

A* Algorithmus

A*

Daniel
Mörtenschlag

Überblick

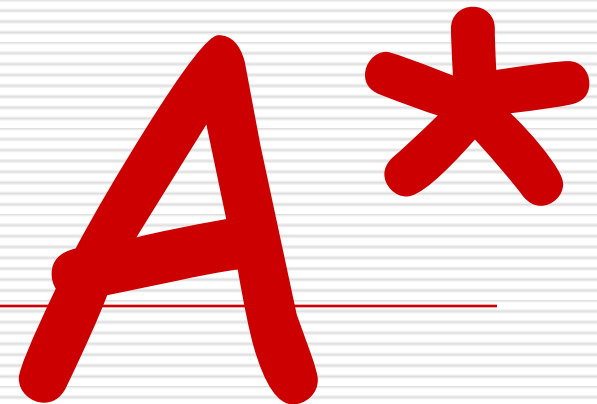
- Problemstellung
- Dijkstra
- A* Algorithmenvarianten
 - Standard Variante
 - IDA
- Problemstellungen
 - Pfadfindungsproblem
 - Schiebepuzzleproblem

A*

Überblick (2)

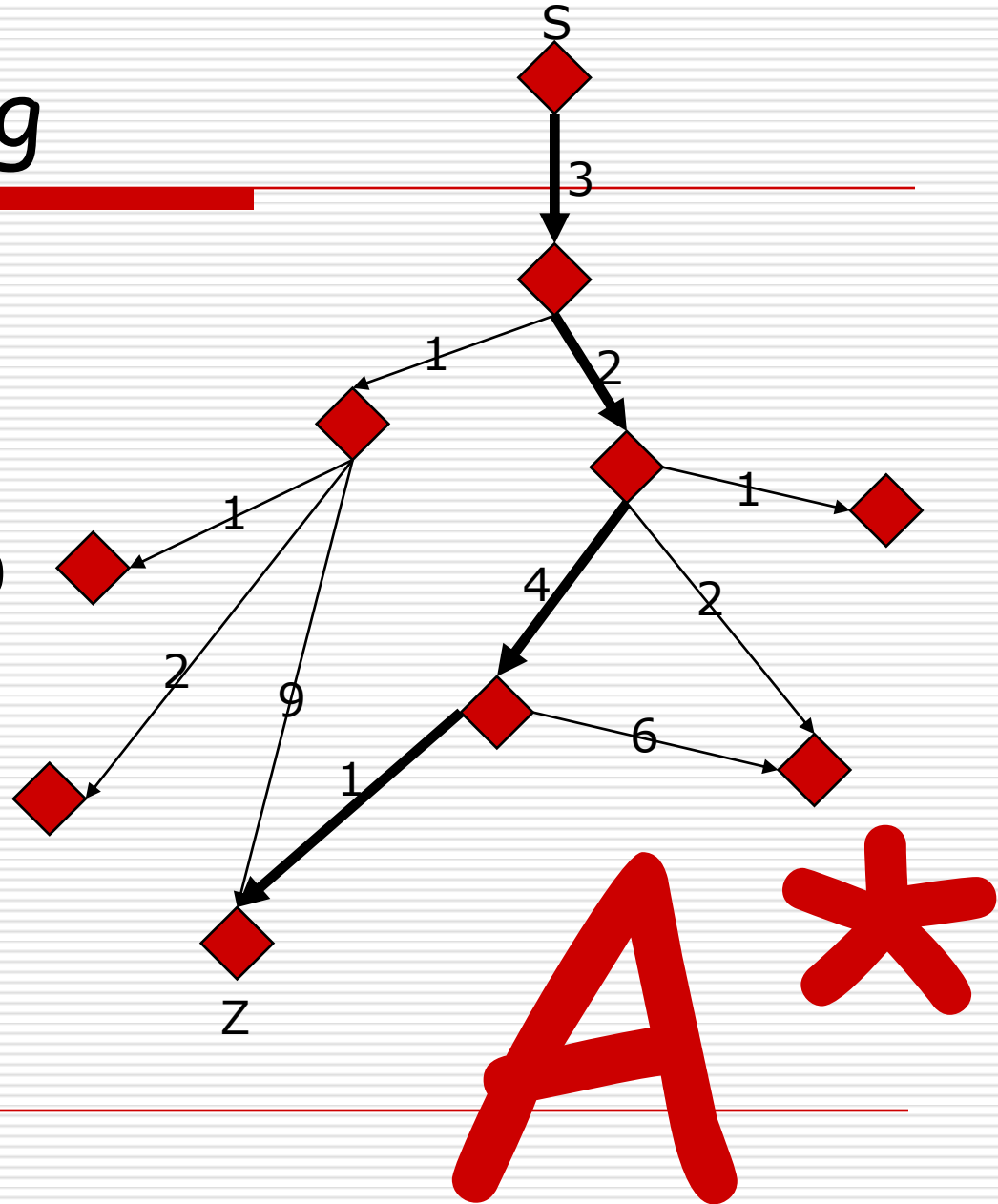
- Gesamtstruktur
- Verteilung auf HL Plugins
 - Algorithmen
 - Probleme
 - Graphendarstellung

- Praktische Vorführung



Problemstellung

- Gegeben
 - Graphen $G[N,E]$
 - Startecke s
 - Zielecke z
 - Alle Gewichte ≥ 0
 - Gesucht
 - Optimaler Kantenzug $s \rightarrow z$
- Kürzeste Wege Problem



