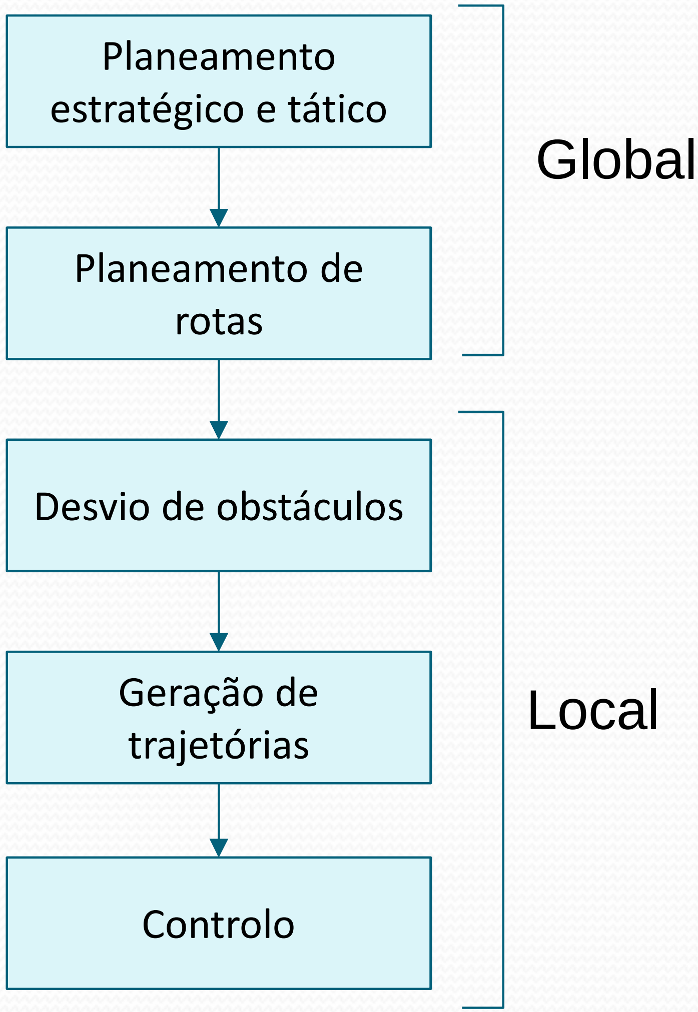
Robótica Espacial

Planeamento de Caminhos em Robôs Móveis

Vítor Santos, Maio 2025
Universidade de Aveiro

Níveis de Planeamento

- Planeamento estratégico e tático
  - e.g. Visitar vários locais numa dada região, por exemplo numa dada sequência.
- Planeamento de rotas
  - e.g. Definir quais os caminhos a fazer para cada região em particular, portanto, com um ponto de partida e um de chegada. Muito se assume neste nível: os caminhos não estão bloqueados ou impedidos por outros agentes, etc.
- Desvio de obstáculos
  - Evitar outros veículos, ultrapassá-los, distâncias a preservar, contornar obstáculos ou bloqueios, etc.
- Geração de trajetórias
  - Definição dos caminhos instantâneos e dos momentos em que devem ser percorridos (velocidades, acelerações), respeitando questões de segurança e conforto para as pessoas e bens transportados.
- Controlo
  - Instruções e leis para os atuadores para procurar respeitar a trajetória planeada fazendo as correções necessárias



Planeamento global vs. local

- Global
  - É preciso ter um destino final para calcular soluções
  - As soluções podem ser de alguma forma otimizadas porque há um conhecimento a priori das diversas possibilidades, por exemplo para evitar mínimos locais
  - As soluções são caminhos que podem ser planeados a priori
- Local
  - O destino final não é, em geral, necessário nem usado no cálculo, apenas destinos próximos ou provisórios sempre em atualização
  - Algumas soluções incorrem no risco de ter mínimos locais
  - Requer sensores para condicionar movimento e evitar obstáculos
  - Requer uma lei de controlo para as correções contínuas de trajetória
- Algumas técnicas reúnem características mistas

Princípio geral do planeamento **global** de caminhos

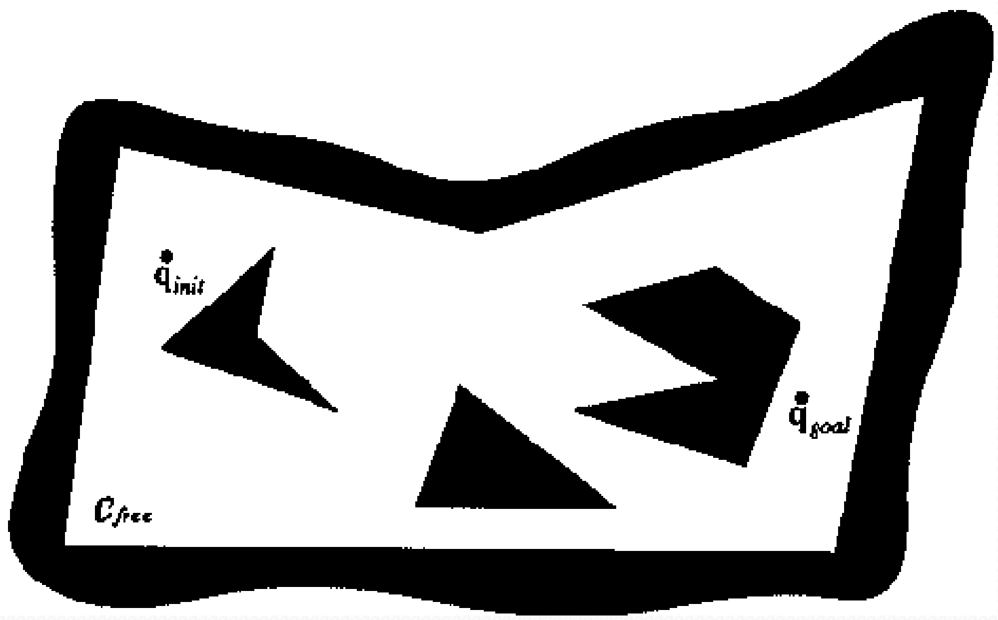
- Os métodos de planeamento global de caminhos incluem 2 passos:
  - **Pre-processamento**: Obter a conectividade do espaço livre numa representação conveniente (um grafo, árvore, etc.)
  - **Processamento de busca**: Fazer uma busca de um caminho na representação (grafo, árvore, ...).
- Famílias principais de métodos
  - Decomposição em células
  - Mapa de estradas
  - Campos de potencial (também usado em planeamento local)
  - Métodos de amostragem (*sampling methods*)
    - Recomendados em especial para espaços multidimensionais

Métodos de decomposição em células

- Célula
  - Região do espaço livre tal que, calcular um caminho entre dois pontos quaisquer, dentro da mesma célula, é um processo rápido (leva sempre o mesmo tempo). Um exemplo é o caso de polígonos convexos.
- Pre-processamento
  - Partição do espaço livre numa coleção de células.
  - Construir o grafo de conectividade representando a relação de "adjacência" entre células.
- Processamento da busca
  - Procurar, no grafo de conectividade, por uma sequência de células adjacentes entre a célula inicial e a célula final.
  - Transformar essa sequência num caminho.

