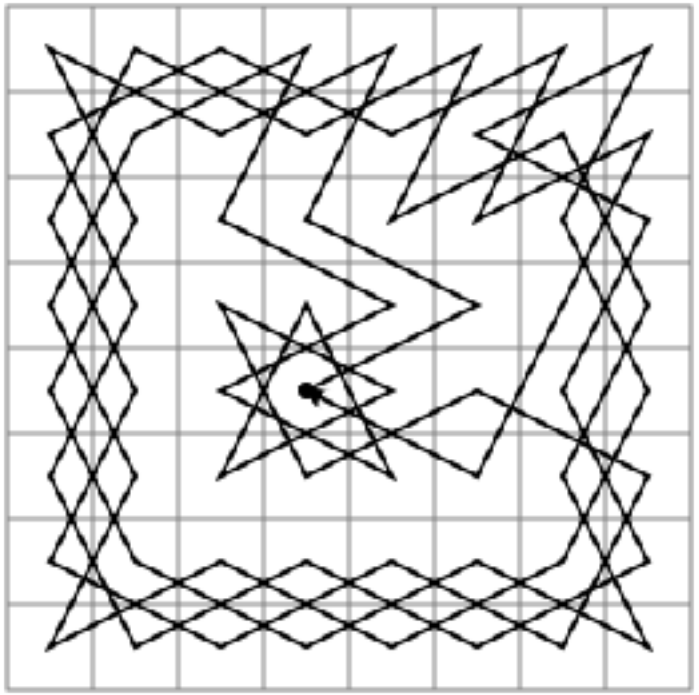
Circuito Hamiltoniano

Brute Force
Greedy
Heuristics

Knight's Tour - Caracterização

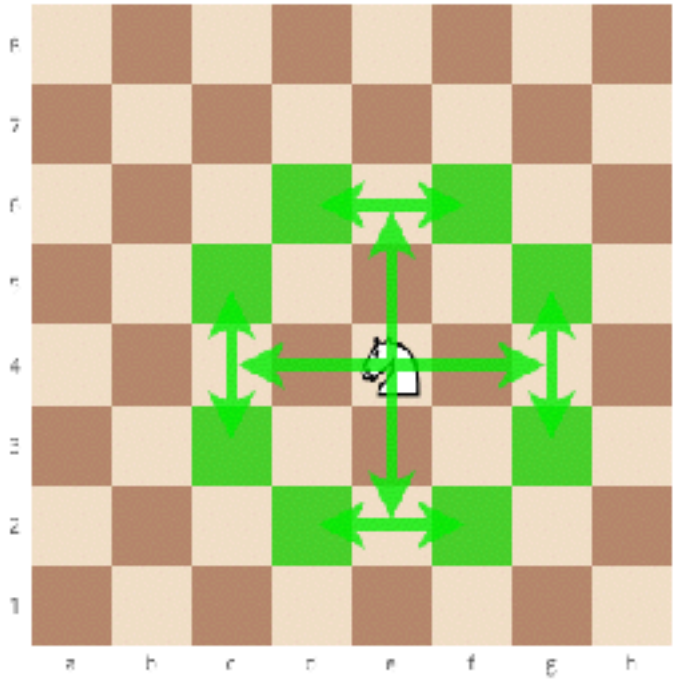


- Na sua formulação original, o problema consiste em determinar um circuito (fechado ou aberto) no qual o cavalo visita todas as posições do tabuleiro de xadrez
- Tal como o TSP, é uma instância do circuito Hamiltoniano mas ao contrário deste é resolúvel com complexidade temporal linear O(n)
- Pode ser visto em termos de grafos como cada jogada possível sendo um ramo e cada posição do tabuleiro um nó

Na imagem, circuito fechado apresentado pelo "The Turk"

