# Collision for 75-step SHA-1: Intensive Parallelization with GPU

E. A. Grechnikov
grechnik@mccme.ru

A. V. Adinetz
adinetz@gmail.com

November 29, 2011

**Abstract**

We present a brief report on the collision search for the reduced SHA-1. With a few improvements to our previous work, directed at efficient parallelization on a GPU cluster, we managed to construct a new collision for 75-step reduced SHA-1 hash function.

## 1 Introduction

Recently we have constructed a new 2-block collision for 72-step SHA-1 and another one for 73-step SHA-1 in [1]. Our previous work has improved the method for the automatic construction suggested by Christophe De Cannière and Christian Rechberger in [2], [3].

In this report we focus on the intensive parallelization with GPUs. As our previous algorithm required large amount of computing power, extending it to GPUs seems natural. This results in some changes in the search of characteristics, and also in implementation. Using GPU, we were able to construct a new 2-block collision for 75-step SHA-1.

## 2 Improvements in the search of characteristics

We refer to [2] for the detailed description of basic method and notations and to [1] for our previous improvements, which are still used.

The search is naturally divided into *generating* part and *testing* part: on the first 16 steps many possible messages are generated and later every message is only tested and can be either accepted or rejected, but does not produce any new messages.

The algorithm is trivially parallelizable. Therefore, the main challenge to its performance on GPU is *coherency*, i.e. similarity of control flow between threads in a warp. We noticed during experiments that coherency can be improved, if the characteristic has many free bit positions in the last few steps of the generating part, and has very little freedoms in the 5 steps which immediately preceed the GPU part.

To take this effect into account, we modify the stage 2 of the method, namely random trials of positions where to impose the - restriction. Namely, we decrease the probability to select a position from the last steps of the generating part, so restrictions tend to concentrate on the first steps. This seems to increase the final workfactor slightly, but the gain from the higher coherency more than compensates this.

We also note that the automatic search of the characteristics can be improved with the following trick. First, we find some characteristics. Second, we look in the first few steps

of the testing part, examine the conditions which increase $P_u$, and copy these conditions to the initial characteristics. Next, we run the automatic search again, starting from the initial characteristics with new conditions. If the number of added conditions is not too large, then the automated search still has considerable freedom and can produce even better characteristics.

# 3 GPU Implementation

Generating part can be naturally viewed as backtracking search, and the testing part is built into the search as a check at the final round. Generating part can be subdivided into CPU part and GPU part. During CPU part, the tree is unfolded to a certain search depth, so that enough *search stacks* is generated to make use of GPU parallelism. The unfold depth is currently selected manually for each characteristic after initial experimentation. Too small a depth will result in insufficient parallelism, and thus underutilization of GPUs, while too large a depth will require too much GPU memory. During experiments we have found out that about 100,000 search stacks per GPU is required to utilize it efficiently.

During GPU part, all stacks are searched in parallel by multiple GPUs. If search is finished for a stack, that stack is removed; no new stacks are generated during GPU part. The main GPU kernel implements backtracking search and testing part. In this kernel, each thread works on a single search stack. For each invocation of the main kernel, each thread does a fixed number of search steps, and then finishes its work. Main kernel also collects statistical data, which include number of nodes traversed and maximum depth reached. Between kernel invocations, checks for maximum depth are performed, and an array of stacks is compacted if necessary.

Computations are distributed on a cluster using MPI. Each MPI process is assigned to a single GPU. Search stacks are distributed in block-cyclic fashion among processes. Each process generates all seach stacks, and discards those which do not belong to it. Initially, there was a global barrier after each invocation of main GPU kernel. However, experimental data has shown that this global barrier can increase runtime by a factor of 2 for certain characteristics. Therefore, in subsequent implementations, there is no global barrier. Instead, master process (which has rank 0) spawns a separate thread for collection of statistics, and other processes report statistical data to that thread after each invocation of the main GPU kernel. Process 0 also spawns another thread, which displays aggregate statistics at regular intervals, every minute by default.

We used NUDA (Nemerle Unified Device Architecture) [4], a set of GPGPU extensions for Nemerle, for implementation. It was chosen because it is open-source and provides a higher-level view of GPU programming. It also handles low-level details, such as data transfer and marshalling of parameters. Internally, NUDA uses OpenCL to generate code for loops sent to GPU.

In our initial implementation, back-tracking search was implemented as a single loop, with stages requiring special handling implemented as conditionals. There were 2 such stages: switching between search rounds, where additional values needed to be precomputed, and testing the generated message. Testing was implemented in a separate function, the loop along testing rounds being fully unrolled using NUDA's `inline` annotation.

