Presentation of the paper

Common Misconceptions Concerning Heuristic Search

by Robert C. Holte

Claudio Marxer

Seminar "Search and Optimization"

HS 2013

Statements

1. More accurate heuristics result in fewer node expansions by A*.
2. A* does fewer node expansions than any other equally informed algorithm that finds optimal solutions.
3. Any admissible heuristic can be turned into a consistent heuristic by a simple technique called pathmax.
4. In search spaces whose operators all have the same cost, A* with the constant heuristic function $h = 0$ is the same as breadth-first search.

Theory: Properties of Heuristic Functions

Definition: Let $\mathcal{S} = \langle S, A, \mathrm{cost}, T, s_0, S_\star \rangle$ be a state space. A **heuristic function** over S is of the following form:

$$h \colon S \to \mathbb{R}_0^+$$

[1] $C(n)$ denotes the actual cost of the cheapest path from n to a goal node.
[2] $\mathrm{cost}(s, t)$ denotes the cost of the cheapest path between s and t.

Theory: Properties of Heuristic Functions

Definition: Let $\mathcal{S} = \langle S, A, \mathrm{cost}, T, s_0, S_\star \rangle$ be a state space. A **heuristic function** over S is of the following form:

$$h \colon S \to \mathbb{R}_0^+$$

Definition: Let h be a heuristic function with $h = 0$ for all goal states.

[1] $C(n)$ denotes the actual cost of the cheapest path from n to a goal node.
[2] $\mathrm{cost}(s, t)$ denotes the cost of the cheapest path between s and t.

Theory: Properties of Heuristic Functions

Definition: Let $\mathcal{S} = \langle S, A, \mathrm{cost}, T, s_0, S_\star \rangle$ be a state space. A **heuristic function** over S is of the following form:

$$h \colon S \to \mathbb{R}_0^+$$

Definition: Let h be a heuristic function with $h = 0$ for all goal states.

- h **admissible** if for all states n:
  $h(n) \leq C(n)$ [1]

[1] $C(n)$ denotes the actual cost of the cheapest path from n to a goal node.
[2] $\mathrm{cost}(s, t)$ denotes the cost of the cheapest path between s and t.

Theory: Properties of Heuristic Functions

Definition: Let $\mathcal{S} = \langle S, A, \mathrm{cost}, T, s_0, S_\star \rangle$ be a state space. A **heuristic function** over S is of the following form:

$$h \colon S \to \mathbb{R}_0^+$$

Definition: Let h be a heuristic function with $h = 0$ for all goal states.

- h **admissible** if for all states n:
  $h(n) \leq C(n)$ [1]
- h **consistent** if for all states s and t:
  $h(s) \leq \mathrm{cost}(s, t) + h(t)$ [2]

[1] $C(n)$ denotes the actual cost of the cheapest path from n to a goal node.
[2] $\mathrm{cost}(s, t)$ denotes the cost of the cheapest path between s and t.

Theory: Properties of Heuristic Functions

Definition: Let $\mathcal{S} = \langle S, A, \mathrm{cost}, T, s_0, S_\star \rangle$ be a state space. A **heuristic function** over S is of the following form:

$$h \colon S \to \mathbb{R}_0^+$$

Definition: Let h be a heuristic function with $h = 0$ for all goal states.

- h **admissible** if for all states n:
  $h(n) \leq C(n)$ [1]
- h **consistent** if for all states s and t:
  $h(s) \leq \mathrm{cost}(s, t) + h(t)$ [2]

h is consistent $\Rightarrow$ h is admissible

[1] $C(n)$ denotes the actual cost of the cheapest path from n to a goal node.
[2] $\mathrm{cost}(s, t)$ denotes the cost of the cheapest path between s and t.

Statement 1: Explanation

More accurate heuristics result in fewer node expansions by A*.

Statement 1: Explanation

More accurate heuristics result in fewer node expansions by A*.

What does more accurate mean?

Statement 1: Explanation

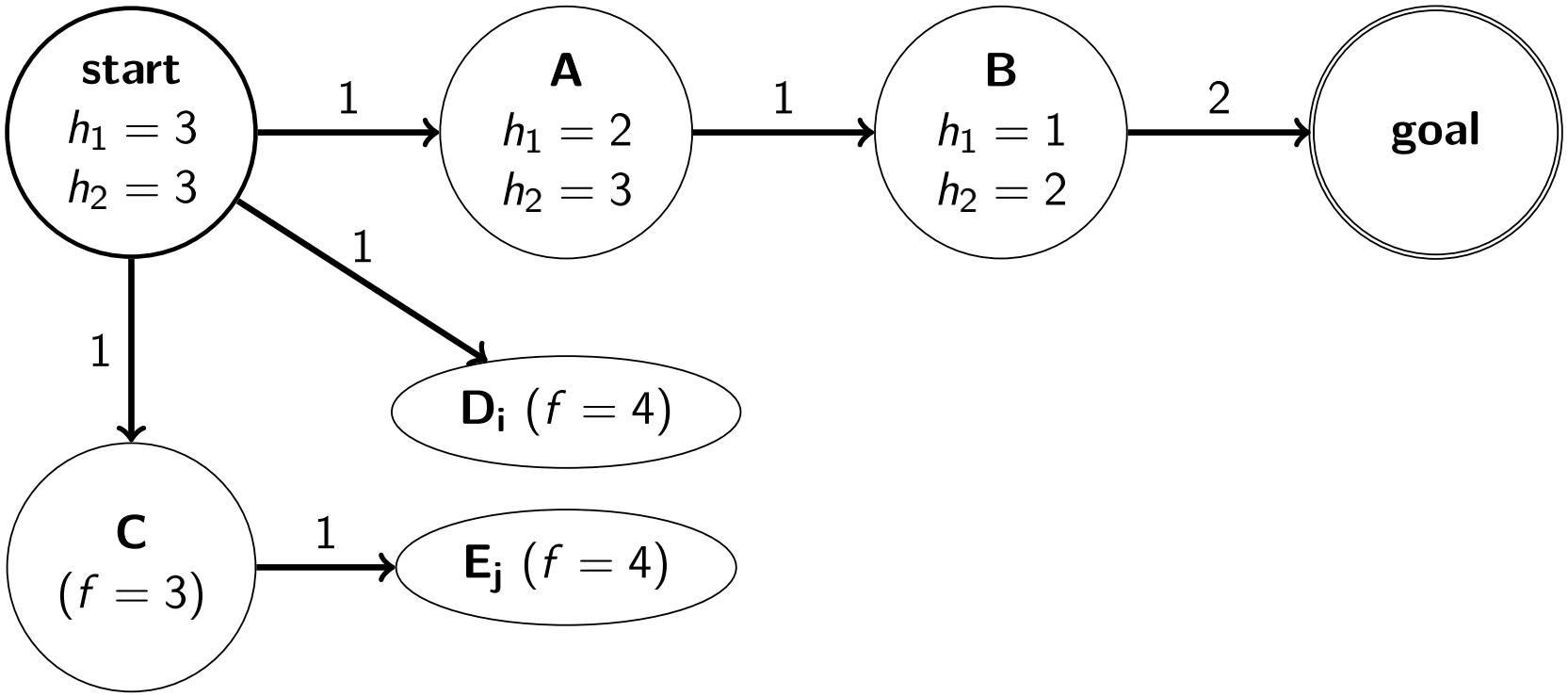
More accurate heuristics result in fewer node expansions by A*.

What does more accurate mean?

Definition: Let h_1 and h_2 be two admissible heuristic functions. h_2 is called **"better than"** h_1 if the following two conditions hold:

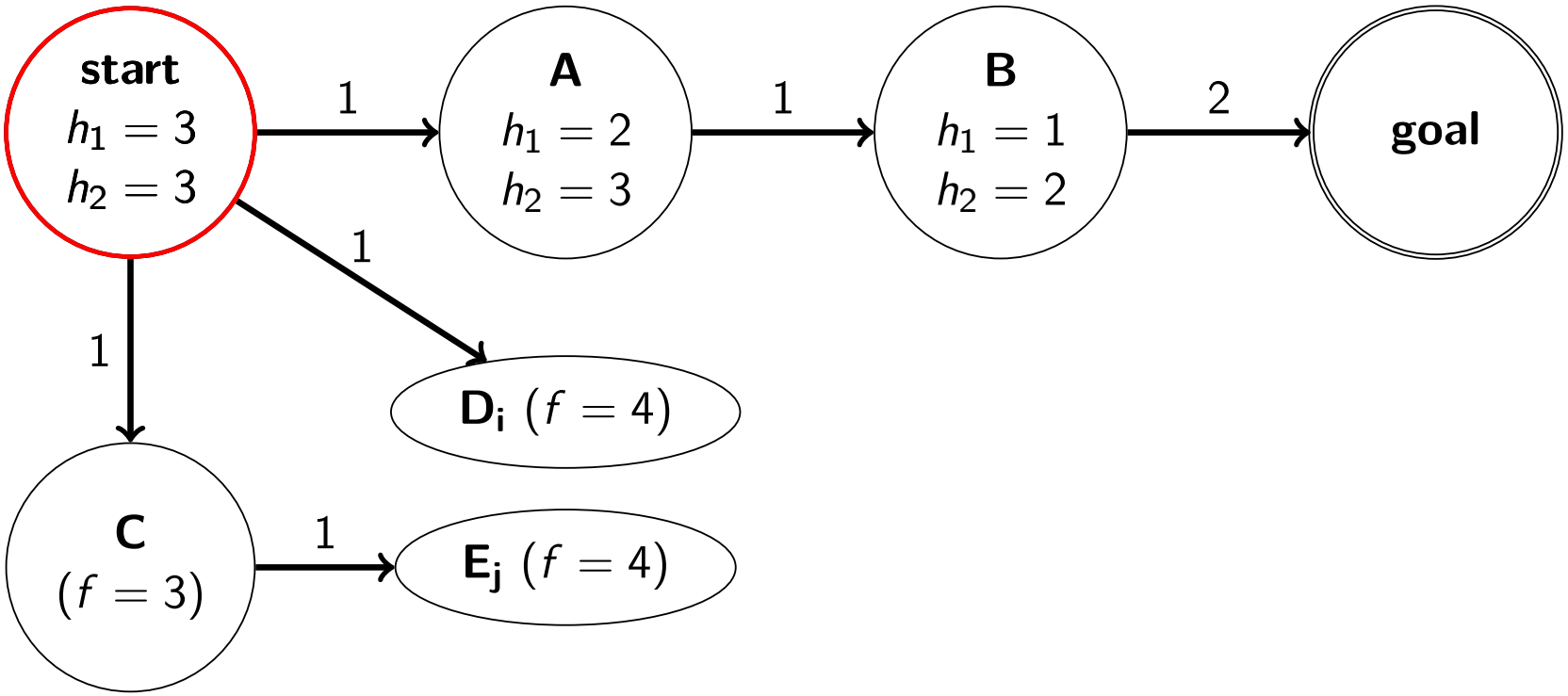
- $\forall s \in S: h_1(s) \leq h_2(s)$
- $\exists s \in S: h_1(s) < h_2(s)$

