Disciplina de Inteligência Artificial e Robótica

Material com atividades de laboratório

Fabricio Barth

https://insper.github.io/ia_bcc/

Conteúdo

1. Inteligência Artificial e Robótica - 2025/2

1. Ementa da disciplina.
2. Plano aula-a-aula da disciplina.
3. Critérios de avaliação.

1.1 Horário das aulas

Segundas e quartas das 07:30 até 09:30.

1.2 Horário de atendimento

Segundas das 16:30 às 18:00 na sala de reunião 03 no 3o andar do prédio P2.

1.3 Informações adicionais sobre os projetos e implementações

- Todas os projetos e implementações irão utilizar a linguagem de programação **Python**. Sendo assim, espera-se que os alunos desta disciplina tenham conhecimento de programação em Python.
- As entregas dos projetos e exercícios serão via Github Classroom

1.4 Contato

Em caso de dúvida ou comentários, favor encaminhar e-mail para fabriciojb@insper.edu.br.

September 1, 2025

2. Introdução

Aplicações modernas apontam para um conceito de Inteligência Artificial (IA) voltado para duas principais características: **autonomia** e **adaptabilidade**. Autonomia é a habilidade de executar tarefas em contextos complexos sem constante intervenção do ser humano e adaptabilidade é a habilidade de melhorar seu desempenho aprendendo com a experiência.

3. Ementa

Introdução à Inteligência Artificial; Definições de Agente Autônomo; Arquitetura computacional de agentes e o laço percepção – planejamento e ação; Caracterização de Ambientes; Resolução de problemas usando espaço de busca; Estratégias de busca; Algoritmos de busca cega e informados; Conceito de Heurística; Teoria de Jogos e Ambientes Competitivos; Aprendizagem por Reforço; Percepção, sensores e incerteza; Noções de visão computacional e reconhecimento de padrões; Aplicação comercial de robôs e usos emergentes, soluções de plataformas robóticas e de software para robôs (R.O.S, OpenCV).

3.1 Objetivos

Ao final da disciplina o estudante será capaz de:

1. Descrever os conceitos, técnicas e métodos para o desenvolvimento de Agentes Autônomos.
2. Identificar quais tipos de problemas podem ser resolvidos através do uso de Agentes Autônomos.
3. Criar soluções para alguns problemas clássicos desta área.
4. Especificar, desenvolver e testar projetos que façam uso de Agentes Autônomos para resolver problemas complexos.
5. Planejar e executar um trabalho em equipe, fornecendo e assimilando devolutivas.

3.2 Conteúdo Programático

1. Definições de Agente Autônomo e resolução de problemas.
2. Estratégias de busca: algoritmos de busca cega e algoritmos informados.
3. Heurísticas.
4. Implementação de agentes autônomos utilizando estratégias de busca.
5. Programação por restrições (CSP).
6. Ambientes competitivos e teoria de jogos.
7. Algoritmo Min-Max e função de utilidade.
8. Implementação de agentes autônomos para ambientes competitivos.
9. Aprendizagem por Reforço.
10. Implementação de agentes autônomos usando aprendizagem por reforço.
11. Algoritmo Q-Learning.
12. Implementações de agentes autônomos usando o projeto Gym.
13. Implementação de um agente robótico.

3.3 Bibliografia Básica

1. NORVIG, P.; RUSSELL, S., Inteligência Artificial, 3ª ed., Campus Elsevier, 2013

3.4 Bibliografia Complementar

1. O'KANE, J., A Gentle introduction to ROS, CreateSpace Publishing, 2013
2. SIEGWART, R.; NOURBAKHSH, I. R.; SCARAMUZZA, D., Introduction to Autonomous Mobile Robots., 2ª ed., MIT Press, 2011
3. SILVER, D.; SINGH S.; PRECUP D.; SUTTON R. Reward is enough. Artificial Intelligence. Vol 299, 2021. Disponível em https://doi.org/10.1016/j.artint.2021.103535.
4. SILVER, D.; HUBERT T.; SCHRITTWIESER, J.; ANTONOGLOU, I.; LAI, M.; GUEZ, A. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. Science 362, 1140-1144 (2018).