Knight's Tour - Formulação Prolog – Brute Force

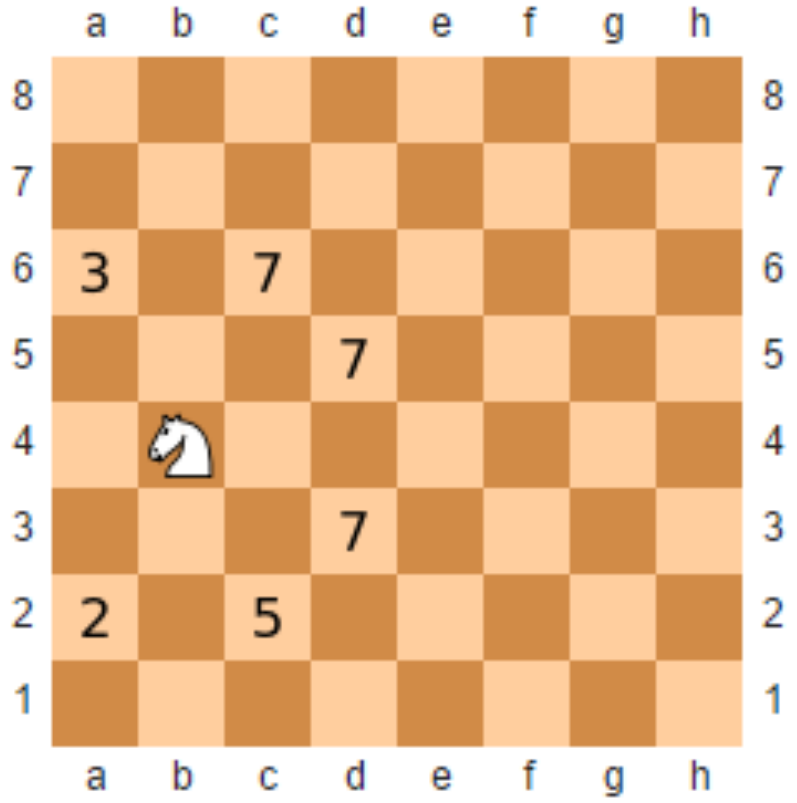
```
table_size(8).
jump((X,Y),(NX,NY)):- NX is X-1, NY is Y-2.
jump((X,Y),(NX,NY)):- NX is X+1, NY is Y-2.
jump((X,Y),(NX,NY)):- NX is X-1, NY is Y+2.
jump((X,Y),(NX,NY)):- NX is X+1, NY is Y+2.
jump((X,Y),(NX,NY)):- NX is X-2, NY is Y+1.
jump((X,Y),(NX,NY)):- NX is X+2, NY is Y+1.
jump((X,Y),(NX,NY)):- NX is X-2, NY is Y-1.
jump((X,Y),(NX,NY)):- NX is X+2, NY is Y-1.

checkInsideTable((X,Y)):-
        table_size(N),
        X>=1, X=<N,
        Y>=1, Y=<N.

%LMoves: lista da posiçoes anteriores
nextValidMove(Pos,LMoves,NewPos):-
      jump(Pos,NewPos),checkInsideTable(NewPos),
      \+ member(NewPos,LMoves).
```



Knight's Tour - Formulação Prolog – Brute Force

```
horseJump(StartPos,LR):-
        table_size(N), NS is N*N-1, horseJump(NS,[StartPos],LR).

horseJump(0,LA,LR):- reverse(LA,LR).

horseJump(NS,LMoves,LR):-
        NS>0,
        LMoves=[Pos|_],
        %brute force
        nextValidMove(Pos,LMoves,NewPos),   NewNS is NS-1,
        horseJump(NewNS,[NewPos|LMoves],LR).

horseJump(NS,_,_):-
        write('-backtracking-'),nl, write(NS),nl,
        fail.
```

Knight's Tour – Heurística de Warnsdorff



- A heurística consiste em determinar o número de jogadas válidas a partir de cada posição adjacente da actual e seleccionar a posição com número de jogadas potenciais mais baixo
- A ideia subjacente é resolver os passos potencialmente mais complicados em primeiro lugar; neste caso a2 seria a posição escolhida
- Esta é uma heurística de construção

