Austen Hsiao  
Programming 1 CS541

**BEST-FIRST**

My best-first implementation uses 3 different cost functions. For the first 2 cases, I used a uniform cost—1 and 5, respectively. As expected, the results are the same and match what we would expect for breadth-first search. For the third case, the cost is determined by a random number between 0 and 5. For all cases and trials, the maximum number of moves was set at 1000.

**Case1 (Cost = 1)**

|  |  |  |
| --- | --- | --- |
| Input | Solution Path | Moves |
| ['1','5','2','4','b','3','7','8','6'] | 1 5 2 4 b 3 7 8 6 --> 1 b 2 4 5 3 7 8 6 --> 1 2 b 4 5 3 7 8 6 --> 1 2 3 4 5 b 7 8 6 --> 1 2 3 4 5 6 7 8 b | 63 |
| ['b','1','2','4','5','3','7','8','6'] | b 1 2 4 5 3 7 8 6 --> 1 b 2 4 5 3 7 8 6 --> 1 2 b 4 5 3 7 8 6 --> 1 2 3 4 5 b 7 8 6 --> 1 2 3 4 5 6 7 8 b | 63 |
| ['b','2','3','1','4','6','7','5','8'] | b 2 3 1 4 6 7 5 8 --> 1 2 3 b 4 6 7 5 8 --> 1 2 3 4 b 6 7 5 8 --> 1 2 3 4 5 6 7 b 8 --> 1 2 3 4 5 6 7 8 b | 42 |
| ['1','2','3','b','8','5','4','7','6'] | 1 2 3 b 8 5 4 7 6 --> 1 2 3 4 8 5 b 7 6 --> 1 2 3 4 8 5 7 b 6 --> 1 2 3 4 b 5 7 8 6 --> 1 2 3 4 5 b 7 8 6 --> 1 2 3 4 5 6 7 8 b | 180 |
| ['2','b','3','1','5','6','4','7','8'] | 2 b 3 1 5 6 4 7 8 --> b 2 3 1 5 6 4 7 8 --> 1 2 3 b 5 6 4 7 8 --> 1 2 3 4 5 6 b 7 8 --> 1 2 3 4 5 6 7 b 8 --> 1 2 3 4 5 6 7 8 b | 255 |
|  | Average moves: 121 |  |

**Case2 (Cost = 5)**

|  |  |  |
| --- | --- | --- |
| Input | Solution Path | Moves |
| [‘1’,’5’,’2’,’4’,’b’,’3’,’7’,’8’,’6’] | 1 5 2 4 b 3 7 8 6 🡪 1 b 2 4 5 3 7 8 6 🡪 1 2 b 4 5 3 7 8 6 🡪 1 2 3 4 5 b 7 8 6 🡪 1 2 3 4 5 6 7 8 b | 63 |
| [‘b’,’1’,’2’,’4’,’5’,’3’,’7’,’8’,’6’] | b 1 2 4 5 3 7 8 6 🡪 1 b 2 4 5 3 7 8 6 🡪 1 2 b 4 5 3 7 8 6 🡪 1 2 3 4 5 b 7 8 6 🡪 1 2 3 4 5 6 7 8 b | 63 |
| [‘b’,’2’,’3’,’1’,’4’,’6’,’7’,’5’,’8’] | b 2 3 1 4 6 7 5 8 🡪 1 2 3 b 4 6 7 5 8 🡪 1 2 3 4 b 6 7 5 8 🡪 1 2 3 4 5 6 7 b 8 🡪 1 2 3 4 5 6 7 8 b | 42 |
| [‘1’,’2’,’3’,’b’,’8’,’5’,’4’,’7’,’6’] | 1 2 3 b 8 5 4 7 6 🡪 1 2 3 4 8 5 b 7 6 🡪 1 2 3 4 8 5 7 b 6 🡪 1 2 3 4 b 5 7 8 6 🡪 1 2 3 4 5 b 7 8 6 🡪 1 2 3 4 5 6 7 8 b | 180 |
| [‘2’,’b’,’3’,’1’,’5’,’6’,’4’,’7’,’8’] | 2 b 3 1 5 6 4 7 8 🡪 b 2 3 1 5 6 4 7 8 🡪 1 2 3 b 5 6 4 7 8 🡪 1 2 3 4 5 6 b 7 8 🡪 1 2 3 4 5 6 7 b 8 🡪 1 2 3 4 5 6 7 8 b | 255 |
|  | Average moves: 121 |  |

