CS5330 Randomized Algorithm Project

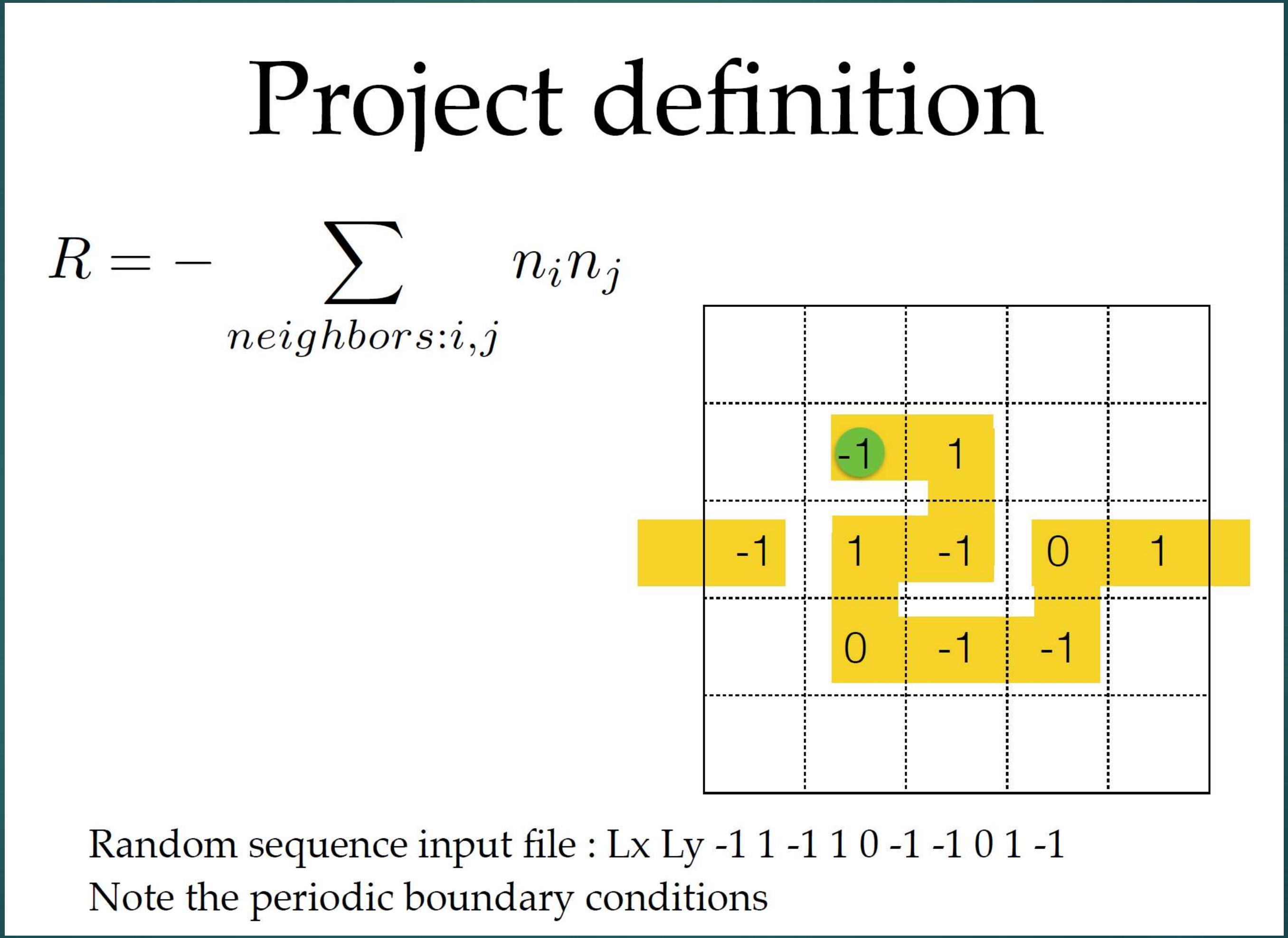
ARNOLD CHRISTOPHER KOROA

A0092101Y

Table of Contents

- Project Definition
- Overall Algorithm
- Invariances
- Score Normalization
- Next-Step Partial Score
- Reachable Cells Heuristics
- Miscellaneous Observations
- Results