July 23, 2024

4. Plano de aula

O plano de aula desta disciplina está divido em **seis** (6) blocos. Para cada bloco as seguintes atividades estão planejadas.

> **ção!**
>
> O programa está sempre sujeito a alterações e adaptações conforme as disciplina é executada.

4.1 Introdução à Inteligência Artificial

Data	Fundamentos / Conteúdo	Dinâmica
11-Ago	Apresentação da disciplina e critérios de avaliação. Introdução à IA e ao conceito de agente autônomo	Dinâmica em grupo
13-Ago	Revisão do conceito de agente autônomo, discussão sobre diferenças de agente autônomo e software convencional, e propriedades de ambientes.	Dinâmica em grupo

O conteúdo associado a este bloco é 1 e 2

4.2 Busca em Espaços de Estados

Data	Fundamentos / Conteúdo	Dinâmica
18-Ago	Resolução de problemas através de espaço de busca.	Exercícios em sala de aula onde os alunos são convidados a definir estado, transição, estado meta e custo da solução encontrada para diversos problemas.
20-Ago	Estratégias de busca. Algoritmos de busca cegos (largura, profundidade, iterativo, custo uniforme). Critérios de comparação entre os algoritmos.	Implementação dos algoritmos de busca e dos agentes autônomos em Python para resolver alguns problemas clássicos da literatura.
01-Set	Estratégias de busca. Algoritmos de busca cegos (largura, profundidade, iterativo, custo uniforme). Critérios de comparação entre os algoritmos.	Implementação dos algoritmos de busca e dos agentes autônomos em Python para resolver alguns problemas clássicos da literatura.
03-Set	Estratégias de busca. Algoritmos de busca cegos (largura, profundidade, iterativo, custo uniforme). Critérios de comparação entre os algoritmos.	Implementação dos algoritmos de busca e dos agentes autônomos em Python para resolver alguns problemas clássicos da literatura.

4.3 Busca Heurística

Data	Fundamentos / Conteúdo	Dinâmica
04-Set	Estratégia de busca. Algoritmos de busca **informados** (busca gananciosa, A^{*}, família subida da montanha). Função heurística. Comparação entre os algoritmos.	Implementação dos algoritmos de busca e dos agentes autônomos em Python para resolver alguns problemas clássicos da literatura.
08-Set	Estratégia de busca. Algoritmos de busca **informados** (busca gananciosa, A^{*}, família subida da montanha). Função heurística. Comparação entre os algoritmos.	Implementação dos algoritmos de busca e dos agentes autônomos em Python para resolver alguns problemas clássicos da literatura.
10-Set	Estratégia de busca. Algoritmos de busca **informados** (busca gananciosa, A^{*}, família subida da montanha). Função heurística. Comparação entre os algoritmos.	Implementação dos algoritmos de busca e dos agentes autônomos em Python para resolver alguns problemas clássicos da literatura.
11-Set	Desenvolvimento de um agente autônomo que atua em um ambiente discreto, determinístico, síncrono, simulado e *single agent*.	Implementação de um projeto, provavelmente, envolvendo algum framework de simulação (i.e., Gym Open AI).

4.4 Ambientes Competitivos

Data	Fundamentos / Conteúdo	Dinâmica
15-Set	Constraint Satisfaction Problems	Implementação de um agente autônomo capaz de identificar estados que satisfazem determinadas restrições.
17-Set	Jogos de tabuleiro, busca competitiva, algoritmo min-max e função de utilidade.	Implementação de um agente autônomo que deverá atuar em um ambiente competitivo, determinístico e completamente observável.
22-Set	**SEMANA DE PROVAS**	**SEMANA DE PROVAS - Prova Intermediária**
24-Set	**SEMANA DE PROVAS**	**SEMANA DE PROVAS - Prova Intermediária**

