Tufts

Lecture 05:
Heuristic Search (A*), II

Artificial Intelligence (CS 131)

1

Admissibility and Consistency

- Admissible heuristic = *never overestimates* actual cost to reach the goal (so never "discourages" search)
- Consistent heuristic: a form of "triangle inequality"

$$\forall n,\ \forall a,\ \forall n',\ h(n) \leq c(n, a, n') + h(n')$$

Heuristic estimate for node n | *Actual cost* to get from n to n' | Heuristic estimate for *next node* n'

- *Theorem*: If a heuristic is consistent, then it is admissible
  - Can be proven by induction on the definition

2

Other Results about Consistent and/or Admissible Heuristics

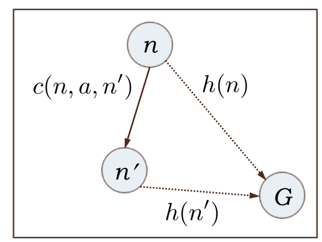
- *Theorem*: If heuristic h is *consistent*, then path costs are monotonic (never decrease)
- *Theorem*: If heuristic h is *admissible*, then A* search is optimal, and always finds best path to a goal
- *Theorem*: If heuristic h is *consistent and admissible*, then A* is maximally efficient; that is, no algorithm will *always* expand fewer nodes than A*

3

If h is Consistent $\Rightarrow$ f is Monotone (Path-costs never decrease)

- A heuristic h is consistent iff:

$$\forall n,\ \forall a,\ \forall n',\ h(n) \leq c(n, a, n') + h(n')$$



- If h is consistent, then:

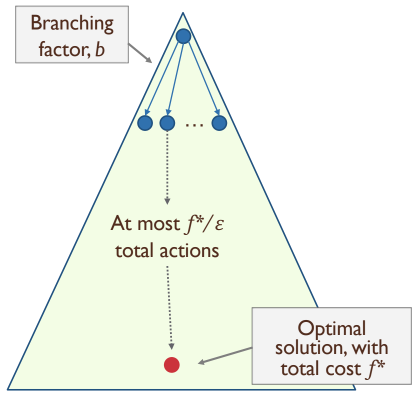
$$\begin{aligned} f(n') &= g(n') + h(n') \\ &= g(n) + c(n, a, n') + h(n') \\ &\geq g(n) + h(n) \\ &= f(n) \end{aligned}$$

  - That is, the cost $f(n) \leq f(n')$ as we go along the path
  - Since this is true of all nodes n, path costs never go down

4

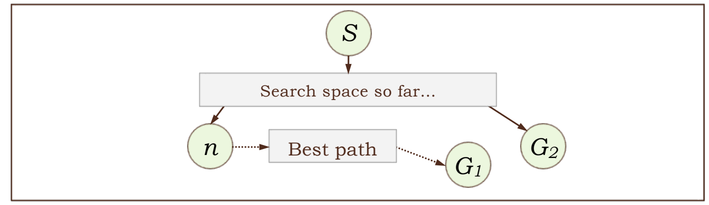
Completeness of A*

- A* is complete unless there are infinitely many nodes n with estimated cost $f(n) < f^*$, where f^* is optimal cost to goal
- A* is therefore complete whenever we have:
  1. A positive lower bound on the cost of actions (i.e., all actions a have cost $c_a \geq \varepsilon > 0$)
  2. A finite branching factor b



5

Optimality of A*



- Suppose suboptimal goal G_2 in frontier, and n is a frontier node on path to optimal goal G_1:

$$\begin{aligned} f(G_2) &= g(G_2) && \text{since } h(G_2) = 0 \\ &> g(G_1) && \text{since } G_2 \text{ is suboptimal} \\ &\geq f(n) && \text{since } h \text{ is admissible (doesn't overestimate)} \end{aligned}$$

- Since $f(G_2) > f(n)$, A* will never select G_2 before n
- Since this is true for *all* nodes n on any path to the optimal goal, we will *always* reach G_1 before expanding G_2

6

A* is Maximally Efficient

- For a given *consistent and admissible* heuristic function, no optimal algorithm is *guaranteed* to do less work
- A* expands *every node necessary* to find the shortest path, and *no other* (aside from ties in f)
- Since it is optimal for any heuristic, the only way to improve on it for basic graph search is to improve the heuristic measure that we are using
  - This can be quite complex in real life
  - Expert domain knowledge is useful when designing heuristic