However, while performance of our initial implementation was good for some characteristics, it was awful for other ones. Specifically, while 60% efficiency was reached on 73-1 characteristic, only 15% was attained on 72-2. Such low performance is due to low coherency between neighbouring threads. We applied two optimizations to increase coherency.

The first optimization was sorting the search stacks after each main kernel invocation. We quickly established that using stable sort was better than unstable sort (e.g. quicksort), as it preserves the order of values with same keys, and thus preserves coherency. We also experimented with various sorting keys, but finally settled on using the value being searched at the current round. On 72-2, stable sorting searched value gave about 45% increase in performance compared to initial implementation. Stable sort was implemented on GPU using radix sorting algorithm [5], and experiments have shown that time spent in sorting stacks is negligible compared to time spent in backtracking search.

The second optimization was replacing the single-loop implementation of backtracking search with triple-loop nest implementation. The innermost loop only traverses nodes and checks for characteristics inside a single round, until either a good node is found, or all nodes of this round have been searched. The second innermost loop implements message testing at the final search round. This allows to check messages without incurring the overhead of round switching. As more that 75% of time is spent in message testing for some characteristics, this improves performance. Finally, the outermost loop performs switching between rounds, and also checks conditions for kernel termination. It was shown that triple-loop configuration is far better in preserving coherency than single-loop configuration. Together, triple-loop and stable sort by round value gave about 2x improvement over initial implementation for 75-1 characteristic.

A number of additional optimizations has been performed. This included using constant memory for storing characteristics, and using shared on-chip memory for storing search stacks. Contrary to our expectations, using shared memory gave only 2.5% improvement over using global memory. This indicates that the memory footprint of the algorithm fits well into L1 cache of NVidia Fermi GPUs, which were used for computations. We also modified the algorithm used for generation of characteristics, as described in the previous section, to provide better coherency. The final version used for 75-round collision search incorporated all optimizations described above.

Because the second run was estimated to take much longer time, support for checkpoints has been integrated into the application. And since the number of available GPUs was expected to fluctuate, checkpointing implementation also supported starting from a checkpoint with different number of processes than the checkpoint was originally generated with. As there is no global synchronization between processes during GPU part, each process writes the checkpoint independently to its file at fixed time intervals, by default once each hour.

# 4 Calculations

Initial experiments and tuning were performed on "GraphIT!" system installed at Research Computing Center, Lomonosov Moscow State University (RCC MSU). This system has 16 nodes, each equipped with 3 NVidia Fermi M2050 GPUs, each having 3 GB of GPU memory. This gives a total of 48 GPUs, however, no more that 30 of them were used for computational experiments.

Final computations were performed on GPU partition of "Lomonosov", currently the most powerful supercomputer in Russia, also installed at RCC MSU. Each GPU node of "Lomonosov" has 2 NVidia Fermi X2070 GPUs, each having 6 GB of GPU memory. As the system is still in beta stage, not all nodes are available for the end users.

For the first block, the estimated work factor was $2^{58}$ traversed nodes. Due to constraints on messages, the characteristic was split into 4 parts, and one of them was chosen for the initial run. The run was performed on 264 GPUs, and took 11000 seconds to find the first collision block. The actual number of nodes traversed was $2^{54.06}$. Here, we count

both search rounds and testing rounds as nodes, though the latter takes about 2.5 more time to compute than the former. For the first block, about 40% were testing rounds, with the rest being search rounds.

The second block has an estimated work factor of $2^{63.01}$ traversed nodes. It started on 320 GPUs, and was restarted from checkpoints multiple times due to node failure or availability of additional GPUs. It finished with 512 GPUs. Based on logs, we estimate that 455 GPUs on average were used. The entire computation took 1904252 seconds, or 22 days and 45 minutes, and traversed a total of $2^{61.92}$ nodes of the search tree. About 58.8% of them were testing rounds, and the rest were search rounds. Computational efficiency during the second run was estimated at 52% (of peak GPU performance). Had the full partition with 1554 GPUs been operational, the entire computation would still require 6.5 days to complete.

In both cases, we had some "luck", as it took less time to find the collision block than it had been originally estimated. Specifically, computation finished about 16 times faster in the first case, and about 2 times faster in the second case. If not for this "luck", the entire computation would have taken 1.5 month to complete.