4.5 Projeto Intermediário

Data	Fundamentos / Conteúdo	Dinâmica
29-Set	Desenvolvimento de um agente autônomo que atua em um ambiente discreto, determinístico, síncrono, simulado e *single agent* ou *multi-agent*.	Desenvolvimento de projeto em sala de aula
01-Out	Desenvolvimento de um agente autônomo que atua em um ambiente discreto, determinístico, síncrono, simulado e *single agent* ou *multi-agent*.	Desenvolvimento de projeto em sala de aula
06-Out	Desenvolvimento de um agente autônomo que atua em um ambiente discreto, determinístico, síncrono, simulado e *single agent* ou *multi-agent*.	Desenvolvimento de projeto em sala de aula

4.6 Aprendendo políticas

Data	Fundamentos / Conteúdo	Dinâmica
08-Out	Definição de aprendizagem por reforço, política de controle e algoritmo Q-Learning.	Discussão em sala. Exercícios em sala de aula envolvendo o ambiente OpenAI Gym. Implementação de agentes autônomos usando o algoritmo Q-Learning.
13-Out	Algoritmo Q-Learning: detalhes e hiperparâmetros. Apresentação do ambiente OpenAI Gym.	Exercícios em sala de aula envolvendo o ambiente OpenAI Gym. Implementação de agentes autônomos usando o algoritmo Q-Learning.
15-Out	Algoritmo Q-Learning: detalhes e hiperparâmetros.	Implementação de agentes autônomos usando o algoritmo Sarsa.
20-Out	Ambientes não-determinísticos. Reinforcement Learning: métodos tabulares	Implementação de agentes autônomos usando o algoritmo Q-Learning e Sarsa
22-Out	Ambientes não-determinísticos. Reinforcement Learning: métodos tabulares	Implementação de agentes autônomos usando o algoritmo Q-Learning e Sarsa

4.7 Robótica

Data	Fundamentos / Conteúdo	Dinâmica
27-Out	Visão geral sobre robótica e framework ROS2	Visão geral sobre robótica e framework ROS2
29-Out	Desenvolvimento de um agente robótico (físico).	Implementação de um projeto envolvendo um robô físico
3-Nov	Desenvolvimento de um agente robótico (físico).	Implementação de um projeto envolvendo um robô físico
5-Nov	Desenvolvimento de um agente robótico (físico).	Implementação de um projeto envolvendo um robô físico
10-Nov	**Avaliação Final da disciplina**	**Avaliação Final da disciplina**
12-Nov	**Avaliação Final da disciplina**	**Avaliação Final da disciplina**

4.8 Reposições de aulas

Neste semestre não teremos aulas nos dias 25/8 e 27/8.

Estas aulas serão repostas nos dias 4/9 e 11/9, respectivamente. Quinta-feira das 14:15 às 16:15.

August 11, 2025

5. Atividades e Critérios de aprovação

Os objetivos de aprendizagem desta disciplina serão avaliados através das seguintes atividades:

Atividade	Peso
N exercícios (APS) sobre agentes autônomos	10%
2 projetos	30%
Avaliação Intermediária	30%
Avaliação Final	30%

O critério para aprovação é:

- nota final superior ou igual a cinco (5);
- a média das avaliações intermediária e final deve ser igual ou maior que cinco (5);
- e 75% de frequência mínima nas aulas.

5.1 *N* exercícios sobre agentes autônomos

Seguem os enunciados que se encaixam nesta categoria:

Descrição	Prazo para entrega
Discussão sobre conceitos gerais sobre IA	13/08

5.2 2 projetos

Seguem os enunciados que se encaixam nesta categoria:

Descrição	Prazo para entrega
TBD	TBD

5.3 Conversão de conceito para valor numérico

O resultado de algumas avaliações poderá adotar conceitos (A+, B,..., I) ao invés de um valor numérico. Para estes casos será utilizada a seguinte tabela de conversão:

