ICFP Programming Contest 2014 – Supermassive Black Hom-set Post-mortem

P. Lepin

What's up with the name?

- Time pressure – wanted to submit ASAP, needed *some* team name.

What's up with the name?

- Time pressure – wanted to submit ASAP, needed *some* team name.
- Last year I played as "By Wadler's Beard!" Should have kept it.

The Plan

Coding directly in GCC assembly possible, but painful. So...

The Plan

Coding directly in GCC assembly possible, but painful. So...

1. Implement the VM.
2. Write a parser for a stand-alone HLL or implement an eDSL.
3. Implement static checks and/or optimizations.
4. Implement codegen.
5. ...?
6. **(end goal)** Implement the Lambda-Man AI.

The Plan

Coding directly in GCC assembly possible, but painful. So...

1. ~~Implement the VM.~~ – there was a web-based implementation
2. Write a parser for a stand-alone HLL ~~or implement an eDSL.~~
3. ~~Implement static checks and/or optimizations.~~
4. Implement codegen.
5. ~~...?~~
6. **(end goal)** Implement the Lambda-Man AI.

The Plan

Coding directly in GCC assembly possible, but painful. So...

1. ~~Implement the VM.~~ – there was a web-based implementation
2. Write a parser for a stand-alone HLL ~~or implement an eDSL.~~
3. ~~Implement static checks and/or optimizations.~~
4. Implement codegen.
5. ~~...?~~
6. **(end goal)** Implement the Lambda-Man AI.

7. Implement symbolic labels on top of GHC assembly.
8. **(end goal)** Implement a Ghost AI (or several).

The Plan

Coding directly in GCC assembly possible, but painful. So...

1. ~~Implement the VM.~~ – there was a web-based implementation
2. Write a parser for a stand-alone HLL ~~or implement an eDSL.~~
3. ~~Implement static checks and/or optimizations.~~
4. Implement codegen.
5. ~~...?~~
6. **(end goal)** Implement the Lambda-Man AI.

7. Implement symbolic labels on top of GHC assembly.
8. **(end goal)** Implement a Ghost AI (or several).

...some of these decisions were extremely myopic.

Perceived Fun Factor

	Tools	AI
Lambda-Man	**FUN!**	**FUN!**
Ghosts	Less fun.	Less fun.

Lisp-machine CPU, compilers, fairly sophisticated AIs – interesting. 8-bit CPUs and severely resource-constrained programs – not so much.

Effort

	Tools	AI
Lambda-Man	~**9 hrs**, 205 sloc	~**25 hrs**, 654 sloc, 3114 instructions compiled
Ghosts	~**3 hrs**, 259 sloc	~**3 hrs**, 91 instruction

Wild Guesstimate Of Impact

	Tools	AI
Lambda-Man	Some.	
Ghosts	**HUGE!**	

Judging by the results of home-brewed tournaments on reddit.

HLL Compiler Targetting GCC

- HLL is Lisp-like, mimicking Scheme and Clojure.

HLL Compiler Targetting GCC

- HLL is Lisp-like, mimicking Scheme and Clojure.

```
467 (def foldl
468     (fun [f init xs]
469         (if [atom? xs]
470             init
471             (recur f (f init (car xs)) (cdr xs)))))
472 (def map-map (fun [f xs] (map (fun [x] (cons x (f x))) xs)))
473 (def map
474     (fun [f xs]
475         (if [atom? xs]
476             NIL
477             (cons (f (car xs)) (map f (cdr xs))))))
```

Figure: Almost looks like the real thing.

HLL Compiler Targetting GCC

- HLL is Lisp-like, mimicking Scheme and Clojure.
- Almost purely functional, `set!` is accepted by the parser, but...

HLL Compiler Targetting GCC

- HLL is Lisp-like, mimicking Scheme and Clojure.
- Almost purely functional, `set!` is accepted by the parser, but...

```
221 cg n env labels (Spec "!1" [Lit x]) = ([Op "LD" [Num x, Num 1]], labels)
222 -- Unsupported: set!
223 cg n env labels expr = (unsafePerformIO $ hPutStrLn stderr $ "WARNING!!! Unable
```

Figure: Not really supported.

HLL Compiler Targetting GCC

- HLL is Lisp-like, mimicking Scheme and Clojure.
- Almost purely functional, `set!` is accepted by the parser, but... is not supported.
- The only effectful part is `do`/`debug`.

HLL Compiler Targetting GCC

- HLL is Lisp-like, mimicking Scheme and Clojure.
- Almost purely functional, `set!` is accepted by the parser, but... is not supported.
- The only effectful part is `do`/`debug`.
- **Many** quirks:

HLL Compiler Targetting GCC

- HLL is Lisp-like, mimicking Scheme and Clojure.
- Almost purely functional, `set!` is accepted by the parser, but... is not supported.
- The only effectful part is `do`/`debug`.
- **Many** quirks:
  - No general TCE, explicit (and unchecked) tail recursion optimization using `recur`.