Project Definition



Project definition

$$R = -\sum_{neighbors:i,j} n_i n_j$$

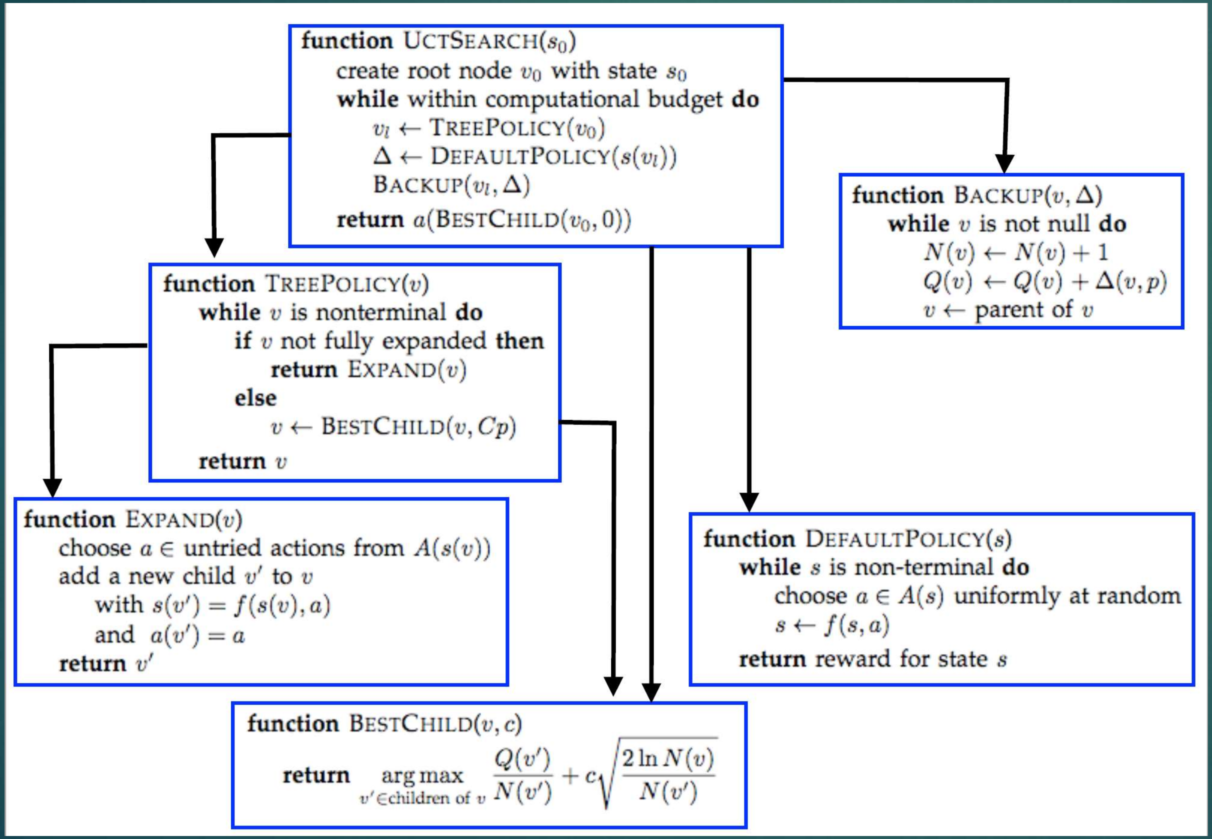
	-1	1		
-1	1	-1	0	1
	0	-1	-1	

Random sequence input file : Lx Ly -1 1 -1 1 0 -1 -1 0 1 -1

Note the periodic boundary conditions

Overall Algorithm

UCT Algorithm (exactly as in class) with ensemble of 5



```
function UCTSEARCH($s_0$)
    create root node $v_0$ with state $s_0$
    while within computational budget do
        $v_l \leftarrow$ TREEPOLICY($v_0$)
        $\Delta \leftarrow$ DEFAULTPOLICY($s(v_l)$)
        BACKUP($v_l, \Delta$)
    return $a$(BESTCHILD($v_0$, 0))
```