Statement 1: Counterexample



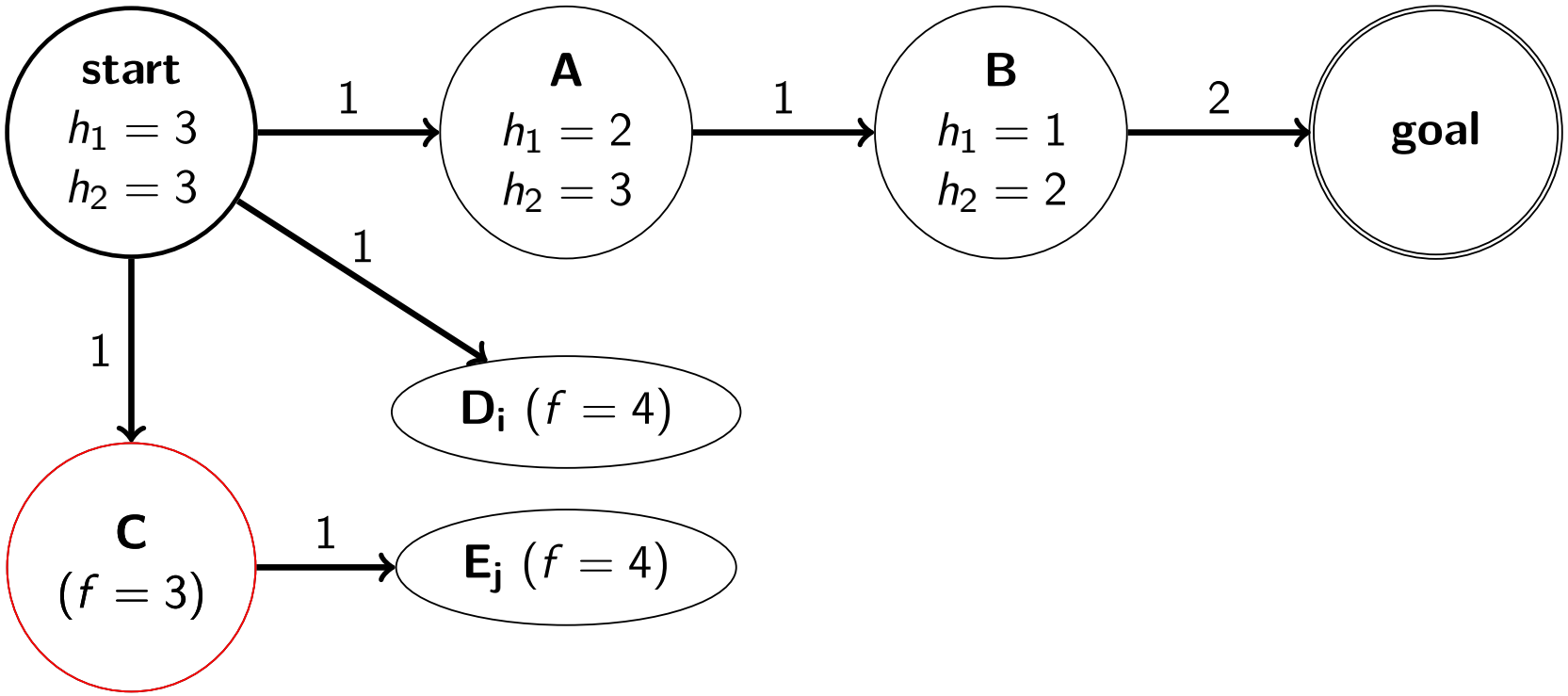
Statement 1: Counterexample

Performing A* with the dominated heuristic function h_1.



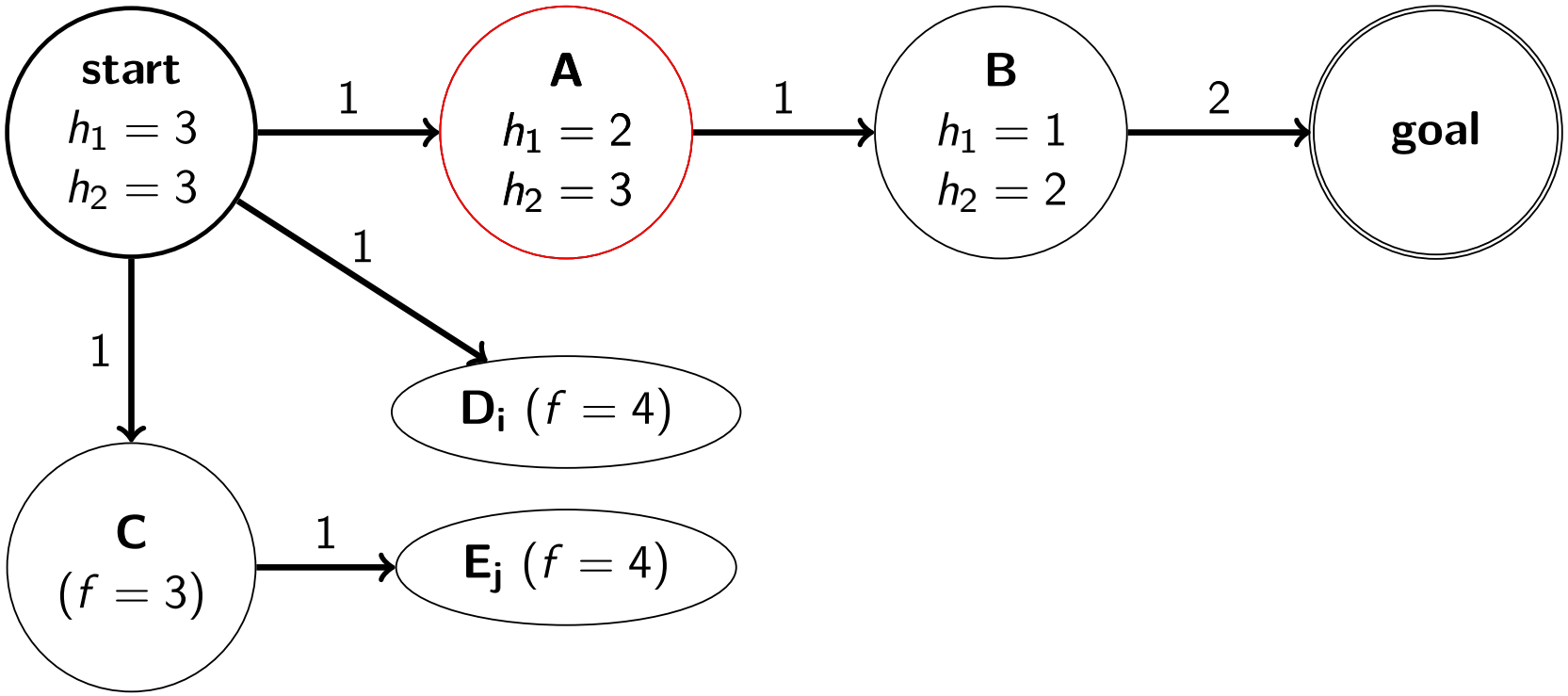
Statement 1: Counterexample

Performing A* with the dominated heuristic function h_1.



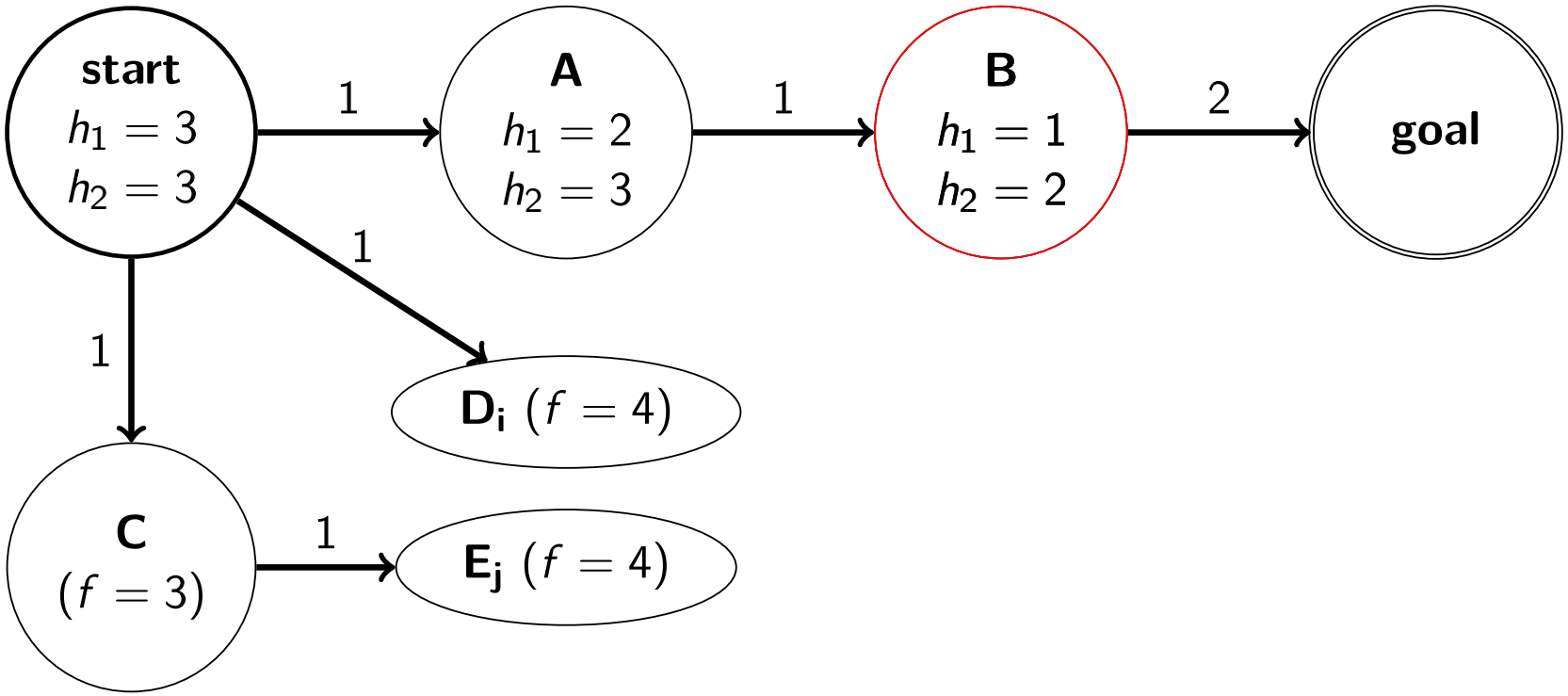
Statement 1: Counterexample

Performing A* with the dominated heuristic function h_1.