HLL Compiler Targetting GCC

- HLL is Lisp-like, mimicking Scheme and Clojure.
- Almost purely functional, `set!` is accepted by the parser, but... is not supported.
- The only effectful part is `do`/`debug`.
- **Many** quirks:
  - No general TCE, explicit (and unchecked) tail recursion optimization using `recur`.
  - Mildly insane function call convention to support `recur` – incompatible with ABI as in spec.

HLL Compiler Targetting GCC

- HLL is Lisp-like, mimicking Scheme and Clojure.
- Almost purely functional, `set!` is accepted by the parser, but... is not supported.
- The only effectful part is `do`/`debug`.
- **Many** quirks:
  - No general TCE, explicit (and unchecked) tail recursion optimization using `recur`.
  - Mildly insane function call convention to support `recur` – incompatible with ABI as in spec.
  - `if`s must be in a tail position - no support for `SEL`.

HLL Compiler Targetting GCC

- HLL is Lisp-like, mimicking Scheme and Clojure.
- Almost purely functional, `set!` is accepted by the parser, but... is not supported.
- The only effectful part is `do`/`debug`.
- **Many** quirks:
  - No general TCE, explicit (and unchecked) tail recursion optimization using `recur`.
  - Mildly insane function call convention to support `recur` – incompatible with ABI as in spec.
  - `if`s must be in a tail position - no support for `SEL`.
  - Built-ins such as `car` or `+` are special forms rather than first-class entities.

HLL Compiler Targetting GCC

- HLL is Lisp-like, mimicking Scheme and Clojure.
- Almost purely functional, `set!` is accepted by the parser, but... is not supported.
- The only effectful part is `do`/`debug`.
- **Many** quirks:
  - No general TCE, explicit (and unchecked) tail recursion optimization using `recur`.
  - Mildly insane function call convention to support `recur` – incompatible with ABI as in spec.
  - `if`s must be in a tail position - no support for `SEL`.
  - Built-ins such as `car` or `+` are special forms rather than first-class entities.
  - No macros.

HLL Compiler Targetting GCC

- HLL is Lisp-like, mimicking Scheme and Clojure.
- Almost purely functional, `set!` is accepted by the parser, but... is not supported.
- The only effectful part is `do`/`debug`.
- **Many** quirks:
  - No general TCE, explicit (and unchecked) tail recursion optimization using `recur`.
  - Mildly insane function call convention to support `recur` – incompatible with ABI as in spec.
  - `if`s must be in a tail position - no support for `SEL`.
  - Built-ins such as `car` or `+` are special forms rather than first-class entities.
  - No macros.
  - Virtually no diagnostics or static checks – a pain to debug.

HLL Compiler Targetting GCC

- HLL is Lisp-like, mimicking Scheme and Clojure.
- Almost purely functional, `set!` is accepted by the parser, but... is not supported.
- The only effectful part is `do`/`debug`.
- **Many** quirks:
  - No general TCE, explicit (and unchecked) tail recursion optimization using `recur`.
  - Mildly insane function call convention to support `recur` – incompatible with ABI as in spec.
  - `if`s must be in a tail position - no support for `SEL`.
  - Built-ins such as `car` or `+` are special forms rather than first-class entities.
  - No macros.
  - Virtually no diagnostics or static checks – a pain to debug.
- Implementing as eDSL could have given some type safety *almost for free*.

HLL Compiler Targetting GCC – Bugs

- A couple of nasty bugs ~30 hrs into the contest.

HLL Compiler Targetting GCC – Bugs

- A couple of nasty bugs ~30 hrs into the contest.

```
212 212  cg n env labels (Spec "do" []) = ([], labels)
213     -cg n env labels (Spec "do" (expr : rest)) = (code' ++ code'', labels')
    213 +cg n env labels (Spec "do" (expr : rest)) = (code' ++ code'', labels'')
214 214      where
```

Figure: The fact that it typechecks doesn't *always* mean it's correct. (But encoding the codegen in a monad *could* have helped.)

HLL Compiler Targetting GCC – Bugs

- A couple of nasty bugs ~30 hrs into the contest.

```
212 212  cg n env labels (Spec "do" []) = ([], labels)
213     -cg n env labels (Spec "do" (expr : rest)) = (code' ++ code'', labels')
    213 +cg n env labels (Spec "do" (expr : rest)) = (code' ++ code'', labels'')
214 214      where
```

Figure: The fact that it typechecks doesn't *always* mean it's correct. (But encoding the codegen in a monad *could* have helped.)

```
208 208  -- In general, never apply to unnamed functions: might be expensive
209     -cg n env labels (Spec "recur" args) = ([Op "LD" [Num 0, Num 0]] ++ code' ++ [Op "LD" [Num 0,
    209 +cg n env labels (Spec "recur" args) = ([Op "LD" ("$@recur" `var_lookup` (n, env))] ++ code'
210 210      where
```

Figure: `recur` only worked – occasionally – by accident.

