Searching is about exploring alternatives.

- **Uninformed search**
  - Does not use heuristics to speed up the search
  - Potentially wastes a lot of time looking at completely unviable solutions
  - “brute force” search

- **Informed search**
  - Does use a heuristic to speed up the search (e.g. manhattan distance in A*)
  - Only explores viable solutions

Most AI search procedures are informed search procedures; the search spaces are usually too big to consider brute force search.
Searching & Planning

Some Path

Search

Depth-First
Hill Climbing

Breadth-First
Beam
Best-First

British Museum
Branch & Bound
Dynamic Programming
A*

Games

Minimax
Alpha-Beta Pruning
Heuristic Pruning
Heuristic Continuation

Source: Artificial Intelligence, P. Winston, Addison-Wesley, 1984
A*: Another Perspective

- Form a queue of partial paths. Let the initial queue consist of the zero-length, zero-step path from the starting position to nowhere.
- Until the queue is empty or the goal is reached, determine if the first path in the queue reaches the goal.
  - If the first path reaches the goal, do nothing.
  - If the first path does not reach the goal:
    - Remove the first path from the queue.
    - Form new paths from the removed path by extending one step.
  - Add the new paths to the queue.
  - Sort the queue by the sum of cost accumulated so far and a lower-bound (manhattan distance bound) of the cost remaining, with least cost paths in front.
    - If two or more paths reach a common node, delete all those paths except for one that reaches the common nodes with the minimum cost.
- If the goal node has been reached, announce success; otherwise announce failure.
Hill-Climb Search

- Enter the starting position on queue
- Until the queue is empty or the goal has been reached, determine if the first element is the goal.
  - If the first element is the goal, do nothing
  - If the first element is not the goal node, remove the first element from the queue, sort the first element’s children, if any, by the estimated remaining distance, and add the sorted set of children to the front of the queue (least cost in front).
- If the goal has been reached, announce success, otherwise announce failure.
Adversarial Search

- All the search procedures considered so far assume that the goal is static.
- However, this is not a realistic assumption in games.
  - In games the opponent reacts to the player’s moves in such a way as to maximize his/hers own gain.
  - We need search procedures that take this into account.
- In adversarial search we want to take the possible reactions of the opponent into account:
  - Assume that opponent is rational, that is, the opponent wants to maximize their own gain.
  - This means when searching for alternatives we want to select alternatives that maximize our own gain but minimize that of the opponent → minimax search
Figure 4-15. Games provide a search environment with a new twist, competition. The nodes represent game situations, and the branches represent the moves that connect them.
Minimax