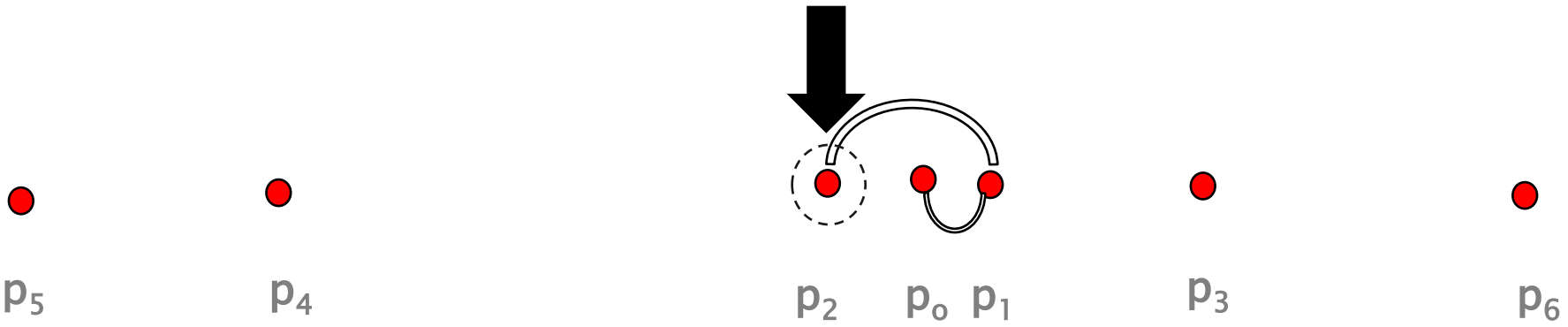
NearestNeighborTSP(P)



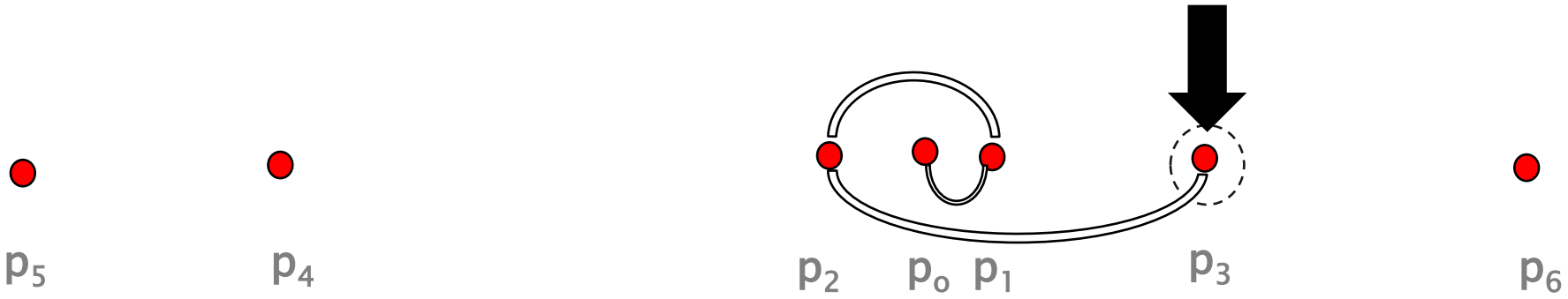
NearestNeighborTSP(P)



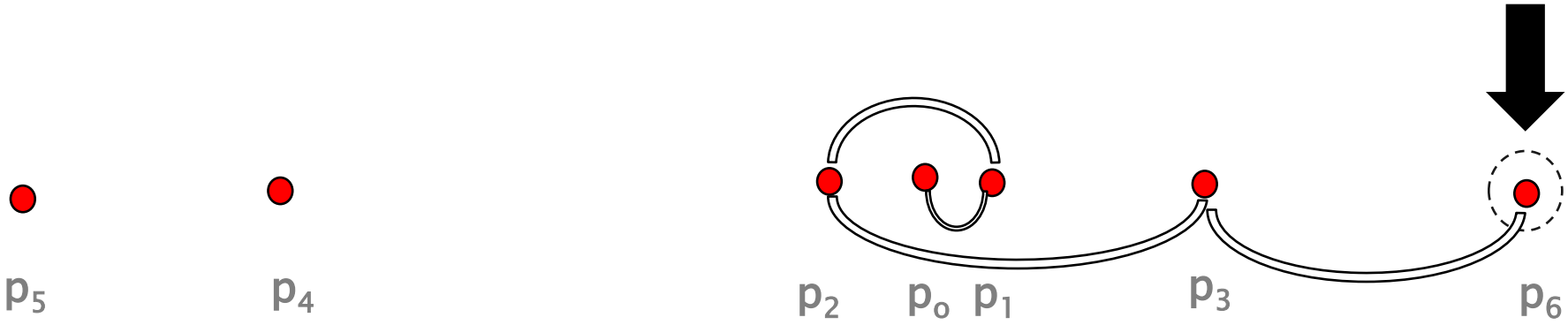
NearestNeighborTSP(P)