**Case3(Cost = random[0,5])**

|  |  |  |
| --- | --- | --- |
| Input | Solution Path | Moves |
| [‘1’,’5’,’2’,’4’,’b’,’3’,’7’,’8’,’6’] | 1 5 2 4 b 3 7 8 6 🡪 1 b 2 4 5 3 7 8 6 🡪 1 2 b 4 5 3 7 8 6 🡪 1 b 2 4 5 3 7 8 6 🡪 1 2 b 4 5 3 7 8 6 🡪 1 b 2 4 5 3 7 8 6 🡪 1 2 b 4 5 3 7 8 6 🡪 1 2 3 4 5 b 7 8 6 🡪 1 2 3 4 5 6 7 8 b | 778 |
| [‘b’,’1’,’2’,’4’,’5’,’3’,’7’,’8’,’6’] | b 1 2 4 5 3 7 8 6 🡪 1 b 2 4 5 3 7 8 6 🡪 1 5 2 4 b 3 7 8 6 🡪 1 b 2 4 5 3 7 8 6 🡪 1 2 b 4 5 3 7 8 6 🡪 1 2 3 4 5 b 7 8 6 🡪 1 2 3 4 5 6 7 8 b | 46 |
| [‘b’,’2’,’3’,’1’,’4’,’6’,’7’,’5’,’8’] | b 2 3 1 4 6 7 5 8 🡪 1 2 3 b 4 6 7 5 8 🡪 1 2 3 7 4 6 b 5 8 🡪 1 2 3 7 4 6 5 b 8 🡪 1 2 3 7 b 6 5 4 8 🡪 1 2 3 b 7 6 5 4 8 🡪 1 2 3 5 7 6 b 4 8 🡪 1 2 3 5 7 6 4 b 8 🡪 1 2 3 5 7 6 b 4 8 🡪 1 2 3 5 7 6 4 b 8 🡪 1 2 3 5 b 6 4 7 8 🡪 1 2 3 5 6 b 4 7 8 🡪 1 2 3 5 b 6 4 7 8 🡪 1 2 3 b 5 6 4 7 8 🡪 1 2 3 4 5 6 b 7 8 🡪 1 2 3 b 5 6 4 7 8 🡪 1 2 3 4 5 6 b 7 8 🡪 1 2 3 b 5 6 4 7 8 🡪 1 2 3 5 b 6 4 7 8 🡪 1 2 3 b 5 6 4 7 8 🡪 1 2 3 5 b 6 4 7 8 🡪 1 2 3 b 5 6 4 7 8 🡪 1 2 3 4 5 6 b 7 8 🡪 1 2 3 b 5 6 4 7 8 🡪 1 2 3 4 5 6 b 7 8 🡪 1 2 3 4 5 6 7 b 8 🡪 1 2 3 4 5 6 7 8 b | 471 |
| [‘1’,’2’,’3’,’b’,’8’,’5’,’4’,’7’,’6’] | 1 2 3 b 8 5 4 7 6 🡪 1 2 3 8 b 5 4 7 6 🡪 1 b 3 8 2 5 4 7 6 🡪 b 1 3 8 2 5 4 7 6 🡪 1 b 3 8 2 5 4 7 6 🡪 1 2 3 8 b 5 4 7 6 🡪 1 2 3 8 5 b 4 7 6 🡪 1 2 3 8 5 6 4 7 b 🡪 1 2 3 8 5 b 4 7 6 🡪 1 2 3 8 b 5 4 7 6 🡪 1 2 3 8 7 5 4 b 6 🡪 1 2 3 8 7 5 4 6 b 🡪 1 2 3 8 7 5 4 b 6 🡪 1 2 3 8 b 5 4 7 6 🡪 1 2 3 8 5 b 4 7 6 🡪 1 2 3 8 b 5 4 7 6 🡪 1 2 3 b 8 5 4 7 6 🡪 1 2 3 4 8 5 b 7 6 🡪 1 2 3 4 8 5 7 b 6 🡪 1 2 3 4 b 5 7 8 6 🡪 1 2 3 4 5 b 7 8 6 🡪 1 2 3 4 5 6 7 8 b | 960 |
| [‘2’,’b’,’3’,’1’,’5’,’6’,’4’,’7’,’8’] | Solution Not Found within 1000 moves | |
|  | Average moves: 451 | |