Lambda-Man AI

- Lightning round AI retained as a fall-back – main AI may not return a move in situations it considers hopeless:
  - Manhattan distance to the closest pill as a value function.
  - Doesn't like being too close to ghosts or visiting recently seen locations (reduces the effect of local minima).

Lambda-Man AI

- Lightning round AI retained as a fall-back – main AI may not return a move in situations it considers hopeless:
  - Manhattan distance to the closest pill as a value function.
  - Doesn't like being too close to ghosts or visiting recently seen locations (reduces the effect of local minima).
- BFS: cuts off on dying, reaching anything edible or exceeding the depth limit.

Lambda-Man AI

- Lightning round AI retained as a fall-back – main AI may not return a move in situations it considers hopeless:
  - Manhattan distance to the closest pill as a value function.
  - Doesn't like being too close to ghosts or visiting recently seen locations (reduces the effect of local minima).
- BFS: cuts off on dying, reaching anything edible or exceeding the depth limit.
  - Being close to ghosts is penalized.

Lambda-Man AI

- Lightning round AI retained as a fall-back – main AI may not return a move in situations it considers hopeless:
  - Manhattan distance to the closest pill as a value function.
  - Doesn't like being too close to ghosts or visiting recently seen locations (reduces the effect of local minima).
- BFS: cuts off on dying, reaching anything edible or exceeding the depth limit.
  - Being close to ghosts is penalized.
  - If nothing edible found, simple heuristic value function kicks in.

Lambda-Man AI

- Lightning round AI retained as a fall-back – main AI may not return a move in situations it considers hopeless:
  - Manhattan distance to the closest pill as a value function.
  - Doesn't like being too close to ghosts or visiting recently seen locations (reduces the effect of local minima).
- BFS: cuts off on dying, reaching anything edible or exceeding the depth limit.
  - Being close to ghosts is penalized.
  - If nothing edible found, simple heuristic value function kicks in.
  - Likes running towards nearest power pill when ghosts are nearby.

Lambda-Man AI

- Lightning round AI retained as a fall-back – main AI may not return a move in situations it considers hopeless:
  - Manhattan distance to the closest pill as a value function.
  - Doesn't like being too close to ghosts or visiting recently seen locations (reduces the effect of local minima).
- BFS: cuts off on dying, reaching anything edible or exceeding the depth limit.
- Data structures: binary search trees (logarithmic access map and IntSet), simple queue from *PFDS*.

Lambda-Man AI

- Lightning round AI retained as a fall-back – main AI may not return a move in situations it considers hopeless:
  - Manhattan distance to the closest pill as a value function.
  - Doesn't like being too close to ghosts or visiting recently seen locations (reduces the effect of local minima).
- BFS: cuts off on dying, reaching anything edible or exceeding the depth limit.
- Data structures: binary search trees (logarithmic access map and IntSet), simple queue from *PFDS*.

```
404 (def q-append
405     (fun [q xs]
406         (q-norm (cons (car q) (foldl (fun [r x] (cons x r)) (cdr q) xs)))))
407 (def q-norm
408     (fun [q]
409         (if [atom? (car q)]
410             (cons (reverse (cdr q)) NIL)
411             q)))
```

Lambda-Man AI

- Lightning round AI retained as a fall-back – main AI may not return a move in situations it considers hopeless:
  - Manhattan distance to the closest pill as a value function.
  - Doesn't like being too close to ghosts or visiting recently seen locations (reduces the effect of local minima).
- BFS: cuts off on dying, reaching anything edible or exceeding the depth limit.
- Data structures: binary search trees (logarithmic access map and IntSet), simple queue from *PFDS*.
- Propagates ghosts (ignoring differences in speed) as long as they have no choice.

Lambda-Man AI

- Lightning round AI retained as a fall-back – main AI may not return a move in situations it considers hopeless:
  - Manhattan distance to the closest pill as a value function.
  - Doesn't like being too close to ghosts or visiting recently seen locations (reduces the effect of local minima).
- BFS: cuts off on dying, reaching anything edible or exceeding the depth limit.
- Data structures: binary search trees (logarithmic access map and IntSet), simple queue from *PFDS*.
- Propagates ghosts (ignoring differences in speed) as long as they have no choice.

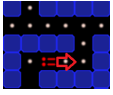


Figure: Initial state.

