Cost function *g*

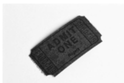
- actual cost from *s* to *v* along the cheapest path found so far
  - exact cost if *G* is a tree
  - can never underestimate the cost if *G* is a general graph
- *f*(*v*) = *g*(*s* ~> *v*) and unit cost → breadth-first search
- *f*(*v*) = –*g*(*s* ~> *v*) and unit cost → depth-first search

Heuristic function *h*

- carries information from outside the graph
- defined for the problem domain
- the closer to the actual cost, the less superfluous vertices are expanded
- *f*(*v*) = *g*(*s* ~> *v*) → cheapest-first search
- *f*(*v*) = *h*(*v* ~> *r*) → best-first search

Admissibility

- let Algorithm A be a best-first search using the evaluation function *f*
- search algorithm is *admissible* if it finds the minimal path (if it exists)
  - if $f = f^*$, Algorithm A is admissible
- Algorithm A* = Algorithm A using an estimate function *h*
  - A* is admissible, if *h* does not overestimate the actual cost



Monotonicity

- *h* is locally admissible → *h* is monotonic
- monotonic heuristic is also admissible
- actual cost is never less than the heuristic cost → *f* will never decrease
- monotonicity → A* finds the shortest path to any vertex the first time it is expanded
  - if a vertex is rediscovered, path will not be shorter
  - simplifies implementation

Optimality

- Optimality theorem: The first path from *s* to *r* found by A* is optimal.
- Proof: textbook p. 105

Informedness

- the more closely *h* approximates h^*, the better A* performs
- if A_1 using h_1 will never expand a vertex that is not also expanded by A_2 using h_2, A_1 is more informed that A_2
- informedness → no other search strategy with *the same amount of outside knowledge* can do less work than A* and be sure of finding the optimal solution

Algorithm A*

- because of monotonicity
  - all weights must be positive
  - closed list can be omitted
- the path is constructed from the mapping π starting from the goal vertex
  - $s \rightarrow \ldots \rightarrow \pi(\pi(\pi(r))) \rightarrow \pi(\pi(r)) \rightarrow \pi(r) \rightarrow r$

Practical considerations

- computing *h*
  - despite the extra vertices expanded, less informed *h* may yield computationally less intensive implementation
- suboptimal solutions
  - by allowing overestimation A* becomes inadmissible, but the results may be good enough for practical purposes

Realizing the movement

- movement through the waypoints
  - unrealistic: does not follow the game world geometry
  - aesthetically displeasing: straight lines and sharp turns
- improvements
  - line-of-sight testing
  - obstacle avoidance
- combining path finding to user-interface
  - real-time response

Recapitulation

1. discretization of the game world
   - grid, navigation mesh
   - waypoints, connections, costs
2. path finding in a graph
   - Algorithm A*
3. realizing the movement
   - geometric corrections
   - aesthetic improvements

Alternatives?

- Although this is the *de facto* approach in (commercial) computer games, are there alternatives?
- possible answers
  - AI processors (unrealistic?)
  - robotics: reactive agents (unintelligent?)
  - analytical approaches (inaccessible?)