The random cost function obviously performed worst. In all three cases, the cost is not a function of any heuristic. Using random numbers as the cost doesn’t make a whole lot of sense but it’s very clear to see that random searching is not a good idea. The conclusion is that best-first search with uniform values (BFS) does an OK job of searching. 255 total moves is not terrible, but for a larger pattern (or a different pattern than the ones tested), this could pose performance issues.

**A\***

My A\* cost functions are equal to each of the heuristics (misplaced, Manhattan, sum) + 5. Basically, I’m treating the base cost for each move as ‘5’, then tacking on a heuristic function.

**Case1 (Cost = 5 + misplaced(n))**

|  |  |  |
| --- | --- | --- |
| Input | Solution Path | Moves |
| ['1','5','2','4','b','3','7','8','6'] | 1 5 2 4 b 3 7 8 6 --> 1 b 2 4 5 3 7 8 6 --> 1 2 b 4 5 3 7 8 6 --> 1 2 3 4 5 b 7 8 6 --> 1 2 3 4 5 6 7 8 b | 33 |
| ['b','1','2','4','5','3','7','8','6'] | b 1 2 4 5 3 7 8 6 --> 1 b 2 4 5 3 7 8 6 --> 1 2 b 4 5 3 7 8 6 --> 1 2 3 4 5 b 7 8 6 --> 1 2 3 4 5 6 7 8 b | 28 |
| ['b','2','3','1','4','6','7','5','8'] | b 2 3 1 4 6 7 5 8 --> 1 2 3 b 4 6 7 5 8 --> 1 2 3 4 b 6 7 5 8 --> 1 2 3 4 5 6 7 b 8 --> 1 2 3 4 5 6 7 8 b | 17 |
| ['1','2','3','b','8','5','4','7','6'] | 1 2 3 b 8 5 4 7 6 --> 1 2 3 4 8 5 b 7 6 --> 1 2 3 4 8 5 7 b 6 --> 1 2 3 4 b 5 7 8 6 --> 1 2 3 4 5 b 7 8 6 --> 1 2 3 4 5 6 7 8 b | 86 |
| ['2','b','3','1','5','6','4','7','8'] | 2 b 3 1 5 6 4 7 8 --> b 2 3 1 5 6 4 7 8 --> 1 2 3 b 5 6 4 7 8 --> 1 2 3 4 5 6 b 7 8 --> 1 2 3 4 5 6 7 b 8 --> 1 2 3 4 5 6 7 8 b | 149 |
|  | Average moves: 63 |  |