Lambda-Man AI

- Lightning round AI retained as a fall-back – main AI may not return a move in situations it considers hopeless:
  - Manhattan distance to the closest pill as a value function.
  - Doesn't like being too close to ghosts or visiting recently seen locations (reduces the effect of local minima).
- BFS: cuts off on dying, reaching anything edible or exceeding the depth limit.
- Data structures: binary search trees (logarithmic access map and IntSet), simple queue from *PFDS*.
- Propagates ghosts (ignoring differences in speed) as long as they have no choice.

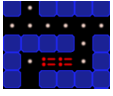


Figure: After 1 step.

Lambda-Man AI

- Lightning round AI retained as a fall-back – main AI may not return a move in situations it considers hopeless:
  - Manhattan distance to the closest pill as a value function.
  - Doesn't like being too close to ghosts or visiting recently seen locations (reduces the effect of local minima).
- BFS: cuts off on dying, reaching anything edible or exceeding the depth limit.
- Data structures: binary search trees (logarithmic access map and IntSet), simple queue from *PFDS*.
- Propagates ghosts (ignoring differences in speed) as long as they have no choice.

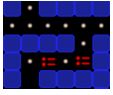


Figure: After 2 steps.

Lambda-Man AI

- Lightning round AI retained as a fall-back – main AI may not return a move in situations it considers hopeless:
  - Manhattan distance to the closest pill as a value function.
  - Doesn't like being too close to ghosts or visiting recently seen locations (reduces the effect of local minima).
- BFS: cuts off on dying, reaching anything edible or exceeding the depth limit.
- Data structures: binary search trees (logarithmic access map and IntSet), simple queue from *PFDS*.
- Propagates ghosts (ignoring differences in speed) as long as they have no choice.

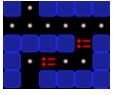


Figure: After 3 steps.

Lambda-Man AI

- Lightning round AI retained as a fall-back – main AI may not return a move in situations it considers hopeless:
  - Manhattan distance to the closest pill as a value function.
  - Doesn't like being too close to ghosts or visiting recently seen locations (reduces the effect of local minima).
- BFS: cuts off on dying, reaching anything edible or exceeding the depth limit.
- Data structures: binary search trees (logarithmic access map and IntSet), simple queue from *PFDS*.
- Propagates ghosts (ignoring differences in speed) as long as they have no choice.

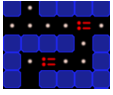


Figure: After 4 or more steps.

Lambda-Man AI

- Lightning round AI retained as a fall-back – main AI may not return a move in situations it considers hopeless:
  - Manhattan distance to the closest pill as a value function.
  - Doesn't like being too close to ghosts or visiting recently seen locations (reduces the effect of local minima).
- BFS: cuts off on dying, reaching anything edible or exceeding the depth limit.
- Data structures: binary search trees (logarithmic access map and IntSet), simple queue from *PFDS*.
- Propagates ghosts (ignoring differences in speed) as long as they have no choice.
- Value of scared ghosts and fruits discounted by time to expiration.

Lambda-Man AI – Quirks

- Termination of search on reaching anything edible leads to “interesting” behavior.

Lambda-Man AI – Quirks

- Termination of search on reaching anything edible leads to “interesting” behavior.
- Scaredy Lambda-Man – pessimistic estimates of ghost movement.

Lambda-Man AI – Quirks

- Termination of search on reaching anything edible leads to “interesting” behavior.
- Scaredy Lambda-Man – pessimistic estimates of ghost movement.
- Very little global preprocessing – computing tunnels as graph edges was tempting.

Lambda-Man AI – Quirks

- Termination of search on reaching anything edible leads to “interesting” behavior.
- Scaredy Lambda-Man – pessimistic estimates of ghost movement.
- Very little global preprocessing – computing tunnels as graph edges was tempting.
- Unnecessary map preprocessing on each step.

Lambda-Man AI – Quirks

- Termination of search on reaching anything edible leads to “interesting” behavior.
- Scaredy Lambda-Man – pessimistic estimates of ghost movement.
- Very little global preprocessing – computing tunnels as graph edges was tempting.
- Unnecessary map preprocessing on each step.
- AI state fragile – failures screw up future behavior.

Lambda-Man AI – Quirks

- Termination of search on reaching anything edible leads to "interesting" behavior.
- Scaredy Lambda-Man – pessimistic estimates of ghost movement.
- Very little global preprocessing – computing tunnels as graph edges was tempting.
- Unnecessary map preprocessing on each step.
- AI state fragile – failures screw up future behavior.

- **Not-a-quirk:** Ghost AIs known, but emulation impractical due to cycle limit.

Lambda-Man AI – Panic

- A few hours before the deadline...

Lambda-Man AI – Panic

- A few hours before the deadline...

Figure: OMG!!! Cycle limit!

Lambda-Man AI – Panic

- A few hours before the deadline...
- Hard to estimate cycles spent sanely from inside the simulation.

Lambda-Man AI – Panic

- A few hours before the deadline...
- Hard to estimate cycles spent sanely from inside the simulation.
- Regretted not having my own VM – web implementation sluggish *and* read-only.