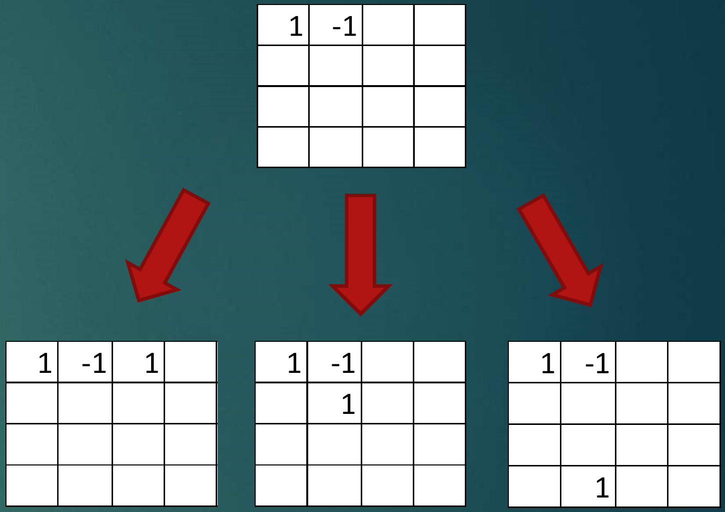
```
function BACKUP($v, \Delta$)
    while $v$ is not null do
        $N(v) \leftarrow N(v) + 1$
        $Q(v) \leftarrow Q(v) + \Delta(v, p)$
        $v \leftarrow$ parent of $v$
```

```
function TREEPOLICY($v$)
    while $v$ is nonterminal do
        if $v$ not fully expanded then
            return EXPAND($v$)
        else
            $v \leftarrow$ BESTCHILD($v, Cp$)
    return $v$
```

```
function EXPAND($v$)
    choose $a \in$ untried actions from $A(s(v))$
    add a new child $v'$ to $v$
        with $s(v') = f(s(v), a)$
        and $a(v') = a$
    return $v'$
```

```
function DEFAULTPOLICY($s$)
    while $s$ is non-terminal do
        choose $a \in A(s)$ uniformly at random
        $s \leftarrow f(s, a)$
    return reward for state $s$
```

```
function BESTCHILD($v, c$)
    return $\underset{v' \in \text{children of } v}{\arg\max} \frac{Q(v')}{N(v')} + c\sqrt{\frac{2 \ln N(v)}{N(v')}}$
```



1	-1		

1	-1	1	

1	-1		
	1		

1	-1		
	1		

Default Policy:

Random Playout (till the end, except 64x64)

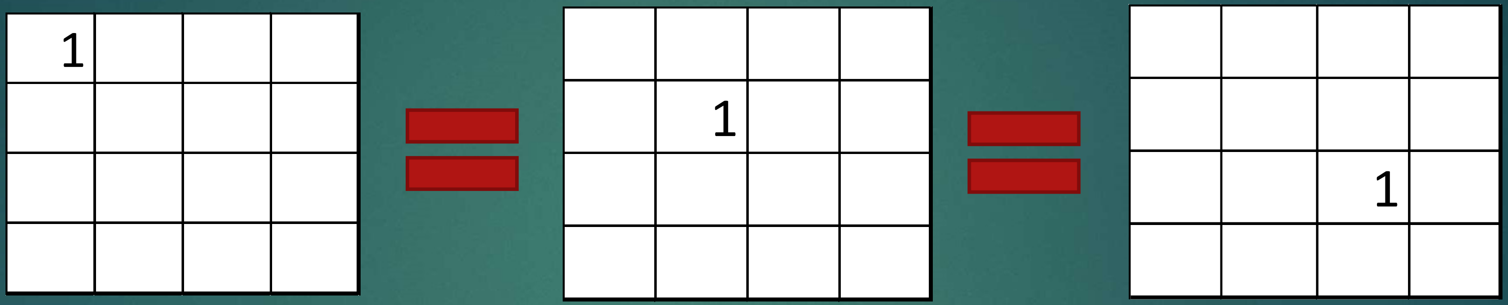
If deadend, try 10 times before giving up

