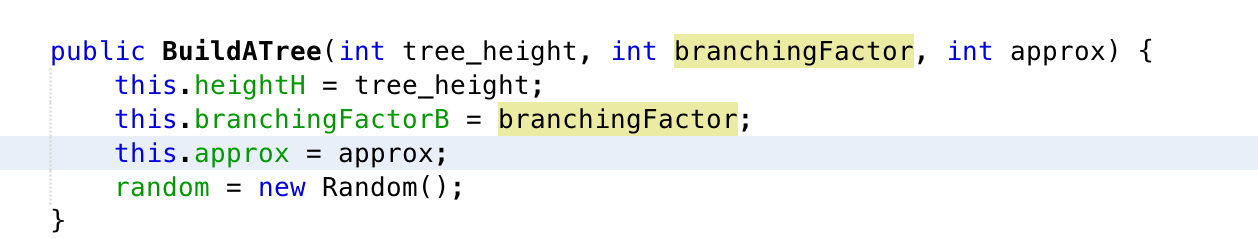
COMP30260 "Artificial Intelligence for Games & Puzzles" 1st Programming Assignment

NegaScout/PVS with Principal Variation Reordering

**Step One: Building a tree**

The class BuildATree’s constructor would take three positive integers as argument :

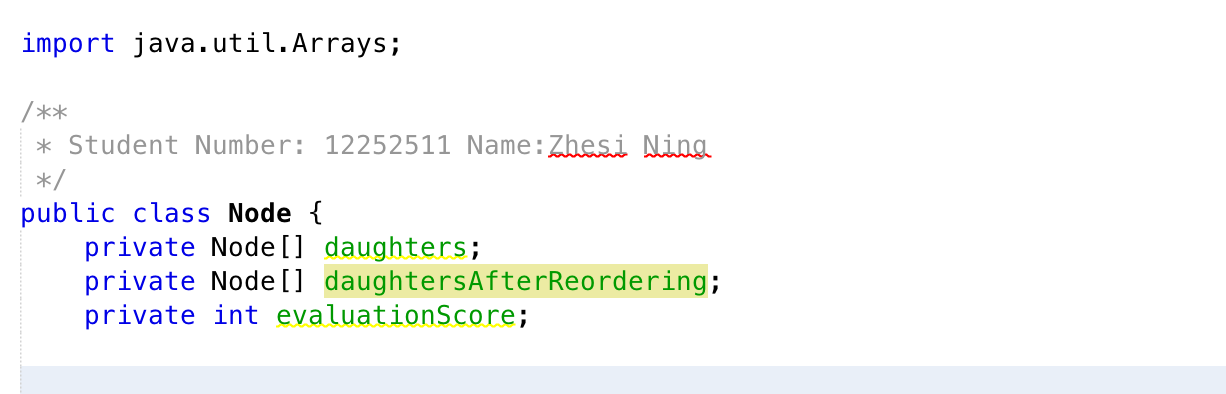
1, branching factor b

2, height h

3, approximation Approx

In the node class, each of the node has its own static evaluation(score) to indicate how good it is.

1, its own static evaluation: **private Node[] daughters;**

****

2, its daughters, in some originally-generated order: **private Node[] daughters;**

3, its daughters, in an order that might get changed by PV reordering during a search: **private Node[] daughtersAfterReordering;**

We can generate a new tree with height h and branching factor b and approx like this “new BuildATree(h, b, approx);”

**Step Two: Negamax-style alpha-beta algorithm with iterative deepening**

Class AB is a simple implantation of negamax alpha-beta algorithm with iterative deepening.

**(a) code to count the number of static evaluations performed:**

The variable “private int noOfEvalutionPerformed ” in class AB and class Pvs is use to count the number of static evaluations performed in both alpha beta search and PVS search.