Lambda-Man AI – Panic

- A few hours before the deadline...
- Hard to estimate cycles spent sanely from inside the simulation.
- Regretted not having my own VM – web implementation sluggish *and* read-only.

Figure: Good luck editing *this*.

Lambda-Man AI – Panic

- A few hours before the deadline...
- Hard to estimate cycles spent sanely from inside the simulation.
- Regretted not having my own VM – web implementation sluggish *and* read-only.
- Regretted not having macros – no easy inlining.

Lambda-Man AI – Panic

- A few hours before the deadline...
- Hard to estimate cycles spent sanely from inside the simulation.
- Regretted not having my own VM – web implementation sluggish *and* read-only.
- Regretted not having macros – no easy inlining.
- Random BFS depth cutoffs based on the map size.

Lambda-Man AI – Panic

- A few hours before the deadline...
- Hard to estimate cycles spent sanely from inside the simulation.
- Regretted not having my own VM – web implementation sluggish *and* read-only.
- Regretted not having macros – no easy inlining.
- Random BFS depth cutoffs based on the map size.
- Optimized by hand.

Lambda-Man AI – Panic

- A few hours before the deadline...
- Hard to estimate cycles spent sanely from inside the simulation.
- Regretted not having my own VM – web implementation sluggish *and* read-only.
- Regretted not having macros – no easy inlining.
- Random BFS depth cutoffs based on the map size.
- Optimized by hand.
  - Non-tail recursive list HOFs – no artifical limitations on stack depth, `RTN`s are cheaper than reversing the accumulator.

```
473 (def map
474     (fun [f xs]
475         (if [atom? xs]
476             NIL
477             (cons (f (car xs)) (map f (cdr xs))))))
478 ;(def map (fun [f xs] (reverse (map-rev NIL f xs))))
479 ;(def map-rev
480 ;    (fun [acc f xs]
481 ;        (if [atom? xs]
482 ;            acc
483 ;            (recur (cons (f (car xs)) acc) f (cdr xs)))))
```

Lambda-Man AI – Panic

- A few hours before the deadline...
- Hard to estimate cycles spent sanely from inside the simulation.
- Regretted not having my own VM – web implementation sluggish *and* read-only.
- Regretted not having macros – no easy inlining.
- Random BFS depth cutoffs based on the map size.
- Optimized by hand.
  - Non-tail recursive list HOFs – no artifical limitations on stack depth, `RTN`s are cheaper than reversing the accumulator.
  - Fused `map`s and `filter`s in a few places.

Lambda-Man AI – Panic

- A few hours before the deadline...
- Hard to estimate cycles spent sanely from inside the simulation.
- Regretted not having my own VM – web implementation sluggish *and* read-only.
- Regretted not having macros – no easy inlining.
- Random BFS depth cutoffs based on the map size.
- Optimized by hand.
  - Non-tail recursive list HOFs – no artifical limitations on stack depth, `RTN`s are cheaper than reversing the accumulator.
  - Fused `map`s and `filter`s in a few places.
  - Eliminated some unneeded intermediate variables.

Lambda-Man AI – Panic

- A few hours before the deadline...
- Hard to estimate cycles spent sanely from inside the simulation.
- Regretted not having my own VM – web implementation sluggish *and* read-only.
- Regretted not having macros – no easy inlining.
- Random BFS depth cutoffs based on the map size.
- Optimized by hand.
  - Non-tail recursive list HOFs – no artifical limitations on stack depth, `RTN`s are cheaper than reversing the accumulator.
  - Fused `map`s and `filter`s in a few places.
  - Eliminated some unneeded intermediate variables.
- Seems to have worked.

Ghost AI

- *Very* limited resources – even accounting for no need to make decisions on most runs.

Ghost AI

- *Very* limited resources – even accounting for no need to make decisions on most runs.
- Wanted something simple and reasonably robust.

Ghost AI

- *Very* limited resources – even accounting for no need to make decisions on most runs.
- Wanted something simple and reasonably robust.
- Tries to minimize L_1 distance to Lambda-Man (*Blinky*/*Chaser*-style).

Ghost AI

- *Very* limited resources – even accounting for no need to make decisions on most runs.
- Wanted something simple and reasonably robust.
- Tries to minimize L_1 distance to Lambda-Man (*Blinky*/*Chaser*-style).
  - **Problem:** Ghosts tend to clump together.

Ghost AI

- *Very* limited resources – even accounting for no need to make decisions on most runs.
- Wanted something simple and reasonably robust.
- Tries to minimize L_1 distance to Lambda-Man (*Blinky*/*Chaser*-style).
  - **Problem:** Ghosts tend to clump together.
  - **Problem:** Can get stuck in dead ends easily.