Dijkstra

- Edsger W. Dijkstra
- Löst → kürzeste Wege Problem
- Greedy Algorithmus

- Eingabeparameter
 - Graph G
 - Startknoten $start$
 - Zielknoten $target$

A *

procedure Dijkstra (*G* : Graph; start, target : Vertex);

Var

predecessor : **table of** Vertex, Vertex;

D : **table of** Vertex, Real;

i, *j* : Vertex;

B : **set of** Vertex;

begin

initKw(start);

B := {start};

while *B* ≠ {} **do begin**

choose *i* ∈ *B* with *D*[*i*] minimum;

if (*i* = target) **then**

exit('target found');

B.delete(*i*);

for each neighbour *j* of *i* **do**

if $D[i] + B[i,j] < D[j]$ **then begin**

$D[j] := D[i] + B[i, j];$

predecessor [*j*] := *i*;

B.add(*j*);

end

end

end

Dijkstra (2)

S (0)	∞	∞	∞
∞	∞	∞	∞
∞	∞	∞	∞
∞	∞	∞	Z

A *

Dijkstra (3)

S (0)	1	∞	∞
1	1.41	∞	∞
∞	∞	∞	∞
∞	∞	∞	Z

A *

Dijkstra(4)

S (0)	1	2	∞
1	1.41	2.41	∞
∞	∞	∞	∞
∞	∞	∞	Z

A *

Dijkstra(5)

S (0)	1	2	∞
1	1.41	2.41	∞
2	2.41	∞	∞
∞	∞	∞	Z

A *

Dijkstra (6)

S (0)	1	2	∞
1	1.41	2.41	∞
2	2.41	2.48	∞
∞	∞	∞	Z

A*

Dijkstra (7)

S (0)	1	2	3
1	1.41	2.41	3.41
2	2.41	2.48	∞
∞	∞	∞	Z

A *

Dijkstra (8)

S (0)	1	2	3
1	1.41	2.41	3.41
2	2.41	2.48	∞
3	3.41	∞	Z

A *

Dijkstra (9)

S (0)	1	2	3
1	1.41	2.41	3.41
2	2.41	2.82	∞
3	3.41	3.82	Z

A *

Dijkstra (10)

S (0)	1	2	3
1	1.41	2.41	3.41
2	2.41	2.82	3.82
3	3.41	3.82	Z

A *

Dijkstra (11)

S (0)	1	2	3
1	1.41	2.41	3.41
2	2.41	2.82	3.82
3	3.41	3.82	Z (4.23)



A* Algorithmenvarianten

- Problem von Dijkstra
 - „Wissen über Graph wird ignoriert“
- „Verbesserung“ von Dijkstra
 - A* Algorithmen
- Peter Hart, Nils Nilson und Betram Raphael

A*

A* Algorithmenvarianten (2)

- Wissen über Graphen verwenden
- Schätzfunktion $f(n)$
 - Zielkosten nie überschätzen (konsistent)
 - $f(z) = 0 \wedge f(n) \geq 0$
- Nur Optimal zu z

A*

Standard Variante

- Erweiterung von Dijkstra
- Eingabeparameter
 - Graph G
 - Vertex start
 - Vertex target
 - Function f

A*

procedure A*-Search (*G* : Graph; *f* : Function; start, target : Vertex);

Var

predecessor : **table of** Vertex, Vertex;

D : **table of** Vertex, Real;

i, *j* : Vertex;

B : **set of** Vertex;

begin

initKw(start);

B := {start};

while *B* ≠ {} **do begin**

choose *i* ∈ *B* with *D*[*i*] + *f*(*i*) minimum;

if (*i* = target) **then**

exit('target found');

B.delete(*i*);

for each neighbour *j* of *i* **do**

if *D*[*i*] + *B*[*i*,*j*] < *D*[*j*] **then begin**

D[*j*] := *D*[*i*] + *B*[*i*, *j*];

predecessor [*j*] := *i*;

if not *b*.Contains(*i*) **then**

B.add(*j*);

end

end

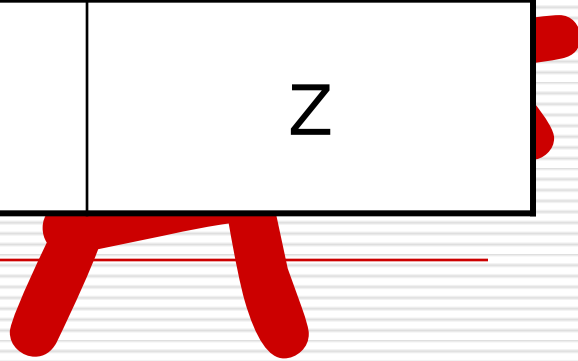
end

Standard Variante (2)

S (0+4.24)	∞	∞	∞
∞	∞	∞	∞
∞	∞	∞	∞
∞	∞	∞	Z


Standard Variante (3)

