Andreas Bach Landgrebe
Computer Science 250: Analysis of Algorithms
February 25, 2015
Laboratory Assignment 5 - Exhaustive Search Eight Queens

Part 1: Implementing Basic Eight Queens &
Part 2: Implementing Advanced Eight Queens

Source Code

```
import java.util.Scanner;


public class eightQueens {

   static int QueensFive[] = new int[5];

   static int QueensEight[] = new int[8];

   static int QueensEleven[] = new int[11];

   static int solutions =0;

   public static boolean isGoodEight(int row, int col) {
      int colLeft=col-1;
      int colRight=col+1;
      for (int i=row-1; i>=0; i--) {
         if (QueensEight[i]==colLeft--) return false;
         if (QueensEight[i]==col) return false;
         if (QueensEight[i]==colRight++) return false;
      } //for
      return true;
   } //isGoodEight

   public static boolean isGoodFive(int row, int col) {
      int colLeft=col-1;
      int colRight=col+1;
      for (int i=row-1; i>=0; i--) {
         if (QueensFive[i]==colLeft--) return false;
         if (QueensFive[i]==col) return false;
         if (QueensFive[i]==colRight++) return false;
      } //for
      return true;
   } //isGoodFive

   public static boolean isGoodEleven(int row, int col) {
      int colLeft=col-1;
      int colRight=col+1;
      for (int i=row-1; i>=0; i--) {
         if (QueensEleven[i]==colLeft--) return false;
         if (QueensEleven[i]==col) return false;
         if (QueensEleven[i]==colRight++) return false;
      } //for
      return true;
```

```
} //isGoodEleven


public static void printBoardEight() {

   for (int col=0; col < 8; col++) {
      for (int j=0; j < 8; j++) {
         if (j==QueensEight[col]) {
            System.out.print("X");
         } else {
            System.out.print(".");
         }//if-else
      }//for
      System.out.println();
   }//for
}//printBoardEight

public static void printBoardFive() {
   for (int col = 0; col < 5; col++) {
      for (int j = 0; j < 5; j++) {
         if (j == QueensFive[col]) {
            System.out.print("X");
         } else {
            System.out.print(".");
         } //if-else
      } //for
      System.out.println();
   } //for
}//printBoardFive

public static void printBoardEleven() {
   for (int col = 0; col < 11; col++) {
      for (int j = 0; j < 11; j++) {
         if (j == QueensEleven[col]) {
            System.out.print("X");
         } else {
            System.out.print(".");
         } //if-else
      } //for
      System.out.println();
   } //for
} //printLevelEleven

public static void tryLevelEight(int Level) {
   for (int i = 0; i < 8; i++) {
      if (isGoodEight(Level,i)) {
         QueensEight[Level]=i;
         if (Level==7) {
            printBoardEight();
            //for (int j=0;j<8;j++) System.out.print(Queens[j]);
            System.out.println();
            solutions++;
         } else {
            tryLevelEight(Level+1);
         } //if-else
      } //if
   }//for
} //tryLevelEight

public static void tryLevelFive(int Level){
   for (int i = 0; i < 5; i++) {
      if(isGoodFive(Level,i)){
         QueensFive[Level]=i;
         if(Level==4) {
            printBoardFive();

            System.out.println();
            solutions++;
         } else{
            tryLevelFive(Level+1);
         } //if-else
      } //if
   } //for
} //tryLevelFive

public static void tryLevelEleven(int Level){
   for (int i = 0; i < 11; i++) {
      if(isGoodEleven(Level,i)){
         QueensEleven[Level]=i;
         if(Level==10) {
            printBoardEleven();

            System.out.println();
            solutions++;
         } else{
            tryLevelEleven(Level+1);
         } //if-else
      } //if
   } //for
} //tryLevelEleven


public static void main(String[] args) {

   Scanner scan = new Scanner(System.in);

   int numberOfQueens;

   System.out.println("Please scan in the number of queens");
```

```
      numberOfQueens = scan.nextInt();

      if(numberOfQueens == 8){
         tryLevelEight(0);
         System.out.println("Number of solutions: "+ solutions);
      } //if
      if(numberOfQueens == 5){
         tryLevelFive(0);
         System.out.println("Number of solutions: "+ solutions);
      } //if
      if(numberOfQueens == 11){
         tryLevelEleven(0);
         System.out.println("Number of solutions: " + solutions);
      } //if
   } //main
} //eightQueens class
```

Output

EightQueens for Part 1:

```
X.......
....X...
.......X
.....X..
..X.....
......X.
.X......
...X....
```

EightQueens for Part 2:

```
X.......
....X...
.......X
.....X..
..X.....
......X.
.X......
...X....

X.......
.....X..
.......X
..X.....
......X.
...X....
.X......
....X...

X.......
......X.
...X....
.....X..
.......X
.X......
....X...
..X.....

X.......
......X.
....X...
.......X
.X......
...X....
.....X..
..X.....
```

```
.X......
...X....
.....X..
.......X
..X.....
X.......
......X.
....X...
```