Decomposição poligonal – I

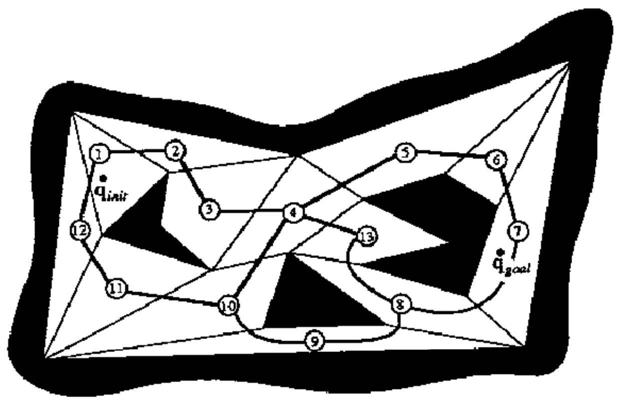
- Problema
  - Planeamento do caminho de qinit para qgoal

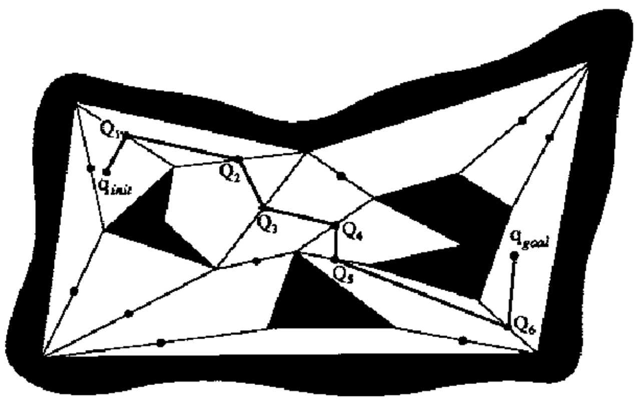


- Método a usar
  - Decomposição poligonal exata em polígonos convexos

Decomposição poligonal - II

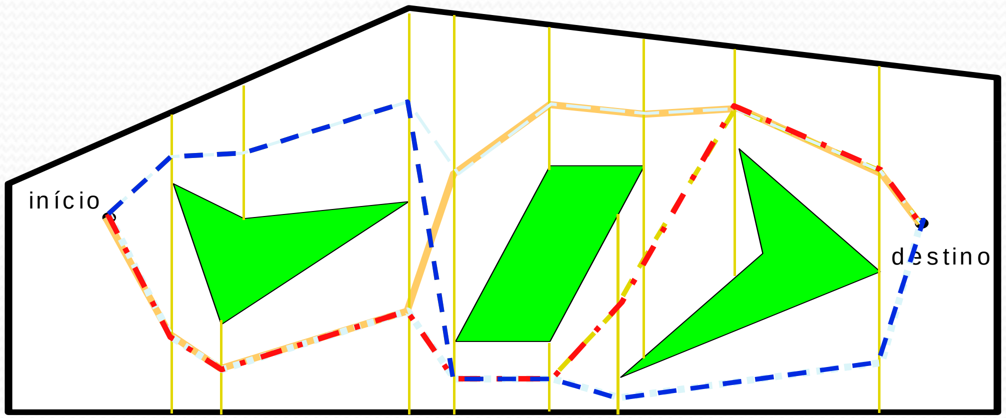
- Estabelecer o grafo de adjacências
- Definir os caminhos
  - Processo 1: grafo dos centros geométricos dos polígonos
  - Processo 2: grafo dos pontos médios dos segmentes de adjacência
- Seleccionar o caminho com base num critério
  - Mais curto, mais seguro...





Decomposição poligonal - III

- A decomposição trapezoidal vertical
  - Polígonos trapezoidais com base nos vértices dos obstáculos e da fronteira do ambiente
  - Grafo dos pontos médios dos segmentos de adjacência
  - Procura de caminhos segundo um critério a definir
- O exemplo ilustrado tem 6 caminhos possíveis

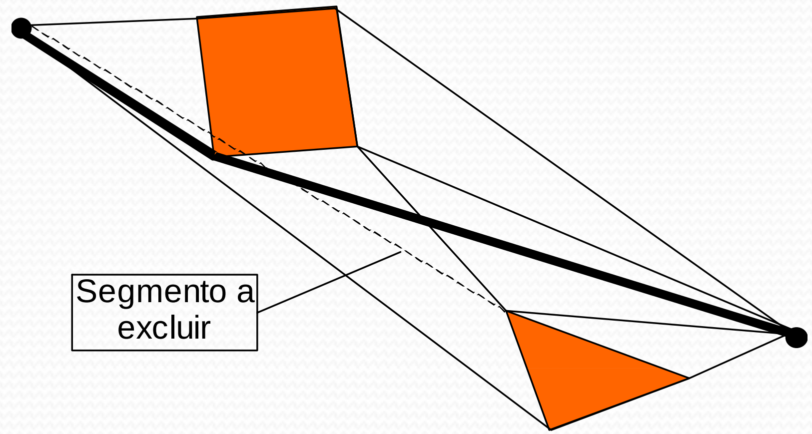


Métodos de mapa de estradas (*roadmaps*)

- Mapa de estradas
  - Rede de curvas a uma dimensão no espaço livre. Idealmente:
    - ...deveria haver uma correspondência de um-para-um entre os componentes desse mapa e os componentes do espaço livre.
    - ...encontrar um caminho ligando qualquer configuração livre a um ponto no mapa de estradas deveria ser rápido.
- Pré-processamento
  - Construir uma rede de curvas (o mapa de estradas) que "traduza" a conectividade do espaço livre.
- Processamento de busca
  - Ligar as configurações inicial e final ao mapa de estradas.
  - Procurar um caminho no mapa de estradas.
- Exemplos de técnicas
  - Grafo de visibilidade
  - Diagrama de Voronoi

Grafo de visibilidade

- Definição
  - Todas as sequências de segmentos que unem vértices de obstáculos sem atravessar nenhum deles, incluindo os pontos de partida e chegada.
- Escolha de caminho
  - Mais curto, menos segmentos, ...
- Caminho final
  - Requer processamento adicional para “afastar” dos obstáculos
- Este método proporciona o caminho mais curto entre partida e destino



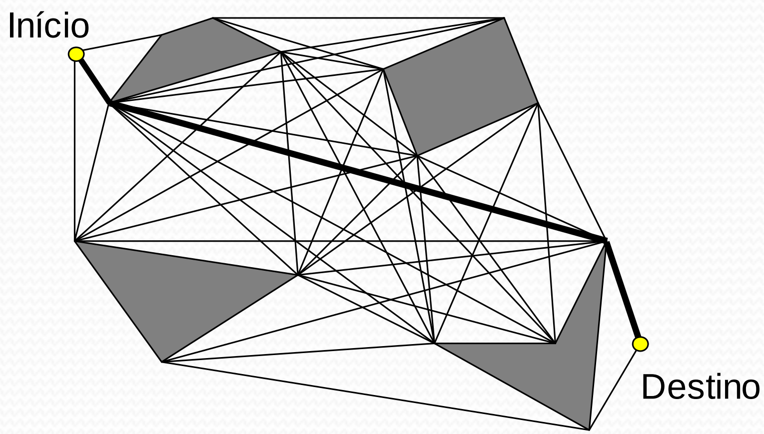
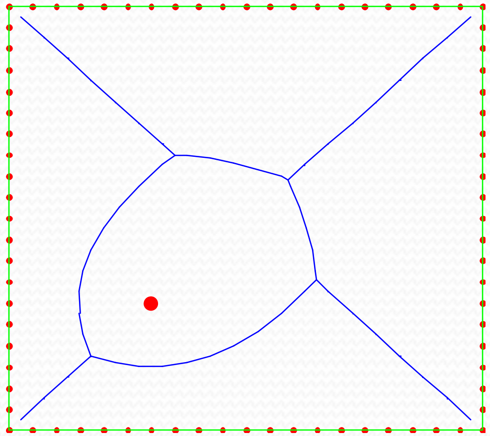