**Case2 (Cost = 5 + Manhattan(n))**

|  |  |  |
| --- | --- | --- |
| Input | Solution Path | Moves |
| ['1','5','2','4','b','3','7','8','6'] | 1 5 2 4 b 3 7 8 6 --> 1 b 2 4 5 3 7 8 6 --> 1 2 b 4 5 3 7 8 6 --> 1 2 3 4 5 b 7 8 6 --> 1 2 3 4 5 6 7 8 b | 5 |
| ['b','1','2','4','5','3','7','8','6'] | b 1 2 4 5 3 7 8 6 --> 1 b 2 4 5 3 7 8 6 --> 1 2 b 4 5 3 7 8 6 --> 1 2 3 4 5 b 7 8 6 --> 1 2 3 4 5 6 7 8 b | 5 |
| ['b','2','3','1','4','6','7','5','8'] | b 2 3 1 4 6 7 5 8 --> 1 2 3 b 4 6 7 5 8 --> 1 2 3 4 b 6 7 5 8 --> 1 2 3 4 5 6 7 b 8 --> 1 2 3 4 5 6 7 8 b | 5 |
| ['1','2','3','b','8','5','4','7','6'] | 1 2 3 b 8 5 4 7 6 --> 1 2 3 4 8 5 b 7 6 --> 1 2 3 4 8 5 7 b 6 --> 1 2 3 4 b 5 7 8 6 --> 1 2 3 4 5 b 7 8 6 --> 1 2 3 4 5 6 7 8 b | 6 |
| ['2','b','3','1','5','6','4','7','8'] | 2 b 3 1 5 6 4 7 8 --> b 2 3 1 5 6 4 7 8 --> 1 2 3 b 5 6 4 7 8 --> 1 2 3 4 5 6 b 7 8 --> 1 2 3 4 5 6 7 b 8 --> 1 2 3 4 5 6 7 8 b | 8 |
|  | Average moves: 6 |  |

**Case3(Cost = 5 + Manhattan(n) + misplaced(n))**

|  |  |  |
| --- | --- | --- |
| Input | Solution Path | Moves |
| ['1','5','2','4','b','3','7','8','6'] | 1 5 2 4 b 3 7 8 6 --> 1 b 2 4 5 3 7 8 6 --> 1 2 b 4 5 3 7 8 6 --> 1 2 3 4 5 b 7 8 6 --> 1 2 3 4 5 6 7 8 b | 5 |
| ['b','1','2','4','5','3','7','8','6'] | b 1 2 4 5 3 7 8 6 --> 1 b 2 4 5 3 7 8 6 --> 1 2 b 4 5 3 7 8 6 --> 1 2 3 4 5 b 7 8 6 --> 1 2 3 4 5 6 7 8 b | 5 |
| ['b','2','3','1','4','6','7','5','8'] | b 2 3 1 4 6 7 5 8 --> 1 2 3 b 4 6 7 5 8 --> 1 2 3 4 b 6 7 5 8 --> 1 2 3 4 5 6 7 b 8 --> 1 2 3 4 5 6 7 8 b | 5 |
| ['1','2','3','b','8','5','4','7','6'] | 1 2 3 b 8 5 4 7 6 --> 1 2 3 4 8 5 b 7 6 --> 1 2 3 4 8 5 7 b 6 --> 1 2 3 4 b 5 7 8 6 --> 1 2 3 4 5 b 7 8 6 --> 1 2 3 4 5 6 7 8 b | 7 |
| ['2','b','3','1','5','6','4','7','8'] | 2 b 3 1 5 6 4 7 8 --> b 2 3 1 5 6 4 7 8 --> 1 2 3 b 5 6 4 7 8 --> 1 2 3 4 5 6 b 7 8 --> 1 2 3 4 5 6 7 b 8 --> 1 2 3 4 5 6 7 8 b | 8 |
|  | Average moves: 6 |  |

The results for A\* are better across the board when compared to best-first. Within A\*, the ‘misplaced’ heuristic performed the worst, followed by the sum of ‘Manhattan’ and ‘misplaced’, then ‘Manhattan’. This suggests that the ‘Manhattan’ heuristic does a better job of determining cost-to-goal than the other 2 functions.