S (0+4.24)	1+3.6	∞	∞
1+3.6	1.41+2.83	∞	∞
∞	∞	∞	∞
∞	∞	∞	Z



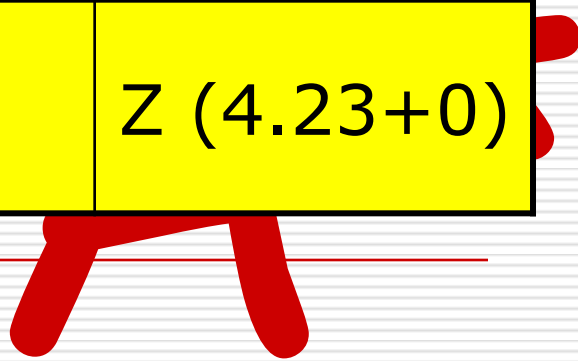
Standard Variante (4)

S (0+4.24)	1+3.6	2.82+3.16	∞
1+3.6	1.41+2.83	2.41+2.24	∞
2.82+3.16	2.41+2.24	2.82+1.41	∞
∞	∞	∞	Z



Standard Variante (5)

S (0+4.24)	1+3.6	2.82+3.16	∞
1+3.6	1.41	2.41+2.24	4.23+2
2.82+3.16	2.41+2.24	2.82	3.82+1
∞	4.23+2	3.82+1	Z (4.23+0)



Iterative Variante (IDA*)

- Erweiterung iterative Tiefensuche
- Geringer Speicherplatzbedarf

- Kostenschranke B

A*

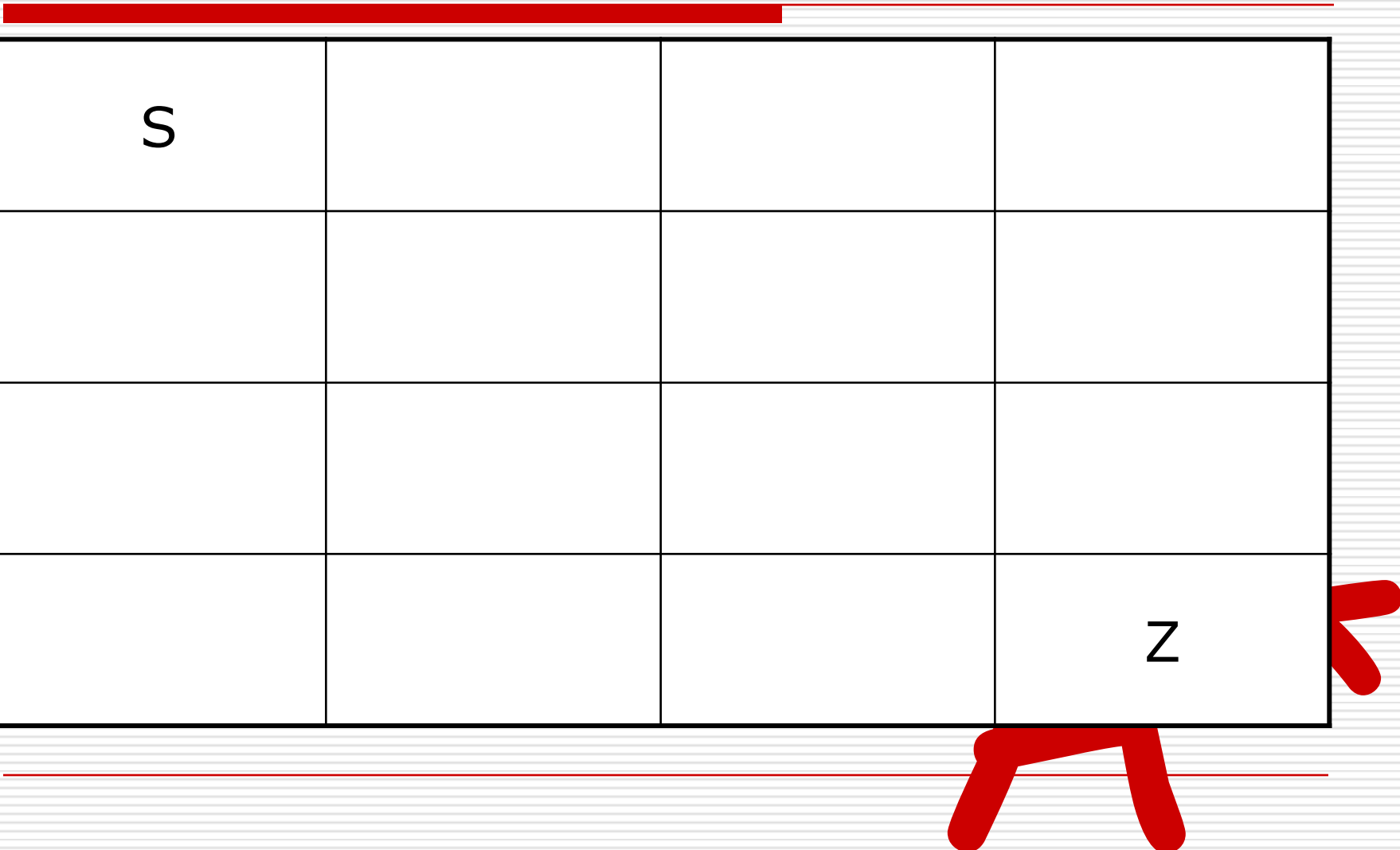
```

procedure IDA*-Search (G : Graph; start, target : Vertex);
var j : Integer; B, BNew, b : Real;
    e : Entry; S : Stack of Entry;
    found : boolean;
begin
    if start = target then exit('start and target are equal');
    B := f(start); found = false;
    while not found and B <  $\infty$  do begin
        S.push(new Entry(start,0)); BNew :=  $\infty$ ;
        while not found and S  $\neq$  {} do begin
            e:=S.head;
            j := nextNeighbour(e.vertex);
            if j  $\neq$  0 then begin
                if j= target and e.value + value(e.vertex, j)  $\leq$  B then begin
                    found:= true;
                    output(j);
                    while S  $\neq$  {} do output (S.pop.vertex);
                end
                else begin
                    b := f(j) + e.value + value(e.vertex, j);
                    if b > B then BNew := min { b, BNew }
                    else S.push(new Entry(j, e.value + value(e.vertex,j)));
                end
            end
        end
        else S.pop;
    end
    B:= BNew;
end
    if not found then output ('Target not found');
end