7

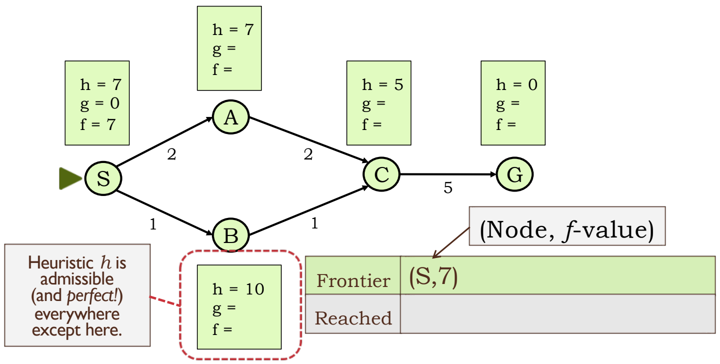
Optimal Efficiency of A*

- Given a *consistent and admissible* heuristic, A* is optimally efficient in that:
  - It never expands a node on the frontier if there is a shorter path to that node
  - That is, it is finding not only the most efficient path to the final goal, but also explores most efficient paths to each unique non-goal node along the way
- If either of these properties are lacking, we have no such guarantees anymore!

8

1. A* is *Not* Optimal (if *h* is Inadmissible)

- It is easy to show that A* can fail to find an optimal solution if it given a non-admissible heuristic, which *overestimates* at least some of the time:

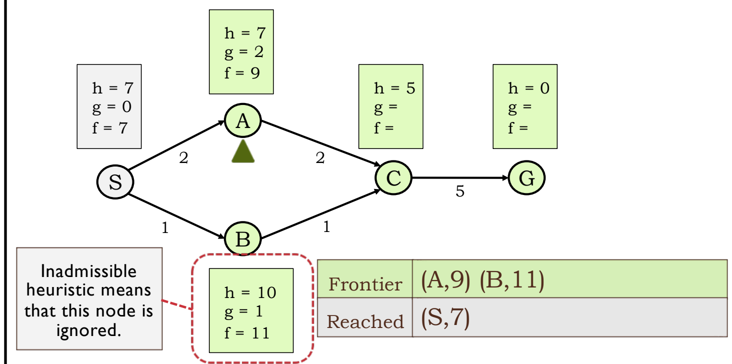


Frontier	(S,7)
Reached	

9

1. A* is *Not* Optimal (if *h* is Inadmissible)

- It is easy to show that A* can fail to find an optimal solution if it given a non-admissible heuristic, which *overestimates* at least some of the time:

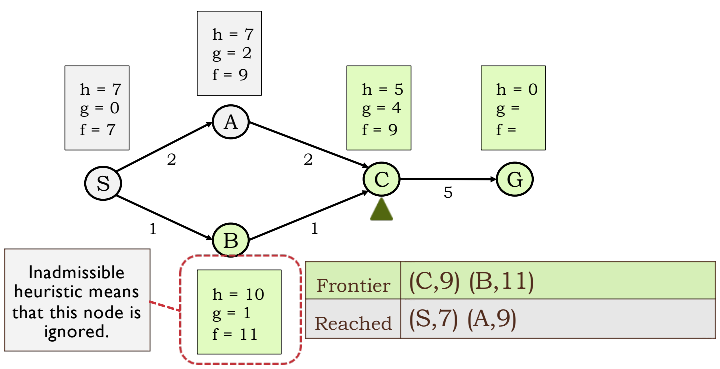


Frontier	(A,9) (B,11)
Reached	(S,7)

10

1. A* is *Not* Optimal (if *h* is Inadmissible)

- It is easy to show that A* can fail to find an optimal solution if it given a non-admissible heuristic, which *overestimates* at least some of the time:

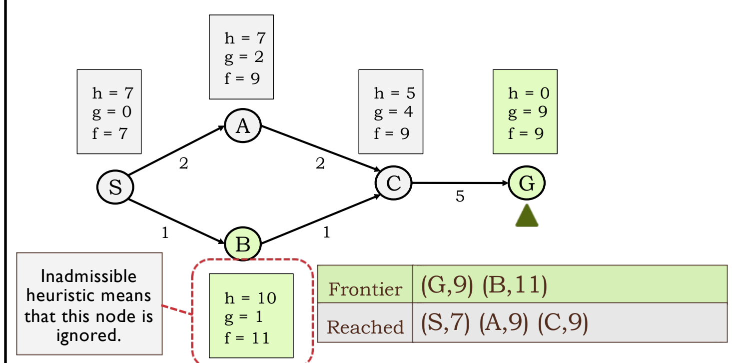


Frontier	(C,9) (B,11)
Reached	(S,7) (A,9)

11

1. A* is *Not* Optimal (if *h* is Inadmissible)

- It is easy to show that A* can fail to find an optimal solution if it given a non-admissible heuristic, which *overestimates* at least some of the time:

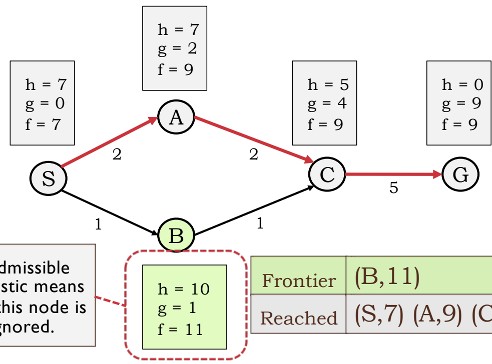


Frontier	(G,9) (B,11)
Reached	(S,7) (A,9) (C,9)

12

1. A* is *Not* Optimal (if *h* is Inadmissible)

- It is easy to show that A* can fail to find an optimal solution if it given a non-admissible heuristic, which *overestimates* at least some of the time:



We end up with a sub-optimal solution, due to the inadmissible, over-estimate heuristic

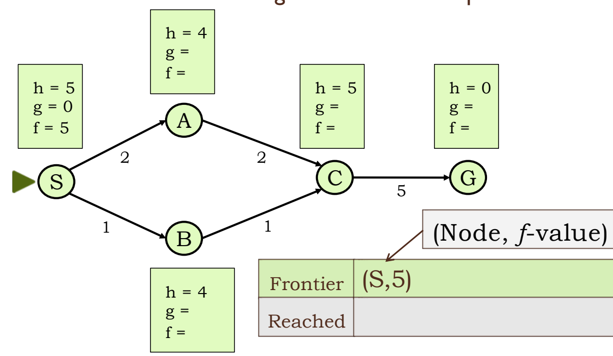
Inadmissible heuristic means that this node is ignored.

Frontier	(B,11)
Reached	(S,7) (A,9) (C,9) (G,9)

13

2. A* is Optimally Efficient (if *h* is Admissible & Consistent)

- Consider following, where we have an admissible and consistent heuristic throughout our search-space:



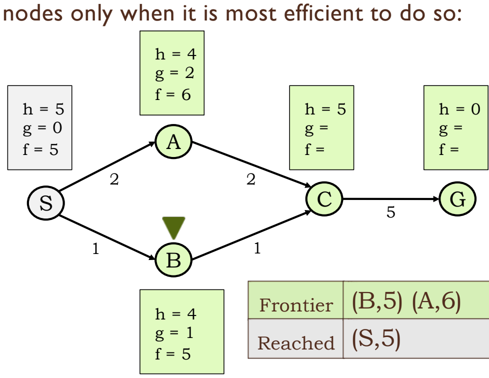
(Node, *f*-value)

Frontier	(S,5)
Reached	

14

2. A* is Optimally Efficient (if *h* is Admissible & Consistent)

- Given the admissible and consistent heuristic, we expand nodes only when it is most efficient to do so:

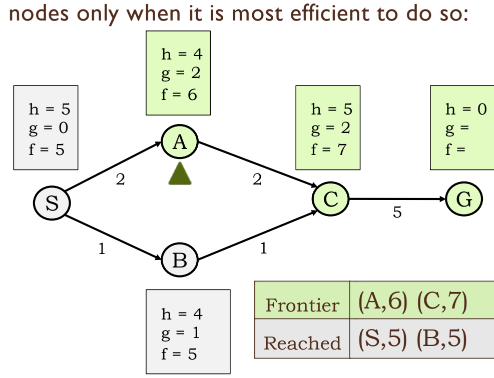


Frontier	(B,5) (A,6)
Reached	(S,5)