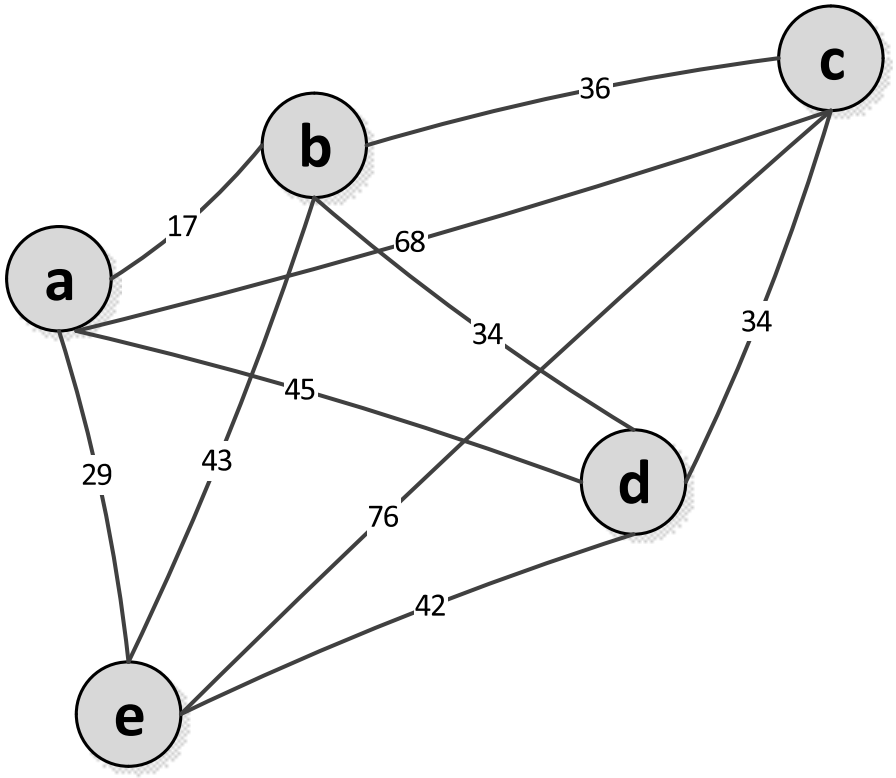
Knight's Tour – Heurística de Warnsdorff

```
validMoves(Pos,LMoves,LValMoves):-
        findall(NewPos,
        nextValidMove(Pos,LMoves,NewPos),
        LValMoves).


countValidMoves(Pos,LMoves,Count):-
        findall(NewPos,
        nextValidMove(Pos,LMoves,NewPos),
        LValMoves),
        length(LValMoves,Count).


%heuristic Warnsdorff
sortedMoves(Pos,LMoves,SortedMoves):-
   validMoves(Pos,LMoves,LNeighbors),
   findall((Count:(Neighbor)),(member(Neighbor,LNeighbors),
            countValidMoves(Neighbor,LMoves,Count)), Moves),
   sort(Moves,SortedMoves).
```

Knight's Tour - Heurística de Warnsdorff

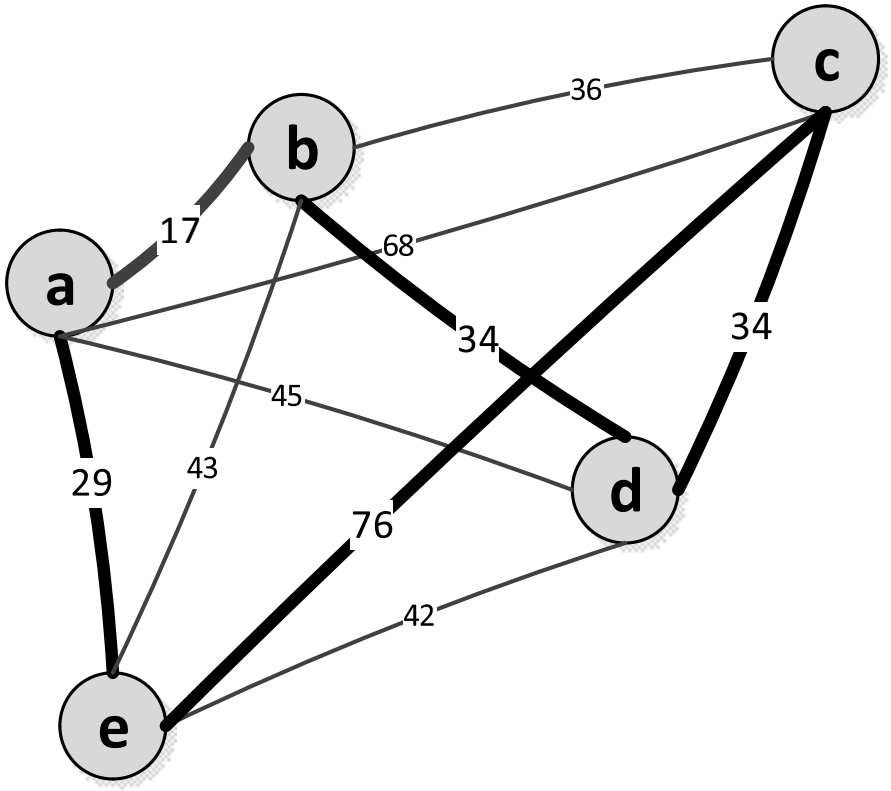
```
horseJump(StartPos,LR):-
        table_size(N), NS is N*N-1, horseJump(NS,[StartPos],LR).

horseJump(0,LA,LR):-   reverse(LA,LR).

horseJump(NS,LMoves,LR):-
        NS>0,
        LMoves=[Pos|_],

        %heuristica Warnsdorff
        sortedMoves(Pos,LMoves,SortedMoves),  member((_:NewPos),SortedMoves),
        NewNS is NS-1,
        % write((NS,' ',NewPos)),nl, get0(_),
         horseJump(NewNS,[NewPos|LMoves],LR).
```

