

# Artificial Intelligence For NLP Lesson-03

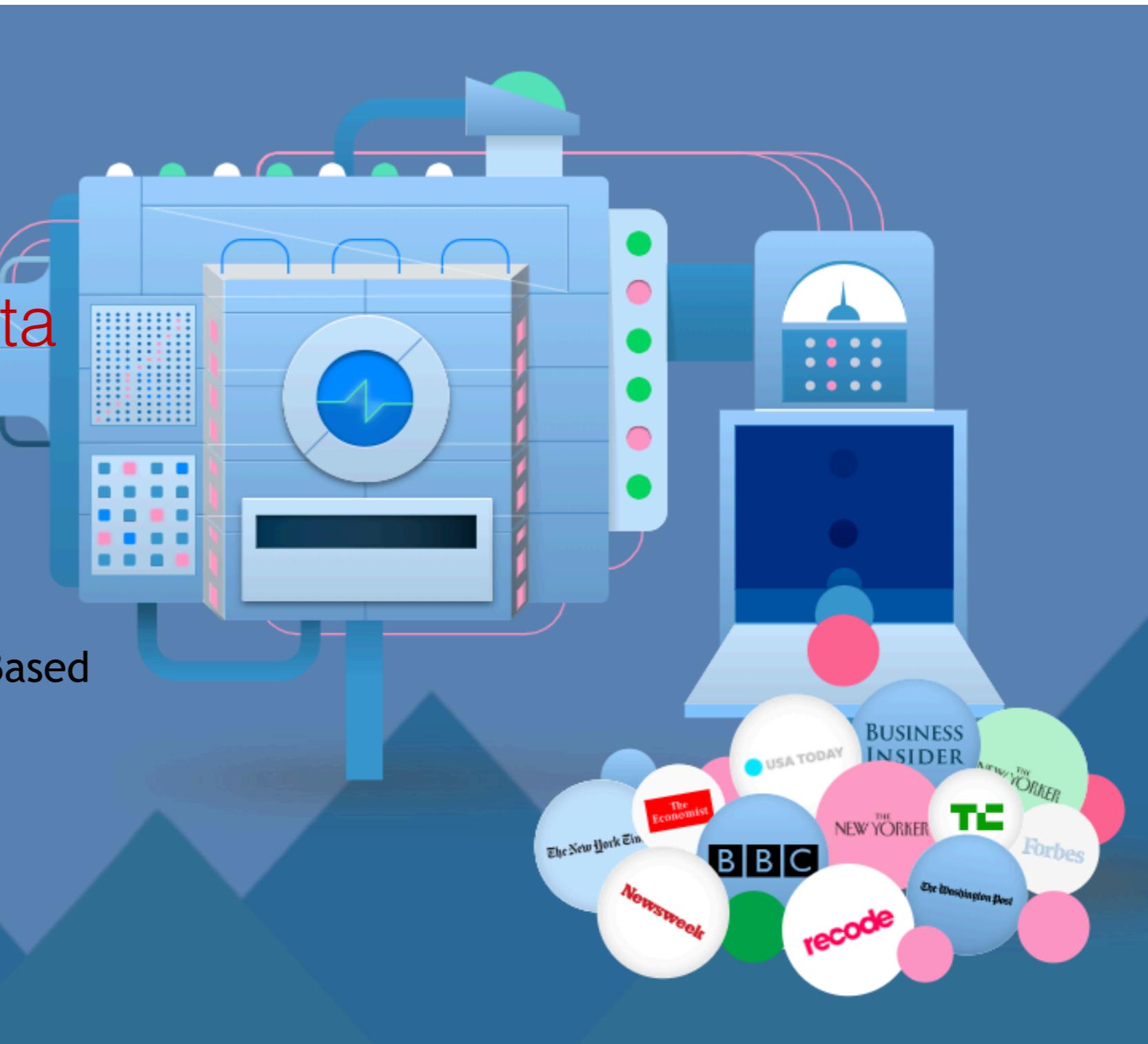
人工智能与自然语言处理  
课程组

2019.July. 13



# AI Paradigm & Data Driven

- 1. Rule Based
- 2. Search Based
- 3. Mathematical or Analytic Based
- 3. Probability Based
- 4. Machine Learning (deep learning) Based