15

2. A* is Optimally Efficient (if *h* is Admissible & Consistent)

- Given the admissible and consistent heuristic, we expand nodes only when it is most efficient to do so:

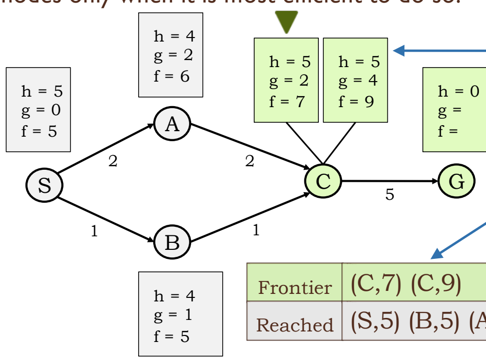


Frontier	(A,6) (C,7)
Reached	(S,5) (B,5)

16

2. A* is Optimally Efficient (if h is Admissible & Consistent)

- Given the admissible and consistent heuristic, we expand nodes only when it is most efficient to do so:



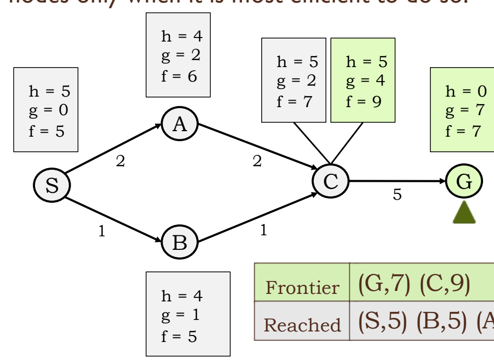
We now have 2 versions of node C in our frontier. We always expand the least-cost version first.

Frontier	(C,7) (C,9)
Reached	(S,5) (B,5) (A,6)

17

2. A* is Optimally Efficient (if h is Admissible & Consistent)

- Given the admissible and consistent heuristic, we expand nodes only when it is most efficient to do so:

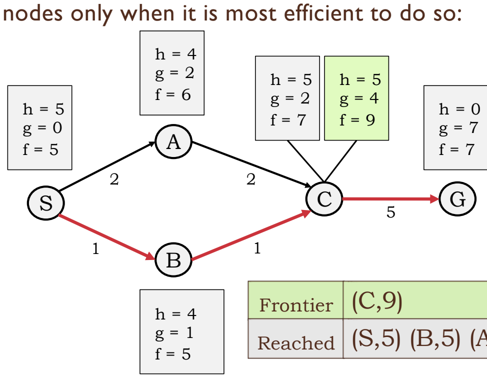


Frontier	(G,7) (C,9)
Reached	(S,5) (B,5) (A,6) (C,7)

18

2. A* is Optimally Efficient (if h is Admissible & Consistent)

- Given the admissible and consistent heuristic, we expand nodes only when it is most efficient to do so:



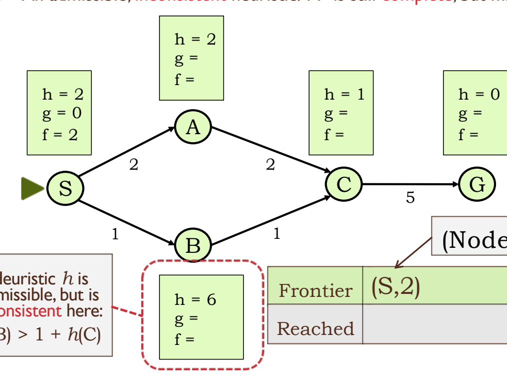
We have found the optimal path to goal G, without ever expanding a sub-optimal path to any node if we didn't need to

Frontier	(C,9)
Reached	(S,5) (B,5) (A,6) (C,7) (G,7)

19

3. A* May *Not* be Optimally Efficient (if h is Admissible, but not Consistent)

- A consistent heuristic is always admissible; but not vice-versa
- An admissible, inconsistent heuristic: A* is still complete, but may be inefficient



