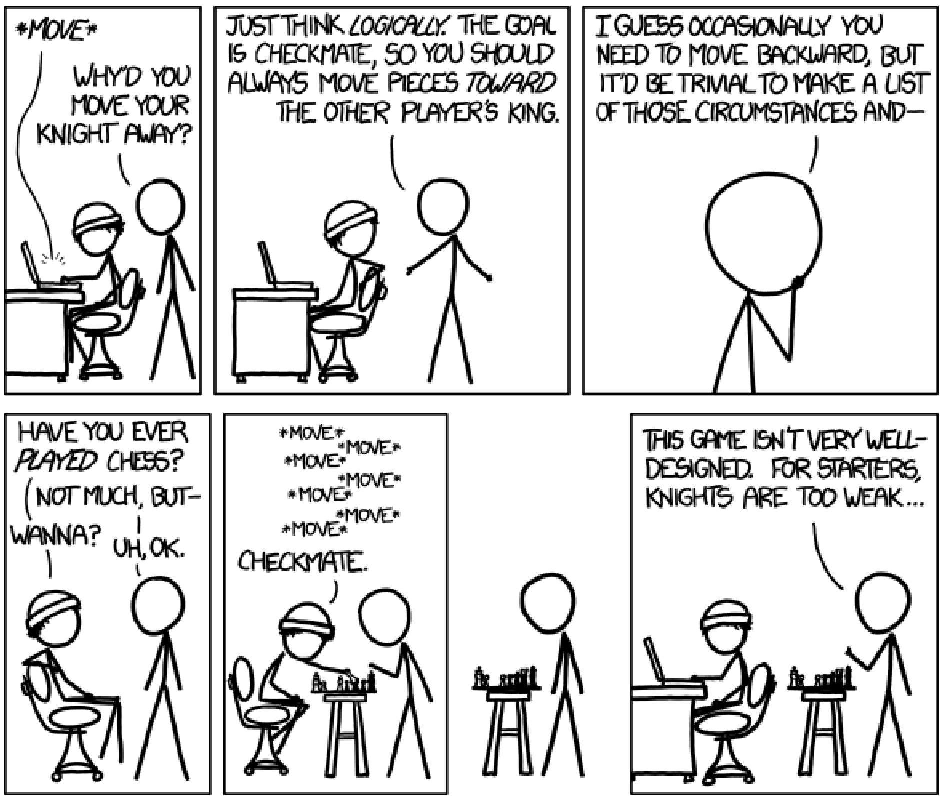
CS52 - Assignment 9

Due Wednesday 5/4 at 7:00pm


MOVE
WHY'D YOU MOVE YOUR KNIGHT AWAY?
JUST THINK *LOGICALLY*. THE GOAL IS CHECKMATE, SO YOU SHOULD ALWAYS MOVE PIECES *TOWARD* THE OTHER PLAYER'S KING.
I GUESS OCCASIONALLY YOU NEED TO MOVE BACKWARD, BUT IT'D BE TRIVIAL TO MAKE A LIST OF THOSE CIRCUMSTANCES AND—
HAVE YOU EVER *PLAYED* CHESS?
NOT MUCH, BUT—
WANNA?
UH, OK.
MOVE *MOVE* *MOVE* *MOVE* *MOVE* *MOVE* *MOVE* *MOVE*
CHECKMATE.
THIS GAME ISN'T VERY WELL-DESIGNED. FOR STARTERS, KNIGHTS ARE TOO WEAK...

`https://xkcd.com/1112/`

For this assignment we will use exhaustive search and branch and bound search to solve five different search problems. Most of the work will be writing the predicate functions, i.e. for exhaustive search a function that takes in a solution and returns `true` or `false` and for branch and bound search a function that takes in a *partial* solution and returns `INCORRECT`, `PENDING` or `CORRECT`. Put your solutions in a file named `assign9.sml` and submit in the usual way.

For this assignment, you may (and in fact, I'd encourage you to) work with a partner. If you work with a partner, you only need to submit one file, just make sure both of your names are at the top of it. **If you choose to do work with a partner, you must both be there whenever you're working on the assignment and you should work on each of the problems collaboratively.** In particular, what is not appropriate is to divide the problems up and then combine your answers into a single submission (or, of course, just have one person do all the work :).

1 Background Reading

We have discussed all of the relevant material for this assignment in class. Like other programming languages, SML does have some built-in functions to help you. For this assignment research the following two built-in functions:

- `ListPair.zip`
- `List.tabulate`

Do a search online to discover what these functions do. They will likely be helpful for this assignment (*hint, hint*).

Starter

I have put together a starter for this assignment that includes a selection of the functions that we have looked at in class that I think will be relevant for this assignment:

`http://www.cs.pomona.edu/~dkauchak/classes/cs52/assignments/assign9/assign9-starter.sml`

Read through all of the code in the starter and make sure that you understand what each function does and the role that it plays in searching.

To include this code in your assignment, add:

```
use "assign9-starter.sml"
```

at the top of your file.

