Problem Solving in Computer Science: Week 2

Scribe on duty: Tatjana Petrov

September 29, 2008

1 Project requirements

Each team should define two functions for the given SSG, and then evaluate them practically and theoretically.

Defining the functions. In order to find the optimal permutation which implies the solution of a given SSG as explained in the previous lecture, define the two polynomial functions *first* and *next* with following inputs and outputs:

	INPUT	OUTPUT
first	SSG	random permutation r_0
next	SSG, random permutation r_i	random permutation r_{i+1}.

Remark: The input for next function may be the history of all the previous permutations. The functions should be defined in such a way that, starting form the first, the iteration of *next* must generate all random permutations until an optimal is found.

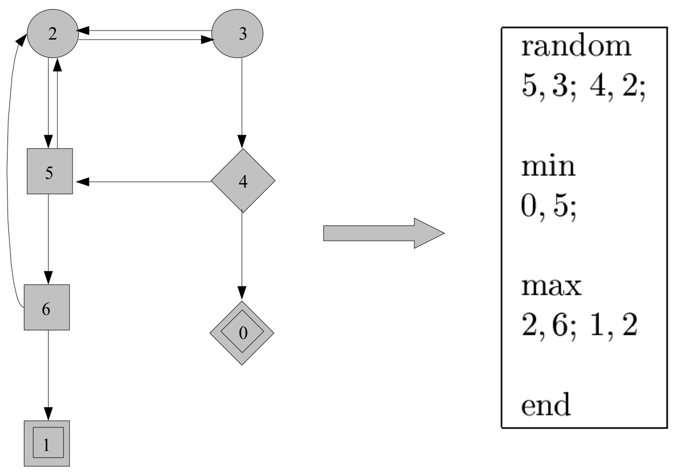
Once we have these functions, we want to evaluate them practically and theoretically.

Practical evaluation. We do not want to measure the efficiency of solving the game, but how long it takes to find the right permutation. We will count *how many iterations* are needed to find the optimal permutation.

Input format for SSG is shown in Figure 1. The nodes of the graph are numbered in the following way: the target *min* node gets number 0, the target *max* node gets number 1. The remaining nodes are enumerated according to their position in the file, where the keywords *random*, *min*, and *max* define the node types. In the SSG shown in Figure 1, we see that the *random* node 2 has a left successor labeled with 4, and right successor labeled with 2. The left successor of *max* node 5 is 2, and the right successor is 6.

Some sample graphs will be put on the web during the week. Each team should implement the functions. Teams will compete based on how many iterations are needed to compute the optimal strategy. The input games for the competition will not be given in advance.

Theoretical evaluation. Once the functions are implemented, make the theoretical evaluation. This might include the following observations:



```
random
5,3; 4,2;

min
0,5;

max
2,6; 1,2

end
```

Figure 1: Format of the input graph

- Try to characterize classes of SSG's for which the algorithm works well (in polynomial time), and classes of SSG's for which the algorithm needs exponentially many iterations;
- Find exactly the examples where the algorithm works poorly; Try to state this as theorems, and to prove them.
- Define a family of examples with certain properties.

Deliverables

1. Program to participate in the competition
2. A $4 - 8$ pages report, including the practical and theoretical evaluations

Progress reports

Each of the groups reported the ideas for finding the optimal permutation.

Remark: All groups were more concentrated on defining the initial permutation. The advice is to focus more on the *next* function.

TO DO: Define the *next* function, such that it is (1) - polynomial, and that (2) - even starting from the random initial permutation it leads to the solution.

Notes on Technical Writing

Discussion about basic rules of technical writing ([1]).

References

[1] Donald E. Knuth, Tracy Larrabee, and Paul M. Roberts *Mathematical Writing*, Stanford University, 1987.