CS322 Fall 1997
Assignment 4

Solution

Question 1

In this question we will investigate using graph searching to design video presentations.

Suppose we represent a node as the term

$vp(To\_Cover, Segs)$

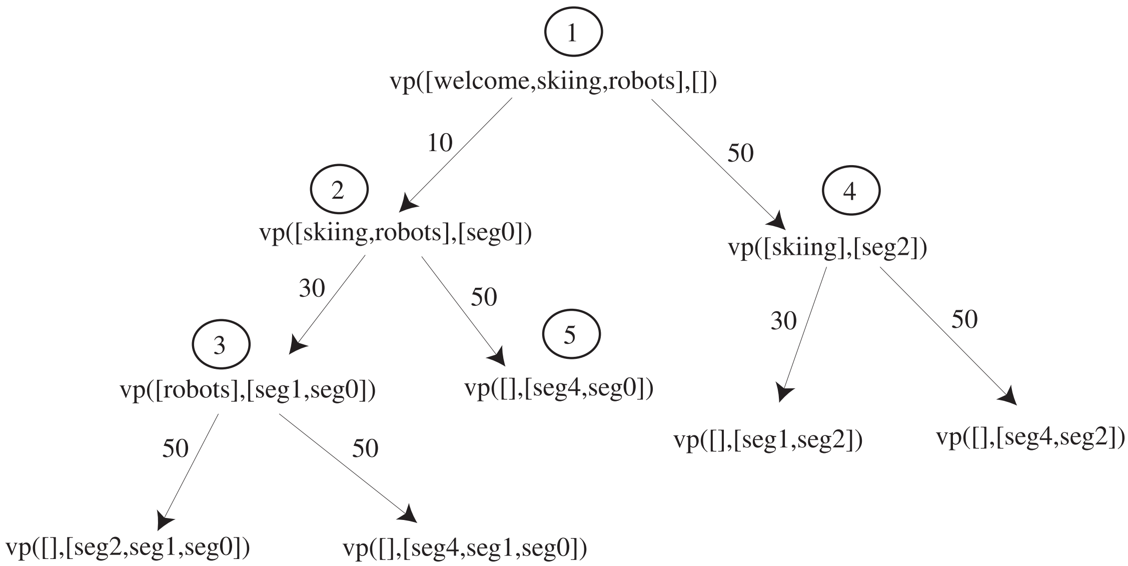
where *Segs* is a list of segments that must be in the presentation, and *To_Cover* is a list of topics that also must be covered, such that none of the segments in *Segs* covers any of the topics in *To_Cover*.

The neighbors of a node are obtained by first selecting a topic from *To_Cover*. There is a neighbor for each segment that covers the selected topic.

Given that the goal is to design a presentation that covers all of the topics in *MustCover*, the starting node is $vp(MustCover, [])$, and the goal nodes are of the form $vp([], Presentation)$. The length of the path from a start node to a goal node the time of the presentation. An optimal presentation is then the shortest presentation that covers all of the topics in *MustCover*.

1. [6 marks] Suppose that the goal is to cover the topics $[welcome, skiing, robots]$. Suppose you always select the leftmost topic to find the neighbors for each node. Draw the search space expanded for a lowest-cost-first search until the first solution is found. This should show all nodes expanded, show which node is a goal node, and show the frontier when the goal was expanded.

   Here is the solution. The nodes are expanded according to the circled numbers. The circled (5) is the first goal node found — this is the shortest presentation that covers all the topics.



2. [6 marks] Give a non-trivial heuristic function h that is an underestimate of the real cost. [Note that $h(n) = 0$ for all n is the trivial heuristic function.] Does it satisfy the monotone restriction for a heuristic function?

   Solution #1: For each topic, t, let $s(t)$ be the length of the smallest segment that covers topic t. Let $h(vp(TC, Segs)) = \max_{t \in TC} s(t)$. That is, we find the topic t for which $s(t)$ is maximum.

   Yes, it satisfies the monotone restriction as the actual distance between any two nodes is the sum of the length of the topics added between the two nodes (the only nodes that have actual distances that those where there is a path from one to the other). The difference between the h-values of the two nodes must be less than the time of the segments added to solve that goal.

   Solution #2: For each segment let the *contribution* of the segment be the time of the segment divided by the number of topics the segment covers. For each topic, t, let $s(t)$ be the smallest contribution for all of the segments that covers the topic. Let $h(vp(TC, Segs)) = \sum_{t \in TC} s(t)$. That is, we sum $s(t)$ for all of the topics t in TC.

   The intuition is that each topic t requires at least $s(t)$ time. Note that we need to divide by the number of topics the segment covers to make sure that we don't double count the time for segments added that cover multiple topics.

   Yes it satisfies the monotone restriction (for similar arguments to those above).

   Both of these solution require one pass through the segment database to build the $s(t)$ function, but once this is built, the heuristic function can be computed in time proportional to the length of the *To_cover* list.

Question 2

The aim of this question is to get a bit more practice writing simple logic programs.

1. [4 marks] Write a relation $remove(E, L, R)$ that is true if R is the resulting list of removing one instance of E from list L. The relation is false if E isn't a member of L.

   Solution:

   ```
   remove(E,[E|R],R).
   remove(E,[H|T],[H|R]) <-
      remove(E,T,R).
   ```

   What are all of the answers to the following queries:

   ask $remove(a, [b, a, d, a], R)$.
   ask $remove(E, [b, a, d, a], R)$.
   ask $remove(E, L, [b, a, d])$.
   ask $remove(p(X), [a, p(a), p(p(a)), p(p(p(a)))], R)$.

   Just run this.

2. [4 marks] Write a relation $subsequence(L1, L2)$ that is true if list $L1$ contains a subset of the elements of $L2$ in the same order.

   Solution:

   ```
   subsequence([],[]).
   subsequence([X|L],[X|R]) <-
      subsequence(L,R).
   subsequence(L,[_|R]) <-
      subsequence(L,R).
   ```

   How many different proofs are there for each of the following queries:

   ask $subsequence([b, a], [b, a, d, a])$.

   There are 2 answers to this because the a can either refer to the first a or the second a.

   ask $subsequence([X, Y], [b, a, d, a])$.

   There are C_2^4 solutions, This is the number of way to choose 2 elements from a set of 4. (You are choosing two positions from the set of all positions).

   ask $subsequence(S, [b, a, d, a])$.

   There are 2^4 solutions. This is the number of subsets of 4 elements.

Question 3

For each of the questions in this assignment, please estimate the amount of time you spent on it. What this a reasonable amount of time for one week in one course?

Solution: Question 1: about 1 hour. Question 2: about 1 hour. Question 3: about 2 minutes. I would expect more difficult assignment for a 3rd-year course.

Last weeks assignment was more difficult, but not unreasonable.