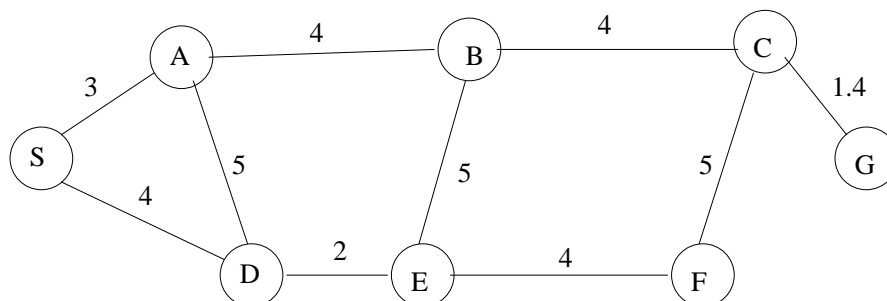


Problem assignment 2

Due: Thursday, September 24, 2009

Problem 1

Consider the following graph that represents road connections between different cities. The weights on links represent driving distances between connected cities. Let S be the initial city and G the destination.



Part a. Show how the uniform search tree works by giving **the order in which nodes are expanded**. Is the path found by the algorithm optimal?

Part b. Assume the following set of the straight line distances between G and other cities.

S	A	B	C	D	E	F
10	10	6	1.4	9	7	2

Show how the greedy search algorithm with the straight-line distance heuristic works. Is the path the algorithm finds optimal?

Part c. Show how the A* with the straight-line distance heuristic works. Is the path found optimal?

Problem 2. Heuristic search for the 8-puzzle problem.

In this problem we will explore heuristic search solutions for the 8-puzzle problem. We will use evaluation-function driven search procedure to incorporate various exploration strategies. The procedure searches the space by expanding the nodes with the minimum value first.

The search-tree NODE structure defined in files *search.c* and *search.h* has been modified and uses three attributes: *g_value*, *h_value* and *f_value* storing the values of $g(n)$, $h(n)$ and $f(n)$ functions respectively. *g_value* is the cost of the path from the initial node to the current node, *h_value* reflects the estimate of the cost from the current node to the goal state, and *f_value* is the estimate of the total path cost through that node n .

Part a. Uniform cost search

You are given code that implements a general path-search procedure driven by the evaluation function values. The procedure is called *f_search* and it is included in file *main2a.c*. The procedure searches the space by expanding the nodes with the minimum *f_value* first.

The code we gave you is flexible in that you can later define your own evaluation and heuristic functions by modifying *h_function* and *f_function* code. The first one computes the *h_value*, the second the *f_value* of the node. *main2a.c* implements the uniform cost search, that is $h(n) = 0$ and $f(n) = g(n)$.

Remark: The uniform search algorithm for the puzzle-8 problem in fact implements the breadth-first search since all operator costs are one. The difference is that we simulate the breadth-first search through a more flexible evaluation-function representation and priority queue operations.

Part b. Uniform cost search with the elimination of redundant nodes

Write *main2b.c* program that enhances the uniform cost search with the value-driven elimination of redundant nodes. The node should be eliminated whenever a node associated with the same state has been expanded before with a lower evaluation function value *f_value*. Use procedure *mark_node_value* defined in *mark.h* and *mark.c* to accomplish this task. Note that these files are different from *mark.c* and *mark.h* files in PS-1. Your program should run on all three test examples and collect the following statistics:

- the total number of nodes expanded;

- the total number of nodes generated;
- the maximum length of the queue structure;
- the length of the solution path (number of moves)

Part c. A* algorithm with the misplaced tile heuristic

The *f_search* function and the program in part b. can be easily turned into an A* algorithm by supplying an appropriate heuristic function $h(n)$.

Write a *main2c.c* program that performs A* search by modifying *main2a.c* with a new *f_function* that computes the *f_value* of the node, and a new *h_function* that computes the estimate of the remaining distance to the solution, or *h_value* of the node. The heuristic function, *h_function*, should compute the number of misplaced tiles of the current configuration (heuristic h_1 on page 106 of the textbook). Use *f_value* marking to eliminate the redundant nodes.

The program should run on all three test examples and collect the same set of statistics as above.

Part d. Greedy search with the misplaced tile heuristic

Write a program *main2d.c* that implements greedy search in which the exploration is driven purely by the misplaced tile heuristic (heuristic h_1 on page 106 of the textbook). Run it on all three test examples and collect the same statistics as above. Do not forget to use *f_value* marking to eliminate the state repeats.

Part e. A* with Manhattan distance heuristic

Write program *main2e.c* that relies on the Manhattan distance heuristic (heuristic h_2 on page 106 of the textbook) to implement A* . You should write the code for the Manhattan distance heuristic and place it in the file *h2.c*. Use *f_value* marking to eliminate the redundant nodes in your programs.

Part f. Analysis of results

Analyze the performance of all methods (parts a through e) in terms of the collected statistics and include the analysis in the report. You should:

- Summarize the results of the methods in different tables, one table for every configuration tested: Uniform cost search with elimination of redundant nodes, A^* with misplaced tile heuristic, A^* with Manhattan distance heuristics, Greedy with misplaced tile.
- Which method is the best in terms of respective statistics? Explain why.
- State which heuristic would you suggest to use and explain why.

In addition, answer the following questions.

- Would A^* work without marking? Why or why not?
- Would greedy method work without marking? Why or why not?
- Assume we create a heuristic function h_3 such that it averages the values of the misplaced tile heuristic (h_1) and the Manhattan distance heuristic (h_2):

$$h_3(n) = \frac{1}{2} [h_1(n) + h_2(n)].$$

Is h_3 an admissible heuristic?