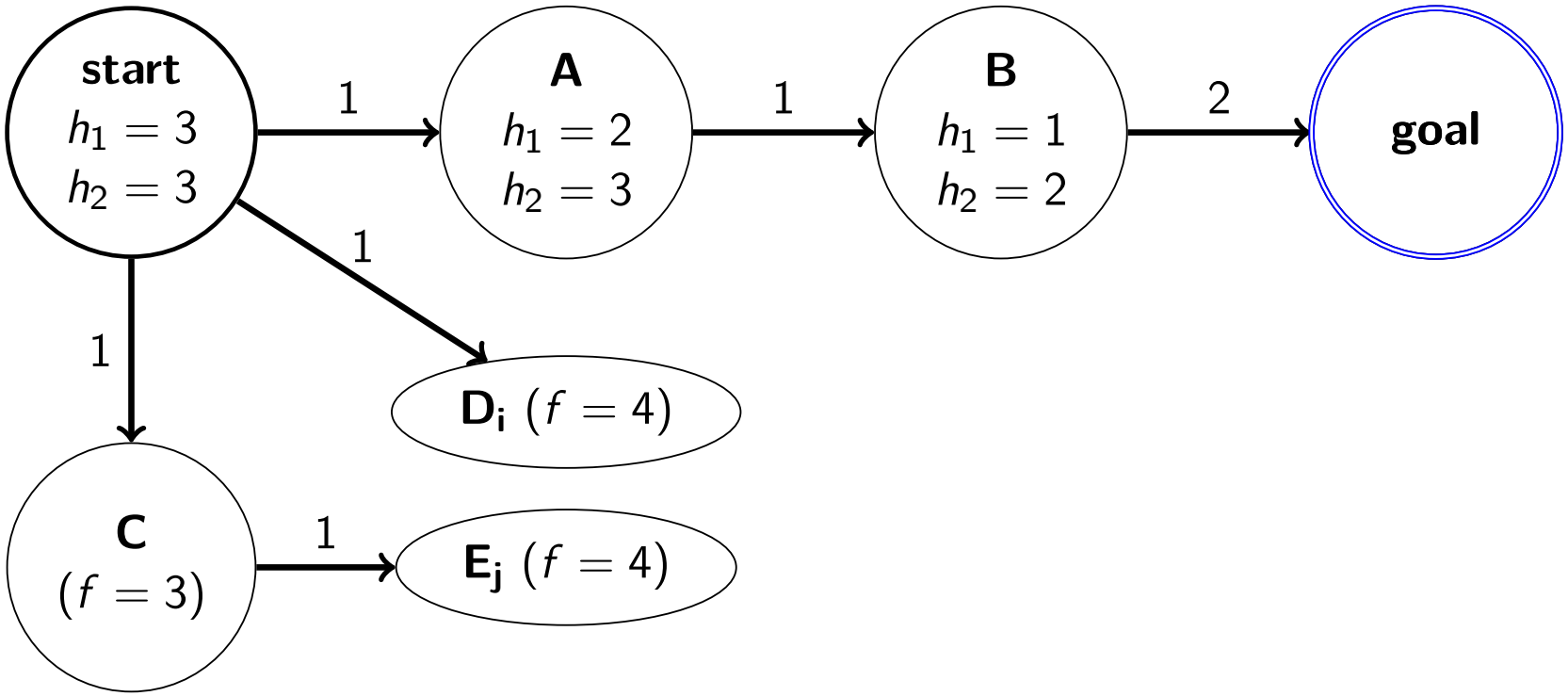
Statement 1: Counterexample

Performing A* with the dominated heuristic function h_1.



Statement 1: Counterexample

Performing A* with the dominated heuristic function h_1.

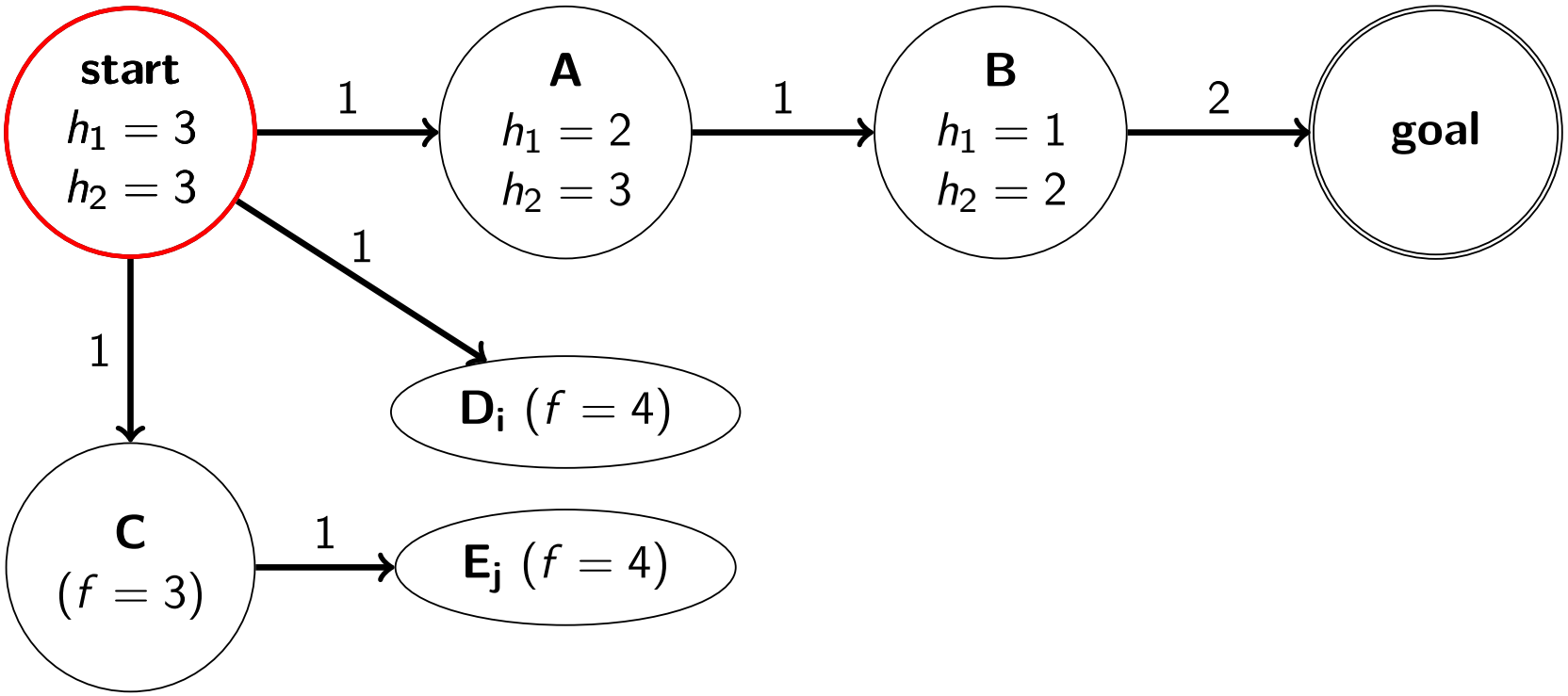


Worst case: 4 node expansions.

Best case: 3 node expansions.

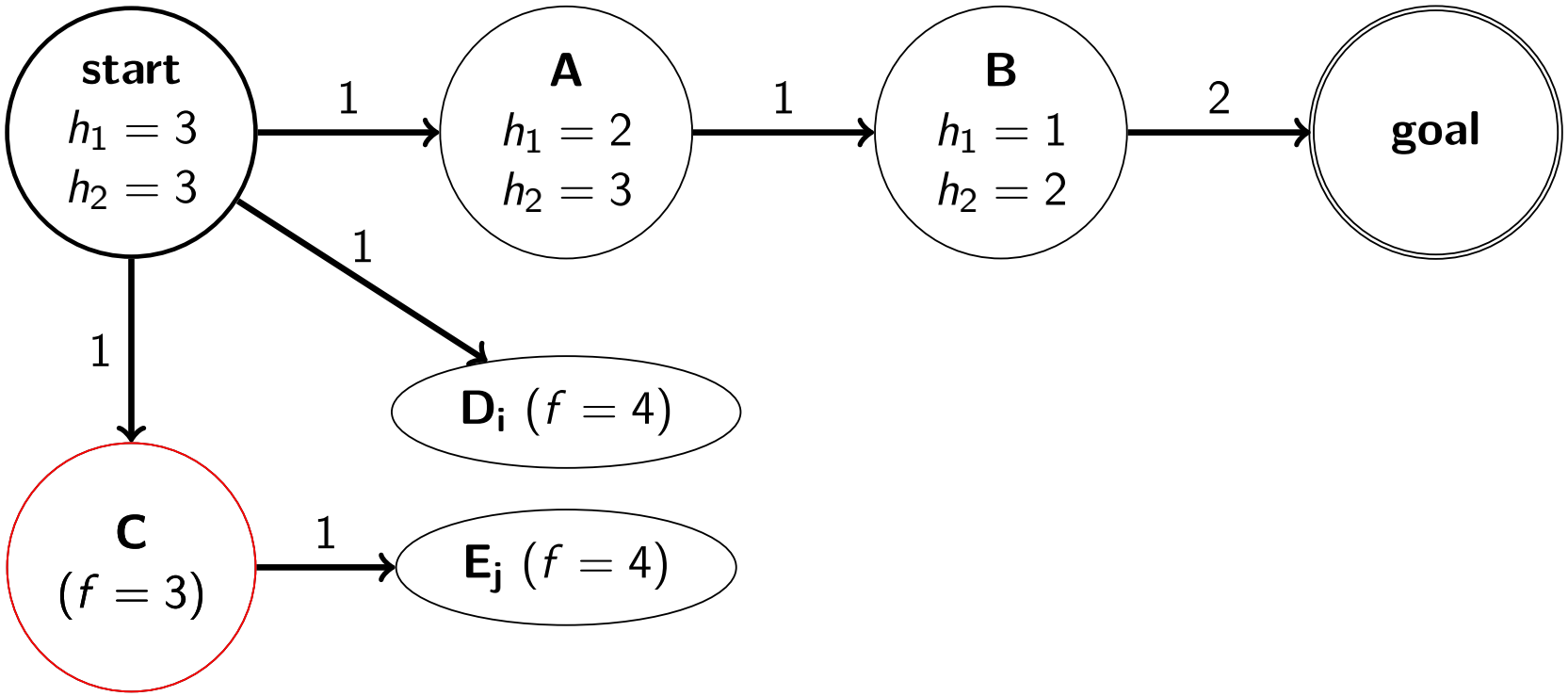
Statement 1: Counterexample

Performing A* with the better heuristic function h_2.