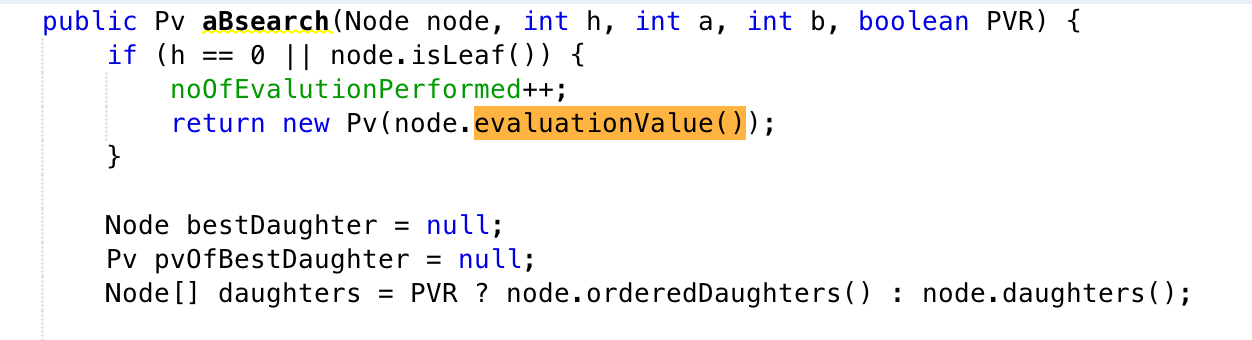
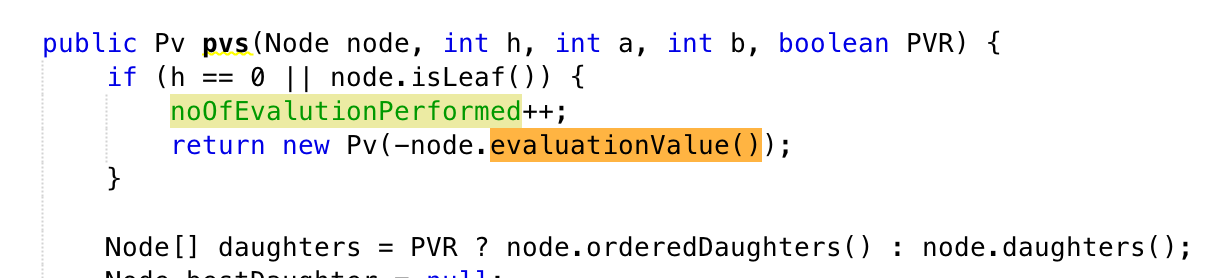
**(b) code to return both a value and a principal variation:**

As the screenshot shown below the method getScore get the score value and the evaluationValue() return the principal variation of the node.

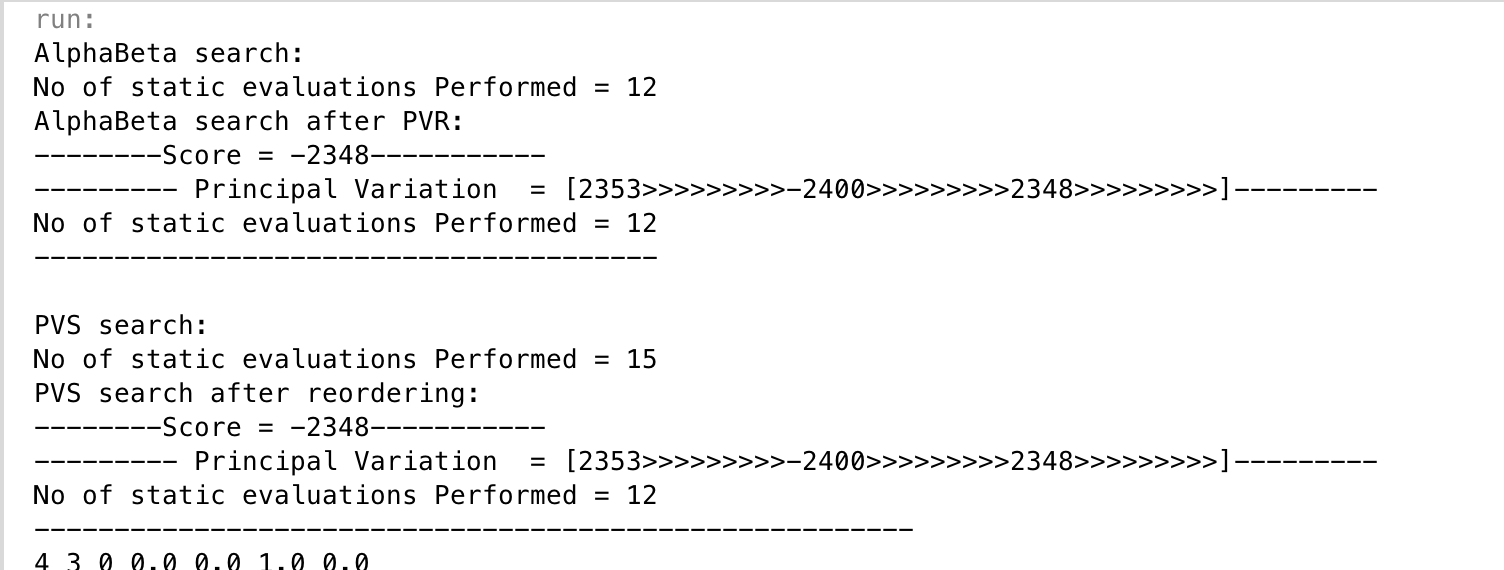
**(c) code to unpick the returned values appropriately:**