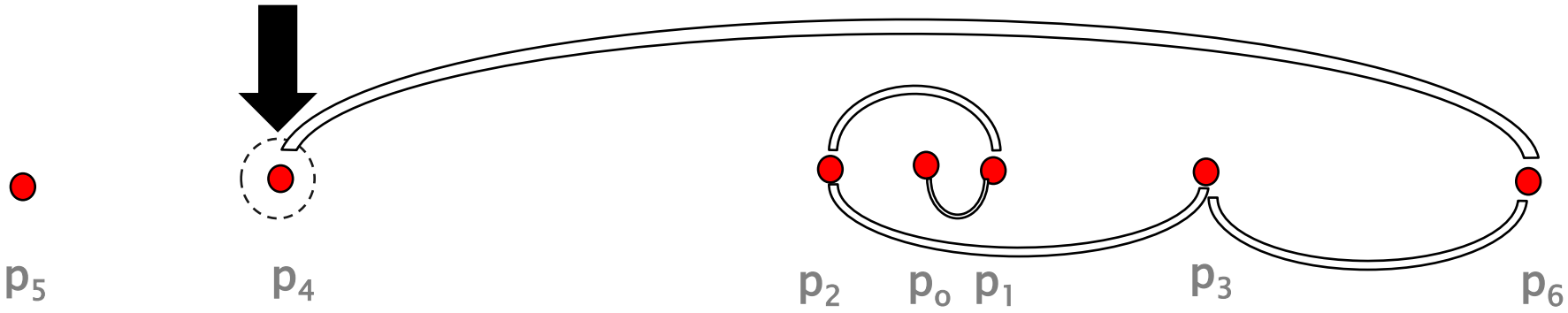
NearestNeighborTSP(P)



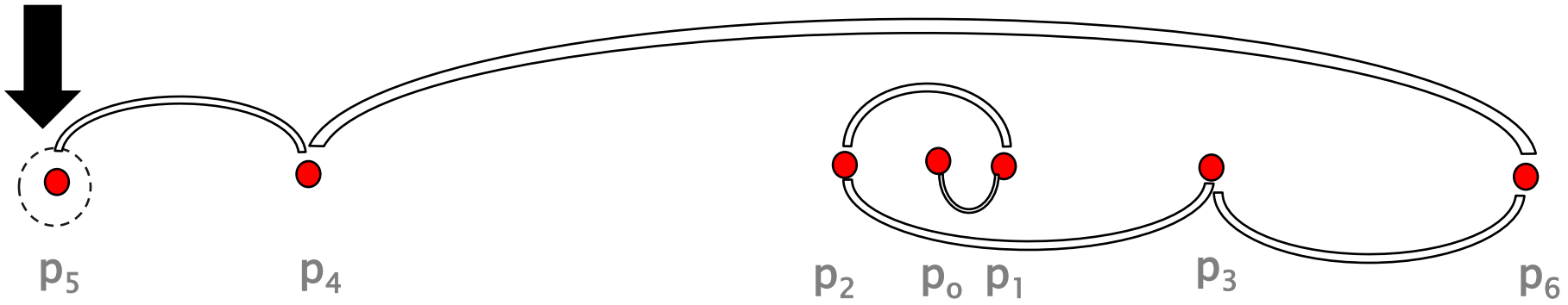
NearestNeighborTSP(P)



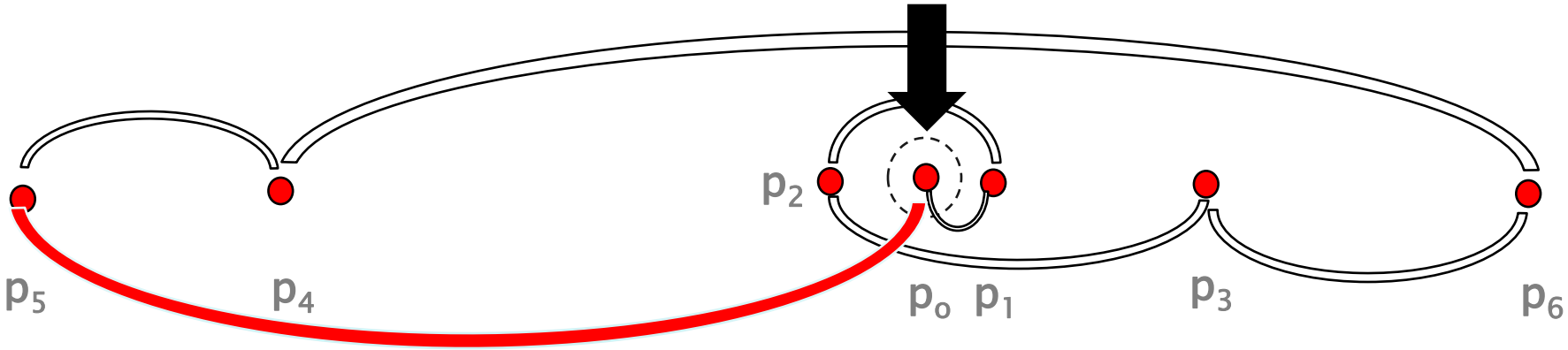
NearestNeighborTSP(P)