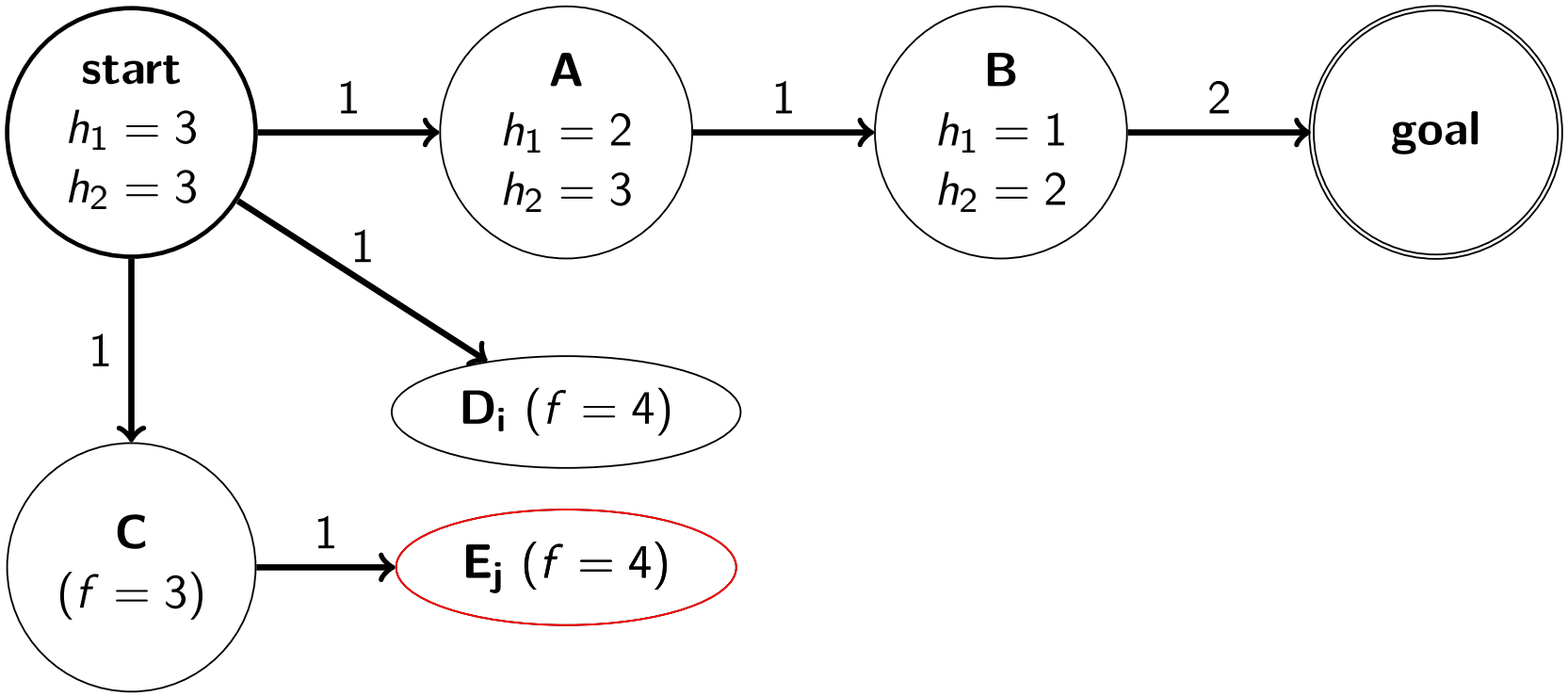
Statement 1: Counterexample

Performing A* with the better heuristic function h_2.



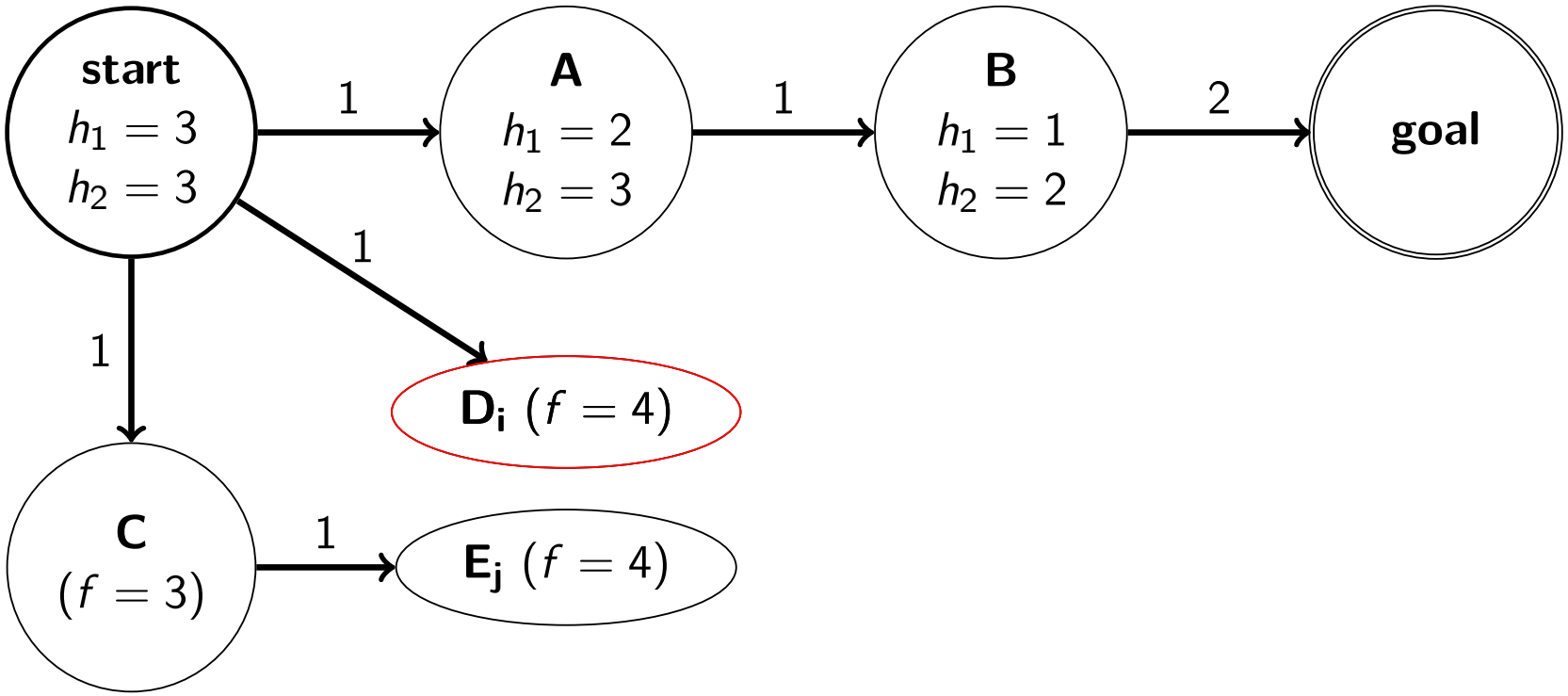
Statement 1: Counterexample

Performing A* with the better heuristic function h_2.



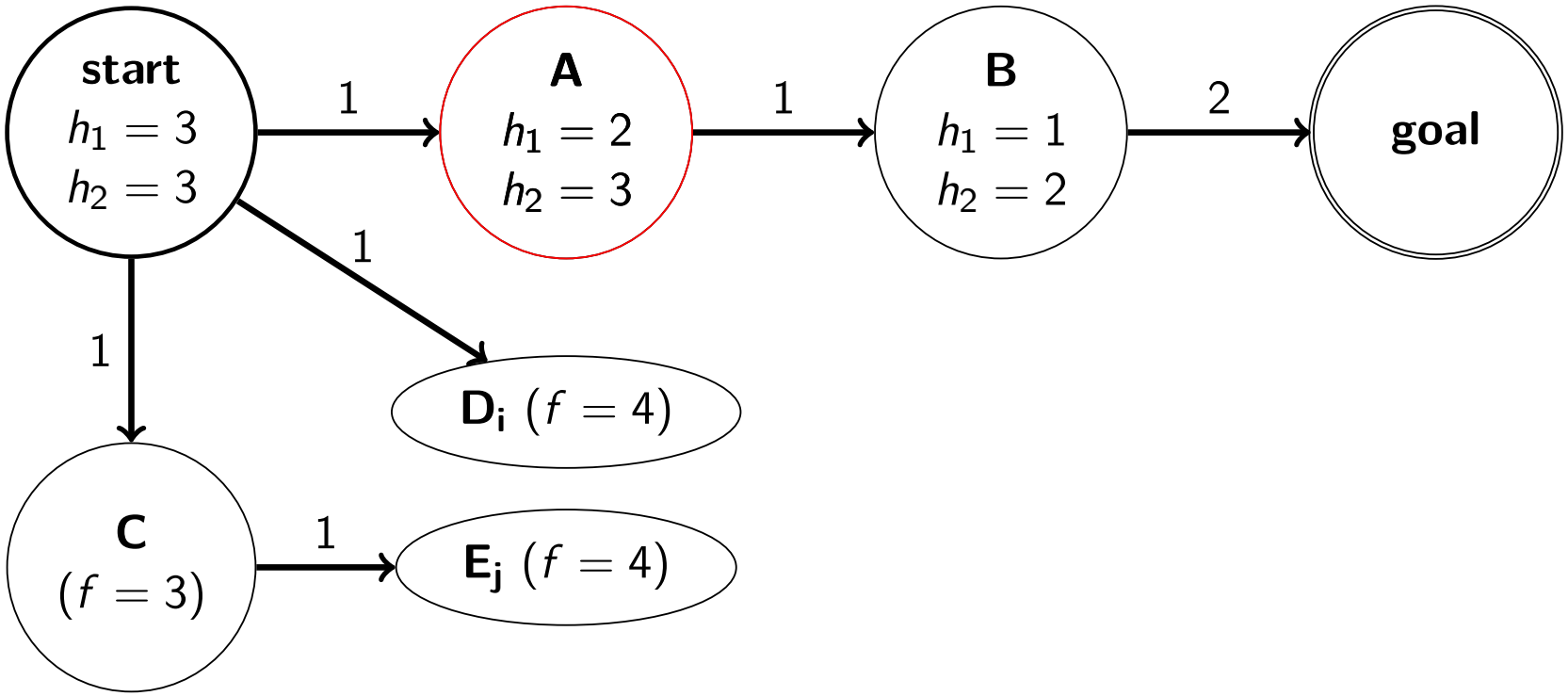
Statement 1: Counterexample

Performing A* with the better heuristic function h_2.