Ghost AI

- *Very* limited resources – even accounting for no need to make decisions on most runs.
- Wanted something simple and reasonably robust.
- Tries to minimize L_1 distance to Lambda-Man (*Blinky*/*Chaser*-style).
- Ghost's index affects tie-breaks: different ghosts favour different directions at intersections.

Ghost AI

- *Very* limited resources – even accounting for no need to make decisions on most runs.
- Wanted something simple and reasonably robust.
- Tries to minimize L_1 distance to Lambda-Man (*Blinky* / *Chaser*-style).
- Ghost's index affects tie-breaks: different ghosts favour different directions at intersections.
- Simple counter: after a few actual decisions in a row resulting in horizontal or vertical move, that axis is excluded.

Ghost AI

- *Very* limited resources – even accounting for no need to make decisions on most runs.
- Wanted something simple and reasonably robust.
- Tries to minimize L_1 distance to Lambda-Man (*Blinky*/*Chaser*-style).
- Ghost's index affects tie-breaks: different ghosts favour different directions at intersections.
- Simple counter: after a few actual decisions in a row resulting in horizontal or vertical move, that axis is excluded.
- Pretty efficient on non-pathological maps: ghosts tend to surround the Lambda-Man.

Sources

- Submission is available on GitHub, link can be found on ICFPC subreddit, along with many interesting submissions and reports from other teams: `http://www.reddit.com/r/icfpcontest`

How To Win The ICFP Programming Contest

What I learned from past failures and how I used it this time – or not

- Try to decipher the hint.

How To Win The ICFP Programming Contest

What I learned from past failures and how I used it this time – or not

- Try to decipher the hint.

Figure: The apparently black background image on the official site has slight brightness variations. This “steganographic message” turned out to be random noise. D’oh!

How To Win The ICFP Programming Contest

What I learned from past failures and how I used it this time – or not

- ~~Try to decipher the hint.~~

How To Win The ICFP Programming Contest

What I learned from past failures and how I used it this time – or not

- ~~Try to decipher the hint.~~
- Do your research.

How To Win The ICFP Programming Contest

What I learned from past failures and how I used it this time – or not

- ~~Try to decipher the hint.~~
- Do your research.
  - There are people smarter than I am on the Net.

How To Win The ICFP Programming Contest

What I learned from past failures and how I used it this time – or not

- ~~Try to decipher the hint.~~
- Do your research.
  - There are people smarter than I am on the Net.
  - SECD – little insight into the problem, the spec was enough.

How To Win The ICFP Programming Contest

What I learned from past failures and how I used it this time – or not

- ~~Try to decipher the hint.~~
- Do your research.
  - There are people smarter than I am on the Net.
  - SECD – little insight into the problem, the spec was enough.
  - Classic ghost AIs – could have mislead me.

How To Win The ICFP Programming Contest

What I learned from past failures and how I used it this time – or not

- ~~Try to decipher the hint.~~
- ~~Do your research.~~

How To Win The ICFP Programming Contest

What I learned from past failures and how I used it this time – or not

- ~~Try to decipher the hint.~~
- ~~Do your research.~~
- If you want to do well – get on a strong team.

How To Win The ICFP Programming Contest

What I learned from past failures and how I used it this time – or not

- ~~Try to decipher the hint.~~
- ~~Do your research.~~
- If you want to do well – get on a strong team.
  - But can be stressful.

How To Win The ICFP Programming Contest

What I learned from past failures and how I used it this time – or not

- ~~Try to decipher the hint.~~
- ~~Do your research.~~
- If you want to do well – get on a strong team.
  - But can be stressful.
  - And I'm in it for the sheer fun of it...

How To Win The ICFP Programming Contest

What I learned from past failures and how I used it this time – or not

- ~~Try to decipher the hint.~~
- ~~Do your research.~~
- ~~If you want to do well – get on a strong team.~~

How To Win The ICFP Programming Contest

What I learned from past failures and how I used it this time – or not

- ~~Try to decipher the hint.~~
- ~~Do your research.~~
- ~~If you want to do well – get on a strong team.~~
- Don't forget to rest, eat and sleep.

How To Win The ICFP Programming Contest

What I learned from past failures and how I used it this time – or not

- ~~Try to decipher the hint.~~
- ~~Do your research.~~
- ~~If you want to do well – get on a strong team.~~
- Don't forget to rest, eat and sleep. On the other hand...

How To Win The ICFP Programming Contest

What I learned from past failures and how I used it this time – or not

- ~~Try to decipher the hint.~~
- ~~Do your research.~~
- ~~If you want to do well – get on a strong team.~~
- ~~Don't forget to rest, eat and sleep.~~

How To Win The ICFP Programming Contest

What I learned from past failures and how I used it this time – or not

- ~~Try to decipher the hint.~~
- ~~Do your research.~~
- ~~If you want to do well – get on a strong team.~~
- ~~Don't forget to rest, eat and sleep.~~
- Random tweaks don't work. Think instead.