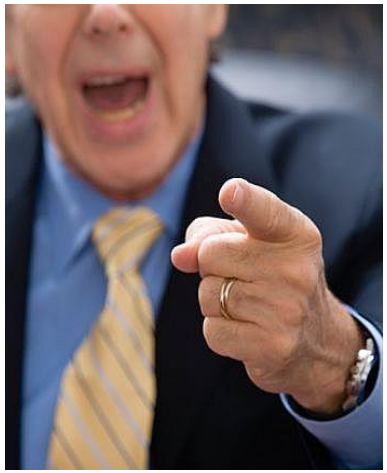
NearestNeighborTSP(P)



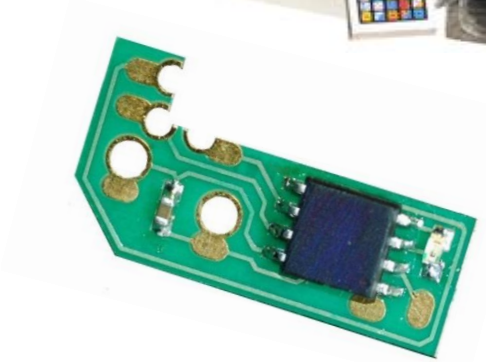
NearestNeighborTSP(P)



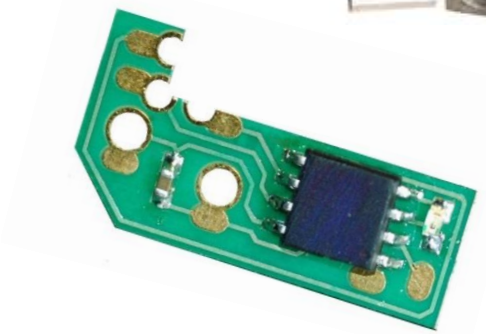
NearestNeighborTSP(P)



Imagine o seu chefe vendo o vai-e-vem do Robot...

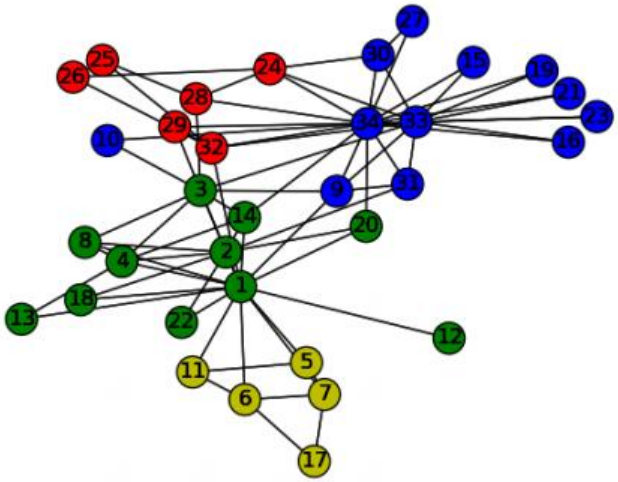


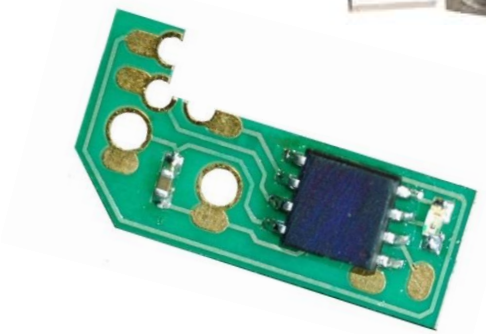
Há algum algoritmo melhor ?



Busca Exaustiva

- ✓Tente todas as possíveis combinações de pontos;
- ✓Selecione dentre estas aquela que tem o menor custo.