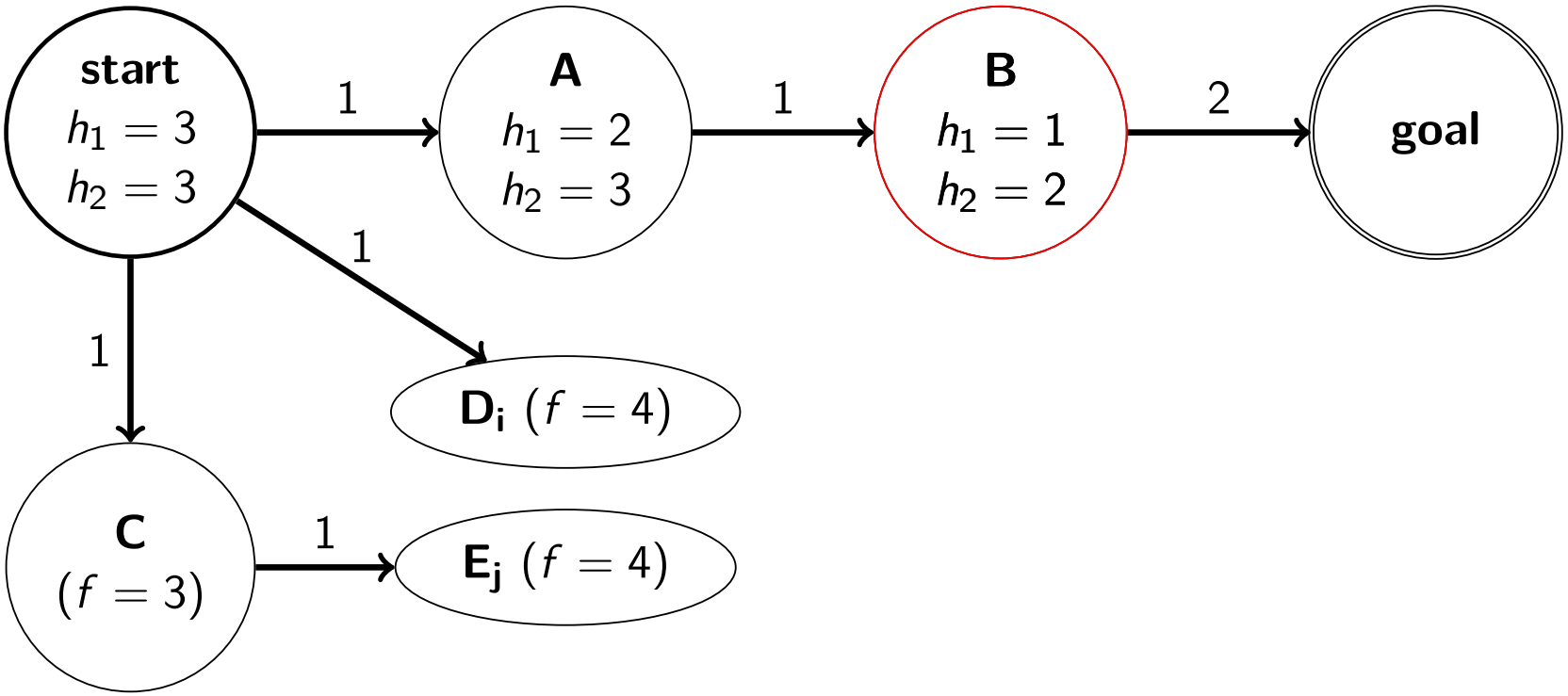
Statement 1: Counterexample

Performing A* with the better heuristic function h_2.



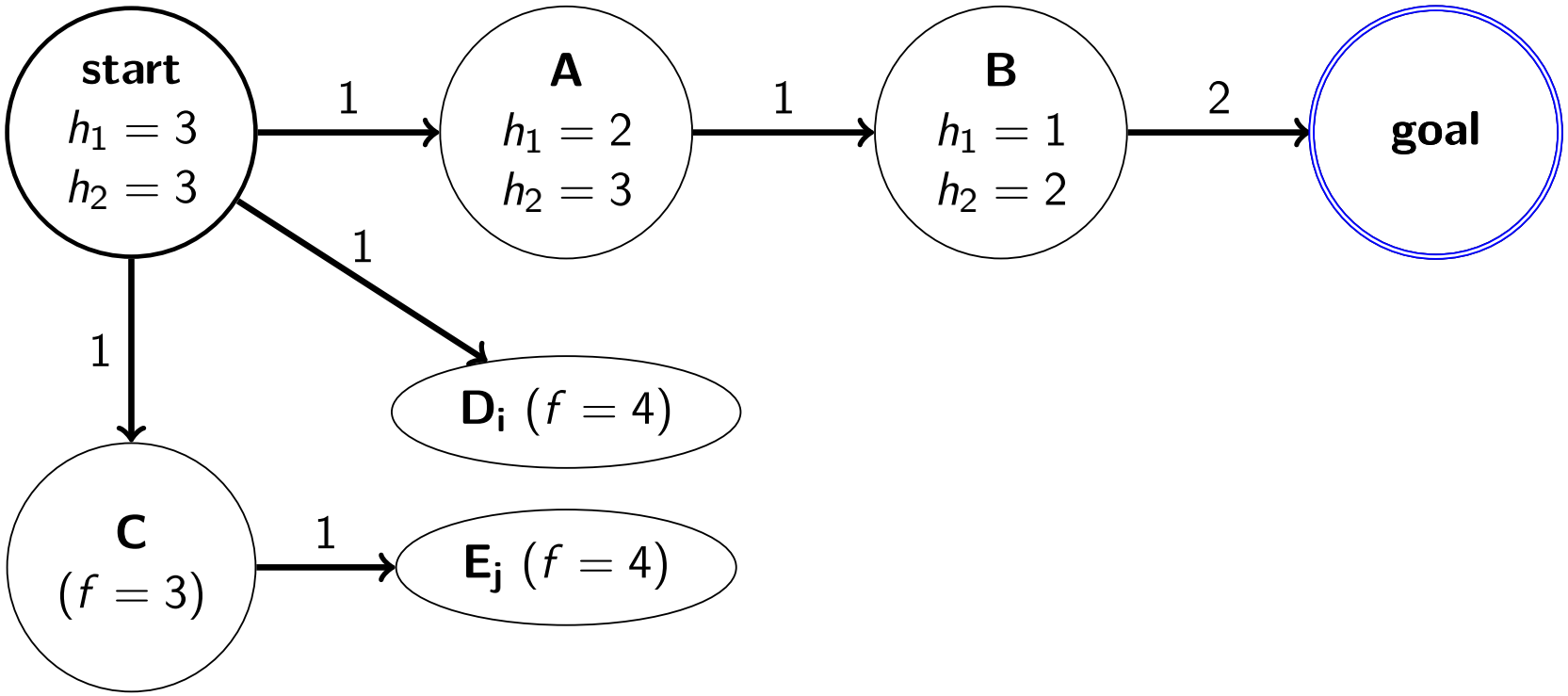
Statement 1: Counterexample

Performing A* with the better heuristic function h_2.



Statement 1: Counterexample

Performing A* with the better heuristic function h_2.



Worst case: $4 + i + j$ node expansions.

Best case: 4 node expansions.

Statement 1: Comparison

Comparision

Heuristic function	**Worst Case**	**Best Case**
h_2 (better)	$4 + i + j$	4
h_1 (dominated)	4	3

Statement 1: Comparison

Comparision

Heuristic function	Worst Case	Best Case
h_2 (better)	$4 + i + j$	4
h_1 (dominated)	4	3

What happened?

- All states not on optimal path: $h_1 = h_2$
- All states on optimal path: $h_2 > h_1$ (by definition)

$\Rightarrow$ Lower (but less accurate) h-values prevent from exploring states which are not on the optimal path.

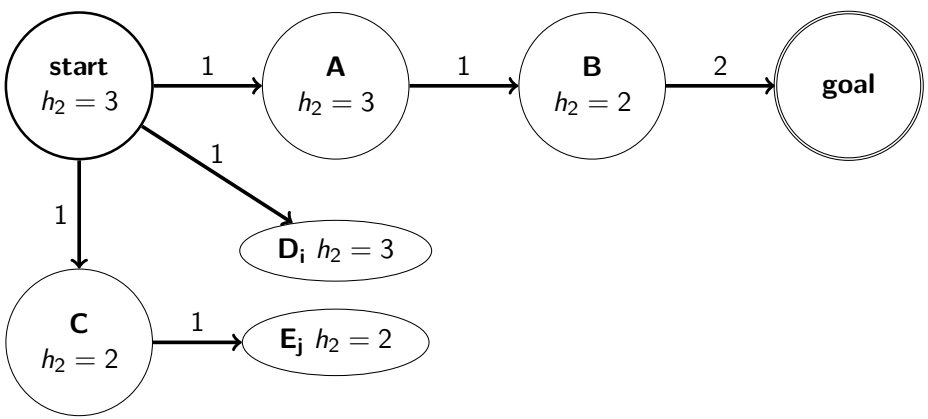
Statement 1: Can tie-breaking help?

Can tie-breaking help?

Can tie-breaking help?

Performing A* with the better heuristic function h_2.