How To Win The ICFP Programming Contest

What I learned from past failures and how I used it this time – or not

- ~~Try to decipher the hint.~~
- ~~Do your research.~~
- ~~If you want to do well – get on a strong team.~~
- ~~Don't forget to rest, eat and sleep.~~
- Random tweaks don't work. Think instead.

```
[comp-thr (fun [m] (if [<= (bstm-w m) 25] 20 (if [<= (bstm-w m) 35] 12 (if [<= (bstm-w m) 50] 8 5))))]
```

Figure: Cutoff depth selection for BFS. Speaks for itself.

How To Win The ICFP Programming Contest

What I learned from past failures and how I used it this time – or not

- ~~Try to decipher the hint.~~
- ~~Do your research.~~
- ~~If you want to do well – get on a strong team.~~
- ~~Don't forget to rest, eat and sleep.~~
- ~~Random tweaks don't work. Think instead.~~

How To Win The ICFP Programming Contest

What I learned from past failures and how I used it this time – or not

- ~~Try to decipher the hint.~~
- ~~Do your research.~~
- ~~If you want to do well – get on a strong team.~~
- ~~Don't forget to rest, eat and sleep.~~
- ~~Random tweaks don't work. Think instead.~~
- Read the spec carefully.

How To Win The ICFP Programming Contest

What I learned from past failures and how I used it this time – or not

- ~~Try to decipher the hint.~~
- ~~Do your research.~~
- ~~If you want to do well – get on a strong team.~~
- ~~Don't forget to rest, eat and sleep.~~
- ~~Random tweaks don't work. Think instead.~~
- Read the spec carefully.
  - Signed my submissions with SHA256 instead of SHA1.

How To Win The ICFP Programming Contest

What I learned from past failures and how I used it this time – or not

- ~~Try to decipher the hint.~~
- ~~Do your research.~~
- ~~If you want to do well – get on a strong team.~~
- ~~Don't forget to rest, eat and sleep.~~
- ~~Random tweaks don't work. Think instead.~~
- Read the spec carefully.
  - Signed my submissions with SHA256 instead of SHA1.
  - Only noticed minutes before the deadline.

How To Win The ICFP Programming Contest

What I learned from past failures and how I used it this time – or not

- ~~Try to decipher the hint.~~
- ~~Do your research.~~
- ~~If you want to do well – get on a strong team.~~
- ~~Don't forget to rest, eat and sleep.~~
- ~~Random tweaks don't work. Think instead.~~
- ~~Read the spec carefully.~~

How To Win The ICFP Programming Contest

What I learned from past failures and how I used it this time – or not

- ~~Try to decipher the hint.~~
- ~~Do your research.~~
- ~~If you want to do well – get on a strong team.~~
- ~~Don't forget to rest, eat and sleep.~~
- ~~Random tweaks don't work. Think instead.~~
- ~~Read the spec carefully.~~
- Use C++.

How To Win The ICFP Programming Contest

What I learned from past failures and how I used it this time – or not

- ~~Try to decipher the hint.~~
- ~~Do your research.~~
- ~~If you want to do well – get on a strong team.~~
- ~~Don't forget to rest, eat and sleep.~~
- ~~Random tweaks don't work. Think instead.~~
- ~~Read the spec carefully.~~
- Use C++.
  - It appears to dominate the field as far as the tools of choice for discriminating hackers are concerned.

How To Win The ICFP Programming Contest

What I learned from past failures and how I used it this time – or not

- ~~Try to decipher the hint.~~
- ~~Do your research.~~
- ~~If you want to do well – get on a strong team.~~
- ~~Don't forget to rest, eat and sleep.~~
- ~~Random tweaks don't work. Think instead.~~
- ~~Read the spec carefully.~~
- Use C++.
  - It appears to dominate the field as far as the tools of choice for discriminating hackers are concerned.
  - Used Haskell this year.

How To Win The ICFP Programming Contest

What I learned from past failures and how I used it this time – or not

- ~~Try to decipher the hint.~~
- ~~Do your research.~~
- ~~If you want to do well – get on a strong team.~~
- ~~Don't forget to rest, eat and sleep.~~
- ~~Random tweaks don't work. Think instead.~~
- ~~Read the spec carefully.~~
- ~~Use C++.~~

How To Win The ICFP Programming Contest

What I learned from past failures and how I used it this time – or not

- ~~Try to decipher the hint.~~
- ~~Do your research.~~
- ~~If you want to do well – get on a strong team.~~
- ~~Don't forget to rest, eat and sleep.~~
- ~~Random tweaks don't work. Think instead.~~
- ~~Read the spec carefully.~~
- ~~Use C++.~~
- All in all, I have no idea how does this work.

Lots of teams from Eastern Europe and Russia for some reason?