A+	A	B	C	D	I
10	9	7	5	4	2

- **Insuficiente (I)**: não entregou, entregou algo que não exigiu esforço ou desviou do objetivo.
- **Em Desenvolvimento (D)**: entregou algo que estava de acordo com o objetivo e exigiu algum esforço, mas possui problemas e limitações importantes. O mínimo aceitável não foi atingido.
- **Básico (C)**: o mínimo aceitável do objetivo medido foi alcançado, mas o desempenho foi abaixo do esperado. É suficiente para aprovação.
- **Esperado (B)**: atingiu o desempenho esperado.
- **Avançado (A)**: o desempenho foi acima do esperado. É algo desejável, mas não é motivo de preocupação se não for alcançado.

August 5, 2025

6. Aulas

6.1 Introdução

6.1.1 Introdução à Inteligência Artificial

Objetivos e estrutura da disciplina

- Qual é o escopo desta disciplina? Ver ementa.
- Como a disciplina está organizada? Ver plano de aulas.
- Quais são os critérios de avaliação? Ver avaliação.

Introdução

O objetivo desta aula é responder as seguintes perguntas:

- O que é inteligência artificial (IA) geral ou profunda?
- O que é inteligência artificial fraca?
- Quais são as principais aplicações de IA?
- Qual é a definição de agente autônomo?
- Qual é a relação de agente autônomo com IA?
- Quais as funcionalidades que um agente autônomo precisa ter e como podemos desenvolver tais funcionalidades?
- O que é IA Generativa?
- Quais são as outras disciplinas do curso que estão relacionadas com este tópico?

Atividade

Forme grupos de 3 a 4 pessoas e discuta as perguntas acima. Elabore uma apresentação para a próxima aula que responda as perguntas listadas acima. Para responder as perguntas, você pode usar os materiais de referência listados abaixo, além de outros materiais que você encontrar. Você também poderá usar sistemas de IA Generativa, como o ChatGPT, para ajudar a responder as perguntas. A apresentação deve ter no máximo 10 minutos.

Material de referência

Material com exemplos de ficção científica:

Material com exemplos de aplicações reais:

Uma apresentação sobre a evolução da IA:

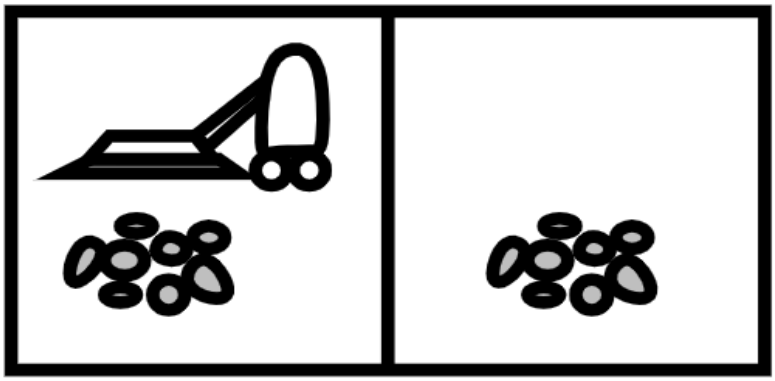
August 5, 2025

6.1.2 Agentes Autônomos

Exemplo do aspirador de pó

Um robô aspirador de pó deve limpar uma casa com duas posições. As operações que ele sabe executar são:

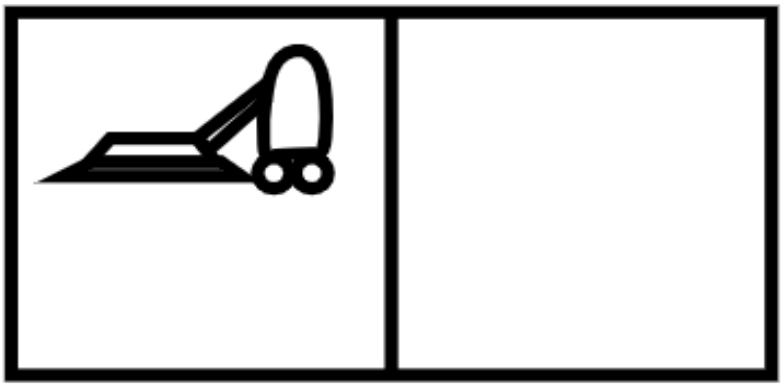
- sugar
- ir para a posição da esquerda
- ir para a posição da direita