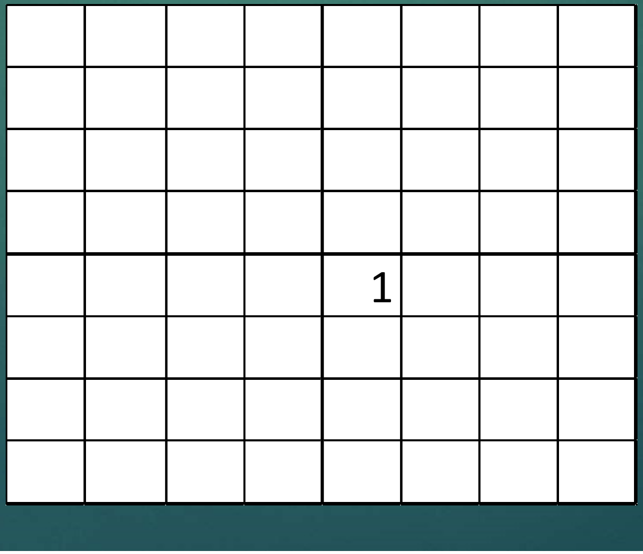
Exploration constant: 0.2

Move to direction with highest vote

Invariances

Translational Invariance



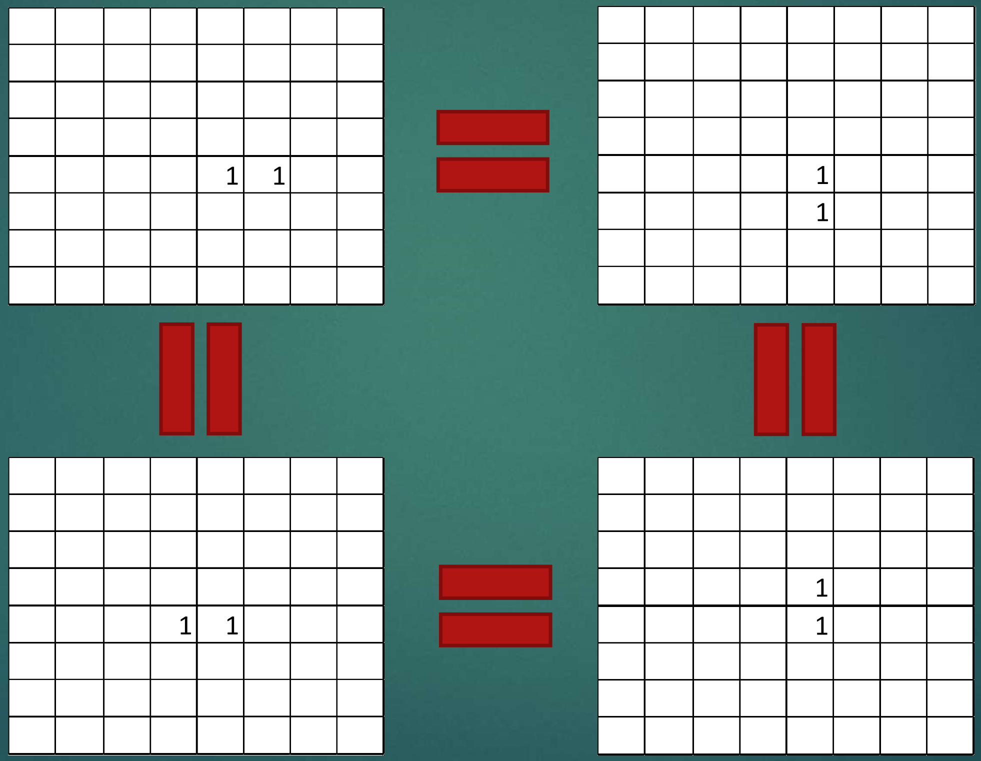


1st move always (0,0)

