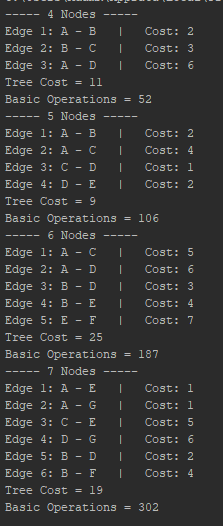
(2)



I had some trouble getting the books version of Prim’s working, my family is currently being affected by the outbreak and I did not have much time for trouble shooting so I implemented it with nested for loops in a “brute force” style. With the nested for loops of my implementation, the time complexity should be of order O(n3) instead of the books order of O(n2).

There are 4 instances of Prim’s in the screenshot:

4 Nodes: 52 operations < 43

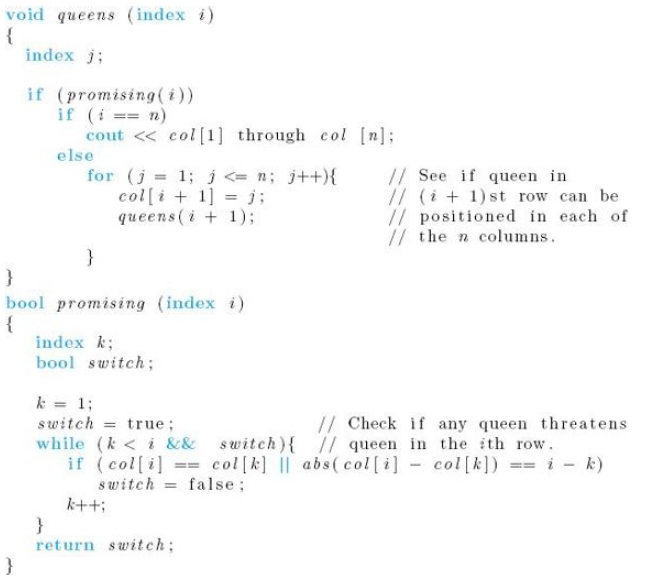
5 Nodes: 106 operations < 53

6 Nodes: 187 operations < 63

7 Nodes: 302 operations < 73

These numbers make sense. I can see that the basic operations will stay under there upper bound and I can also see the increase of operations with the increase of the input.

(4)



The promising function could still be used in this case. I would make the following changes to queens():

bool queens(index i){

index j;

if(promising(i)){

if (i == n){

cout << col[i] through col[n];

return true; // this should allow the function to return when a solution is found, not continuing

}else{

for (j = 1; j <= n; ++j){

col[i + 1] = j;

if(queens(i + 1)){ // this should return true from the recursive call

return true;

}

}

return false;

}}}