Em **grupo** de 4 pessoas responda as seguintes perguntas (tempo de 15 minutos):

- O que é relevante representar nos estados do mundo? Como os estados são estruturados (estrutura de dados) e qual o significado dela (dos campos)?
- Quais são os estados possíveis do mundo do aspirador e as suas transições?
- Quais são as consequências das ações sobre os estados? As ações são determinísticas?
- É possível desenhar o **grafo** com todos os estados e transições para este problema?
- Apresente uma solução possível. Uma sequência de ações que fazem o robô sair do estado inicial e chegar no estado final.

Considere como estado final a situação ilustrada abaixo:



Prepararem-se para eventualmente apresentar as suas respostas.

Conceitos que devem ser explorados neste exercício

Representação de um problema na forma de **estado**, **transição**, **grafo**

February 10, 2023

6.1.3 Implementando um Aspirador de Pó

Importante!

As orientações sobre a entrega deste exercício estão abaixo. Este é um exercício individual. Os alunos poderão discutir sobre o problema, mas a implementação deve ser feita individualmente. Não pode ser utilizado nenhum tipo de ferramenta de Inteligência Artificial para a resolução destes problemas.

Configuração do ambiente

Para executar esta atividade você terá que criar um projeto e como dependência deste projeto instalar a biblioteca AIGYM:

```
pip install aigyminsper
```

Esta biblioteca implementa diversos algoritmos de busca que serão utilizados ao longo da disciplina. Além disso, esta biblioteca permite o desenvolvimento de agentes inteligentes através de uma interface padrão.

Exercício: Aspirador de Pó

Para este exercício vamos utilizar o arquivo AgentSpecification.py.

- Faça uma cópia deste arquivo e chame ele de `AspiradorPo.py`, por exemplo.
- No método `__init__` defina a estrutura de dados que você especificou na aula passada.
- No método `successors` codifique a execução das ações segundo o que você especificou na aula passada. **Dica**: cada novo estado é uma instância de um novo objeto da classe que é adicionado na lista retornada pelo método.
- Codifique o comportamento do método `is_goal`. Ele deve retornar `True` quando encontrar um estado igual a meta.
- O médoto `description()` retorna uma descrição do problema que está sendo resolvido. Por favor, altere o que está descrito no método.
- Faça um retorno para o método `env()`. Peça para o professor explicar em sala de aula.
- Altere o método `main()` para que o mesmo chame a sua classe.

Conceitos que devem ser explorados neste exercício

Representação de um problema na forma de **estado**, **transição**, **grafo**

Conceitos que devem ser explorados neste exercício

Algoritmos de Busca, começando com o algoritmo de busca em largura.

Exercício 2: Aspirador de Pó com 4 quartos.

Vamos implementar um aspirador de pó que atua em um ambiente com 4 quartos? Um quadrado (2×2)?

Mas, antes de implementar usando uma estrutura similar a descrita acima, responda as questões abaixo:

- O que é relevante representar nos estados do mundo? Como os estados são estruturados (estrutura de dados) e qual o significado dela (dos campos)?
- Mostre como ficam representados os estados inicial e final segundo a representação adotada.
- Quais as operações sobre os estados? (detalhe como cada operação irá alterar os estados e quais as condições para cada operação ser executada)
- Qual a estimativa do tamanho do espaço de busca (número de estados possíveis)?

Entrega

A entrega deste exercício deve ser feita no Github Classroom: https://classroom.github.com/a/cRCLHC5C. A entrega é individual e deve ser feita até o meio dia do dia 20/08 até às 10 horas da manhã. O projeto base existente no Github classroom tem um arquivo de teste. Você deve implementar o código de forma que todos os testes passem.