```

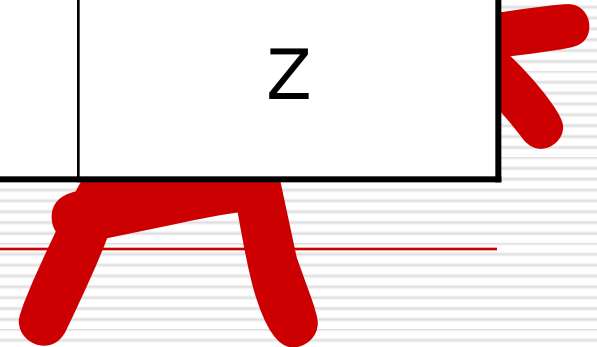
Iterative Variante (IDA*) (2)

S			
			Z



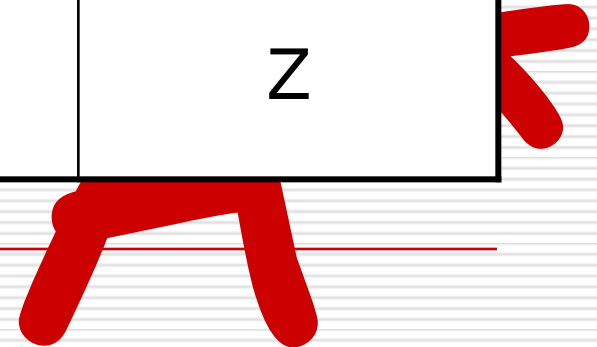
Iterative Variante (IDA*) (3)

S	1+3.6		B = 4.24
			Z



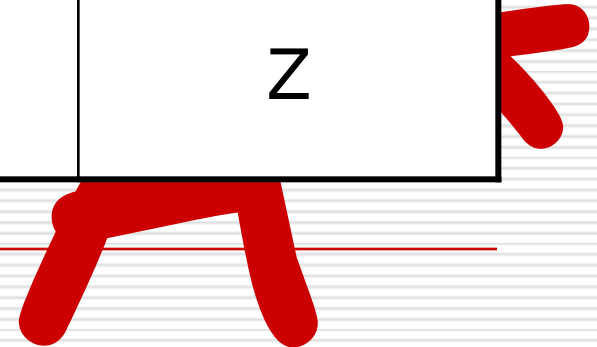
Iterative Variante (IDA*) (4)

S			B = 4.24
	1.41+2.82		
			Z



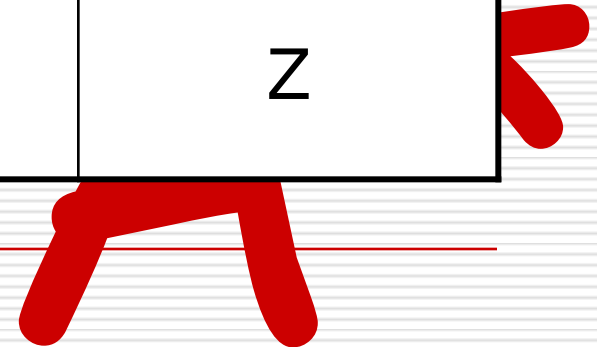
Iterative Variante (IDA*) (5)

S	2.41+3.6		B = 4.24
	1.41		
			Z



Iterative Variante (IDA*) (6)

S		2.82+3.16	B = 4.24
	1.41		
			Z

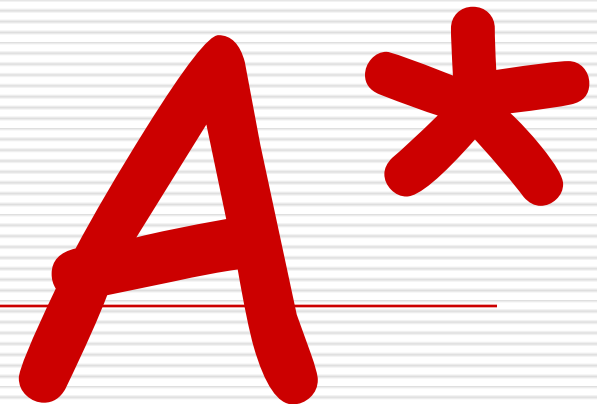


Problemstellungen

□ Immer ein kürzestes Wege Problem

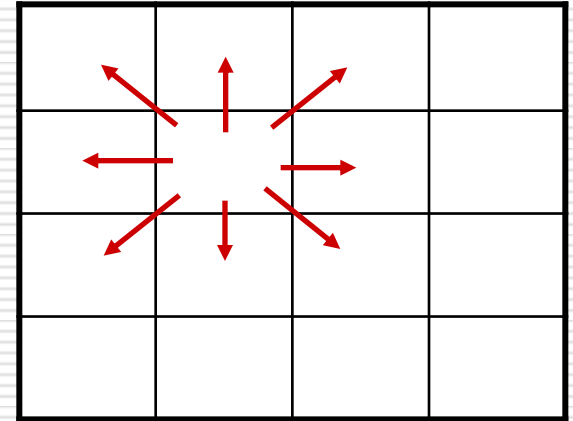
- Graphen $G[N,E]$
- Startecke s
- Zielecke z
- Schätzfunktion $f(n)$