Diagrama de Voronoi - I

- Definição
  - Lugar geométrico que representa a região mais afastada dos obstáculos e do ambiente
- Exemplos numa sala rectangular
  - Um obstáculo pontual e as paredes da sala
  - Dois obstáculos pontuais e as paredes da sala



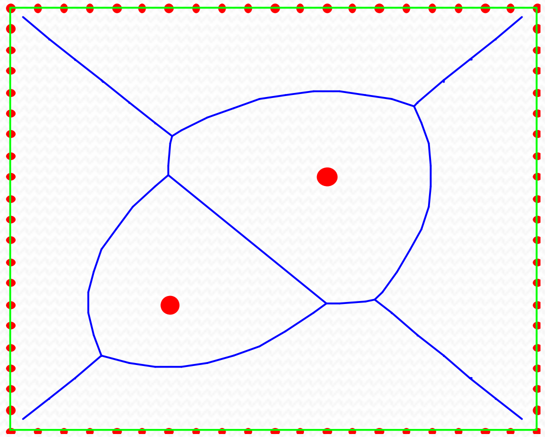
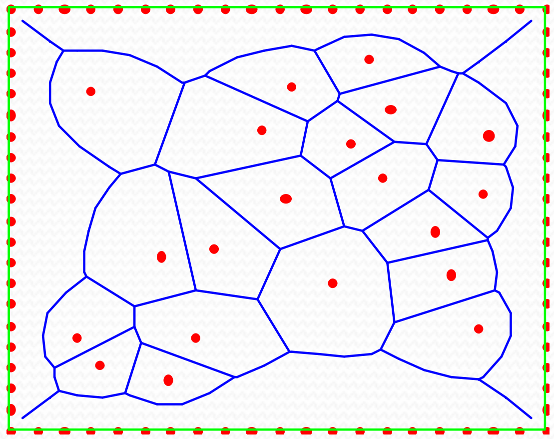
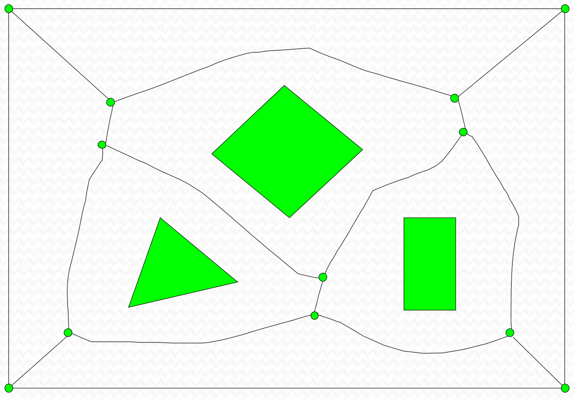


Diagrama de Voronoi - II

- Exemplo com 20 obstáculos pontuais
- Diagrama de Voronoi generalizado
  - Obstáculos não pontuais



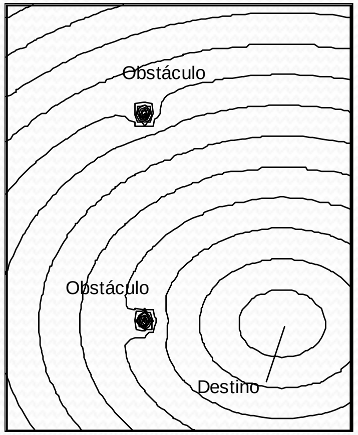


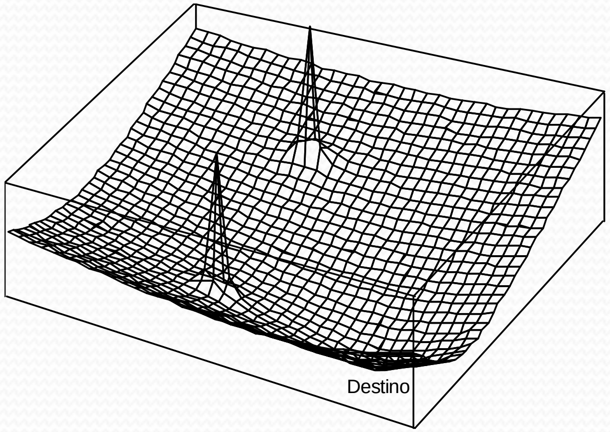
Campos de potencial

- Definição
  - Função definida sobre a região de espaço livre
  - Os obstáculos repelem e o destino atrai
- Metodologia
  - Pre-processamento
    - Definir a função (o campo de potencial) sobre o espaço de configuração do robô.
    - Definir e colocar uma grelha sobre o espaço de configuração do robô para efeitos práticos da busca.
  - Processamento da busca
    - Procura um caminho sobre a grelha usando o campo de potencial como heurística.

Campos de potencial – II

- Exemplo do conceito
  - Dois obstáculos pontuais
    - Em cuja vizinhança aumenta uma dada função de custo (“potencial”)
  - Um objetivo
    - Em cuja vizinhança diminui a função de custo (“potencial”)
- O problema maior é:
  - Risco de mínimos locais do campo de potencial





Utilização dos Campos de Potencial

- Planeamento Global (como apresentado)
  - Os mínimos locais podem ser previstos e resolvidos a priori (planeamento do caminho)
- Planeamento local
  - Para definição do movimento instantâneo
  - Destino como ponto **atrator**
  - Medidas sensoriais como **repulsores** de obstáculos
  - Problema dos mínimos locais mais complexo de contornar
  - *A abordar mais tarde...*

Problemática do planeamento de caminhos

- Adotar a metodologia
  - Decomposição em células ou ...
  - ... Obtenção do mapa de estradas
- Estabelecer a conectividade
  - Definir entidades (pontos, regiões) como nós de **grafos**
  - Relacionar com arcos de grafos as adjacências entre os nós
  - Formular matematicamente a conectividade com uma matriz de adjacências e/ou de capacidades (custos)
- Obter o caminho final
  - Algoritmos de **busca** de caminhos