Busca Exaustiva

$d = \infty$

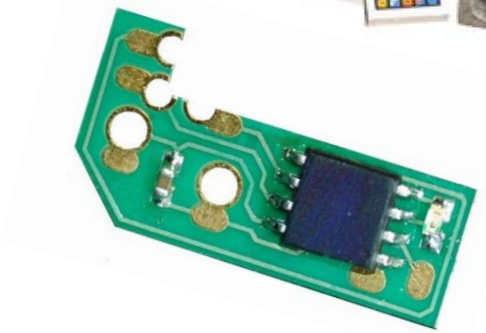
For each of the n! permutations i of the n points
    If (cost(Π_i) $\leq$ d) then

        $d = cost(\Pi_i)$ and $P_{min} = \Pi_i$

Return Pmin

Busca Exaustiva

- Desde que todas as combinações estão sendo consideradas, o algoritmo é correto.

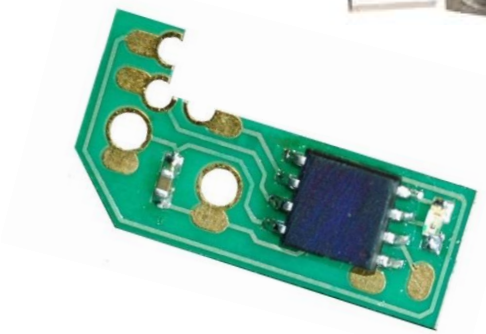


Busca Exaustiva

No entanto ...

Busca Exaustiva

O algoritmo é extremamente lento . . .



Busca Exaustiva

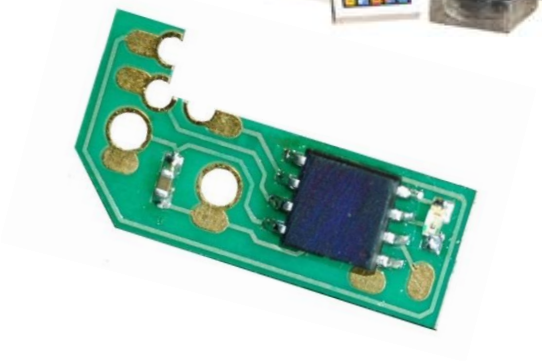
- ✓ Para 20 pontos:
- ✓ 20! = 2.432.902.008.176.640.000 combinações

Busca Exaustiva

- ✓ Para circuitos reais, onde n≈ 1000 pontos
- ✓ Esqueça este algoritmo...

TSP – Travelling Salesman Problem

- ✓ O problema apresentado é clássico na Computação.
- ✓ Denomina-se TSP – Travelling Salesman Problem;
- ✓ Caixeiro Viajante



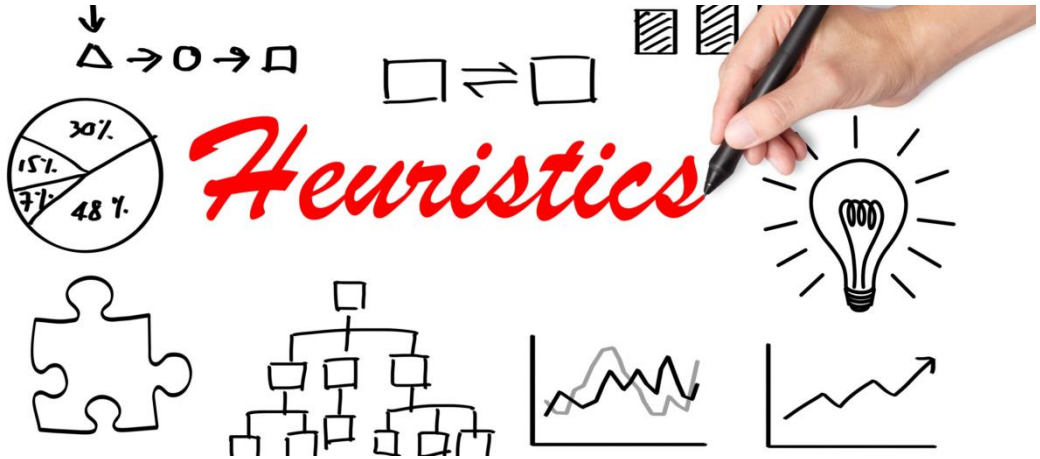
TSP – Travelling Salesman Problem

- ✓ Não existe algoritmo correto e eficiente para este problema.
- ✓ O problema é atacado com **Heurísticas**.

Heurísticas

- ✓ Na Ciência da Computação, busca-se criar algoritmos com tempo de execução aceitável e ser uma solução ótima para o problema em todas as suas instâncias;

- ✓ Um algoritmo heurístico não cumpre uma dessas propriedades, podendo ser ou um algoritmo que encontra boas soluções a maioria das vezes, mas não tem garantias de que sempre as encontrará.



Heurística – Caixeiro Viajante

https://www.youtube.com/watch?v=QTMo_El_tMU