Exercício adicional: Aspirador de Pó com poltrona.

Se você já terminou o exercício acima então você pode implementar uma solução para esta situação adicional. Este exercício é opcional e não será considerado para a nota.

Neste problema os quartos da casa podem possuir poltronas. Para limpar o quarto da casa que tem poltrona o agente deve antes de executar a ação limpar `virar` a poltrona. Claro que depois de limpar o agente deve também `desvirar` a poltrona para deixar o quarto em ordem.

Para este problema considere um ambiente (casa) também com 4 quartos (um quadrado 2×2).

Mas, antes de implementar usando uma estrutura similar a descrita acima, responda as questões abaixo:

- O que é relevante representar nos estados do mundo? Como os estados são estruturados (estrutura de dados) e qual o significado dela (dos campos)?
- Mostre como ficam representados os estados inicial e final segundo a representação adotada.
- Quais as operações sobre os estados? (detalhe como cada operação irá alterar os estados e quais as condições para cada operação ser executada)
- Qual a estimativa do tamanho do espaço de busca (número de estados possíveis)?

August 17, 2025

6.2 Busca em espaço de estados

6.2.1 Alterando os algoritmos de busca

Nas aulas anteriores implementamos algumas versões do problema do aspirador de pó. Para a atividade de hoje vamos considerar a implementação mais simples, que é a que considera um aspirador de pó em um ambiente com 2 quartos.

Completando o método `env`

Nesta implementação você vai adicionar o seguinte comportamento no médoto `env()`:

```
def env(self):
    return json.dumps(self.__dict__)
```

Alterando a chamada do método `search`

Na hora de chamar o método `search` altere o código de:

```
    algorithm = BuscaLargura()
    result = algorithm.search(state)
```

para:

```
    algorithm = BuscaLargura()
    result = algorithm.search(state, trace=True)
```

adicionando o parâmetro `trace=True`. Tente entender o que acontece: (i) qual o resultado encontrado, (ii) qual o conteúdo do log impresso.

Alterando o algoritmo de busca

Depois disto, altere o código de:

```
    algorithm = BuscaLargura()
    result = algorithm.search(state, trace=True)
```

para:

```
    algorithm = BuscaProfundidade()
    result = algorithm.search(state, trace=True, m=10)
```

Tente entender o que acontece: (i) qual o resultado encontrado, (ii) qual o conteúdo do log impresso. Por que os resultados são diferentes?

Objetivo desta aula

O objetivo desta aula é introduzir o conceito de algoritmo de busca, mais especificamente, usando os algoritmo de busca em largura e o algortimo de busca em profundidade.

Material de referência

February 11, 2025

6.2.2 Algoritmos de Busca em Largura, Profundidade e Profundidade Iterativa

Nas aulas anteriores implementamos um agente aspirador de pó. Na última aula executamos os algoritmos de Busca em Largura e Profundidade e discutimos alguns conceitos relacionados a esses algoritmos. Nesta aula vamos explicar com mais detalhes e de forma mais estruturada como esses algoritmos funcionam.

Objetivo desta aula

O objetivo desta aula é entender como os algoritmos de busca em largura, profundidade e profundidade iterativa funcionam.

Ao final desta aula...

Ao final desta aula você deverá saber a diferença entre os algoritmos de busca:

- em largura;
- em profundidade, e;
- em profundidade iterativa.

Além disso...

Além disso você também deverá saber como avaliar um algoritmo de busca, quais indicadores utilizar, e, consequentemente, saber como aplicá-los nos problemas apresentados.

Material utilizado

O material utilizado para esta aula está nos slides 17 até 28 do conjunto de slides abaixo:

Atividade de laboratório

Um robô elevador está programado para transportar cargas dentro de um armazém vertical. Ele começa no térreo (andar 0) e pode realizar duas operações para se mover entre os andares:

- Subir 1 andar: o elevador move-se do andar $A = A+1$.
- Subir 2 andares: o elevador move-se do andar $A = A+2$.