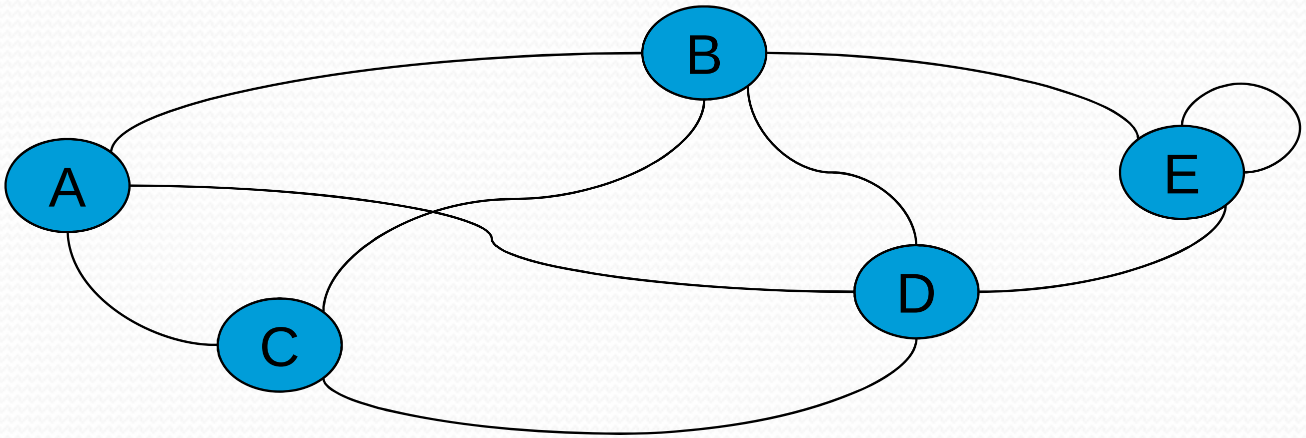
- Existem abordagens onde os diversos passos são integrados ou operados com variantes.

Ferramentas computacionais

- Como expressar a conectividade
  - **Grafos** e árvores
  - **Matriz de adjacências** (e/ou de custos)
- Algoritmos de busca
  - Tradicionais / "força bruta"
  - **Dijkstra**
  - **A*** (*A-star*)
  - etc.

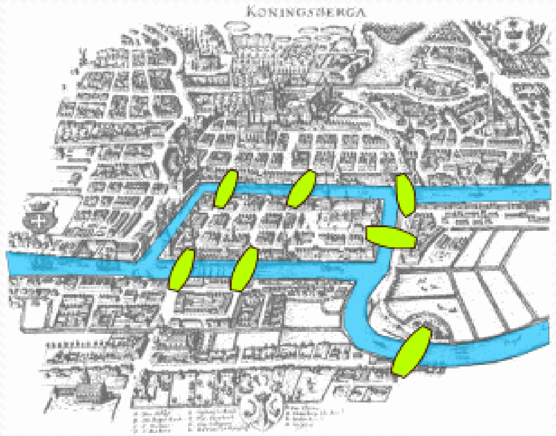
Grafo

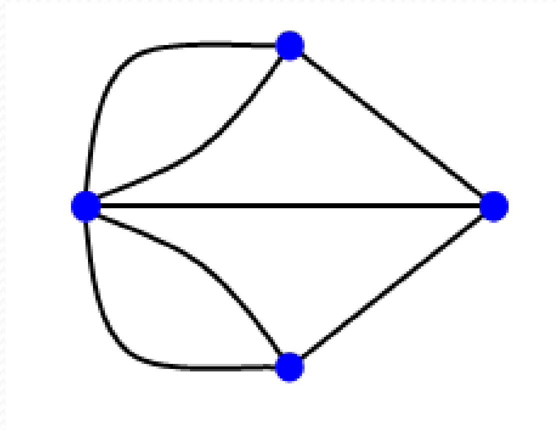
- Tema de estudo da Teoria dos Grafos
- Informalmente, consiste num conjunto de objetos (pontos, nós ou vértices) ligados entre si através de arcos (ligações, ramos ou arcos)
- Inúmeros problemas podem ser formulados recorrendo a grafos, como é o caso do planeamento de caminhos



Nascimento da teoria dos Grafos

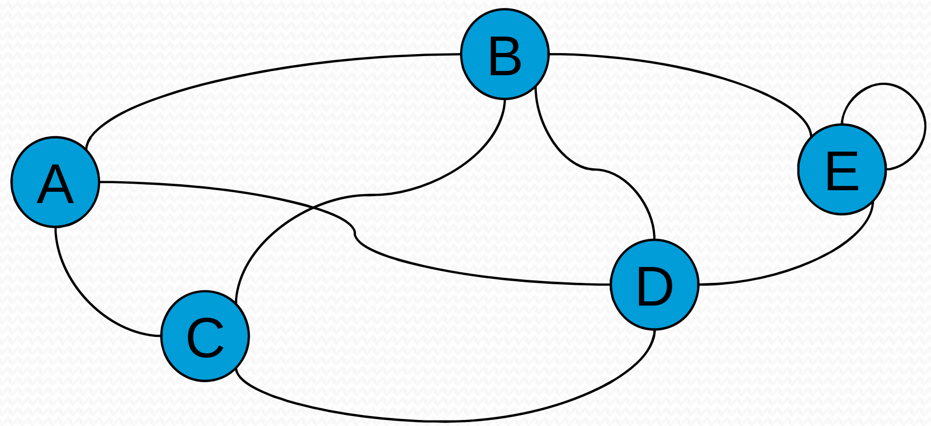
- As 7 pontes de Königsberg (No enclave Russo entre Polónia e Lituânia)
- Problema: passar uma só vez em cada uma das sete pontes e voltar ao ponto de partida!
- Em 1736, Euler formulou um grafo e demonstrou que o problema era impossível!
  - O grafo de Euler tinha sete arcos representando as sete pontes que se juntavam nas 4 regiões em que o rio Pregel dividia a cidade