Heuristic h is admissible, but is inconsistent here: $h(B) > 1 + h(C)$

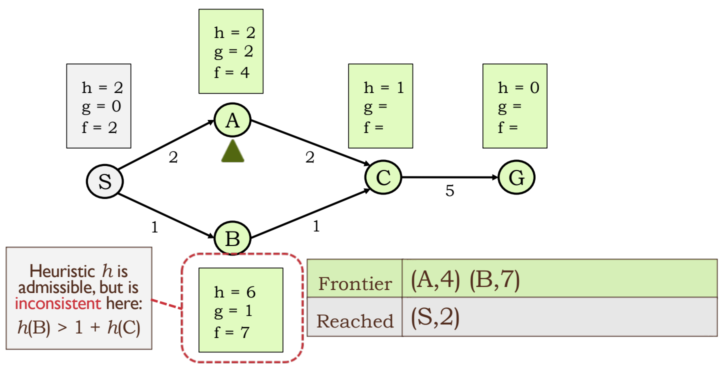
(Node, f-value)

Frontier	(S,2)
Reached	

20

3. A* May *Not* be Optimally Efficient (if *h* is Admissible, but not Consistent)

- A consistent heuristic is always admissible; but not vice-versa
- An admissible, inconsistent heuristic: A* is still complete, but may be inefficient

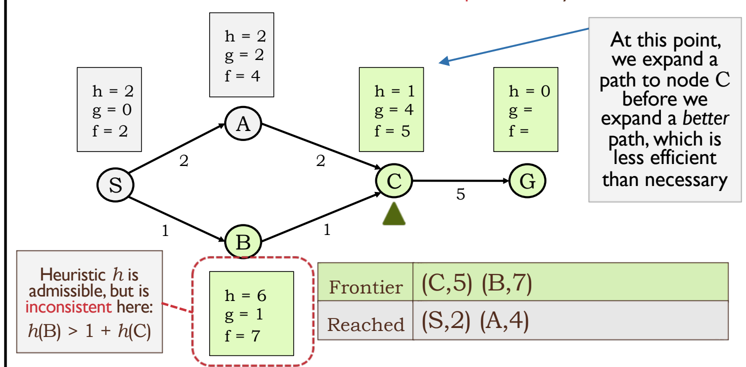


Frontier	(A,4) (B,7)
Reached	(S,2)

21

3. A* May *Not* be Optimally Efficient (if *h* is Admissible, but not Consistent)

- A consistent heuristic is always admissible; but not vice-versa
- An admissible, inconsistent heuristic: A* is still complete, but may be inefficient

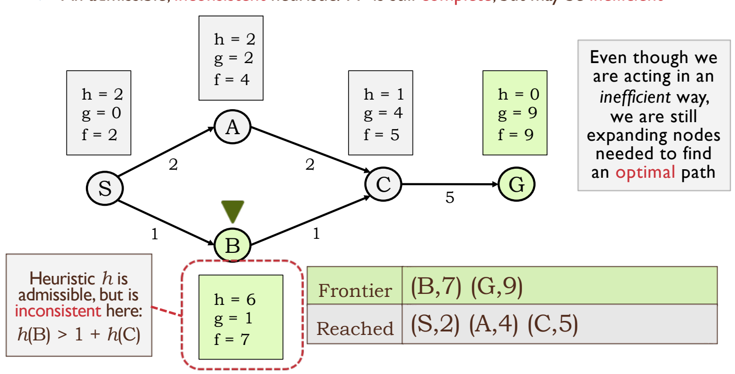


Frontier	(C,5) (B,7)
Reached	(S,2) (A,4)

22

3. A* May *Not* be Optimally Efficient (if *h* is Admissible, but not Consistent)

- A consistent heuristic is always admissible; but not vice-versa
- An admissible, inconsistent heuristic: A* is still complete, but may be inefficient

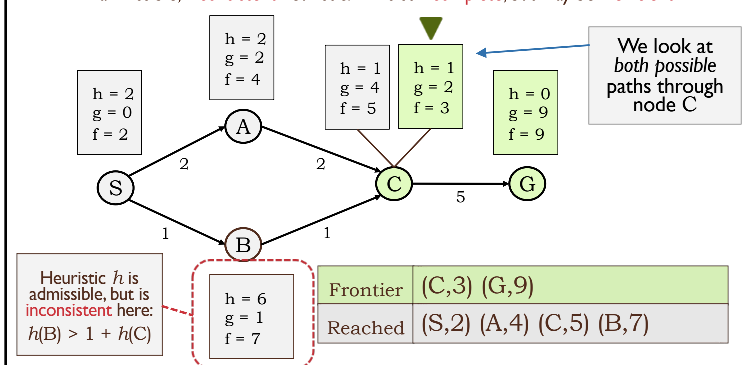


Frontier	(B,7) (G,9)
Reached	(S,2) (A,4) (C,5)