Comparing current GPU implementation with the previous parallel search implementation in [1], we estimate a single GPU to be as fast as 39 x86 CPU cores. Thus the equivalent number of CPU cores for the second computation would be 17745, which is about the same number used in our previous computation. However, given the demand for CPUs on "Lomonosov", it was highly unlikely that we could get so many cores for several weeks. The demand for GPUs was much lower, and it was therefore possible to get the required number of GPU cores for the required time.

# 5 Acknowledgements

We are thankful to Research Computing Center of Moscow State University, for providing us with access to computational resources we required. We are also thankful to support team of "T-Platforms", and particularly to Anton Korzh for assistance in solving problems related to hardware and drivers.

# References

[1] E. A. Grechnikov. *Collisions for 72-step and 73-step SHA-1: Improvements in the Method of Characteristics*. Cryptology ePrint Archive: Report 2010/413, available at `http://eprint.iacr.org/2010/413`.

[2] Christophe De Cannière and Christian Rechberger. *Finding SHA-1 Characteristics: General Results and Applications.* In Proceedings of ASIACRYPT, volume 4284 of LNCS, pages 1–20. Springer, 2006.

[3] Christophe De Cannière, Florian Mendel, and Christian Rechberger. *Collisions for 70-step SHA-1: On the Full Cost of Collision Search.* In Proceedings of Selected Areas in Cryptography, volume 4876 of LNCS, pages 56-73. Springer, 2007.

[4] Andrew V. Adinetz. *NUDA Programmer's Guide*. URL: `http://nuda.sf.net`.

[5] Nadathur Satish, Changkyu Kim, Jatin Chhugani, Anthony D. Nguyen, Victor W. Lee, Daehyun Kim, and Pradeep Dubey. *Fast sort on CPUs and GPUs: a case for bandwidth oblivious SIMD sort.* In Proceedings of the 2010 international conference

on Management of data (SIGMOD '10), pages 351-362. ACM, New York, NY, USA, 2010.

# A Collision for the 75-step reduced SHA-1

**Table 1. Example of a 75-step SHA-1 collision using the standard IV**

| $i$ | Message 1, first block | | | | Message 1, second block | | | |
|---|---|---|---|---|---|---|---|---|
| 1–4 | F01EE8EE | BDDFF313 | B2F59EE4 | BB37F2BB | F072633F | 0D32226A | DFF74459 | 98507743 |
| 5–8 | 2F472A36 | 1C052F6A | 96403EF0 | F144298B | EEFE63DD | FE10D5C5 | AFE33902 | EF74984E |
| 9–12 | DAF5519C | 7A90DD71 | 2BF3718E | A7E3DE6D | 350272F7 | DB382ABC | 155B0414 | B800179D |
| 13–16 | EFFA975E | 9B00AA95 | 6056E3EE | 2BA4483A | 18ECD4BC | 15497213 | 1505284C | 60C4F869 |
| $i$ | Message 2, first block | | | | Message 2, second block | | | |
| 1–4 | 001EE884 | 3DDFF353 | 22F59E94 | 0B37F2E8 | 00726355 | 8D32222A | 4FF74429 | 28507710 |
| 5–8 | 1F472A3E | 1C052F29 | 46403E82 | 4144299B | DEFE63D5 | FE10D586 | 7FE33970 | 5F74985E |
| 9–12 | 2AF551FE | BA90DD33 | 2BF371BE | 47E3DE2F | C5027295 | 1B382AFE | 155B0424 | 580017DF |
| 13–16 | CFFA973E | 7B00AAD4 | 4056E3BE | EBA4487B | 38ECD4DC | F5497252 | 3505281C | A0C4F828 |
| $i$ | XOR-differences are the same for both blocks | | | | | | | |
| 1–4 | F000006A | 80000040 | 90000070 | B0000053 | F000006A | 80000040 | 90000070 | B0000053 |
| 5–8 | 30000008 | 00000043 | D0000072 | B0000010 | 30000008 | 00000043 | D0000072 | B0000010 |
| 9–12 | F0000062 | C0000042 | 00000030 | E0000042 | F0000062 | C0000042 | 00000030 | E0000042 |
| 13–16 | 20000060 | E0000041 | 20000050 | C0000041 | 20000060 | E0000041 | 20000050 | C0000041 |
| $i$ | The colliding hash values | | | | | | | |
| 1–5 | 3DF7F21E | 130079F3 | C2E6EFFF | FD9C4141 | 9AA8723A | | | |