□ Basisklasse: ProblemBase



Pfadfindungsproblem

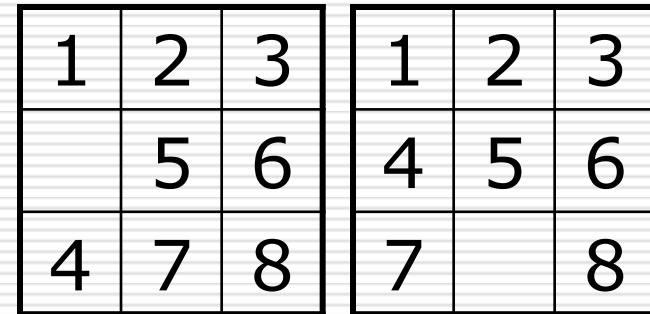
- Knoten = Position
 - X/Y Koordinaten
- Kante = Übergänge zw. Positionen
- Schätzfunktionen
 - Euklid
 - Manhattan
- Plugin: PFP



Schiebepuzzleproblem

- Zustandsraumproblem
- Knoten = Puzzlezustand
- Kante = Schiebebewegung
- Schätzfunktion
 - Falsche Blöcke
 - Summierter Manhattanabstand
- Plugin: TPP

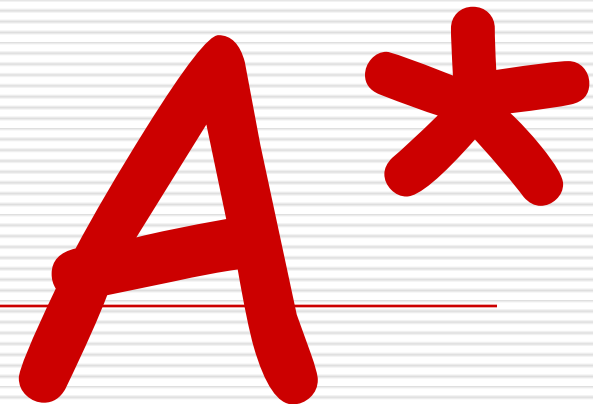
1	2	3
4	5	6
	7	8



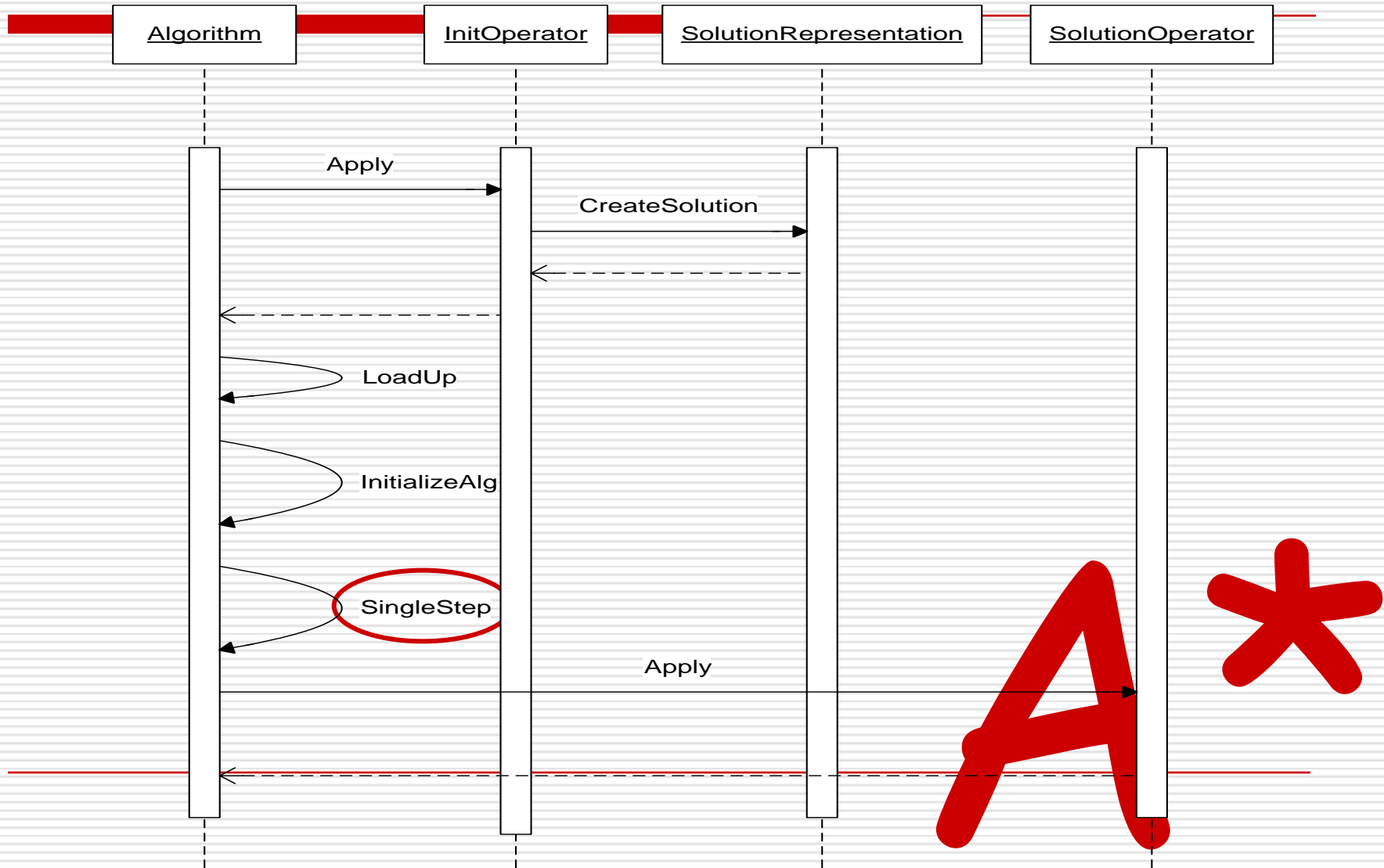
A*

Gesamtstruktur

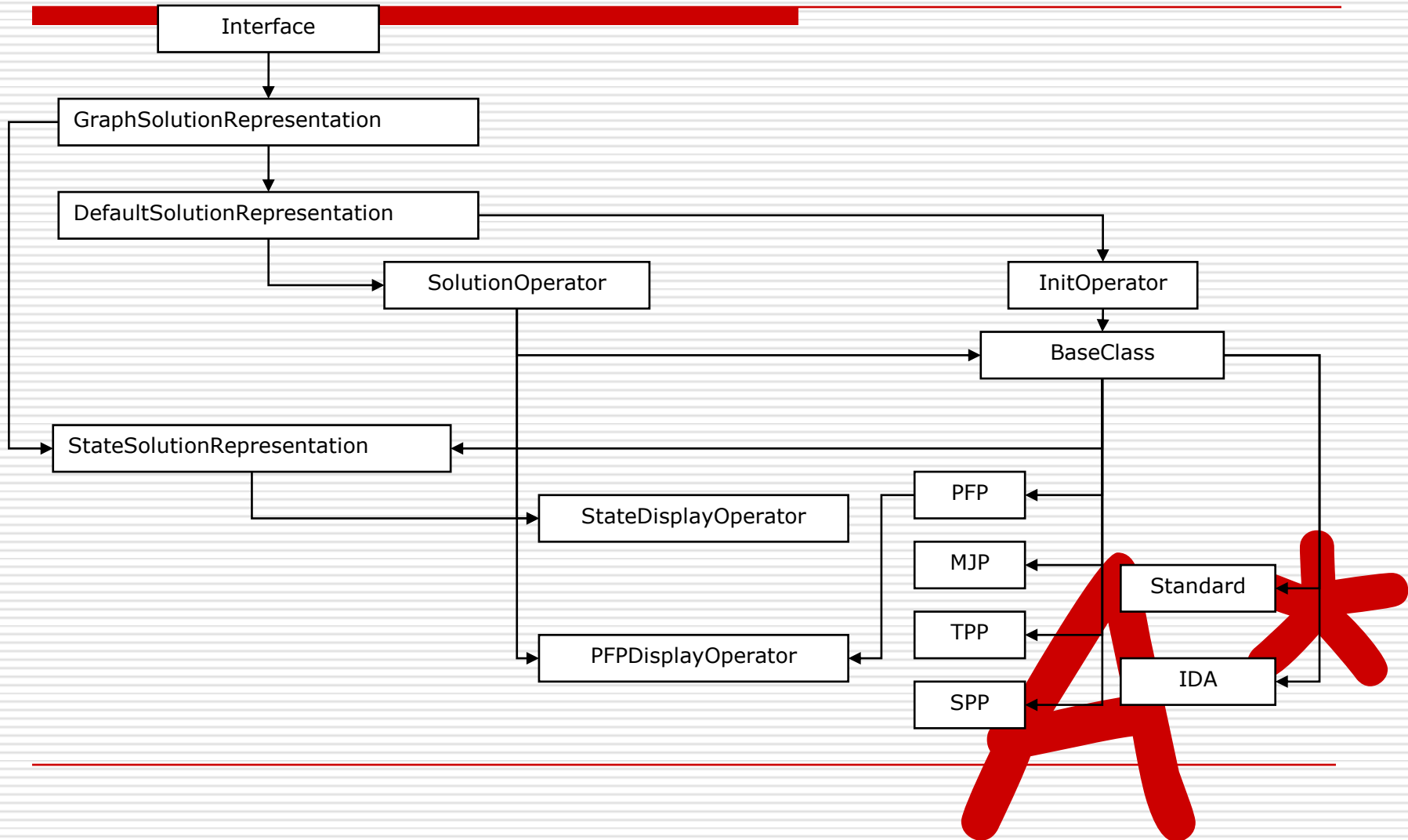
- SolutionRepresentation
 - Erzeugt Lösungen
- Problem
 - Startknoten
 - Finished
 - Graphen
 - Schätzfunktionen
- Algorithmen
 - InitOperator
 - SingleStep
 - SolutionOperator