O objetivo do robô é alcançar um andar específico A_{f}, definido pelo usuário.

Implemente um agente chamado `Elevador` que resolve este problema utilizando os algoritmos de busca em largura, busca em profundidade e busca em profundidade iterativa.

Este robô sabe executar duas ações: "subir 1 andar" e "subir 2 andares". Na sua implementação você deve usar exatamente este nome para as ações. No método `successors` você deve primeiro adicionar a ação "subir 2 andares" e depois a ação "subir 1 andar".

Este exercício é composto por algumas partes: implementando o agente com os algoritmos e avaliando o comportamento do agente.

IMPLEMENTANDO O AGENTE

Você deve implementar um arquivo `Elevador.py` com uma função `main(inicio, objetivo, algoritmo)` que recebe 3 parâmetros: o andar de início, o andar objetivo e o algoritmo que será utilizado. O parâmetro algoritmo aceita três valores: BL, BP, BPI. Estes valores representam os algoritmos de busca em largura, busca em profundidade e busca em profundidade iterativa, respectivamente. A sua implementação deve passar por todos os testes do arquivo `test_elevador.py`.

Esta primeira parte é pré-requisito da segunda parte. Se na entrega esta parte não estiver implementada, a segunda parte não será avaliada.

AVALIANDO O COMPORTAMENTO DO AGENTE

Utilize este código para testar os três algoritmos vistos até o momento (busca em largura, busca em profundidade e busca em profundiade iterativo) variando o estado final de \(1\) até \(50\). No caso do algoritmo em profundidade, teste duas versões (\(m = 10\) e \(m = 100\)).

Armazene o tempo de processamento criando uma tabela similar a esta:

Algoritmo	**Objetivo**	**Tempo de processamento em segundos**
Busca em Largura	1	0.000036
Busca em Largua	\(\cdots\)	\(\cdots\)
Busca em Largura	10	0.000378
Busca em Largura	50	\(\cdots\)
Busca em Profundidade com \(m= 10\)	1	0.000033
Busca em Produndidade com \(m= 10\)	\(\cdots\)	\(\cdots\)
Busca em Produndidade com \(m= 100\)	\(\cdots\)	\(\cdots\)
Busca em Profundidade Iterativa	1	\(\cdots\)
Busca em Profundidade Iterativa	\(\cdots\)	\(\cdots\)
Busca em Profundidade Iterativa	50	\(\cdots\)

Em alguns casos a combinação do algoritmo com o objetivo não fornece um resultado. Você deve informar na tabela estes casos. Nestas situações você deve colocar o tempo de processamento como NaN, ou seja, valor faltante. De forma alguma você pode colocar o valor zero nestas situações.

Este tipo de medida pode variar de máquina para máquina. Portanto, não se preocupe em comparar os seus resultados com os resultados dos seus colegas. O importante é que você consiga perceber a diferença entre os algoritmos.

Além disso, na mesma máquina, o tempo de processamento pode variar de uma execução para outra. Portanto, para cada combinação de algoritmo e objetivo, execute o teste 5 vezes e utilize a média dos tempos como o tempo final.

Utilize os conhecimentos adquiridos na disciplina de Ciência de Dados do semestre passado e faça um *plot* destes dados em um único gráfico.

Em um documento, coloque a tabela, o gráfico e responda as seguintes perguntas:

- Segundo o que discutimos em sala de aula, quais destes algoritmos são **ótimos**? Os resultado encontrados neste exercício são coerentes com está informação? Justifique a sua resposta.
- Segundo o que discutimos em sala de aula, quais destes algoritmos são **completos**? Os resultado encontrados neste exercício são coerentes com está informação? Justifique a sua resposta.
- Teve algum algoritmo que travou por falta de memória no seu computador? Se sim, qual é a explicação?

ENTREGA

Esta atividade deve ser feita em duplas e deve ser submetida via Github Classroom. O link para a atividade é https://classroom.github.com/a/VDgTzBon. O prazo para entrega é **04/09/2025**.