First Solution for Five Queens:

```
X....
..X..
....X
.X...
...X.
```

First Solution for Eleven Queens:

```
X..........
..X........
....X......
......X....
........X..
..........X
.X.........
...X.......
.....X.....
.......X...
.........X.
```

Part 3: While You Have Some Downtime

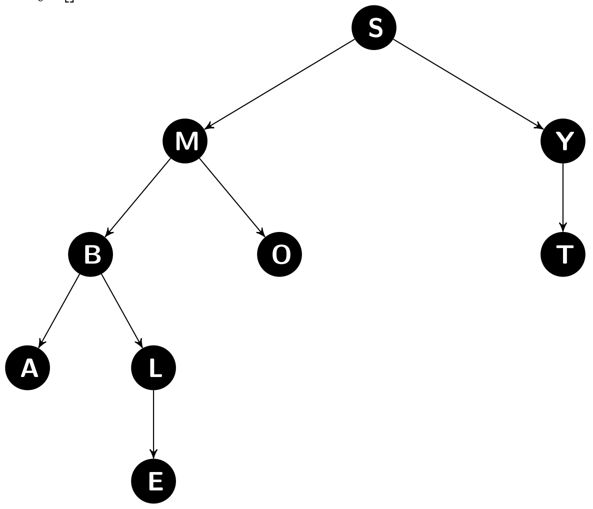
1. Trace the process of inserting the keys EIGHTQUEENS into a binary search symbol table. Each key is associated with the value corresponding to the index of the letter in the string. List the final set of Key-Value pairs after all letter are inserted.
   Keys[]
   E
   EI
   EIG
   EIGH
   EIGHT
   EIGHQT
   EIGHQTU
   EIGHQTU
   EIGHQTU
   EIGHNQTU
   EIGHNQSTU

   Value[]
   0
   0 1
   0 1 2
   0 1 2 3
   0 1 2 3 4
   0 1 2 3 5 4
   0 1 2 3 5 4 6
   7 1 2 3 5 4 6
   8 1 2 3 5 4 6
   8 1 2 3 9 5 4 6
   8 1 2 3 9 5 10 4 6

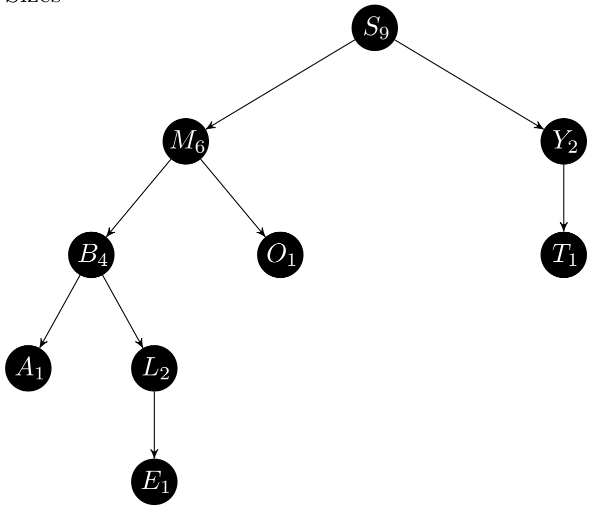
2. Which symbol table implementation (Sequential or Binary Search) would you choose for an application that runs 10^3 put() operations and 10^6 get() operations?
   10^3 put() operations: Sequential
   10^6 get() operations: Binary Search

3. Trace the process of inserting keys SYMBOLTABLE into a binary search tree. Each key is associated with the value corresponding to the index of the letter in the string. List the final set of Key-Value pairs after all letters are inserted.

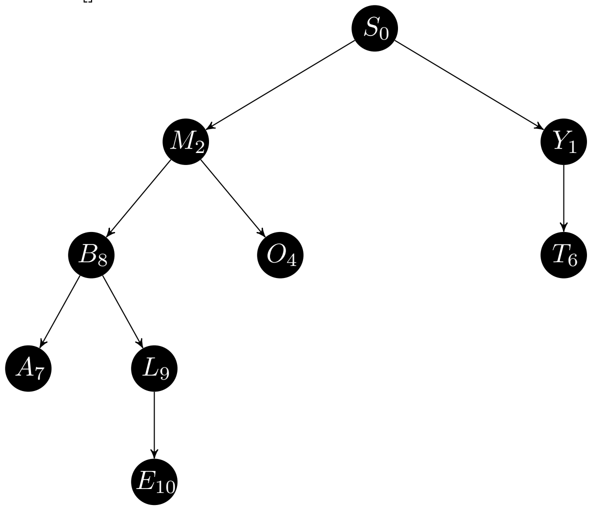
   Keys[]



   Sizes



Values[]


S_0
M_2
Y_1
B_8
O_4
T_6
A_7
L_9
E_{10}