# B Characteristics for the 75-step reduced SHA-1

**Table 2. Characteristics used for the first block of the 75-step collision**

| $i$ | $\nabla A_i$ | $\nabla W_i$ | $F_W$ | $P_u(i)$ | $P_c(i)$ | $N_s(i)$ |
|---|---|---|---|---|---|---|
| -4: | 00001111010010111000011111000011 | | | | | |
| -3: | 01000000110010010101000111011000 | | | | | |
| -2: | 01100010111010110111001111111010 | | | | | |
| -1: | 11101111110011011010101110001001 | | | | | |
| 0: | 01100111010001010010001100000001 | uuuu000000----------10-01uu0u1u0 | 10 | -0.28 | 0.00 | 0.00 |
| 1: | 100n111111-------0--00-1u01n0nn1 | u0111-----------------110n010011 | 10 | -0.07 | -0.06 | 0.06 |
| 2: | n0011111000--------u--nn010100n1 | u01u00-----------------01uun0100 | 6 | -0.33 | -0.31 | 9.87 |
| 3: | 0nnn0000010--0unu-u--0-001unnnn0 | u0uu10--0--------------01n1u10uu | 3 | 0.00 | 0.00 | 15.54 |
| 4: | 1un1uu011unu1u0-000-10-111110u10 | 00un11110----------------011n110 | 2 | -4.19 | -4.19 | 18.54 |
| 5: | nnnn00uun011000nu00-0-1101uu0010 | 000111000000-------------u1010un | 2 | -2.34 | -2.13 | 16.35 |
| 6: | 0100n0101n0111-1001u0-u000uuu011 | un0u0110----------------uuu00n0 | 2 | 0.00 | 0.00 | 16.01 |
| 7: | 100nuuuu100uunn-1u1n-1u1uu1u1101 | u1uu00010----------------00n1011 | 2 | 0.00 | 0.00 | 18.01 |
| 8: | 1111010unnnnn00n11n0-0-001nn1001 | uunu1010111101----0------nn111n0 | 0 | 0.00 | 0.00 | 20.01 |
| 9: | n0001101110011nu000111nn1u1uuu0n | nu1110101001000---------0u1100n1 | 0 | 0.00 | 0.00 | 20.01 |
| 10: | n11100010010101n011n100100u10110 | 001010111111-------------0nn1110 | 5 | -2.00 | -1.00 | 20.01 |
| 11: | 11101011110----un1-unnn1nnn1n11n | unu0011----------------0-u1011n1 | 11 | -1.01 | -0.54 | 23.01 |
| 12: | 11101101---------10--11010011000 | 11u0-------------0---1-1-un11110 | 9 | -1.00 | -0.98 | 33.00 |
| 13: | 1111101101-------01--01011010n11 | unn110----------------0-01n01010u | 10 | -4.00 | -0.83 | 41.00 |
| 14: | n00111010------------0-000110001 | 01u00--------1---------11u1n1110 | 7 | -6.00 | -0.66 | 47.00 |
| 15: | 001111-0111------------n01111101 | nn1010---------0---------n11101n | 10 | -1.00 | 0.00 | 48.00 |
| 16: | n-011--0-----------------1110u01 | nnn001-----------------0nnn01u0 | 0 | -1.00 | -0.01 | 57.00 |
| 17: | --1--1-----------------0111-un0 | n0u1-------------------0011011un | 0 | -2.00 | -0.68 | 56.00 |
| 18: | u-0-0--------------------1---1-- | nn1110-----------------110n11n1 | 0 | 0.00 | 0.00 | 54.00 |
| 19: | -------------------------0-0-1 | 0nn01------------------0u11010n | 0 | -1.00 | -0.91 | 54.00 |
| 20: | n------------------------------ | uu011------------------11un11n0 | 0 | -1.00 | -1.00 | 53.00 |
| 21: | ------------------------------ | 10n11------------------0111001n | 0 | -1.00 | -1.00 | 52.00 |
| 22: | ------------------------------ | nu00-------------------111011n0 | 0 | -2.00 | -2.00 | 51.00 |
| 23: | ----------------------------n- | uuu1-------------------0u0110n0 | 0 | -2.00 | -2.00 | 49.00 |
| 24: | ----------------------------n- | 1un1-------------------1u0011n0 | 0 | -1.00 | -1.00 | 47.00 |
| 25: | ------------------------------ | n100-------------------001111n1 | 0 | -1.00 | -1.00 | 46.00 |
| 26: | ------------------------------ | 00001------------------10001010 | 0 | 0.00 | 0.00 | 45.00 |
| 27: | ------------------------------ | 110--------------------10000001 | 0 | 0.00 | -0.00 | 45.00 |
| 28: | ------------------------------ | n01110-----------------11000010 | 0 | 0.00 | -0.00 | 45.00 |
| 29: | ------------------------------ | 0100-------------------101011u0 | 0 | -1.00 | -1.00 | 45.00 |