Allow wrap-around

Invariances

Rotational Invariance



2nd move always RIGHT

Invariances

Reflective Invariance

				1	1		
					1		

=

					1		
				1	1		

				1	1	1	
						1	

=

						1	
				1	1	1	

First vertical movement always DOWN

Score Normalization

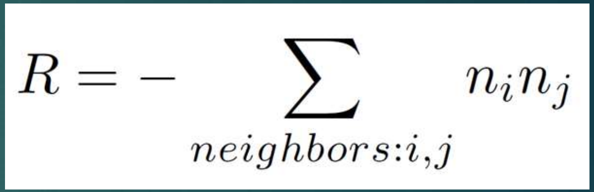
Make UCT exploitation/exploration ratio more efficient

[0, 1, -1, 0, 0, 1]

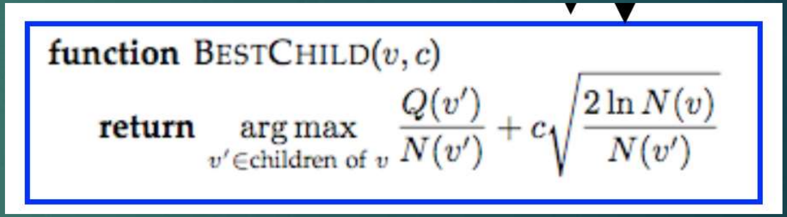
Norm = #non-0 entries = 3

Max score = norm = 3

Min score = - norm = -3



$$R = -\sum_{neighbors:i,j} n_i n_j$$



```
function BESTCHILD($v, c$)
    return $\underset{v' \in \text{children of } v}{\arg\max} \frac{Q(v')}{N(v')} + c\sqrt{\frac{2 \ln N(v)}{N(v')}}$
```

Normalized score = (score + norm) / (2 * norm)

Score -3: (-3 + 3) / (2 * 3) = 0

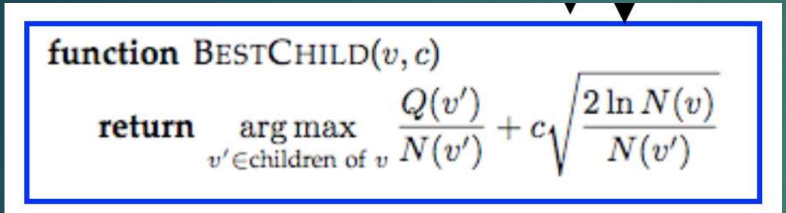
Score 1: (1 + 3) / (2 * 3) = 0.67

Score 3: (3 + 3) / (2 * 3) = 1

Deadend: 0 (unlikely to get absolute lowest score anyway)

Next-Step Partial Score

At the start, any direction is “the same”



```
function BESTCHILD($v, c$)
    return $\underset{v' \in \text{children of } v}{\arg\max} \frac{Q(v')}{N(v')} + c\sqrt{\frac{2 \ln N(v)}{N(v')}}$
```

+ 0.1 * normalized partial score for child

Normalized on norm so far

- This makes the effect lessens over time

Norm so far = 5:
+1: 6/10 = 0.6
-1: 4/10 = 0.4
Difference: 0.2

Norm so far = 10:
+1: 11/20 = 0.55
-1: 9/20 = 0.45
Difference: 0.1

Norm so far = 20:
+1: 21/40 = 0.52
-1: 19/40 = 0.475
Difference: 0.045

Reachable Cells Heuristics

Use BFS To detect if max cells reachable >= length of remaining sequence

x	x	x	x	x	x	x	x
x							x
x			cur				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
x							x
x		nxt	cur				x
x	19		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

max{19,19,19} = 19

x	x	x	x	x	x	x	x
x			nxt		7		x
x			cur				x
x	12		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

max{12 ,0,7} = 12

x	x	x	x	x	x	x	x
x							x
x			cur	nxt			x
x	19		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

max{19,19,0} = 19

Reachable Cells Heuristics

It’s just a heuristic: it might fail!

x	x	x	x	x
x				x
x	x	nxt	x	x
		cur	x	x

Reachable = 3 but
only 2 is usable

Reachable Cells Heuristics

It’s just a heuristic: it might fail!