- In 2012 the organizers expressed puzzlement with a somewhat skewed geographical distribution of teams.

Lots of teams from Eastern Europe and Russia for some reason?

- In 2012 the organizers expressed puzzlement with a somewhat skewed geographical distribution of teams.
- Dmitry Astapov's ICFPC reports (`http://users.livejournal.com/_adept_/tag/icfpc`) – in Russian

Lots of teams from Eastern Europe and Russia for some reason?

- In 2012 the organizers expressed puzzlement with a somewhat skewed geographical distribution of teams.
- Dmitry Astapov's ICFPC reports (`http://users.livejournal.com/_adept_/tag/icfpc`) – in Russian
- Better reading than most fiction.

Lots of teams from Eastern Europe and Russia for some reason?

- In 2012 the organizers expressed puzzlement with a somewhat skewed geographical distribution of teams.
- Dmitry Astapov's ICFPC reports (`http://users.livejournal.com/_adept_/tag/icfpc`) – in Russian
- Better reading than most fiction.
- The 2006 one stands out in particular (and so did the contest itself).

Lots of teams from Eastern Europe and Russia for some reason?

- In 2012 the organizers expressed puzzlement with a somewhat skewed geographical distribution of teams.
- Dmitry Astapov's ICFPC reports (`http://users.livejournal.com/_adept_/tag/icfpc`) – in Russian
- Better reading than most fiction.
- The 2006 one stands out in particular (and so did the contest itself).
- Many folks who read those reports were instantly hooked and wanted to give it a shot – myself included.

Lots of teams from Eastern Europe and Russia for some reason?

- In 2012 the organizers expressed puzzlement with a somewhat skewed geographical distribution of teams.
- Dmitry Astapov's ICFPC reports (`http://users.livejournal.com/_adept_/tag/icfpc`) – in Russian
- Better reading than most fiction.
- The 2006 one stands out in particular (and so did the contest itself).
- Many folks who read those reports were instantly hooked and wanted to give it a shot – myself included.
- Now you know whom to blame.

Compilers Are Scary Stuff

- Considered to be a dark art among the uninitiated.

Compilers Are Scary Stuff

- Considered to be a dark art among the uninitiated.
- Dr. Alex Aiken and his team are running an open Compilers class on Coursera – `https://www.coursera.org/course/compilers`

Compilers Are Scary Stuff

- Considered to be a dark art among the uninitiated.
- Dr. Alex Aiken and his team are running an open Compilers class on Coursera – `https://www.coursera.org/course/compilers`
- It's a good one. Get through it and you'll never be scared of writing a compiler again.

Compilers Are Scary Stuff

- Considered to be a dark art among the uninitiated.
- Dr. Alex Aiken and his team are running an open Compilers class on Coursera – `https://www.coursera.org/course/compilers`
- It's a good one. Get through it and you'll never be scared of writing a compiler again.
- **Correction:** writing a *toy* compiler.

Compilers Are Scary Stuff

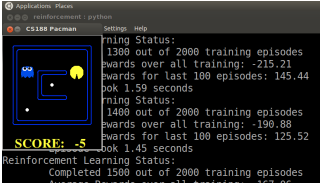
- Considered to be a dark art among the uninitiated.
- Dr. Alex Aiken and his team are running an open Compilers class on Coursera – `https://www.coursera.org/course/compilers`
- It's a good one. Get through it and you'll never be scared of writing a compiler again.
- **Correction:** writing a *toy* compiler.
- Thankfully, the GHC and GCC in this contest were not the Real Thing.

AI Is A Bit Scary Too

- Dr. Dan Klein and others developed an Artificial Intelligence class on edX – `https://www.edx.org/course/uc-berkeleyx/uc-berkeleyx-cs188-1x-artificial-579`

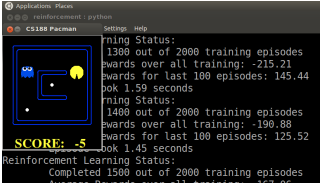
AI Is A Bit Scary Too

- Dr. Dan Klein and others developed an Artificial Intelligence class on edX – `https://www.edx.org/course/uc-berkeleyx/uc-berkeleyx-cs188-1x-artificial-579`
- Curiously, it uses Pac-Man in examples and assignments a lot.



AI Is A Bit Scary Too

- Dr. Dan Klein and others developed an Artificial Intelligence class on edX – `https://www.edx.org/course/uc-berkeleyx/uc-berkeleyx-cs188-1x-artificial-579`
- Curiously, it uses Pac-Man in examples and assignments a lot.



- I'm sure that's sheer coincidence and had *nothing* to do with my performance here.

This was even more awesome than ICFP contests usually are.
Heartfelt thanks to the organizers.
And thank you for listening.

And remember – for the next year, Haskell is the programming tool of choice for discriminating hackers.