## Some utilities for NLP

- 1. Similarity: Edit Distance, Word Distance
- 2. Key words
- 3. Name Entity Recognition
- 4. Dependency Parsing
- 5. Topic Model

# The Problem of Search

- Heuristic Function
- Search Problem, Search Tree

```

def search(graph, concat_func):
    seen = set()
    need_visited = ['1']

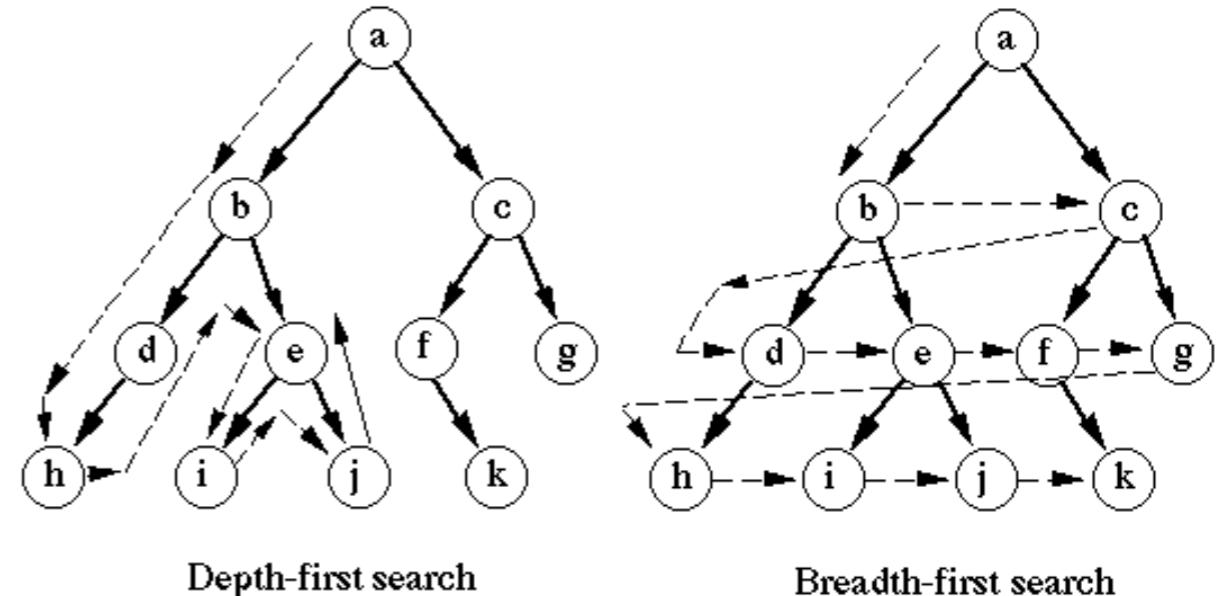
    while need_visited:
        node = need_visited.pop(0)
        if node in seen: continue
        print('    I am looking at : {}'.format(node))
        seen.add(node)
        new_discovered = graph[node]