Status after two expansions.

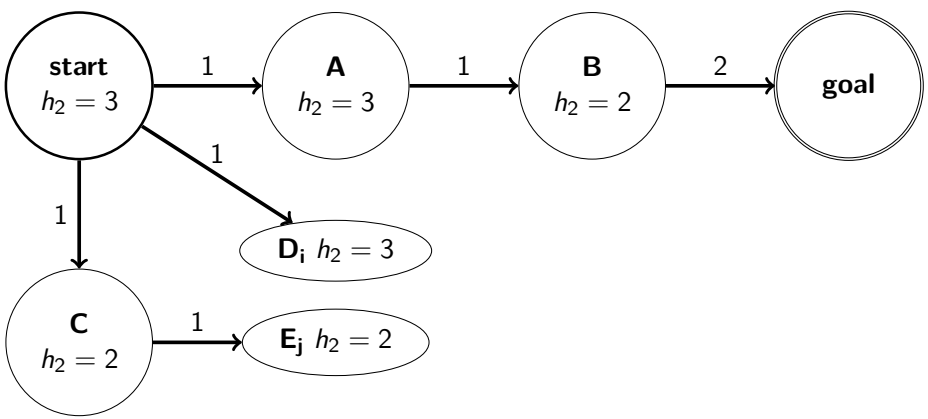


state	$f = g + h$
~~*start*~~	$3 = 0 + 3$
A	$4 = 1 + 3$
~~C~~	$3 = 1 + 2$
D_i	$4 = 1 + 3$
E_j	$4 = 2 + 2$

Can tie-breaking help?

Performing A* with the better heuristic function h_2.

Status after two expansions.



state	$f = g + h$
~~*start*~~	$3 = 0 + 3$
A	$4 = 1 + 3$
~~C~~	$3 = 1 + 2$
D_i	$4 = 1 + 3$
E_j	$4 = 2 + 2$

$\rightarrow$ **No, tie-breaking cannot help!**

Statement 2: Explanation

A* does fewer node expansions that any other equally informed algorithm that finds optimal solutions.

Statement 2: Explanation

A* does fewer node expansions that any other equally informed algorithm that finds optimal solutions.

$\rightarrow$ False if used heuristic is admissible, but not consistent.

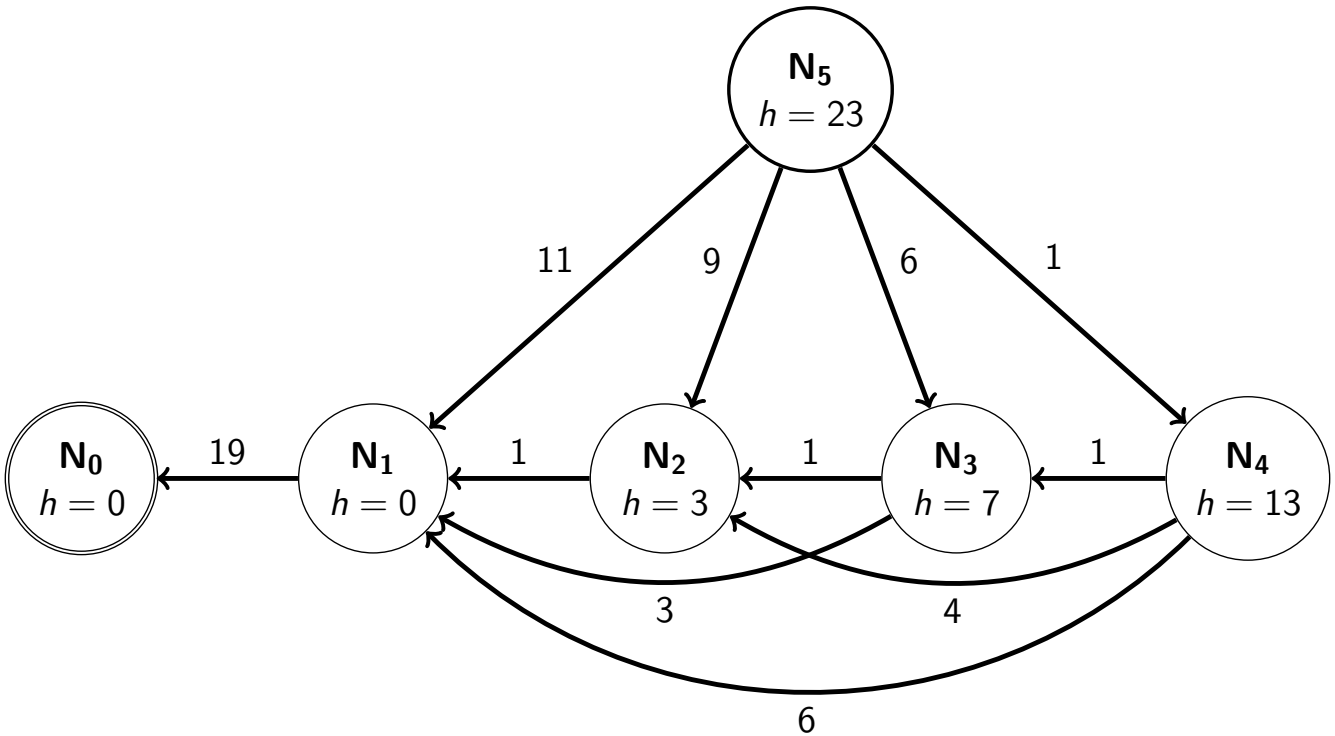
Proof sketch:

- State space with admissible, inconsistent heuristic.
- Show: A* does $\mathcal{O}(2^{i-1})$ node expansions.
- Introduce: Variant of A* which does $\mathcal{O}(i^2)$ node expansions.

$\Rightarrow$ A* is not always optimal w.r.t number of node expansions.

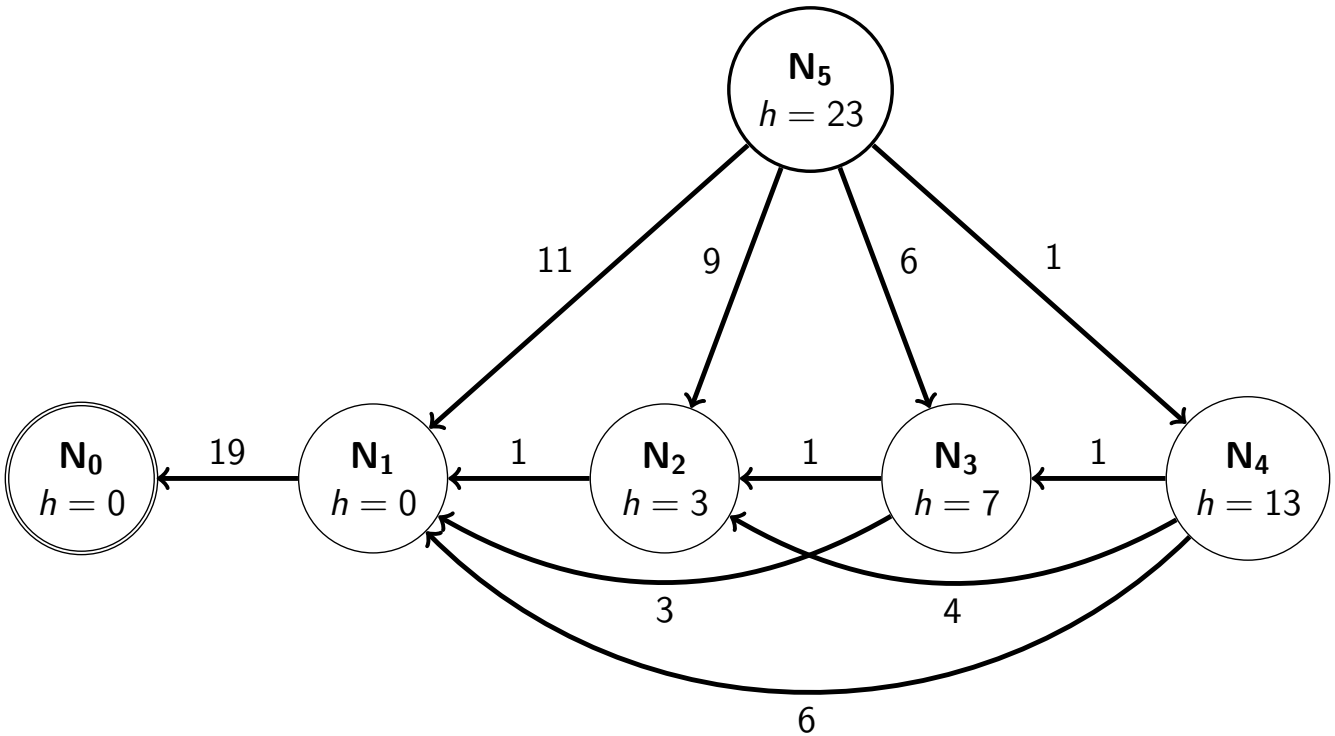
Statement 2: Counterexample (Martelli Graph G_5)

Martelli graph family G_i ($\forall i \geq 3$) with $i+1$ nodes and $\mathcal{O}(2^i)$ expansions (A*).



Statement 2: Counterexample (Martelli Graph G_5)

Martelli graph family G_i ($\forall i \geq 3$) with $i+1$ nodes and $\mathcal{O}(2^i)$ expansions (A*).



Inconsistency: E.g. $h(N_2) = 3$, $h(N_1) = 0$ and $cost(N_1, N_2) = 1$ but:

$$h(N_2) \leq cost(N_1, N_2) + h(N_1)$$

Statement 2: Algorithm B

B (variation of A*) finds optimal solution with less node expansions than A* in G_5.

Idea:

- Global variable F: Remembers highest occured f-value.
- Modified node fetching:
  - **If** (minimum f-value in open list $\geq F$) **then** same state as A*
  - **Else** state with minimum g-value among states with $f < F$

Statement 2: Algorithm B

B (variation of A*) finds optimal solution with less node expansions than A* in G_5.

Idea:

- Global variable F: Remembers highest occured f-value.
- Modified node fetching:
  - **If** (minimum f-value in open list $\geq F$) **then** same state as A*
  - **Else** state with minimum g-value among states with $f < F$

Hence:

No state is expanded twice, until F has increased.

$\Rightarrow$ Less expansions of nodes.

Statement 3: Explanation

Any admissible heuristic can be turned into a consistent heuristic by a simple technique called pathmax.