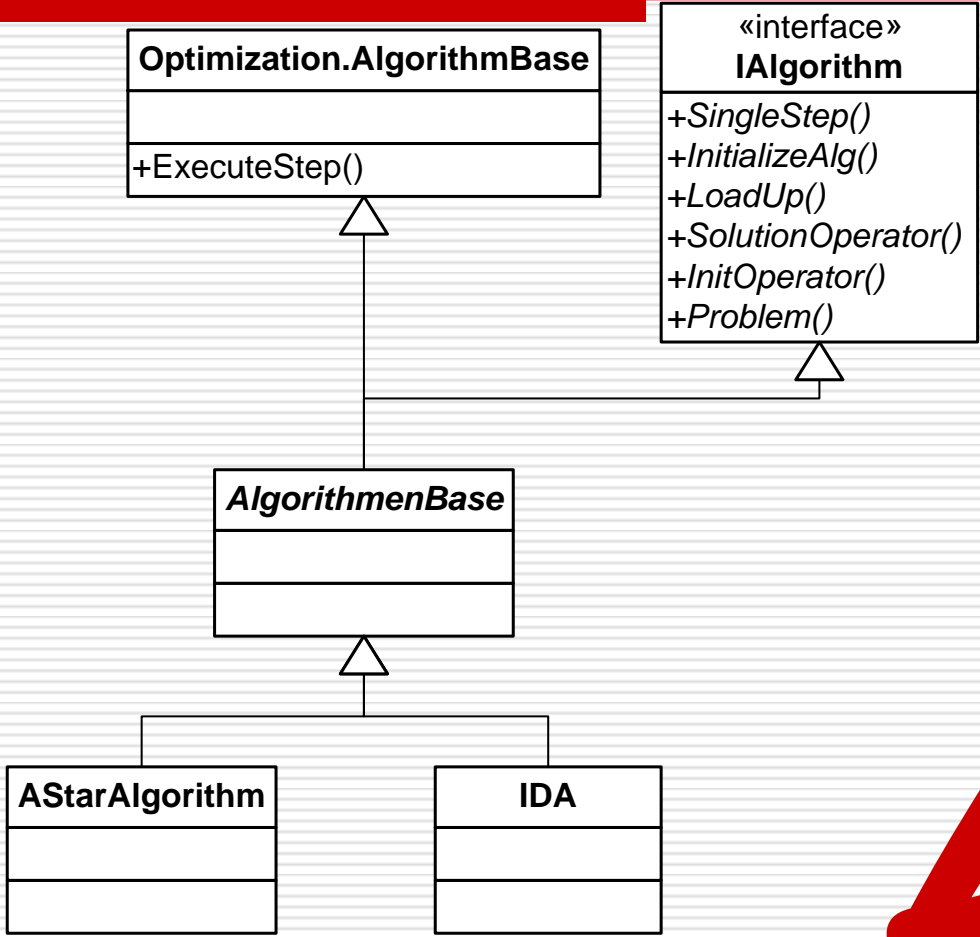
Gesamtstruktur (2)