TSP – Traveling Salesman Problem



- O Problema do Caixeiro Viajante consiste em determinar um circuito (fechado) que permite visitar um conjunto de cidades sem repetições com o mais baixo custo possível
- Este problema é NP-hard
- Existem múltiplas heurísticas (de construção e optimização) que permitem obter soluções sub-óptimas

TSP – Heurística de construção greedy

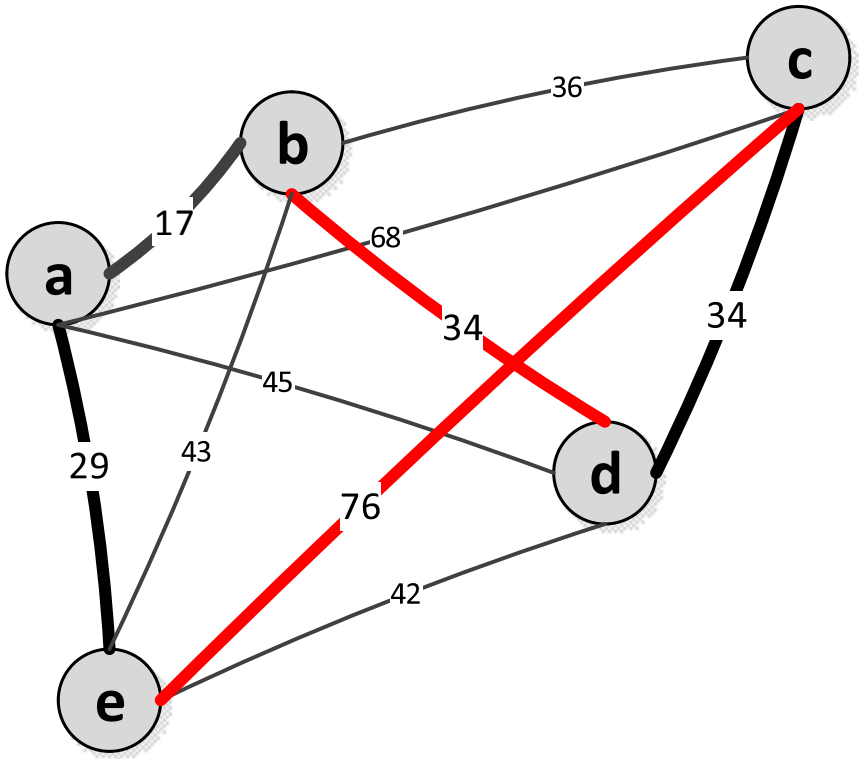


- A heurística greedy (gulosa) consiste em determinar o nó mais próximo do actual e transitar para esse nó
- Em média esta abordagem produz, em tempo linear, soluções 25% mais caras que a solução óptima

Solução: [a,b,d,c,e,a]

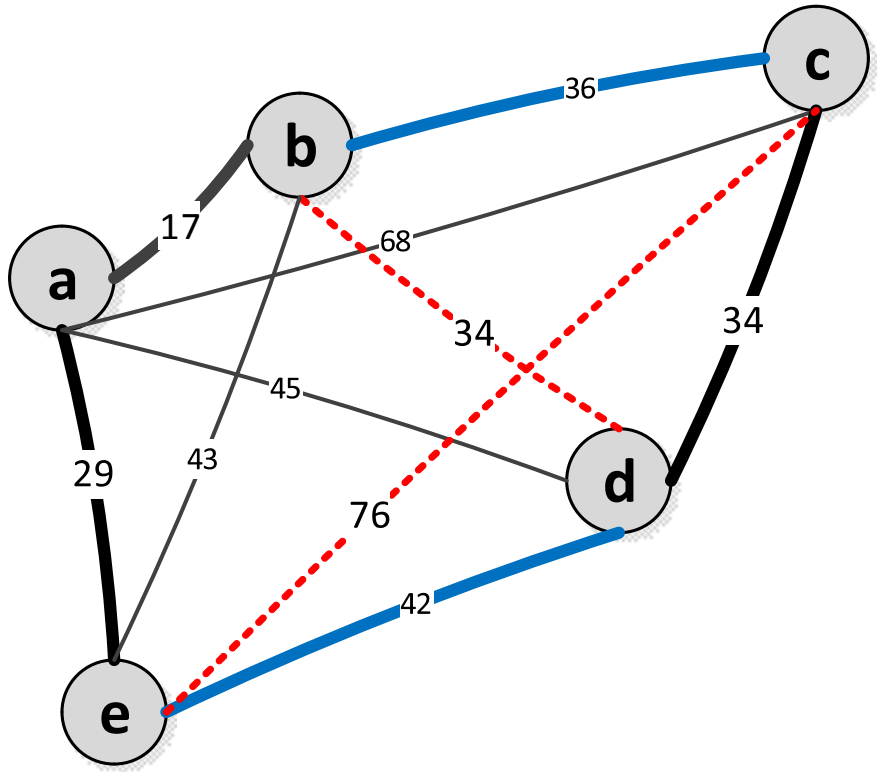
Custo: 17+34+34+76+29 = 190

TSP – Heurística de optimização 2-opt



- A existência de cruzamentos na solução garante que não é óptima
- A remoção dos cruzamentos optimiza obrigatoriamente a solução
- A heurística 2-opt remove pares de segmentos cruzados enquanto existirem cruzamentos na solução
- A complexidade temporal deste algoritmo é O(n^2)
- Não conduz obrigatoriamente a uma solução óptima