RUBRICA DE AVALIAÇÃO

Conceito	Descrição
A+	Submeteu um documento que apresenta a tabela completa, responde todas as perguntas de forma correta e apresenta um único *plot* que sumariza todos os dados de forma correta e objetiva.
A	Submeteu um documento que apresenta a tabela completa, responde todas as perguntas de forma correta e apresenta o *plot*, mas este plot poderia ser melhor feito.
C	Submeteu um documento que apresenta a tabela, mas não responde todas as perguntas de forma correta ou não apresenta o *plot*.
D	A implementação passou por todos os testes.
I	A implementação não passou por todos os testes.

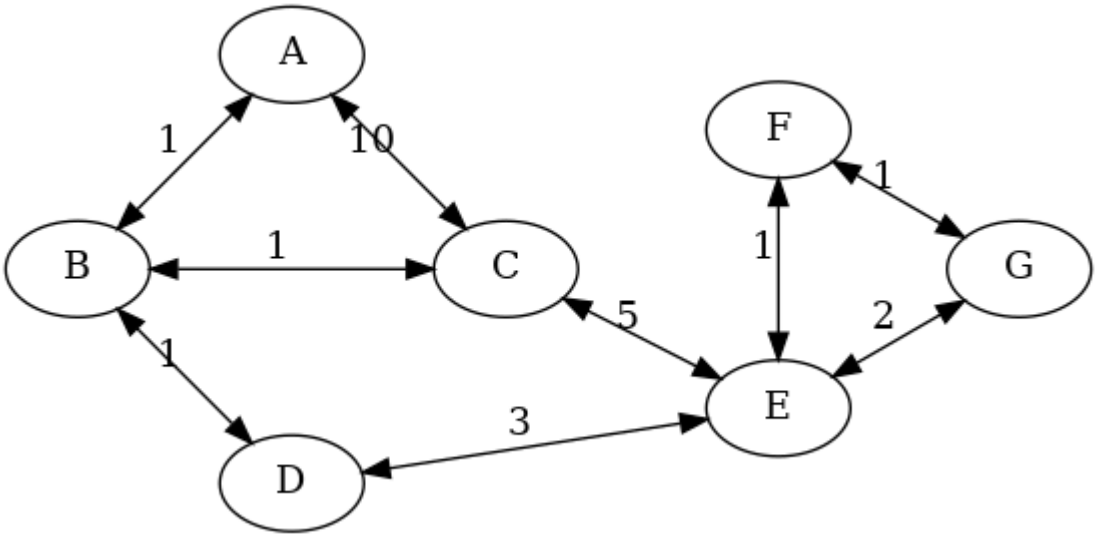
September 2, 2025

6.2.3 Algoritmo de custo uniforme

E se as ações têm custos diferentes? Será que podemos usar algoritmos de busca em largura, profundidade ou profundidade iterativa?

Procurando caminhos em um mapa

Considere o mapa abaixo:



Os \(nodos = \{A,B,C,D,E,F,G\}\) são locais. As arestas são as ligações entre os locais. Cada aresta tem um custo. Por exemplo, O caminho entre A e B tem custo 1. O caminho entre A e C tem custo 10. Este é um grafo bi-direcional. Ou seja, O custo de A para C é o mesmo de C para A.

Implemente uma solução que é capaz de encontrar o menor caminho entre qualquer par de locais do mapa.

Crie um repositório a partir do github classroom. O link é: https://classroom.github.com/a/qU_2zIpW. Neste repositório já existem alguns arquivos. Leia estes arquivos, inclusive o de teste, para entender o que é esperado.

Quando você estiver com o código pronto, faça o push para o repositório.

Esta é uma atividade individual. Deve ser entregue até o dia 06/09/2025 às 23:30.

Referências

O material utilizado para esta aula está nos slides 30 até 32 do conjunto de slides abaixo:

September 2, 2025

7. Referências

7.1 Introdução à Inteligência Artificial

January 30, 2024

7.2 Algoritmos de Busca

February 10, 2023

None