Verteilung auf HL Plugins

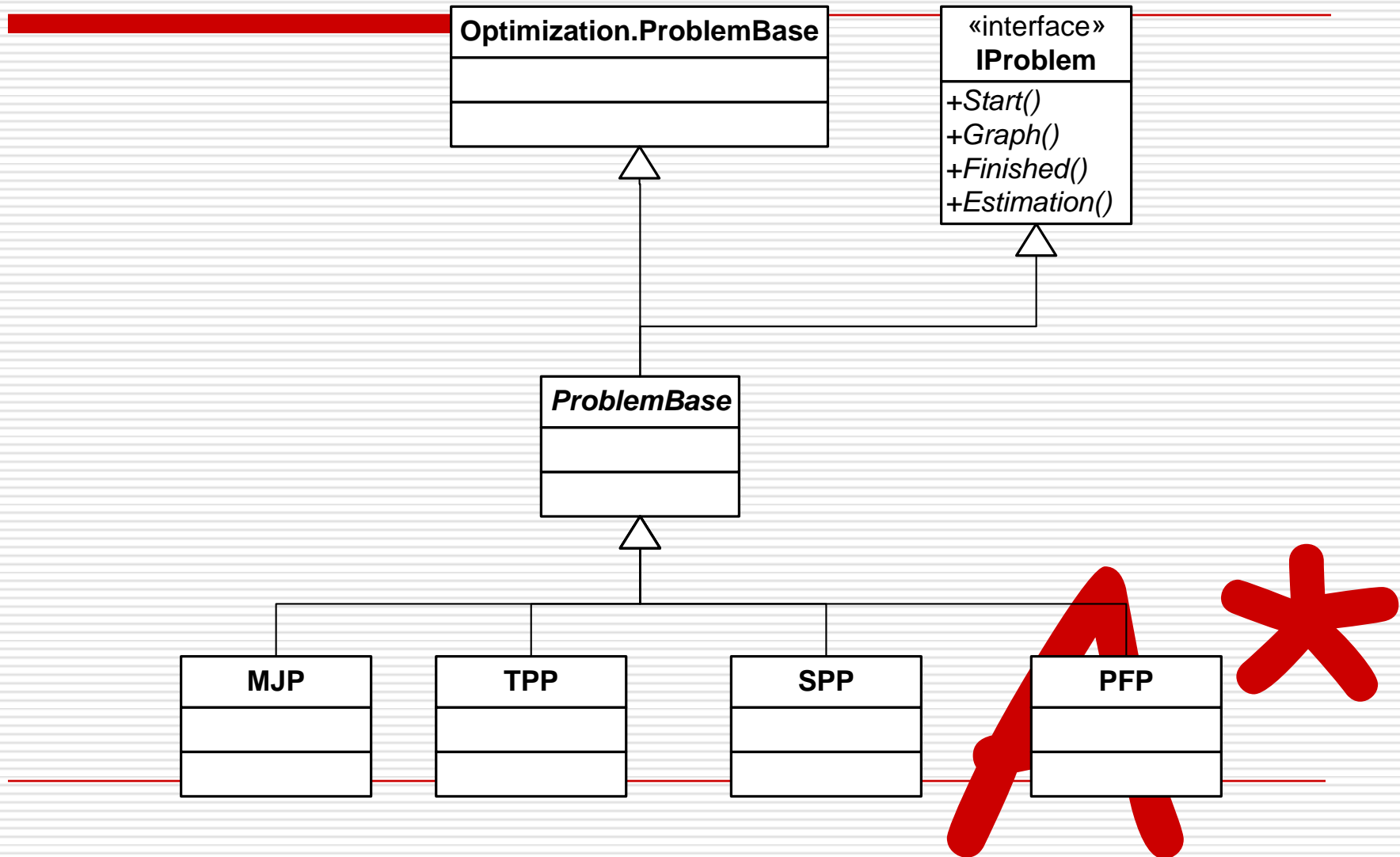


Algorithmen

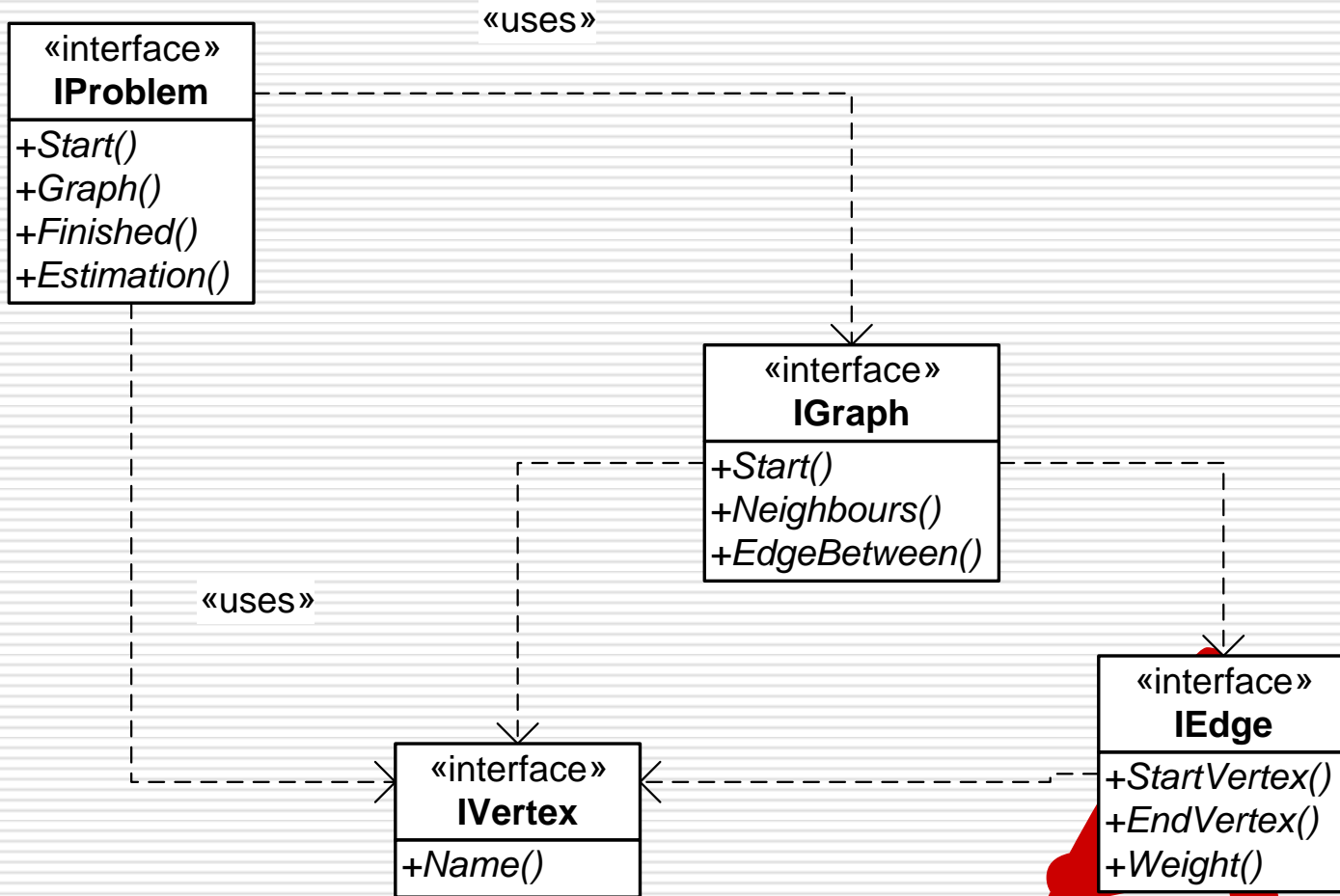


A *

Probleme



Graphendarstellung



Praktische Vorführung

A *