        need_visited = concat_func(new_discovered, need_visited)

    def treat_new_discover_more_important(new_discovered, need_visited):
        return new_discovered + need_visited

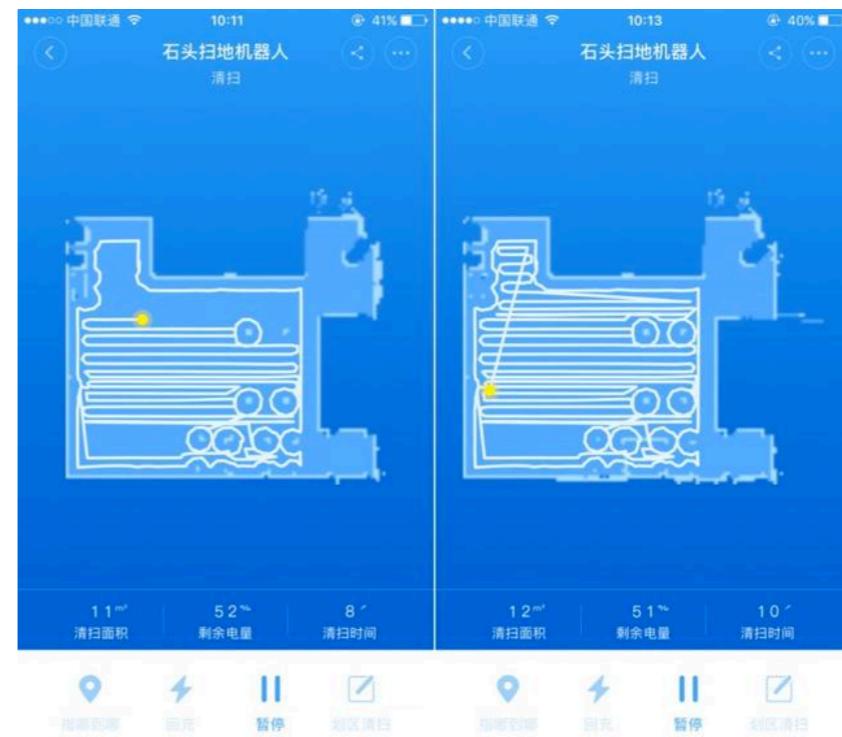
    def treat_already_discovered_more_important(new_discovered, need_visited):
        return need_visited + new_discovered

dfs = partial(search, concat_func=treat_new_discover_more_important)
bfs = partial(search, concat_func=treat_already_discovered_more_important)

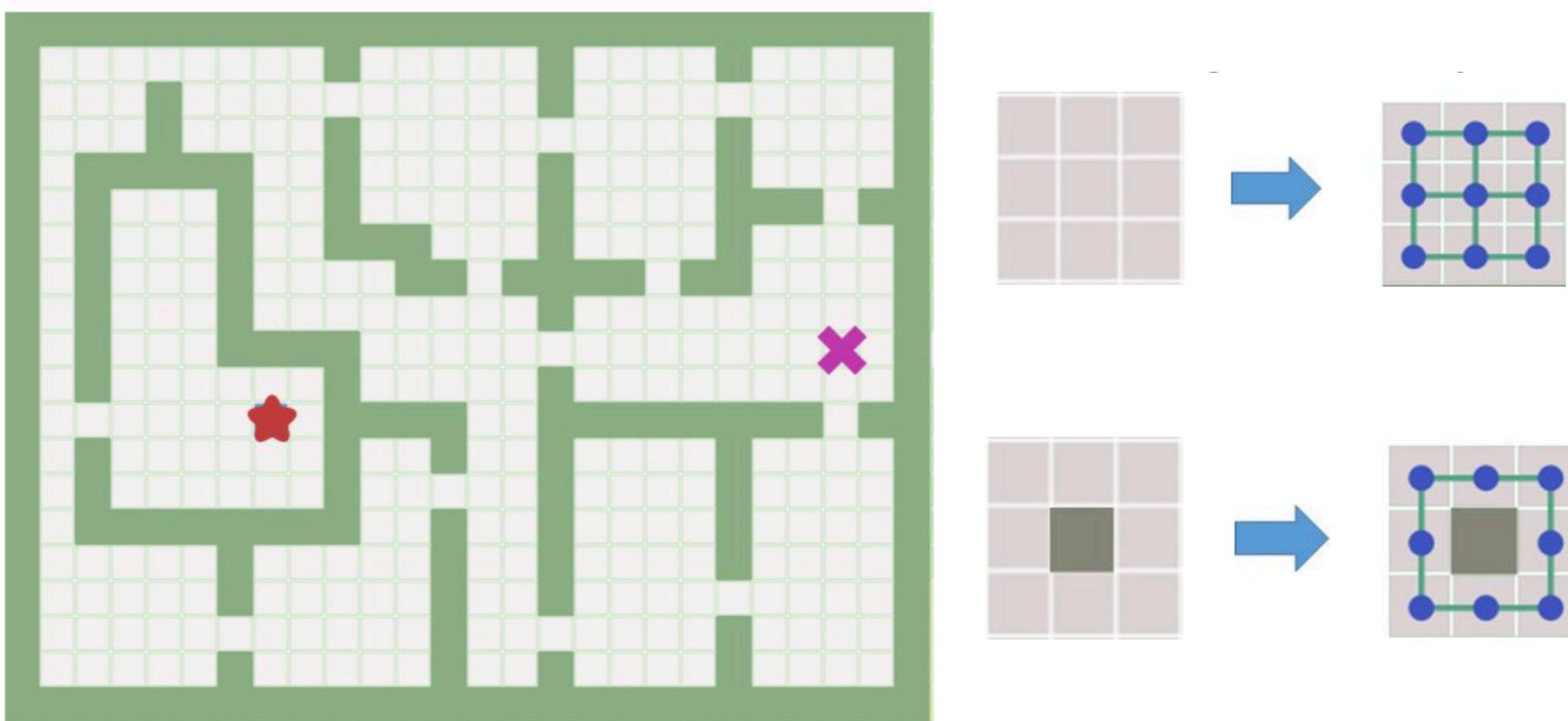
```