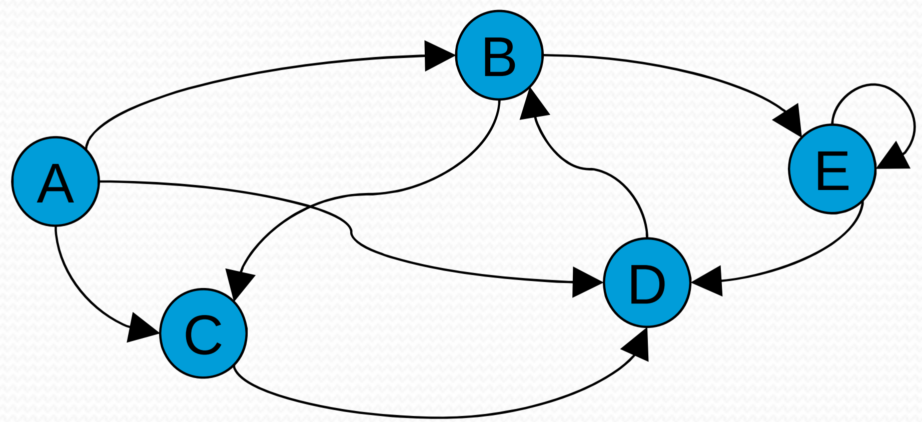




Grafos – Algumas definições base

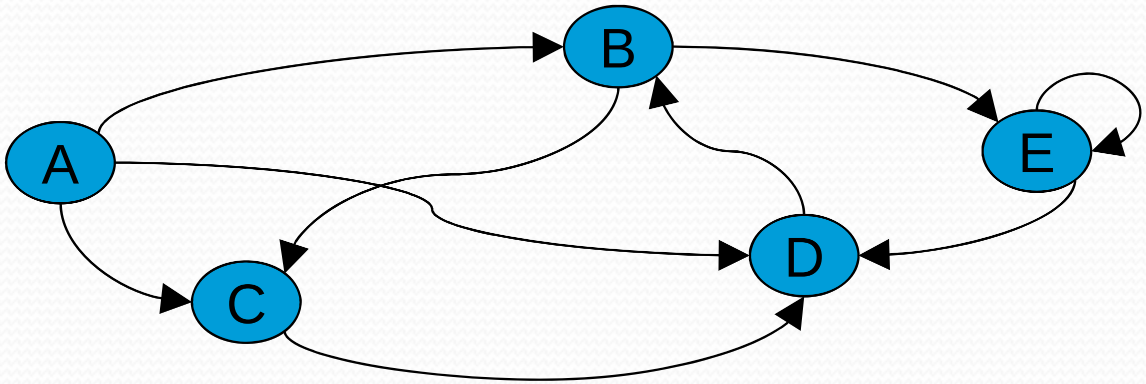
- Os grafos podem ser:
  - não orientados (*non-directed graph*)
    - Não há sentido definido entre os nós
  - orientados (*directed graph* ou *digraph*)
    - Há um sentido definido entre os nós





Grafos e sua conectividade - definições

- Trajeto (*walk*)
  - Sequência alternada de nós e arcos, começando e acabando em nós. Exemplos:
    - A, B, C, D
    - A, C, D, B, E, D
- O comprimento de um trajeto
  - Número de arcos que o compõem
- Caminho (*path*)
  - Todo o trajeto sem vértices repetidos. Ex. A, B, C, D
- Ciclo
  - trajeto que se inicia e termina no mesmo nó. Ex. B,C,D,B
- Anel
  - Arco que liga um nó a si próprio. Caso no nó E ilustrado.
- Distância entre dois nós
  - O menor comprimento de todos os caminhos entre esses nós



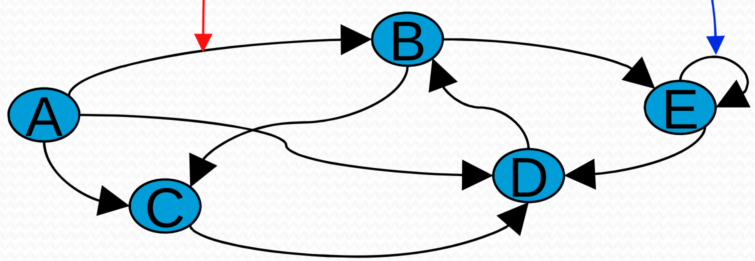
Matriz de Adjacências

- A matriz de adjacências (MA) traduz totalmente a conectividade de um grafo.
- Usualmente nas linhas indica-se o nó de partida e nas colunas o nó de chegada
- O número indica a conectividade
  - **0** para nós não ligados
  - **Outros** valores para nós ligados (1 é o usado numa matriz de adjacências)
- Um anel surge como um valor não nulo no respetivo elemento da diagonal principal da matriz

Para o nó:

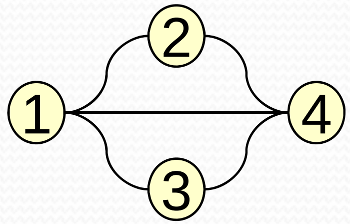
Do nó:

	A	B	C	D	E
A	0	1	1	0	0
B	0	0	1	0	1
C	0	0	0	1	0
D	0	1	0	0	0
E	0	0	0	1	1



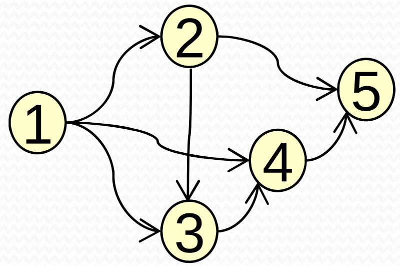
Propriedades da Matriz de Adjacências

- Num grafo **não orientado** a matriz é simétrica
- Num grafo sem anéis a diagonal principal é toda de zeros



0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0

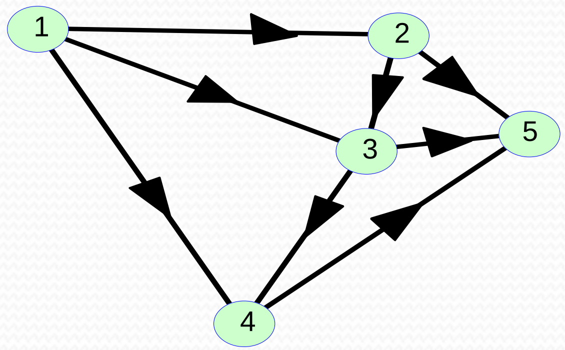
- Num grafo orientado, nos elementos simétricos na matriz, quando um é não nulo o outro é nulo.
- Num grafo orientado em que as ligações entre nós só seguem uma ordem crescente, a matriz é triangular superior
- Um nó de partida cria uma coluna de zeros e um nó terminal cria uma linha de zeros



0	1	1	1	0
0	0	1	0	1
0	0	0	1	1
0	0	0	0	1
0	0	0	0	0

Potenciação da Matriz de Adjacências

- Se A for matriz de adjacência de um grafo então:
  - A^N é a matriz das ligações de comprimento N,
  - e representa as formas de ligar quaisquer par de nós em N passos



A=

0	1	1	1	0
0	0	1	0	1
0	0	0	1	1
0	0	0	0	1
0	0	0	0	0

A^2=

0	0	1	1	3
0	0	0	1	1
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

Três caminhos de 2 passos para ir de 1 a 5: 125 ; 135 ;145

A^3=

0	0	0	1	2
0	0	0	0	1
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

Dois caminhos de 3 passos para ir de 1 a 5: 1235 ; 1345

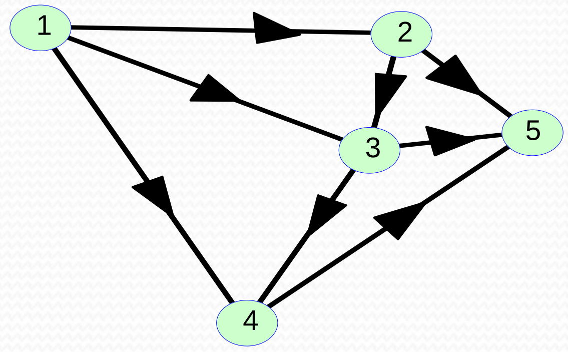
A^4=

0	0	0	0	1
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

Um caminho de 4 passos para ir de 1 a 5: 12345

Potenciação da Matriz – parte II

- Num grafo com matriz de adjacências A é possível determinar se dois nós podem ser ligados com, no máximo, k passos pela seguinte expressão:
  - $S_k = A + A^2 + A^3 + A^4 + \ldots + A^k$
- No exemplo há 6 caminhos com 4 ou menos passos para ir do nó 1 para o nó 5, e só 3 caminhos para ir do nó 1 para o nó 4, etc.!
- NB: O método só indica o número de caminhos, mas não indica diretamente quais são esses caminhos!