TSP – Heurística de optimização 2-opt



- Detecta-se um cruzamento entre os segmentos **b-d** e **c-e**
- Eliminar esses segmentos
- Arbitrariamente seleccionar um dos segmentos e a partir de um dos nós desse segmento seleccionar o nó mais próximo do outro segmento eliminado; criar as novas ligações
- Solução: [a,b,c,d,e,a]

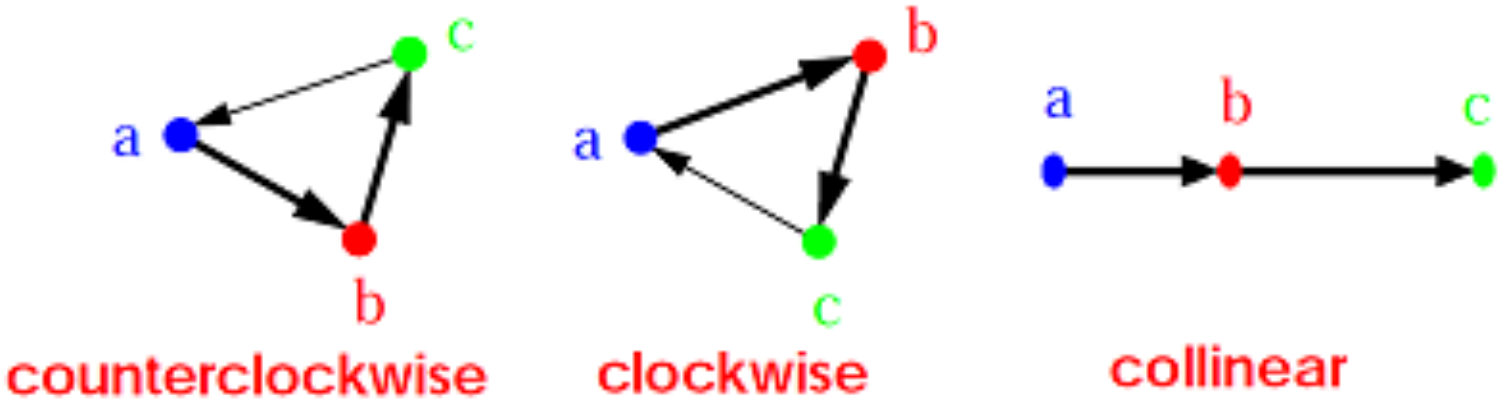
  Custo: 17+36+34+42+29 = 158

Determinação de segmentos cruzados

Dados 2 segmentos (p1, q1) e (p2, q2), é possível determinar se estes se interceptam.

A orientação de um tripleto de pontos pode ser:

sentido-ponteiro-relógio, sentido-contrário-ponteiro-relógio ou colinear.

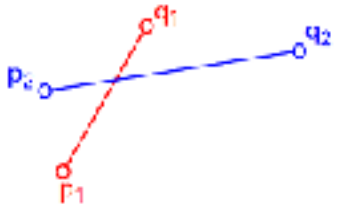


http://www.cdn.geeksforgeeks.org/check-if-two-given-line-segments-intersect/

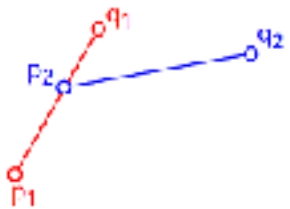
Determinação de segmentos cruzados

Dois segmentos (p1,q1) e (p2,q2) interceptam-se sse uma das seguintes condições se verifica.

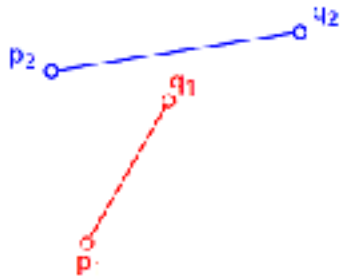
1. *Caso Geral:*
– (p1, q1, p2) e (p1, q1, q2) têm orientações diferentes e
– (p2, q2, p1) e (p2, q2, q1) têm orientações diferentes.