23

3. A* May *Not* be Optimally Efficient (if *h* is Admissible, but not Consistent)

- A consistent heuristic is always admissible; but not vice-versa
- An admissible, inconsistent heuristic: A* is still complete, but may be inefficient

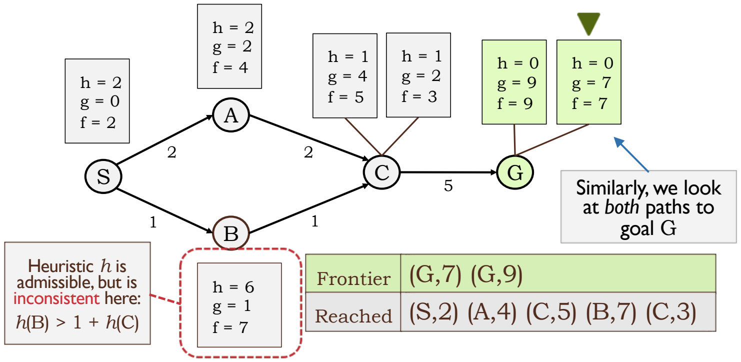


Frontier	(C,3) (G,9)
Reached	(S,2) (A,4) (C,5) (B,7)

24

3. A* May *Not* be Optimally Efficient (if h is Admissible, but not Consistent)

- A consistent heuristic is always admissible; but not vice-versa
- An admissible, inconsistent heuristic: A* is still complete, but may be inefficient

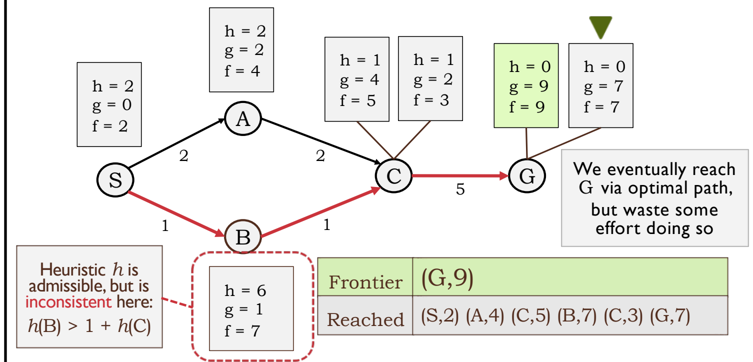


Frontier	(G,7) (G,9)
Reached	(S,2) (A,4) (C,5) (B,7) (C,3)

25

3. A* May *Not* be Optimally Efficient (if h is Admissible, but not Consistent)

- A consistent heuristic is always admissible; but not vice-versa
- An admissible, inconsistent heuristic: A* is still complete, but may be inefficient



Frontier	(G,9)
Reached	(S,2) (A,4) (C,5) (B,7) (C,3) (G,7)

26

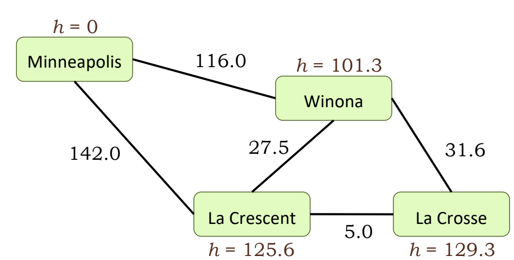
Another Source of Inefficiency

- A* is maximally efficient in general, of all algorithms that expand nodes based solely upon path costs and a given heuristic estimate function
  - This ***does not*** mean it can't be improved, however
- In particular, a naïve implementation of A* allows nodes to *repeat* in partial solution paths, even though a solution that back-tracks *never* makes sense when we have non-decreasing, monotonic path costs