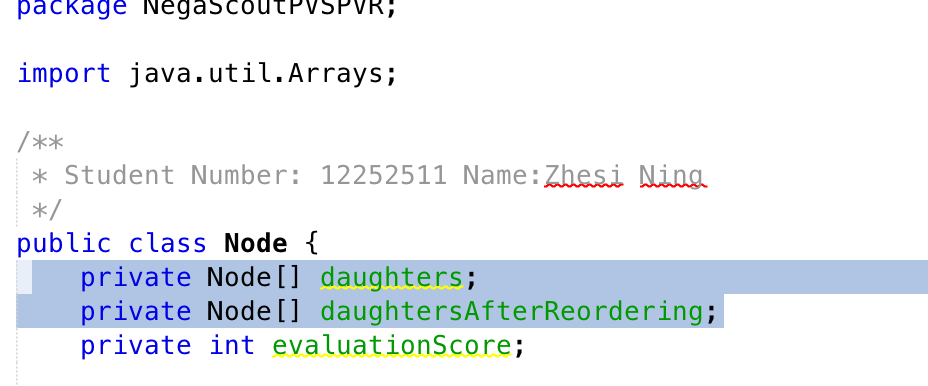
There is a toString method in class Pv where it can unpick the returned values.

**(d) a parameter indicating whether or not to use the modified daughter:**

****

Both alpha beta search and PVS search take a Boolean value PVR in the search method where if PVR is true then this search will use the modified daughter after the reordering which so that the best-seen daughter is in first position. Otherwise just a standard search with the original tree.



**Step Three: Principal Variation Reordering**

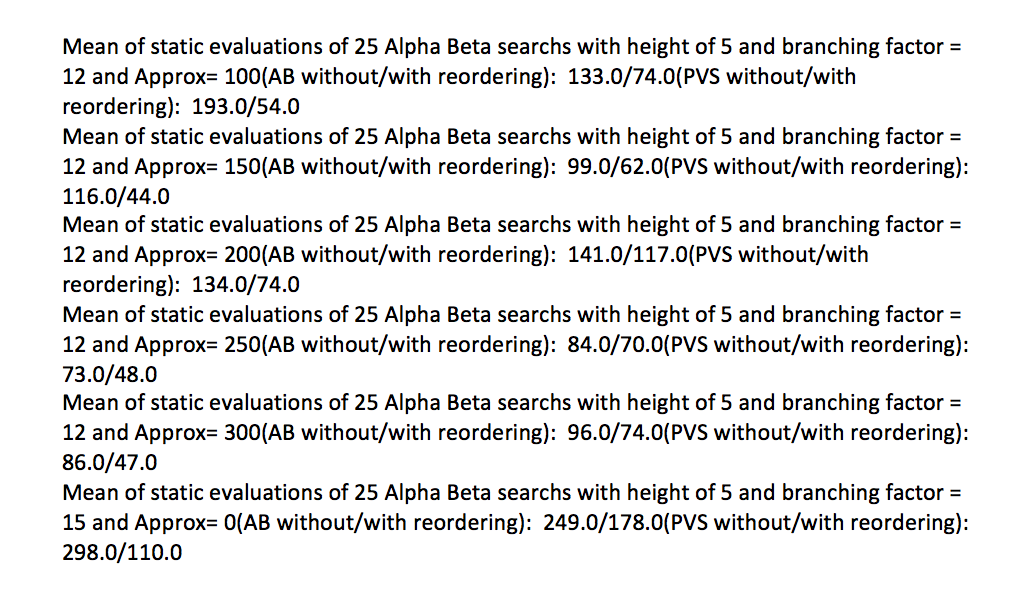
In class Node there are two array: one is all the daughters of the node and the other one is its daughters after performing Principal Variation Reordering.

**Step Four: PVS with iterative deepening**

Class Pvs is a simple implantation of PVS search algorithm which calls the alpha beta search but try with a narrower window every time, this increase the chance of a cut off. The algorithm using iterative deepening as well.

**Step Five: Experiment and Report**

In this step, I carried out a numbers of difference search using both alpha beta search and PVS search with/without PV reordering. For each parameter combination below we generate 25 trees

-tree height, from 4 to 6 in steps of 1

- branching factor, from 3 to 21 in steps of 3

- Approx, from 0 to 300 in steps of 50

then we calculate the means of those 25 results for each combination, and compare the difference between them. Below image gives a rough idea what the output data looks like(it took more than 20min to run).

We can import the data into excel and plot some graph

The graph above legend in orange are search after PV reordering and those in blue are seach without reordering. The graph below legend in blue are alpha beta search and those in orange are PVS searches.

As we obsereve from the graph, generally speaking both alpha beta and pvs search has better proformance while using the tree after PV reordering. This is thanks to the principal variation from an interior node is a sequence whose first element is the best children therefore we have a better chance to cut-off(purn) the tree. We also noticed that bigger the branching factor and deeper the tree height, the proference differece between with and without pv reordering will be larger.

Then for the proformance differnece between alpha beta search and PVS search, since PVS search is calling the alpha-beta algorithm with varying values for its alpha and beta and it is “trying its luck” and hopeing to get a chance to purn the rest of the tree in order to get a better proformance. However, without PV reordering, the proformance increace are pretty small since the score of the children are distributed quite randomly. But after we reordering the children we can see a significant improvement for the proformance since the best children will always at the first place and the seach will go from the left to the right of the tree. Therefore, we can maximise the chance to purn the subtree.