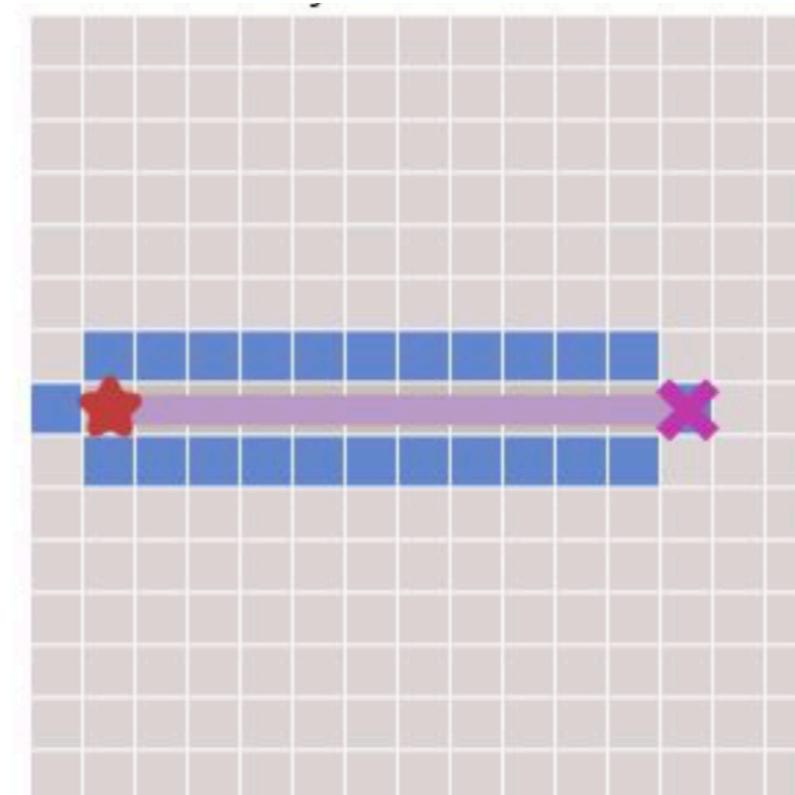
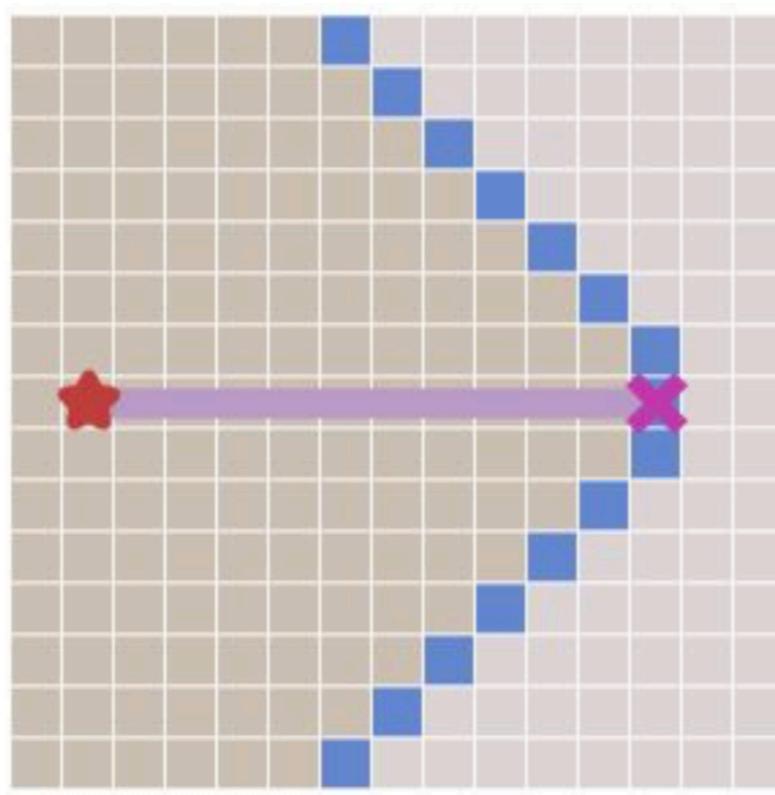
# Applications