Statement 3: Explanation

Any admissible heuristic can be turned into a consistent heuristic by a simple technique called pathmax.

Idea behind pathmax:
On a path to a node n the f-value of the nodes should never decrease.
$\rightarrow$ Maximum f-value seen on the path is used.

Statement 3: Explanation

Any admissible heuristic can be turned into a consistent heuristic by a simple technique called pathmax.

Idea behind pathmax:
On a path to a node n the f-value of the nodes should never decrease.
$\rightarrow$ Maximum f-value seen on the path is used.

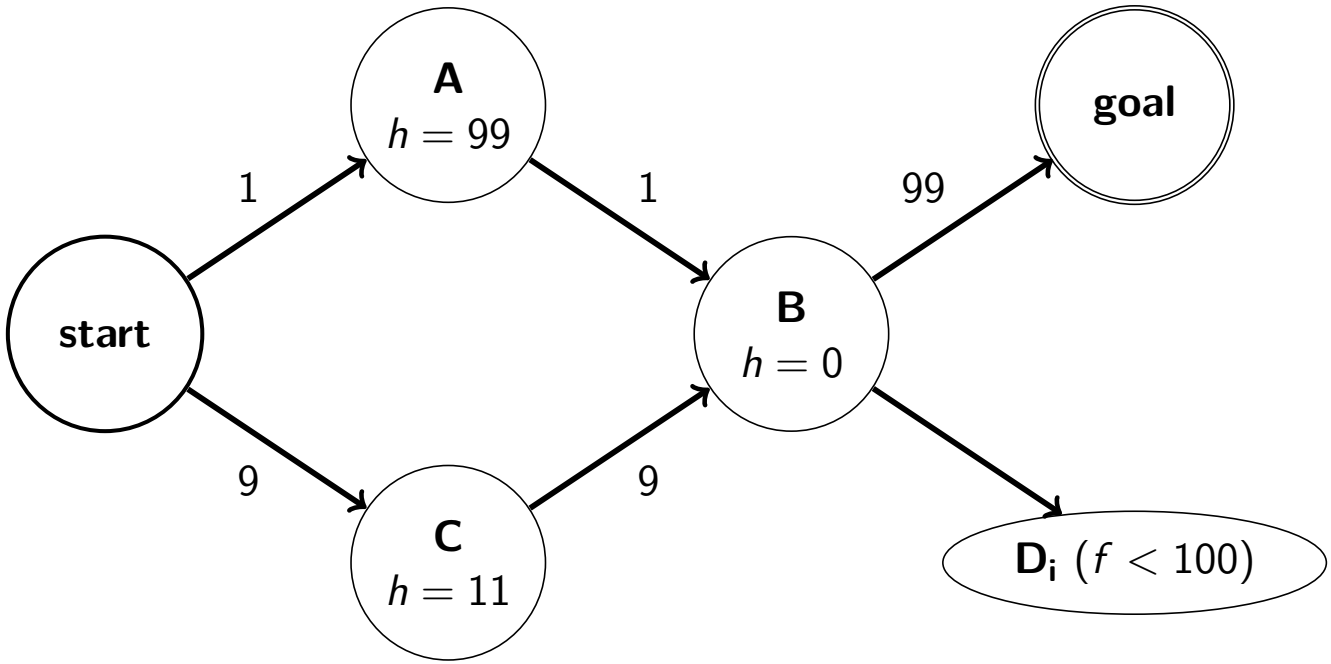
Connection to consistency?
Path through s. t lies later on that path, with decreasing $f(t)$:

$$h(s) \leq cost(s,t) + h(t) \quad \text{(without pathmax)}$$

But: Pathmax can help by prevent from decreasing $f(t)$.

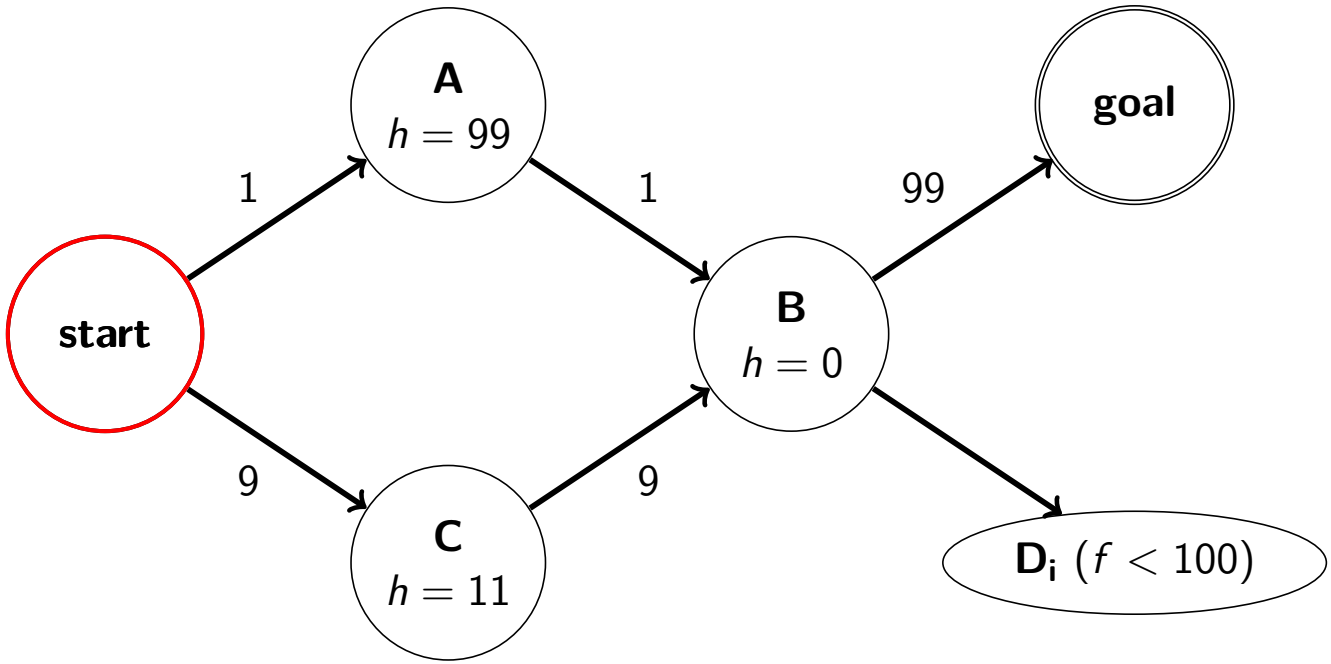
Statement 3: Counterexample

Reminder: A* with consistent heuristic never reopens a state.



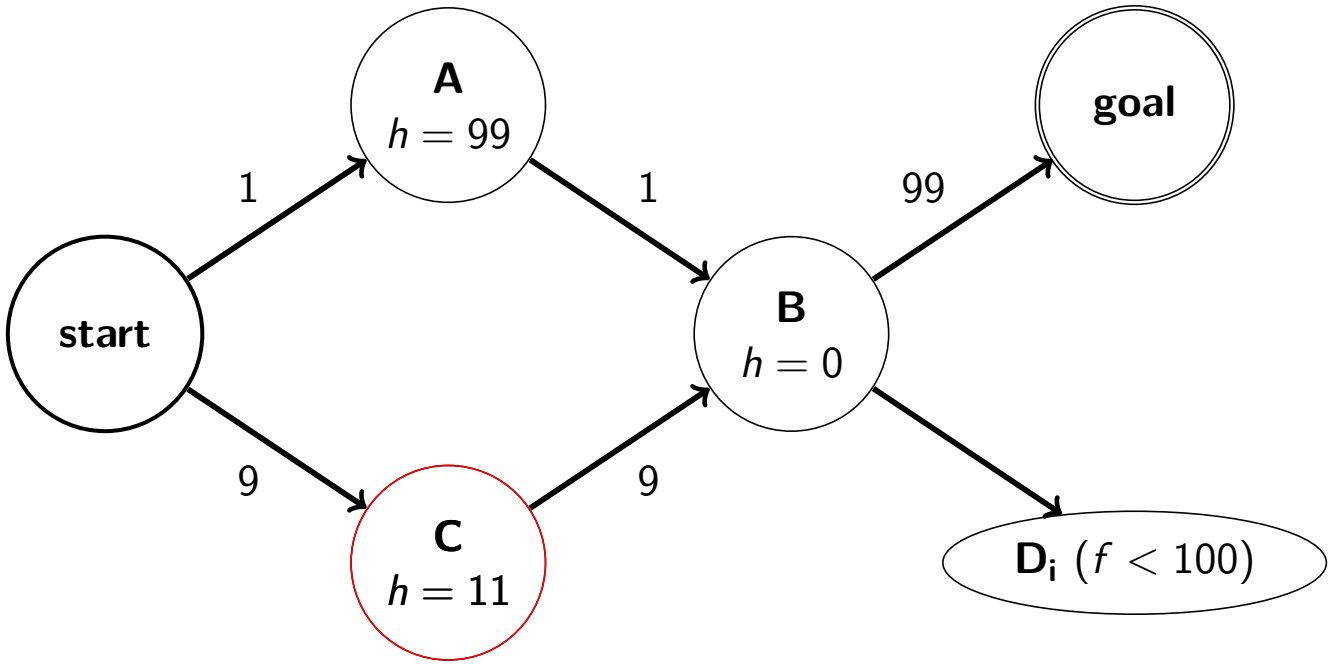
Statement 3: Counterexample

Reminder: A* with consistent heuristic never reopens a state.