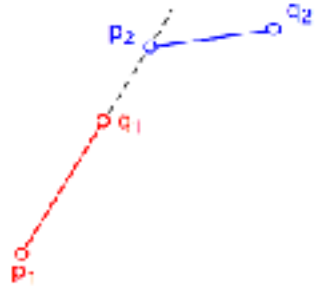
Example 1: Orientations of (p1, q1, p2) and (p1, q1, q2) are different. Orientations of (p2, q2, p1) and (p2, q2, q1) are also different



Example 2: Orientations of (p1, q1, p2) and (p1, q1, q2) are different. Orientations of (p2, q2, p1) and (p2, q2, q1) are also different



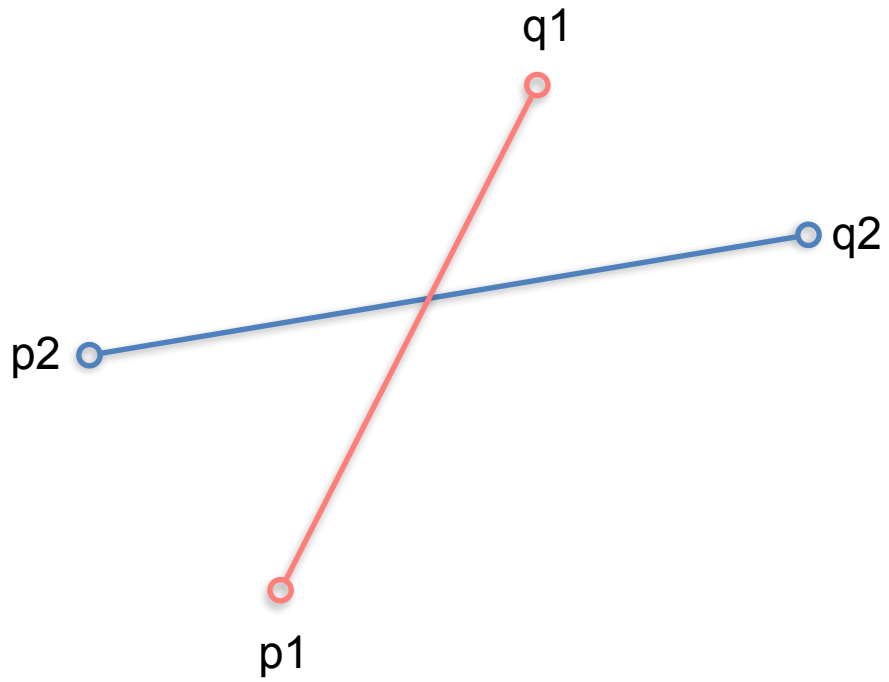
Example 3: Orientations of (p1, q1, p2) and (p1, q1, q2) are different. Orientations of (p2, q2, p1) and (p2, q2, q1) are same



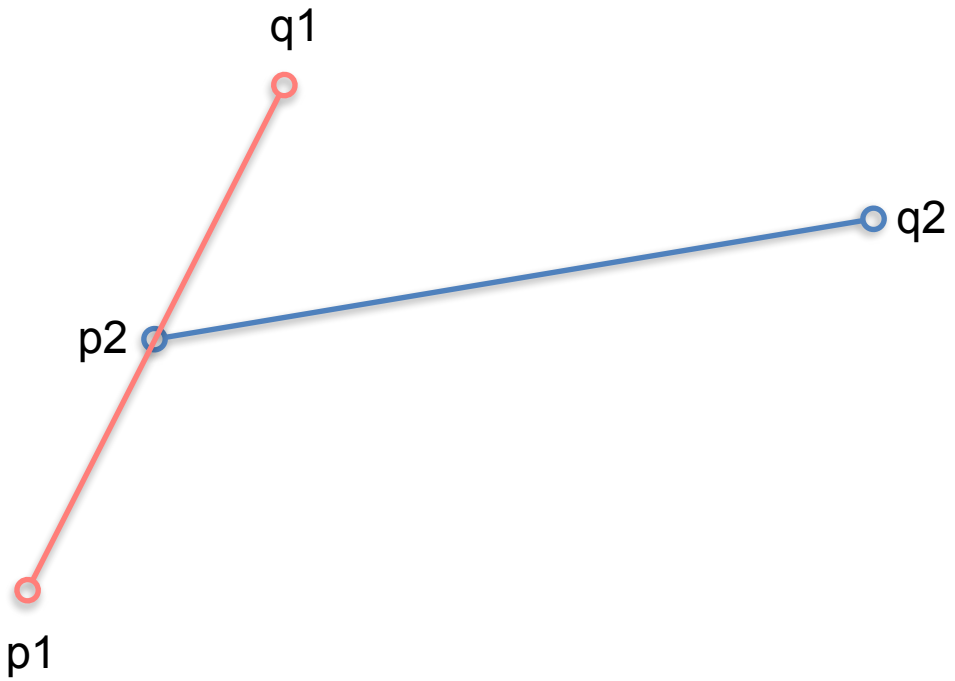
Example 4: Orientations of (p1, q1, p2) and (q1, q1, q2) are different. Orientations of (p2, q2, p1) and (p2, q2, q1) are same