In class, I also discussed a few other helper functions on lazy lists and the `bbSearch` data types. Feel free to copy and paste these into your assignment if you find them useful.

The Problems

We will be trying to solve five different problems in this assignment in different ways (some we've seen already):

– Roomers

Baker, Cooper, Fletcher, Miller, and Smith live on different floors of an apartment house that contains only five floors. Baker does not live on the top floor. Fletcher does not live on either the top or the bottom floor. Miller lives on a higher floor than does Cooper. Smith does not live on a floor adjacent to Fletcher's. Fletcher does not live on a floor adjacent to Cooper's. Where does everyone live?

– Liars

Five highly-competitive computer science majors took an examination. They did not want to reveal the results and agreed that each one would make one true statement and one false one. Here is what they said.

> Daxter: "I aced it. Tingle was second."
>
> Navi: "I was second. Tingle was fourth."
>
> Raiden: "Navi was second. I was only third."
>
> Tingle: "I was third, and Waluigi was last."
>
> Waluigi: "I was next-to-last. Raiden did the best."

What was the actual order of the students' examination scores?

– ***N*** queens

Find a way to place N queens on an $N \times N$ chess board so that no queen is attacking another. Equivalently, place the queens so that no two are on the same row, the same column, or the same diagonal.

– Yachts

Each of five wealthy friends has one daughter and one yacht. Each yacht owner has named his yacht after someone *else's* daughter. Sir Barnacle's yacht is the Gabrielle, Mr. Moore owns the Lorna; Mr. Hall the Rosalind. The Melissa, owned by Colonel Downing, is named after Sir Barnacle's daughter. Mary Ann's father is Mr. Hall. Gabrielle's father owns the yacht that is named after Dr. Parker's daughter. For each father, find the name of the daughter and the yacht.

– Pomona College Computer Science Halloween

Professors Bruce, Bull, Chen, Greenberg, and Wu went trick-or-treating together. Their group included two ghosts, a wizard, Batman, and Robin. At the end of the night, each of them counted the number of pieces of candy they collected. They each collected a different amount of candy: 63, 77, 83, 54, and 84. They also each used a different sorting algorithm to sort their candy from the following sorting algorithms: BubbleSort, InsertionSort, QuickSort, MergeSort, and a Hash Table. Figure out the costume worn by each professor, the amount of candy each collected, and what method they used to sort their candy.

> The ghost with less candy used a Hash Table.
>
> The person with the most candy used InsertionSort.
>
> Professors Bull and Chen collected a total of 161 pieces of candy.
>
> A ghost collected 77 pieces of candy.
>
> 54 pieces of candy were sorted using BubbleSort.
>
> Robin collected 83 pieces of candy.
>
> Professors Bull and Wu collected a total of 140 pieces of candy.
>
> The wizard collected the most pieces of candy.
>
> Professor Bruce used MergeSort.

Part I: Basic Exhaustive Search

Recall from class the basic steps for solving a problem with exhaustive search:

- Come up with a state representation.
- Use the `producer` function to obtain a `lazy_list` of all potential solutions.
- Design a predicate function that returns `true` or `false` to tell if a potential solution is actually correct.
- Use `lazyFilter` and `lazyToList` to create an ordinary list of *all* solutions.

In all cases, your code should actually calculate the solution. Do not simply provide a function that will calculate the solution.

1. [**2 points**] Finding roommates is exhausting

   Solve the *Roomers* problem using exhaustive search. You should print out the solution(s) into the following form, with the names in alphabetical order.

   ```
   [("baker",3), ("cooper",5), ("fletcher",2),
    ("miller",1), ("smith",4)]
   ```

   Hint: The `ListPair.zip` function you already researched may be helpful :)

2. [**2 points**] Liar's poker anyone?

   Solve the *Liars* problem using exhaustive search. Express the solution(s) in the same form as the previous problem.

3. [**4 points**] Chess master

   Write a function to solve the general N queens problem using exhaustive search. A solution will be a list in which the jth element is the row of the Queen in the jth column.

   Hint: Queens on squares (u, v) and (w, x) are on the same diagonal if $u + v = w + x$ or if $u - v = w - x$.

   (a) Apply it to 3 and show the solutions (*hint*, there won't be any :).
   (b) Apply it to 5 and show the solutions.
   (c) Apply it to 8 and *count the number of solutions.*

4. [**3 points**] Are you getting good at this yet?

   Solve the *Yachts* problem using exhaustive search. Specify a solution as a list of triples—father, daughter, yacht.

   Think a bit about how you are going to represent a state. There is a straightforward way to represent a state such that we can use the "odometer" formulation like we have for the previous problems.

   Hint: Each father be assigned to one of five daughters and each father will also be assigned to one of five yachts.

Part II: Branch and Bound Search

Recall from class the basic steps for solving a problem with branch and bound search:

- Come up with a state representation.
- Use the `bbProducer` function to obtain a `bbSearch` of all potential solutions. The `bbOdoIncr` and `bbOdoExt` functions will be helpful here.
- Design a predicate function that takes a *partial* solution and returns a `bbResult` (i.e. `CORRECT, PENDING` or `INCORRECT`.
- Use `bbSolve` to find the solution(s).