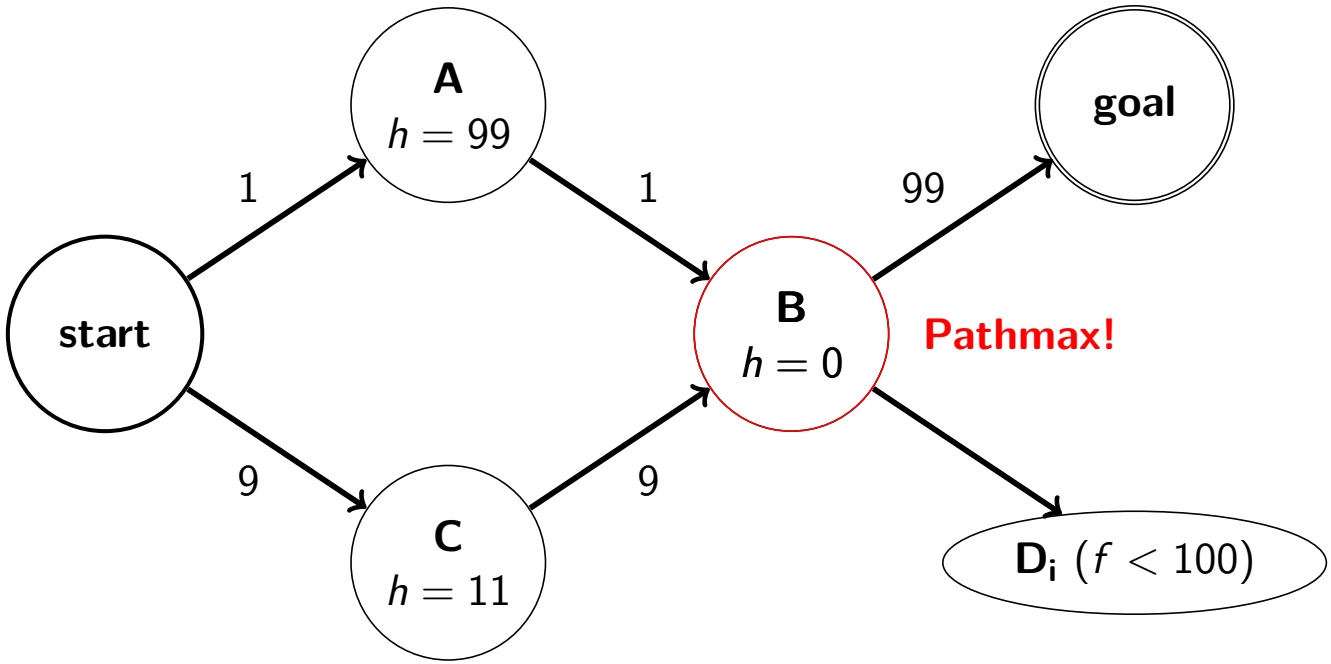
Statement 3: Counterexample

Reminder: A* with consistent heuristic never reopens a state.



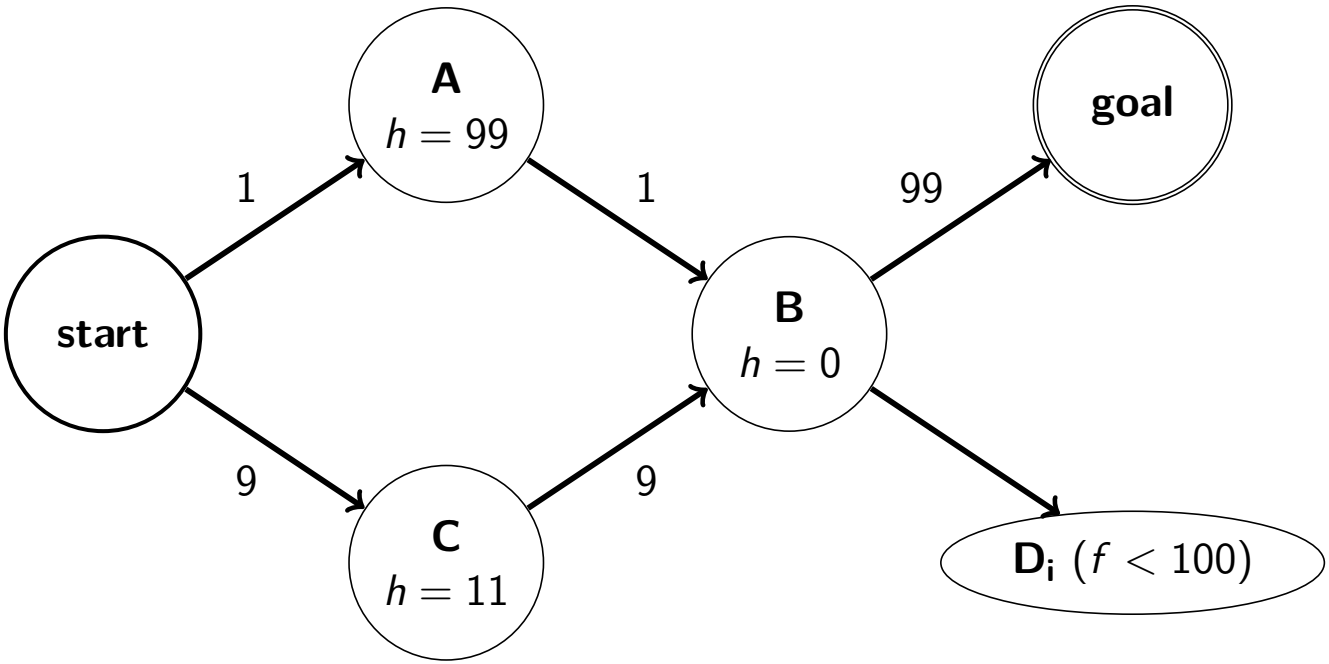
Statement 3: Counterexample

Reminder: A* with consistent heuristic never reopens a state.



Statement 3: Counterexample

Reminder: A* with consistent heuristic never reopens a state.



$\rightarrow$ B is expanded after reaching through a suboptimal path.
$\rightarrow$ B is reopened later reaching through an optimal path.
$\Rightarrow$ Contradiction to reminder.

Statement 4: Explanation

In search spaces whose operators all have the same cost
A* with the constant heuristic function $h = 0$
is the same as breadth-first search.

Statement 4: Explanation

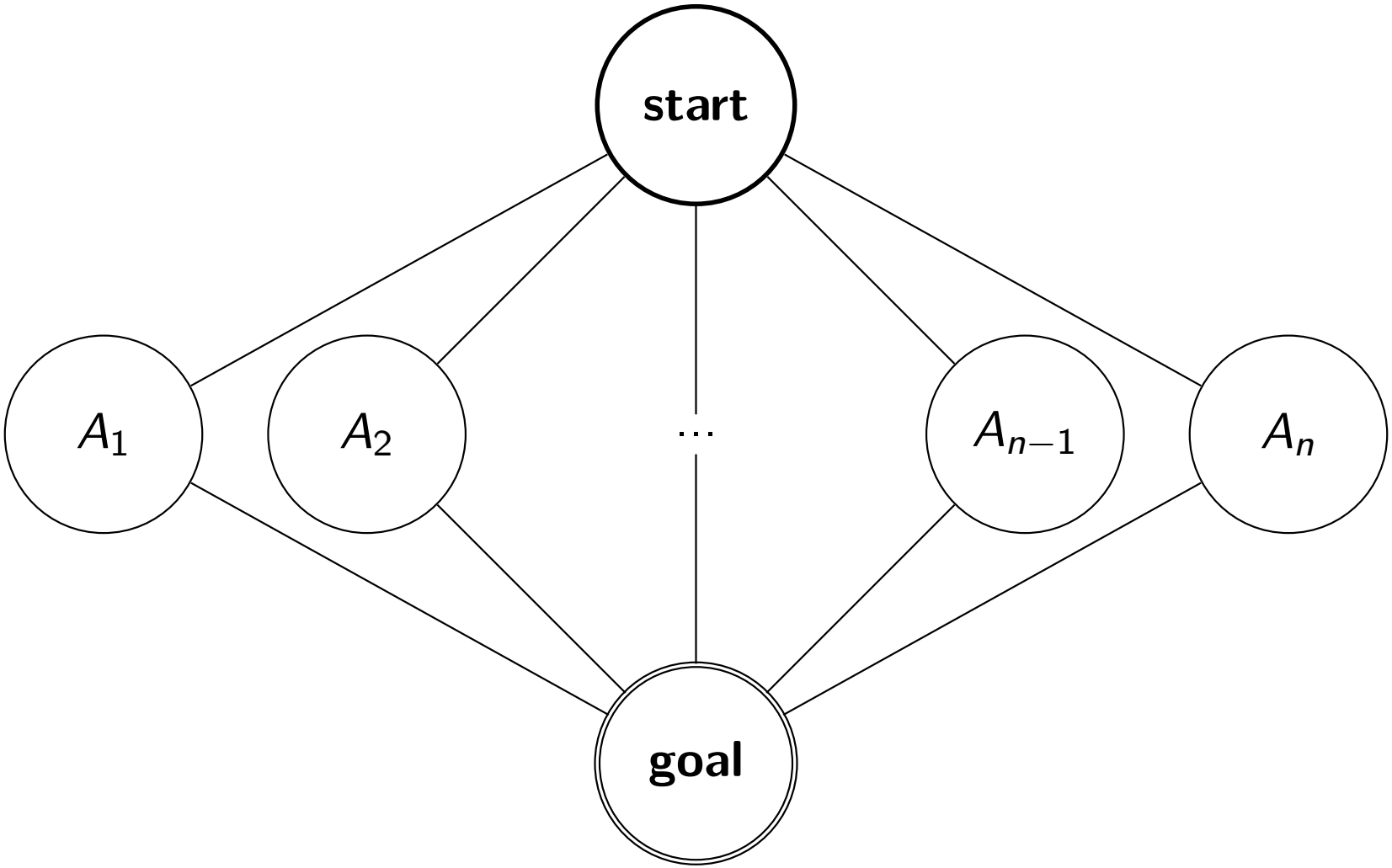
In search spaces whose operators all have the same cost
A* with the constant heuristic function $h = 0$
is the same as breadth-first search.

Where does this assumption come from?

Algorithm General framework for search algorithms

```
1:  OpenList = {root}
2:  ClosedList = {}
3:  while OpenList is not empty do
4:      n := fetch node from OpenList; /* to be specified*/
5:      if n ∉ ClosedList then
6:          if n is a goal then
7:              return plan to n from the root;
8:          save n in the ClosedList;
9:          expand n;
10:         save successors of n in OpenList;
11: return no solution;
```

Statement 4: Example



State space with uniform cost and constant heuristic function $h = 0$

Conclusion

Reformulated version of the discussed statements

1. Better heuristics can result in more search.

Conclusion

Reformulated version of the discussed statements

1. Better heuristics can result in more search.
2. A* is not always optimal with respect to the number of node expansions.

Conclusion

Reformulated version of the discussed statements

1. Better heuristics can result in more search.
2. A* is not always optimal with respect to the number of node expansions.
3. Pathmax does not make heuristics consistent.

Conclusion

Reformulated version of the discussed statements

1. Better heuristics can result in more search.
2. A* is not always optimal with respect to the number of node expansions.
3. Pathmax does not make heuristics consistent.
4. A* with $h = 0$ and uniform costs is not breadth-first search.

Questions?