http://www.cdn.geeksforgeeks.org/check-if-two-given-line-segments-intersect/

Determinação de segmentos cruzados

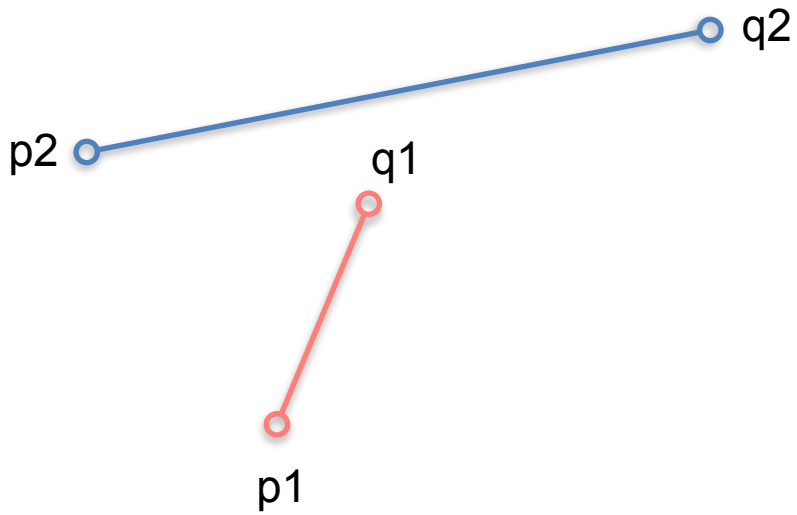


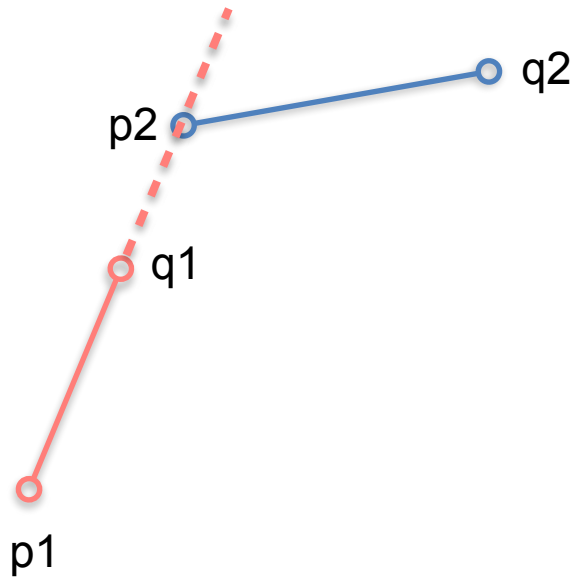
Segmentos cruzados



Orientações de
(**p1**,**q1**,**p2**) e (**p1**,**q1**,**q2**) diferentes
(**p2**,**q2**,**p1**) e (**p2**,**q2**,**q1**) diferentes

Determinação de segmentos cruzados





Segmentos não cruzados

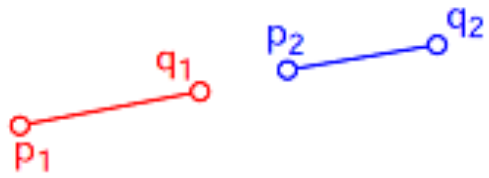
Orientações de
(**p1**,**q1**,**p2**) e (**p1**,**q1**,**q2**) diferentes
(**p2**,**q2**,**p1**) e (**p2**,**q2**,**q1**) iguais

Determinação de segmentos cruzados

Dois segmentos (p1,q1) e (p2,q2) interceptam-se sse uma das seguintes condições se verifica.