27

Another Source of Inefficiency

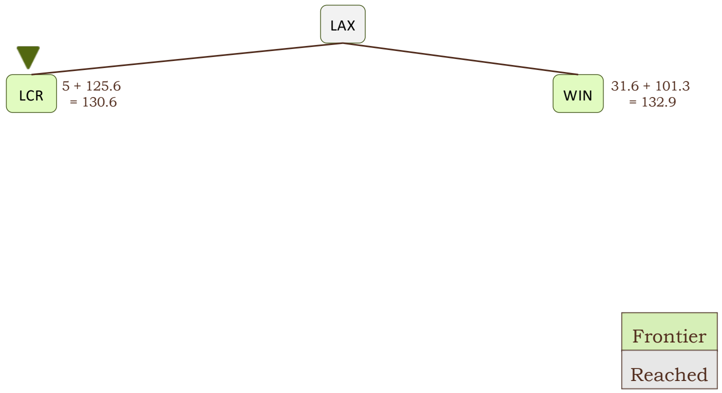
- Consider the following set of cities
- We calculate all heuristic values as minimum geographical distance to target city (Minneapolis), based on cities' latitude and longitude



28

Another Source of Inefficiency

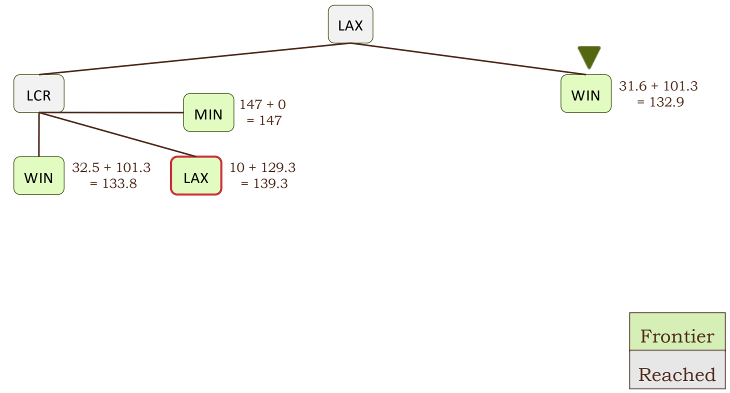
- If we start from La Crosse, and *allow repeats*, we generate unnecessary search branches:



29

Another Source of Inefficiency

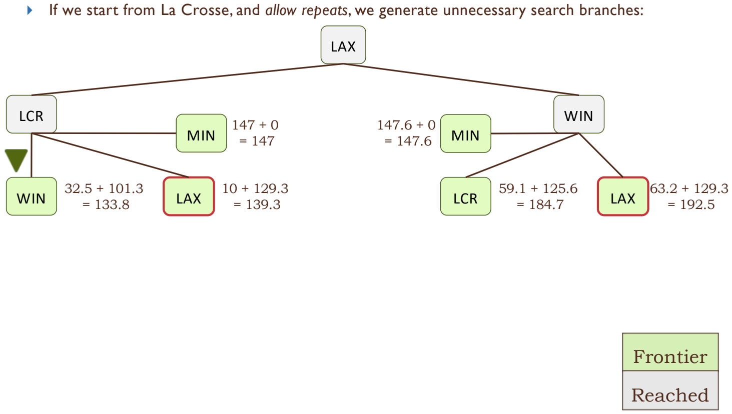
- If we start from La Crosse, and *allow repeats*, we generate unnecessary search branches:



30

Another Source of Inefficiency

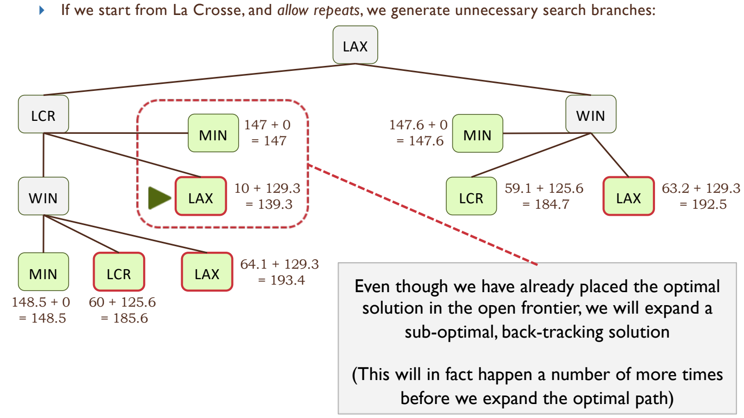
- If we start from La Crosse, and *allow repeats*, we generate unnecessary search branches:



31

Another Source of Inefficiency

- If we start from La Crosse, and *allow repeats*, we generate unnecessary search branches:



32