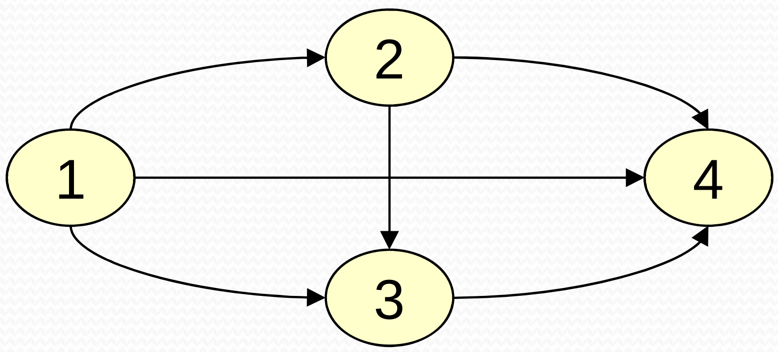


$S_4 =$

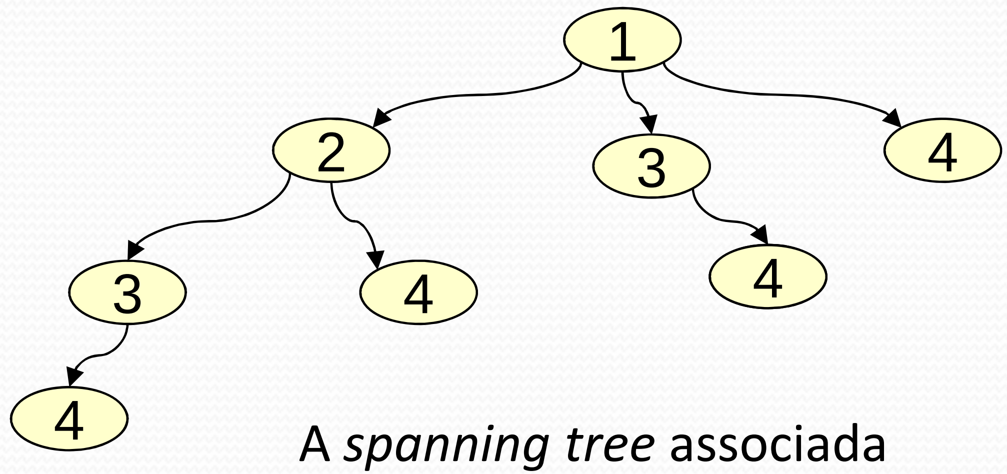
0	1	2	3	6
0	0	1	1	3
0	0	0	1	2
0	0	0	0	1
0	0	0	0	0

Spanning Tree

- Expansão total de um grafo numa árvore
  - Por haver muitos algoritmos adequados a organizações em árvore, pode ser vantajoso representar um grafo de forma expandida numa árvore
  - Elimina ciclos (mas pode repetir nós eventualmente)
- Há ainda a *spanning tree* mínima
  - Onde a soma total dos pesos dos arcos é mínima



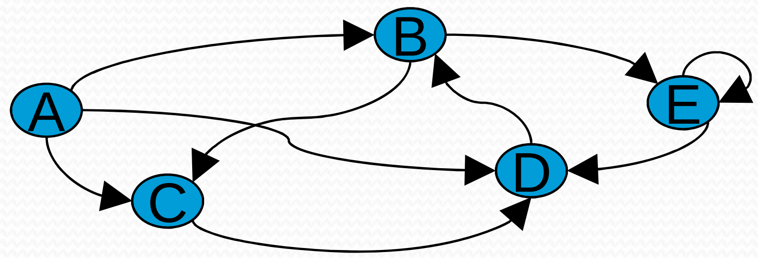
Um grafo



A *spanning tree* associada

Lista de Adjacências

- A LA é uma enumeração (lista) dos nós adjacentes a cada um dos nós de um grafo.
  - A -> B, C, D
  - B -> C, E
  - C -> D
  - D -> B
  - E -> D, E
- Comparativamente à Matriz de Adjacências, pode apresentar vantagens ou desvantagens:
  - Num grafo esparso a LA é mais vantajosa (não é preciso representar as ligações ausentes)
  - É preferível a LA quando se procura uma sequência de adjacências (aonde liga um nó)
  - Se se pretender apenas saber se dois nós são adjacentes a MA é preferível

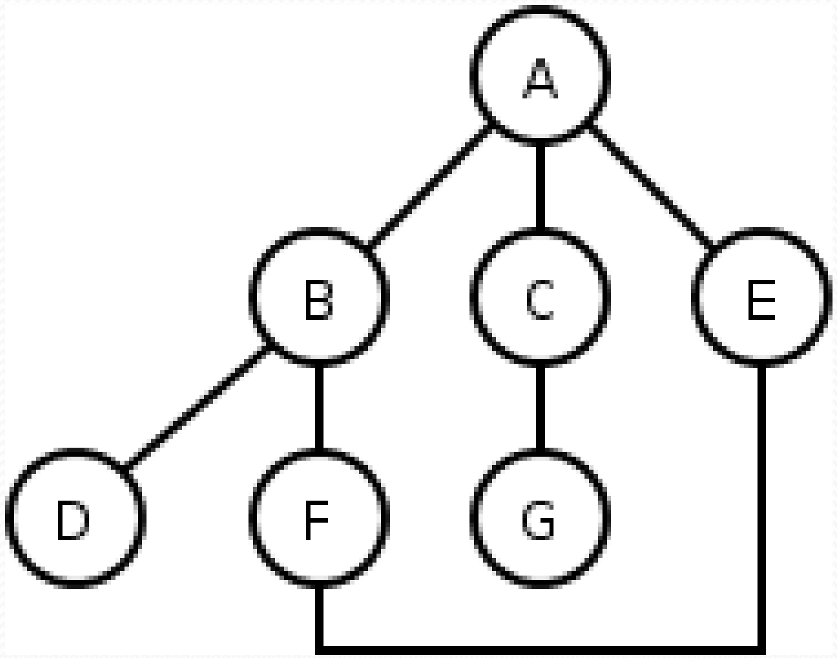


Algoritmos de busca

- A busca de um determinado caminho num grafo faz-se recorrendo à Lista de Adjacências ou à Matriz de Adjacências
- Há 3 categorias principais de algoritmos de busca em grafos:
  - Busca em profundidade (*Depth-first search*) - DFS
  - Busca transversal (*Breadth-fist search*) - BFS
  - Busca do melhor primeiro (*Best-first search*)
    - Algoritmo de Dijkstra
    - Algoritmo A*
    - etc.