5. [**3 points**] Checkmate?

   Solve the N queens problem using branch and bound search.

6. [**4 points**] This problem will give you nightmares

   Solve the *Pomona College Computer Science Halloween* problem. Specify a solution as a list of 4-tuples: professor, costume, candy-count, and sorting algorithm.

   Hints/observations:

   - Make sure you are returning `INCORRECT` as soon as you can tell from a partial solution.
   - Like the *Yacht* problem, think about how you will represent the state space. It should be similar to the yacht problem except now each professor has associated with three things instead of two. The predicate function may be easier to write depending on the representation you choose, so think about it a bit.
   - In class, we implemented an `extend` function that added a 1 to the head/front of the list. You could also right an `extend` function that adds a 1 to the end of the list. Using this version of extend can make the predicate function easier since SML does pattern matching on the front of the list.
   - You can do partial pattern matches in the predicate function, i.e. match a list of length 1...2...3...15. For example, consider a predicate function that wants to check if the first entry is 1 and the third entry is a 3 using branch and bound. We could write it as something like:

     ```
     fun myPred [] = PENDING
         | myPred [x, y, z] = if x = 1 andalso z = 3 then CORRECT else INCORRECT
         | myPred (x::xs) = if x = 1 then PENDING else INCORRECT;
     ```

     The first pattern matches any list of length three and will check the complete constraint. If the list isn't of length three, then the second pattern will check if the first entry is a 1 and it will check those that aren't of length 3. The critical thing is the last pattern where we say we don't care how much of the list is filled in, we just care about the first entry. This can be generalized, for example we might write:

```
someOtherPred (x::_::_::y::xs) = ...
```

which would allow us to pattern match against all lists with 4 or more entries and we're only interested in asking questions about the first and fourth entry.

Just make sure you list your patterns from most completed to least completed since SML will try and match them from top to bottom.

- There is no pretty solution to this problem. Break your predicate function into a number of sub-functions corresponding to checking each of the nine conditions. It's too hard to do in a single function. This will help you organize things and will help with debugging.

Part III: Analysis

One solution to Problem 6 took about 4×10^{-3} seconds. We expect that branch-and-bound will be *much, much* faster than exhaustive search. In this part, you are to carry out a calculation that estimates how much faster branch-and-bound actually is.

7. [**2 points**] What fun would the assignment be without a little math

   In the Pomona College Computer Science Halloween problem, there are 5^5 ways to assign costumes to professors, another 5^5 ways to assign candy-counts, and still another 5^5 ways to assign sorting algorithms. Therefore, there are $(5^5)^3$ candidates to test in the exhaustive search approach.

   An experiment with a branch-and-bound solution to the Pomona College Computer Science Halloween problem concluded that a test of a single candidate solution took about one microsecond. (A microsecond is one millionth of a second.) One might think that a test of a candidate using exhaustive search will take somewhat longer, because one would have to test *all* the conditions every time, but let us assume that each test takes one microsecond.

   Compute how long it will take to solve the Pomona College Computer Science Halloween problem using *exhaustive search*. A rough estimate is sufficient. Express your value as a small integral number of some common unit of time—seconds, hours, days, centuries or the like. Place your solution in a comment in the file you submit.

When you're done

Double check the following things:

- Make sure your code runs without any errors (i.e. `use "assign8.sml"` does not execute an error). If you didn't finish a function and it gives an error, just comment it out and leave a note.
- Make sure that your functions match the specifications *exactly*, i.e. the names should be exactly as written (including casing) and make sure your function takes the appropriate number of parameters and is curried/uncurried appropriately.

- Make sure you have used proper style and formatting. See the course readings for more information on this. Be informative and consistent with your formatting!
- Make sure you've properly commented your code. You should include:
  - A comment header at the top of the file with your name, the date, the assignment number, etc.
  - Each problem should be delimited by comment stating the problem number.
  - Each function should have a comment above it explaining what the function does.
  - Complicating or unusual lines in functions should also be commented.

  Don't go overboard with commenting, but do be conscientious about it.

When you're ready to submit, upload your assignment via the online submission mechanism. You may submit as many times as you'd like up until the deadline. We will only grade the most recent submission.

Grading

Part I	11
Part II	7
Part III	2
comments/style	3
Total	23

Acknowledgements

The Eight Queens problem is part of the folklore of Computer Science. The Pomona College Computer Science Halloween problem was created by the departmental liaisons (many of whom were mentors for this course!) for the Halloween party on October 25, 2013. The others are patterned after problems that appeared in Abelson and Sussman, *Structure and Interpretation of Computer Programs,* second edition, M.I.T. Press, 1996. The authors attribute Roomers to Dinesman, *Superior Mathematical Puzzles,* Simon and Schuster, 1968; Liars to Phillips, *The Sphinx Problem Book,* Faber and Faber, 1934; and Yachts to a booklet "Problematical Recreations," published in the 1960's by Litton Industries.