2. *Caso Particular*
– (p1, q1, p2), (p1, q1, q2), (p2, q2, p1), e (p2, q2, q1) são todos co-lineares e
– a projecções de x de (p1, q1) e (p2, q2) interceptam-se
– as projecções de y de (p1, q1) e (p2, q2) interceptam-se

Example 1: All points are colinear. The x-projections of (p1, q1) and (p2, q2) intersect. The y-projections of (p1, q1) and (p2, q2) intersect



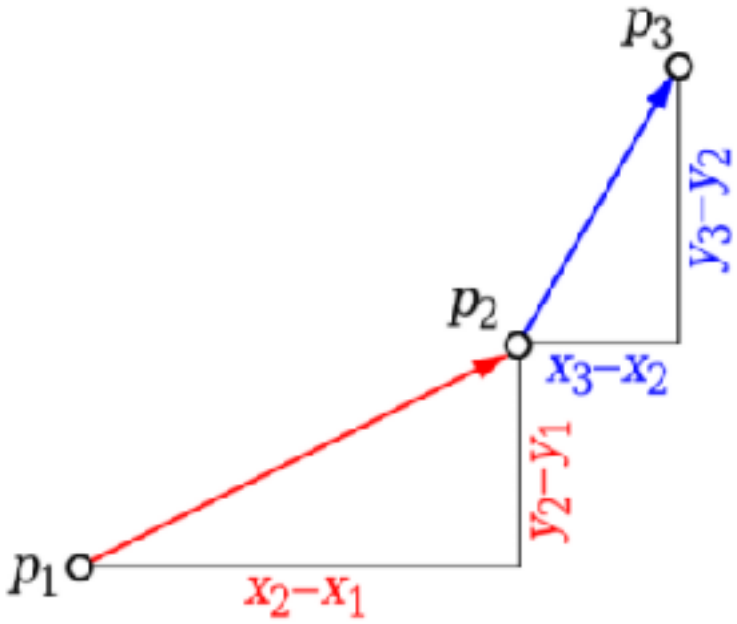
Example 2: All points are colinear. The x-projections of (p1, q1) and (p2, q2) do not intersect. The y-projections of (p1, q1) and do not (p2, q2) intersect

http://www.cdn.geeksforgeeks.org/check-if-two-given-line-segments-intersect/

Determinação da orientação de segmentos

Inclinação do segmento (p1, p2): σ = (y2 - y1)/(x2 - x1)
Inclinação do segmento (p2, p3): τ = (y3 - y2)/(x3 - x2)

σ < τ - orientação contrária ao sentido dos ponteiros do relógio
σ = τ - orientação colinear
σ > τ - orientação no sentido dos ponteiros do relógio



A orientação dos dois segmentos depende do sinal da expressão:

(y2 - y1)*(x3 - x2) - (y3 - y2)*(x2 - x1)

< 0 - σ < τ, i.e., contrária ao sentido dos ponteiros do relógio
0 - σ = τ, i.e., colinear
>0 - σ > τ, i.e., sentido dos ponteiros do relógio

in http://www.geeksforgeeks.org/orientation-3-ordered-points/

Determinação de segmentos cruzados

```
% To find orientation of ordered triplet (p, q, r).
% It returns following values:
% 0 --> p, q and r are colinear
% 1 --> Clockwise
% 2 --> Counterclockwise

orientation((PX,PY), (QX,QY), (RX,RY), Orientation):-
      Val is (QY - PY) * (RX - QX) - (QX - PX) * (RY - QY),
      (
            Val == 0, !, Orientation is 0;
            Val > 0, !, Orientation is 1;
            Orientation is 2
      ).
```

Código Prolog p/ determinar a intersecção de 2 segmentos pode ser encontrada no moodle da disciplina.

Determinação de segmentos cruzados

```
% Find the four orientations needed for general and special cases

orientation4cases(P1,Q1,P2,Q2,Or1,Or2,Or3,Or4):-
    orientation(P1, Q1, P2,Or1),
    orientation(P1, Q1, Q2,Or2),
    orientation(P2, Q2, P1,Or3),
    orientation(P2, Q2, Q1,Or4).

% Given 3 colinear points p, q, r, checks if point q lies on line segment 'pr'
% onSegment(P, Q, R)

onSegment((PX,PY), (QX,QY), (RX,RY)):-
    QX =< max(PX,RX),
    QX >= min(PX,RX),
    QY =< max(PY,RY),
    QY >= min(PY,RY).
```

```
doIntersect(P1,Q1,P2,Q2):-
    % Find the four orientations needed for general and special cases
    orientation4cases(P1,Q1,P2,Q2,Or1,Or2,Or3,Or4),
    (
    % General case
    Or1 \== Or2 , Or3 \== Or4,!;

    % Special Cases
    % p1, q1 and p2 are colinear and p2 lies on segment p1q1
    Or1 == 0, onSegment(P1, P2, Q1),!;

    % p1, q1 and p2 are colinear and q2 lies on segment p1q1
    Or2 == 0, onSegment(P1, Q2, Q1),!;

    % p2, q2 and p1 are colinear and p1 lies on segment p2q2
    Or3 == 0, onSegment(P2, P1, Q2),!;

     % p2, q2 and q1 are colinear and q1 lies on segment p2q2
    Or4 == 0, onSegment(P2, Q1, Q2),!
    ).
```