x	x	x	x	x	x	x	x
x							x
x		nxt	cur				x
x			x	x	x		x
x		19	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

Reachable = 19 and possible to use all 19

x	x	x	x	x	x	x	x
x							x
x			cur	nxt			x
x			x	x	x	NG	x
x		19	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

Reachable = 19 but only possible to use 18

- For any nxt position with enough reachable cells try 10 times
  - If have at least 1 valid path take it otherwise give up and give 0 normalized score (importance sampling)
  - UCT then will take care of remaining deadends
- Still useful: pure random walk on 16x16 always succeed within 10 tries
- This detects deadend configurations very early

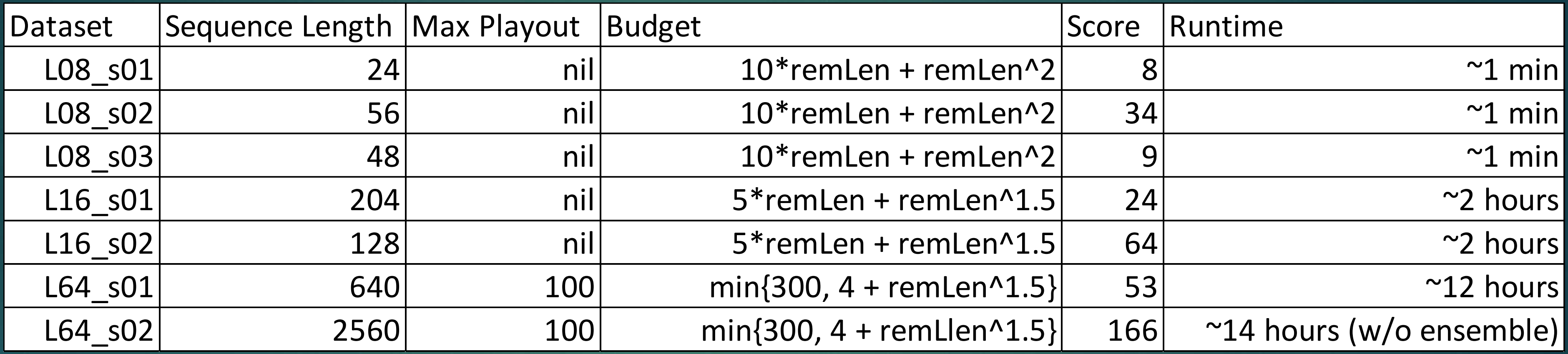
Miscellaneous Observations

- Memorize Highest Random Playout
  - During default policy random walk memorize grid with highest score
  - Usually gives 1~2 extra points

- Maximum Lookahead
  - Normally do total playout for default policy
  - This takes too long for 64x64
  - Limit it to 100 step lookahead

- Immutable Data Structure
  - I notice I copy the grids multiple times to create different nodes, do BFS, etc
  - Might be good to have immutable DS that can represent minor differences more efficiently (maybe use Scala instead of Java)

Results



Dataset	Sequence Length	Max Playout	Budget	Score	Runtime
L08_s01	24	nil	10*remLen + remLen^2	8	~1 min
L08_s02	56	nil	10*remLen + remLen^2	34	~1 min
L08_s03	48	nil	10*remLen + remLen^2	9	~1 min
L16_s01	204	nil	5*remLen + remLen^1.5	24	~2 hours
L16_s02	128	nil	5*remLen + remLen^1.5	64	~2 hours
L64_s01	640	100	min{300, 4 + remLen^1.5}	53	~12 hours
L64_s02	2560	100	min{300, 4 + remLlen^1.5}	166	~14 hours (w/o ensemble)

- L64_s01 without ensemble (~2 hours) obtained 10 less points
  - Ensemble does help
  - But maybe just because it basically takes 5x more samples
- L64_s02 with ensemble of 5 would have taken 60~70 hours
  - I only had ~24 hours left!
  - After deadline tried with ensemble of 3
    - OutOfMainMemoryException after almost 2 days