Exemplificação básica de busca

- Busca iniciando no ponto A e escolhendo primeiro as ligações da esquerda para a direita
- Transversal (*Breadth-first*) (BFS)
  - Ordem de visita:
    - A, B, C, E, D, F, G
- Profundidade (*Depth-first*) (DFS)
  - Ordem de visita:
    - A, B, D, F, C, G, E



Breadth-first search (BFS)

- Intuitivamente, inicia-se na raiz (nó de partida) e verifica os vizinhos todos (nós a que se liga)
- É um algoritmo não informado que percorre todos os nós: é exaustivo e não usa nenhuma heurística.
- Os nós a testar são colocados numa lista ligada com a propriedade de abertos (*open*) enquanto não são verificados mas depois de verificados passam a ter o estatuto de fechados (*closed*)
- O algoritmo é:
  - Completo – se existir o nó procurado, ele encontra-o!
  - Ótimo – consegue encontrar o número mínimo de passos para atingir o objectivo (percorrê-los todos!)
  - Tem complexidade linear com a dimensão do grafo (nodos+ligações)...
    - ...no espaço, em termos de ocupação de memória, ...
    - ... e no tempo computacional

Algoritmos *Greedy* (gulosos ou vorazes)

- Algoritmos (de busca) que, em cada passo, escolhem a opção localmente mais vantajosa
  - Comportamento esperado no Traveling Salesman:
    - Em cada nó, um algoritmo guloso escolheria o próximo nó baseado na menor distância a todos os nós ligados dali para a frente.
- Alguns podem não dar soluções ótimas globalmente, mas podem ser mais eficientes no tempo de busca.

Algoritmo de Dijkstra – Introdução

- Resolve o problema do caminho mais curto num grafo
- Concebido para grafos orientados ponderados de pesos **não negativos**
- Determina as distâncias de um nó a todos os outros do grafo
- Termina quando todas as distâncias foram calculadas
- Também é um algoritmo do tipo “greedy”

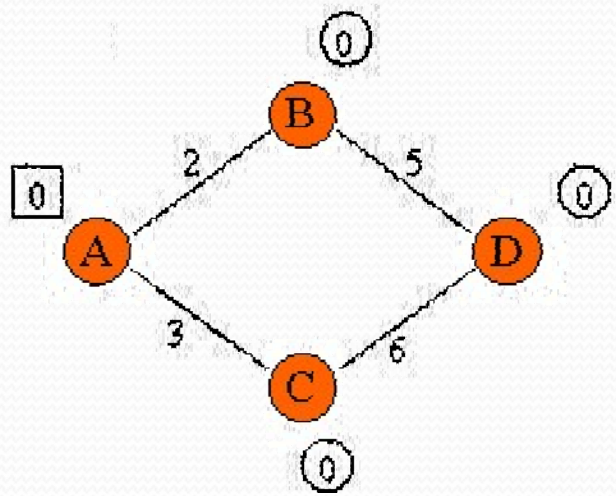
Algoritmo de Dijkstra – I

- Princípios:
  - Criar etiquetas associadas a cada nó relacionadas com distância (custo)
  - Há etiquetas temporárias e permanentes (ou *open* & *closed*)
  - As temporárias são para nós ainda não passados e as permanentes são para nós já passados
  - Ilustração:
    - O nó A tem uma etiqueta temporária de valor 0
    - O nó B tem uma etiqueta definitiva de valor 5

- Como é óbvio, é necessário conhecer o grafo todo de antemão (nós e arcos)
- Embora não haja o conceito de nó de destino final

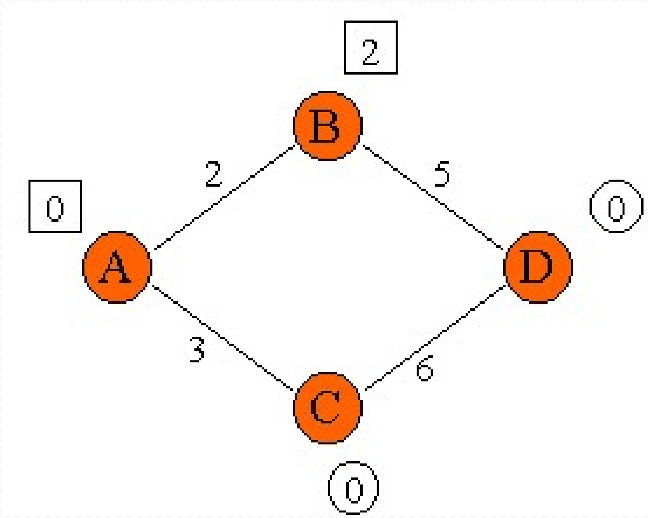
Algoritmo de Dijkstra – II

- Começa por inicializar um vértice do grafo com a etiqueta zero definitiva (A - partida) e todos os outros (B, C, D) com valor temporário de zero



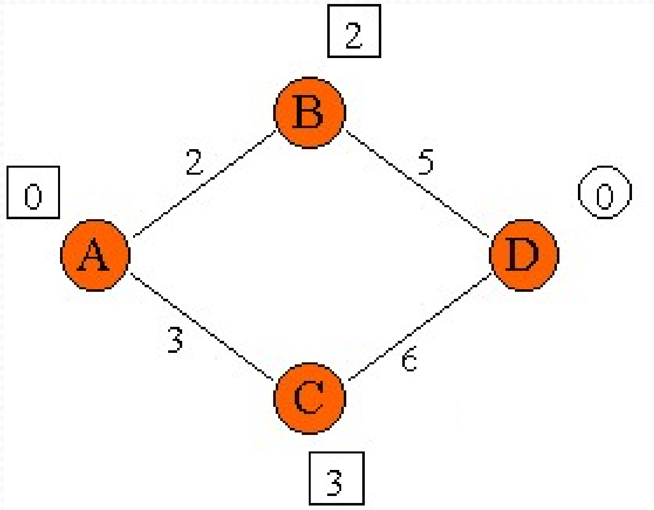
Algoritmo de Dijkstra – III

- Partindo do nó com etiqueta permanente corrente (A, neste caso) procura a ligação com menor custo que tenha etiqueta temporária (seria B, dado que C teria um custo maior: 2 vs. 3)
- A etiqueta de B passa a permanente com o valor de A + custo da ligação AB, ou seja, 0+2=2



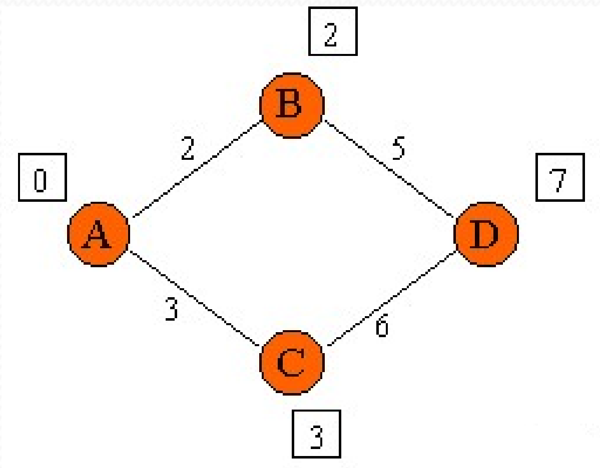
Algoritmo de Dijkstra – IV

- Procurar o nó temporário com menor custo desde o nó permanente A ou B.
- O nó encontrado é C dado que o custo é 3. Se fosse D, seria 7 (2+5)!



Algoritmo de Dijkstra – V

- Repete o processo, resultando no nó D a ter como etiqueta permanente 7 (=2+5), e não 9 (6+3) pelas razões do menor custo.
- Estende o processo até todos os nós terem etiqueta permanente.