- The *start* state
- The *goal* state
- The *successors*
- The strategy that determines the order in which we search.



# From Breadth First Search to Best First Search



A *heuristic* is an approximate measure of how close you are to the target

- Find the person-2-person
- Connection in a social web.



# A\* search

At each iteration of its main loop, A\* needs to determine which of its paths to extend. It does so based on the cost of the path and an estimate of the cost required to extend the path all the way to the goal. Specifically, A\* selects the path that minimizes

$$f(n) = g(n) + h(n)$$

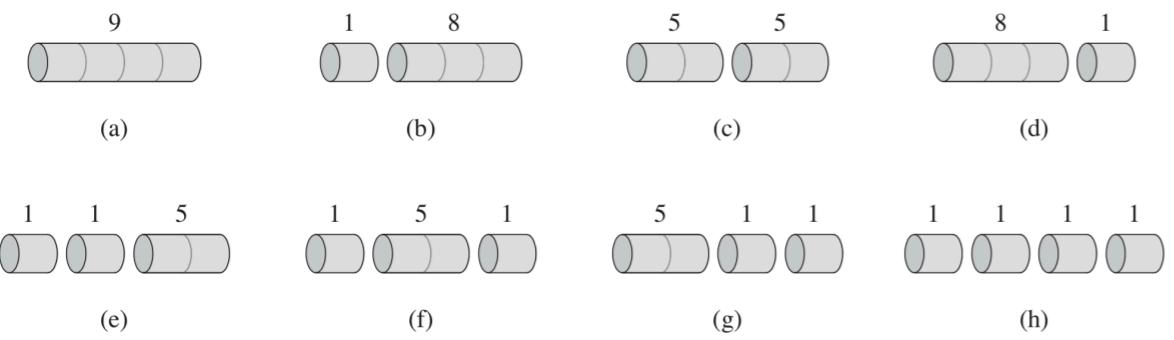
Like BFS, it finds the shortest path, and like Greedy Best First, it's fast.

Each iteration, A\* chooses the node on the frontier which minimizes:

steps from source + approximate steps to target

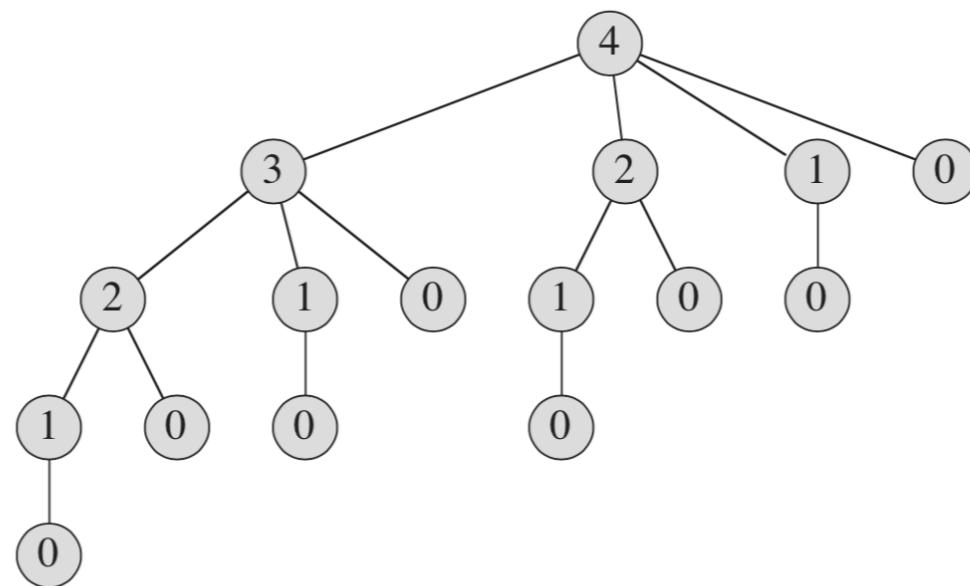
# Dynamic Programming

- 1. Rob Cutting Problem
- 2. Edit-Distance Problem
- 3. Key Characteristics for Dynamic Programming
- 4. The Travel Salesman Problem



$\frac{\text{length } i}{\text{price } p_i}$	1	2	3	4	5	6	7	8	9	10
	1	5	8	9	10	17	17	20	24	30

$$r_n = \max(p_n, r_1 + r_{n-1}, r_2 + r_{n-2}, \dots, r_{n-1} + r_1) .$$



Programming Solution For This

Problem  
2: Edit  
Distance

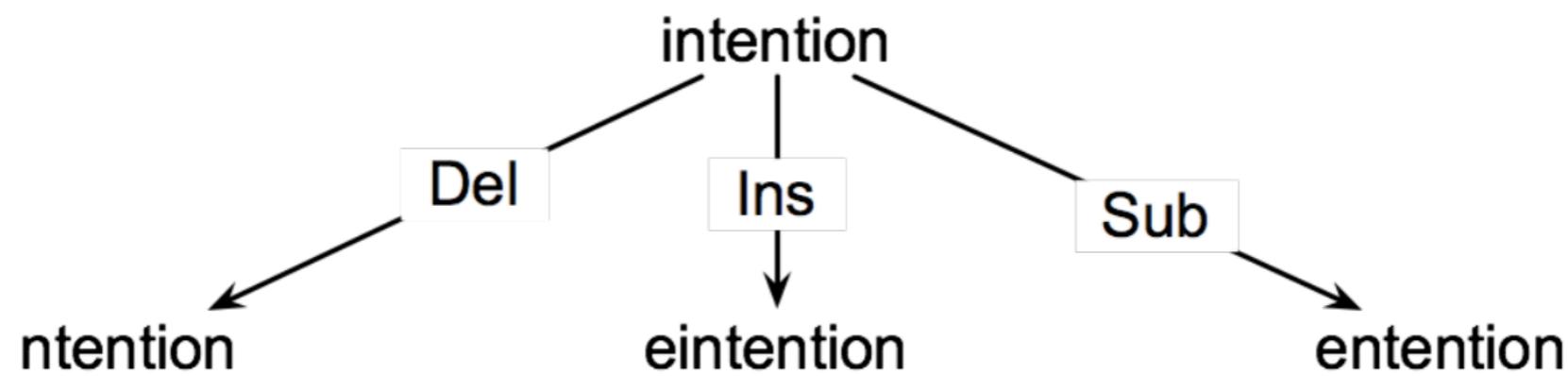
I N T E \* N T I O N  
| | | | | | | |  
\* E X E C U T I O N

- Insertion
- Deletion
- Substitution

# How similar are two strings?

- Spell Correction
    - The user typed “biejing”
    - --biejie? (別介)
    - --beijing? (北京)
    - --beijin? (北金)
  - Evaluating Machine Translation and speech recognition
    - Spokesman confirms senior government adviser was shot.
    - Spokesman said the senior adviser was shot dead.

# Search Graph is Huge



# Defining Min Edit Distance

- For two strings
  - X of length  $n$
  - Y of length  $m$
- We define  $D(i, j)$ 
  - the edit distance between  $X[1..i]$  and  $Y[1..j]$
  - The edit distance between X and Y is thus  $D(n, m)$

# Defining Min Edit Distance (Levenshtein)

- Initialization

- $D(i, 0) = i$
- $D(0, j) = j$

- $D(i, j) = \min \left\{ \begin{array}{l} D(i - 1, j) + 1 \\ D(i, j - 1) + 1 \\ D(i - 1, j - 1) + 2 \text{ if } X(i) \neq Y(i) \text{ else } D(i - 1, j - 1) \end{array} \right.$