



SOLVE EIGHT QUEENS PUZZLE WITH SAS[®] MACRO

Jian Dai, Clinovo, Sunnyvale CA

WUSS 2011

Annual Conference October 2011



TABLE OF CONTENTS

1. ABSTRACT	3
2. INTRODUCTION	3
3. ALGORITHM	4
4. SAS SOLUTION BY USING MACRO9	5
5. CONCLUSION	6
6. REFERENCES	6

1. ABSTRACT

We will demonstrate in this paper how to solve eight queens puzzle by using recursive SAS macro to implement the depth-first search algorithm with backtracking.

2. INTRODUCTION

“The Little SAS Book”¹ contains an excellent example to contrast the difference between SAS as a programming language and C++, and perfectly illustrates how hard dataset processing can be for a general-purposed programming language. It takes 28 non-blank lines of C++ to code a 5-line SAS program when it comes to read in a delimited data file and print it out. This is a good example to showcase that SAS is a fourth-generation programming language² at a higher level of abstraction and with stronger statement power.

We want to look into this contrast from another perspective; we will describe how SAS is a perfect language to handle complicated data structure, and how easy implementing sophisticated algorithms becomes. This is the continuation of our exploration of recursive SAS macro³, adding the ingredient of backtracking. The model problem we choose is the famous eight queens puzzle. To start, let’s review the following concepts:

RECURSION⁴

A function or subroutine is recursive if it invokes itself directly or indirectly. The classical example is the computation of factorial $n!$.

DEPTH-FIRST SEARCH⁵

As an example, take a tree in the sense of a data structure. Depth-first search is the search that visits a subtree of a node following the visit to this node.

BACKTRACKING

“Backtracking is a general algorithm for finding all (or some) solutions to some computational problem that incrementally builds candidates to the solutions, and abandons each partial candidate c (“backtracks”) as soon as it determines that c cannot possibly be completed to a valid solution.”⁶

EIGHT QUEENS PUZZLE⁷

Eight queens puzzle *per se* is about putting eight queens on the chess board such that there is no pair of queens that can attack each other. This puzzle provides a classical example in the field of computation for studying recursion, depth-first search and backtracking.

3. ALGORITHM

We solve eight queens puzzle based on the permutation of number 1, 2, 3, 4, 5, 6, 7 and 8. Mathematically, such permutations form an algebraic structure called permutation group of eight elements. Each permutation is a map

$$\sigma : \{1,2,3,4,5,6,7,8\} \rightarrow \{1,2,3,4,5,6,7,8\}, i \mapsto \sigma(i), \forall i \in \{1,2,3,4,5,6,7,8\}$$

such that

$$\forall i \neq j, \sigma(i) \neq \sigma(j)$$

where i and j are two integers between 1 and 8. In total, there are $8!$ permutations.

Each solution to eight queens puzzle can be represented as a permutation satisfying one more condition

$$\forall i \neq j, |i - j| \neq |\sigma(i) - \sigma(j)|$$

This condition is the key for backtracking. We took the following Javascript solution⁸ as our pseudo code to describe the algorithm for SAS:

```
<body onload="javascript:cnt=0;
function backTrack(trial,next){
  if (trial.length==0){return true;}
  else {
    for (var i in trial){
      if (Math.abs(trial.length-i)==Math.abs(next-trial[i])){
        return false;
      }
    }
    return true;
  }
}
function perm(p,l){
  if (l.length==0){
    cnt++;document.write(cnt+':'+p+'<br/>')
  }
  else {
    for (var i in l){
      if (backTrack(p,l[i])){perm(
        p.concat(l[i]),
        l.slice(0,l.indexOf(l[i])).concat(
          l.slice(l.indexOf(l[i])+1,l.length)
        )
      );
    }
  }
}
perm([], [1,2,3,4,5,6,7,8]) "></body>
```

4. SAS SOLUTION BY USING MACRO⁹

With all of the previous preparation, it is not very hard to write down the following SAS implementation of the recursive solution for the eight queens puzzle.

```

%Macro FirstOf(List);%Scan(&List,1)%Mend;
%Macro RestOf(List);
  %Local lth;
  %Let lth=%Length(%FirstOf(&List));
  %If %Length(&List)>&lth %Then %Left(%Substr(&List,%Eval(1+&lth)));
%Mend;

%Macro OkToAdd(Element,At=,To=,StartAt=);
  %If &To eq %str() or &Element eq %str() %Then 1;
  %Else %If %Sysfunc(Abs(%Eval(%FirstOf(&To)-&Element)))=
    %Sysfunc(Abs(%Eval(&At-&StartAt))) %Then %Do; 0 %Return;%End;
  %Else
%OkToAdd(&Element,At=&At,To=%RestOf(&To),StartAt=%eval(1+&StartAt));
%Mend;

%Macro qIter(PartialSolution=,List=,Level=,CounterName=);
  %Local item preFix suffix;
  %If &List eq %str() %Then %Do;%Let &CounterName=%eval(1+&&&CounterName);
    %Put &&&CounterName [&PartialSolution];
  %End;
  %Else %Do;
    %let preFix=;%let item=%FirstOf(&List);%let suffix=%RestOf(&List);
    %Do %Until (&preFix eq &List);
      %If %OkToAdd(&item,At=&Level,To=&PartialSolution,StartAt=1) %Then
        %qIter(PartialSolution=&PartialSolution &item,
          List=&preFix &suffix,
          Level=%eval(&Level+1),
          CounterName=&CounterName
        );
      %let preFix=&preFix &item;%let item=%FirstOf(&suffix);
      %let suffix=%RestOf(&suffix);
    %End;
  %End;
%Mend;

%let c=0;
%qIter(PartialSolution=,List=1 2 3 4 5 6 7 8,Level=1,CounterName=c)

```

5. CONCLUSION

By using the example of the eight queens puzzle, we show that SAS macro can be used to code complicated algorithms and to handle sophisticated data structure as good as a general purposed programming language.

6. REFERENCES

- ¹Lora D. Delwiche, Susan J. Slaughter “The Little SAS Book: A Primer, Third Edition” 2003 SAS Institute Inc.
- ²http://en.wikipedia.org/wiki/Fourth-generation_programming_language
- ³Jian Dai “[Permutation via Recursive SAS Macro](#)” PharmaSUG 2011
- ⁴http://en.wikipedia.org/wiki/Recursion_%28computer_science%29
- ⁵William Ford, William Topp “Data structure with C++” 1996 Prentice Hall Inc.
- ⁶<http://en.wikipedia.org/wiki/Backtracking>
- ⁷http://en.wikipedia.org/wiki/Eight_queens_puzzle
- ⁸http://tech.groups.yahoo.com/group/sas_academy/message/437
- ⁹http://tech.groups.yahoo.com/group/sas_academy/message/429

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors at:

Name: Jian Dai

Enterprise: Clinovo

Address: 1208 East Arques Avenue

City, State ZIP: Sunnyvale CA 94085

Work Phone: 408-416-1636

E-mail: jian.dai@clinovo.com

Web: www.clinovo.com <http://blog.clinovo.com/> http://tech.groups.yahoo.com/group/sas_academy/

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.