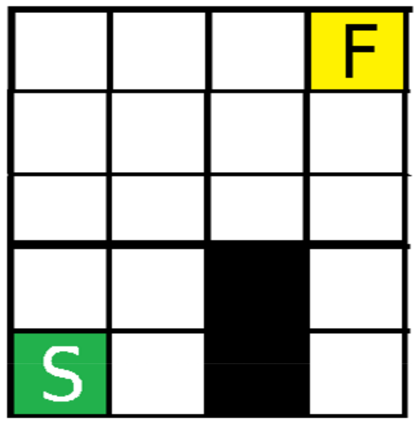


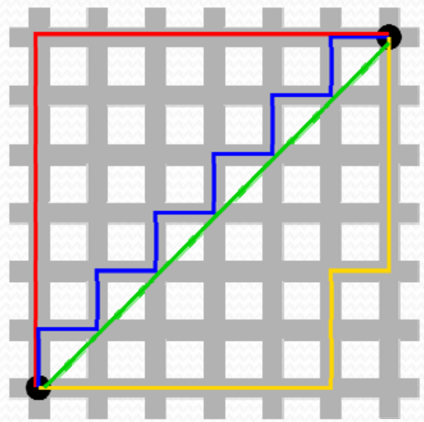
Algoritmo A*

- É equivalente ao Algoritmo de Dijkstra mas é dotado de uma heurística
- A heurística é, em geral, uma função **h(n)** que permite calcular (estimando) a distância do nó corrente ao nó final
  - Por exemplo, se se souberem coordenadas métricas dos nós pode-se usar uma distância cartesiana como estimativa (ou outra distância, como *city-block*, etc.)
  - Ou seja, não se usa apenas a informação da distância (acumulada) ao nó de partida **g(n)**, mas também a distância estimada ao nó de chegada, que seria **h(n)** (a heurística); isto é, usa o valor **g(n)+h(n)** em vez de apenas **g(n)** como faz o algoritmo de Dijkstra para atribuir as “etiquetas” aos nós
- Em resumo, Dijkstra não usa nenhuma informação do destino
  - porque calcula o caminho mais curto entre quais pares de nós no grafo,
  - mas A* leva em linha de conta o de destino e calcula o caminho mais curto (baseado na heurística)
- Com a heurística acertada, é o algoritmo genérico mais eficiente de busca. (usado em jogos, puzzles, etc...)
- A heurística tem de ser admissível, isto é, não pode sobreestimar o valor do custo ao destino!

Exemplo com A*

- O grande desafio do A* é definir o valor da heurística h(n) em cada nó.
- Seja o problema de procurar o caminho num mapa em grelha de S até F.
  - Cada célula será um nó do grafo.
  - As células pretas são obstáculos e não são consideradas
- Para heurística usa-se a distância de Manhattan da célula corrente ao destino
  - $h(n)=|x_n - x_F| + |y_n - y_F|$
  - Segue as linhas horizontais e verticais
  - A distância Euclidiana (a verde) seria mais precisa mas computacionalmente mais exigente!

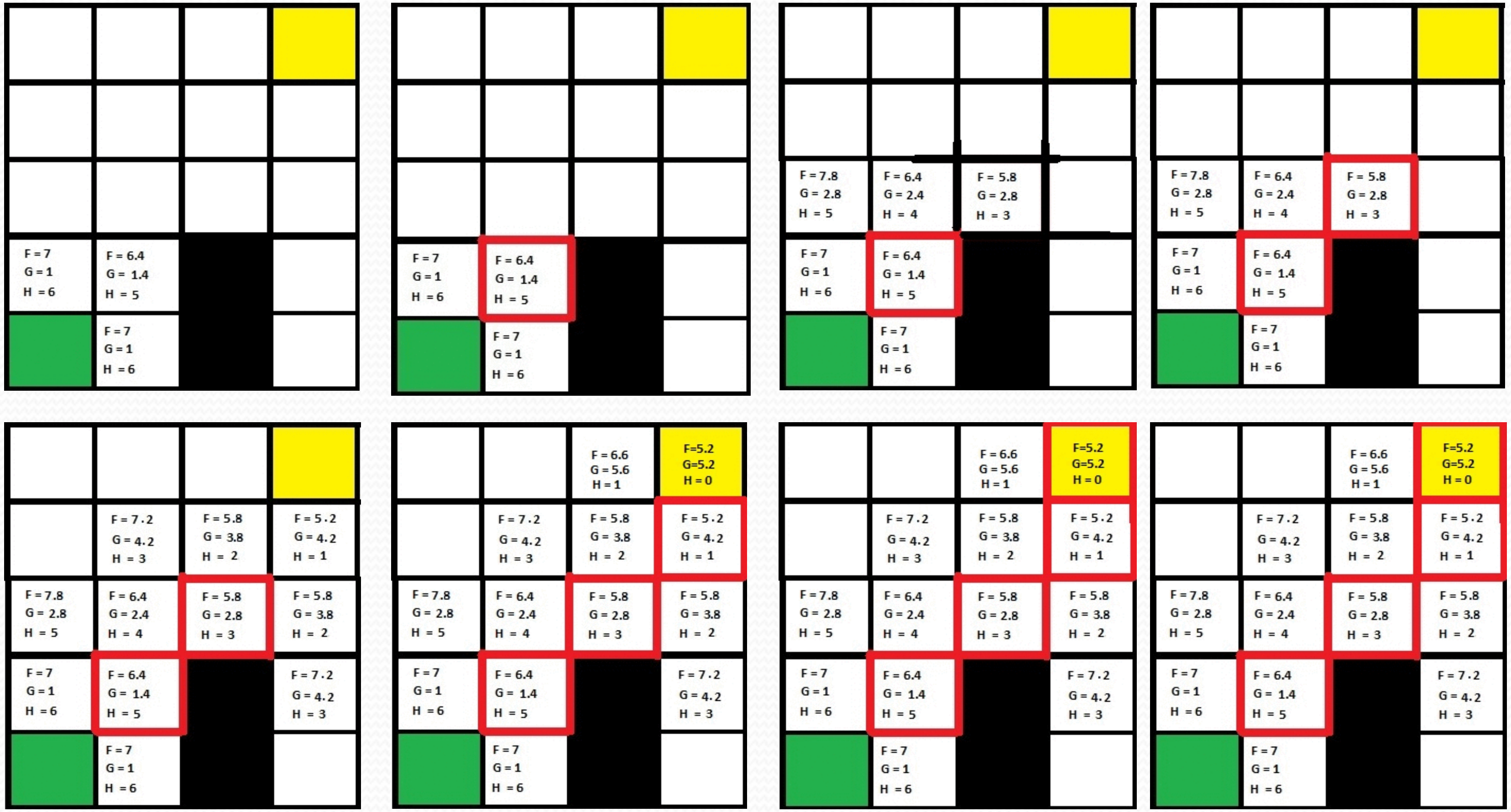




https://brilliant.org/wiki/a-star-search/

Exemplo com A* - Sequência de cálculo do caminho

- Unidade de distâncias na horizontal e vertical =1; na diagonal = 1.4 (≈√2)
- Junto de cada célula corrente indicam-se os valores de F=G+H de todos os vizinhos
- H é inteiro (distância de Manhattan) e G tem múltiplas parcelas de 1 e 1.4
- A próxima célula do caminho é a que tem o menor F dos vizinhos que ainda não pertencem ao caminho.



https://brilliant.org/wiki/a-star-search/