**DATA FOR 15-puzzle**

\*All cost functions corresponding to each case are the same as before. The move limit has been increased from 1,000 to 10,000**.**

**BEST-FIRST**

**CASE1**

|  |  |
| --- | --- |
| Input | Moves |
| ['b','1','2','3','5','6','7','4','9','10','11','8','13','14',15',12'] | 772 |
| ['1','2','3','4','5','b','6','8','9',11','7','12','13','10','14','15'] | 1705 |
| ['1','2','3','4','5','6','7','8','9','10','12','b','13','14','11','15'] | 34 |
| ['1','2','3','4','5','10','6','7','9','14',11','8','13','b','15','12'] | 538 |
| ['1','2','3','4','5','b','6','7','9',10','11','8','13','14','15','12'] | 167 |
| Average | 643 |

**CASE2**

|  |  |
| --- | --- |
| Input | Moves |
| ['b','1','2','3','5','6','7','4','9','10','11','8','13','14',15',12'] | 772 |
| ['1','2','3','4','5','b','6','8','9',11','7','12','13','10','14','15'] | 1705 |
| ['1','2','3','4','5','6','7','8','9','10','12','b','13','14','11','15'] | 34 |
| ['1','2','3','4','5','10','6','7','9','14',11','8','13','b','15','12'] | 538 |
| ['1','2','3','4','5','b','6','7','9',10','11','8','13','14','15','12'] | 167 |
| Average | 643 |

**CASE3**

|  |  |
| --- | --- |
| Input | Moves |
| ['b','1','2','3','5','6','7','4','9','10','11','8','13','14',15',12'] | Solution not found |
| ['1','2','3','4','5','b','6','8','9',11','7','12','13','10','14','15'] | Solution not found |
| ['1','2','3','4','5','6','7','8','9','10','12','b','13','14','11','15'] | Solution not found |
| ['1','2','3','4','5','10','6','7','9','14',11','8','13','b','15','12'] | Solution not found |
| ['1','2','3','4','5','b','6','7','9',10','11','8','13','14','15','12'] | 2528 |
| Average | 2528 |

**A\***

**CASE1**

|  |  |
| --- | --- |
| Input | Moves |
| ['b','1','2','3','5','6','7','4','9','10','11','8','13','14',15',12'] | 260 |
| ['1','2','3','4','5','b','6','8','9',11','7','12','13','10','14','15'] | 652 |
| ['1','2','3','4','5','6','7','8','9','10','12','b','13','14','11','15'] | 15 |
| ['1','2','3','4','5','10','6','7','9','14',11','8','13','b','15','12'] | 253 |
| ['1','2','3','4','5','b','6','7','9',10','11','8','13','14','15','12'] | 57 |
| Average | 248 |

**CASE2**

|  |  |
| --- | --- |
| Input | Moves |
| ['b','1','2','3','5','6','7','4','9','10','11','8','13','14',15',12'] | 7 |
| ['1','2','3','4','5','b','6','8','9',11','7','12','13','10','14','15'] | 10 |
| ['1','2','3','4','5','6','7','8','9','10','12','b','13','14','11','15'] | 5 |
| ['1','2','3','4','5','10','6','7','9','14',11','8','13','b','15','12'] | 7 |
| ['1','2','3','4','5','b','6','7','9',10','11','8','13','14','15','12'] | 5 |
| Average | 7 |

**CASE3**

|  |  |
| --- | --- |
| Input | Moves |
| ['b','1','2','3','5','6','7','4','9','10','11','8','13','14',15',12'] | 7 |
| ['1','2','3','4','5','b','6','8','9',11','7','12','13','10','14','15'] | 11 |
| ['1','2','3','4','5','6','7','8','9','10','12','b','13','14','11','15'] | 5 |
| ['1','2','3','4','5','10','6','7','9','14',11','8','13','b','15','12'] | 9 |
| ['1','2','3','4','5','b','6','7','9',10','11','8','13','14','15','12'] | 5 |
| Average | 8 |

**Conclusion:**

A\* is better than best-first, especially if we have a good cost-to-goal function

END