CSCI 104L Lecture 22: A* Search

A* search is a heuristic search. That means that it uses “rules of thumb” to often improve the runtime. It never runs worse than Dijkstra, and always finds the correct solution, but it requires more information (which you may not have readily available).

Figure 1: Use A* to find the shortest path for Pac-man to reach the power pellet.

• A* modifies Dijkstra’s so that we always next explore the node with smallest \( d[v]+h[v] \), where \( h[v] \) is our estimate of how far \( v \) is from the destination.

• This only works if our heuristic never over-estimates.

• Our heuristic, when it is wrong, must underestimate. We want it to be as accurate as possible however.

The two extremes:

• \( h(v) = 0 \).

• \( h(v) = \) actual distance.

A simple heuristic for A* is Manhattan Distance. This works well for the 16-tile puzzle, and Pac-man. There are better heuristics, but Manhattan Distance is good and simple enough for you to code up yourself.
Balanced Binary Search Trees

**Question 1.** How would you insert a new value into a (non-balanced) binary search tree?

**Question 2.** For the integers 1 through 7, is there an order you can insert them into an (initially empty) binary search tree such that any search will be efficient?

**Question 3.** For the same set, is there an order that will provide a “bad” binary search tree?

Deleting from a Binary Search Tree

Consider the following binary search tree:

```
  44
 /   \
17    62
|    /  \
32  50    78
|  /    |  /    |
48 54  88
```

**Question 4.** Starting from the above tree, what does the tree look like if we remove 32?

**Question 5.** What if we had deleted 17 instead?

**Question 6.** What if we had deleted 44 instead?

**Question 7.** What if we had deleted 62 instead?