| 30: | ----------------------------u- | 11-0-------------------10n001011 | 0 | 0.00 | 0.00 | 44.00 |
| 31: | ------------------------------ | 10100------------------10011101 | 0 | -2.00 | -2.00 | 44.00 |
| 32: | ----------------------------n- | n10--------------------10u010011 | 0 | 0.00 | 0.00 | 42.00 |
| 33: | ------------------------------ | n-1--------------------001011001 | 0 | -2.00 | -2.00 | 42.00 |
| 34: | ----------------------------n- | 1001-------------------00u101100 | 0 | 0.00 | 0.00 | 40.00 |
| 35: | ------------------------------ | n0-1-------------------11010100n | 0 | -2.00 | -2.00 | 40.00 |
| 36: | --------------------------u--- | -0-1-------------------10nu10111 | 0 | 0.00 | 0.00 | 38.00 |
| 37: | ------------------------------ | n11--------------------1110100nu | 0 | -1.00 | -1.00 | 38.00 |
| 38: | ------------------------------ | 1x0--------------------1000001u0 | 0 | -2.00 | -1.00 | 37.00 |
| 39: | ----------------------------u- | nx0--------------------11n01111- | 0 | -1.00 | 0.00 | 35.00 |
| 40: | ------------------------------ | uu---------------------1100010n1 | 0 | -1.00 | -1.00 | 34.00 |
| 41: | ------------------------------ | x10-1------------------0111000110 | 0 | -1.00 | -1.00 | 33.00 |
| 42: | ------------------------------ | x1---------------------1011101-1 | 0 | -1.00 | -1.00 | 32.00 |
| 43: | ------------------------------ | u----1-----------------0000111n0 | 0 | -1.00 | -1.00 | 31.00 |
| 44: | ----------------------------n- | 0----------------------00u11000- | 0 | 0.00 | 0.00 | 30.00 |
| 45: | ------------------------------ | 1--1-------------------1100101-u- | 0 | -1.00 | -1.00 | 30.00 |
| 46: | ------------------------------ | x--11------------------1010000101 | 0 | -1.00 | -1.00 | 29.00 |
| 47: | ------------------------------ | n-1-0-0----------------0010101-1 | 0 | -1.00 | -1.00 | 28.00 |
| 48: | ------------------------------ | x-0--------------------110011-0-1 | 0 | 0.00 | 0.00 | 27.00 |
| 49: | ------------------------------ | ---0-------------------01011001n- | 0 | -1.00 | -0.42 | 27.00 |
| 50: | ----------------------------n- | ----------------------001u110-1- | 0 | 0.00 | 0.00 | 26.00 |
| 51: | ------------------------------ | -1---------------------10101-1-0- | 0 | -2.00 | -1.42 | 26.00 |
| 52: | ----------------------------n- | x01--------------------100u0010-- | 0 | -1.00 | -1.00 | 24.00 |
| 53: | ------------------------------ | n----------------------110010-1u- | 0 | -2.00 | -2.00 | 23.00 |
| 54: | ------------------------------ | 0----------------------0101-0-1-- | 0 | -1.00 | -1.00 | 21.00 |
| 55: | ------------------------------ | x0---------------------0010001--- | 0 | -1.00 | -1.00 | 20.00 |
| 56: | ------------------------------ | x--0-------------------01110-0--- | 0 | 0.00 | 0.00 | 19.00 |
| 57: | ------------------------------ | -----------------------111-1-0--0 | 0 | 0.00 | 0.00 | 19.00 |
| 58: | ------------------------------ | 0----------------------101111---0 | 0 | 0.00 | 0.00 | 19.00 |
| 59: | ------------------------------ | 0-1--------------------0010-11-1- | 0 | 0.00 | 0.00 | 19.00 |
| 60: | ------------------------------ | ----------------------10-0-1--0- | 0 | 0.00 | -0.00 | 19.00 |
| 61: | ------------------------------ | ----------------------01010----1 | 0 | 0.00 | 0.00 | 19.00 |
| 62: | ------------------------------ | -0-------------------0001-0----- | 0 | 0.00 | 0.00 | 19.00 |
| 63: | ------------------------------ | ----------------------1-0-0--n-- | 0 | -1.00 | -0.19 | 19.00 |
| 64: | ----------------------------n-- | ---------------------110u0---10- | 0 | 0.00 | 0.00 | 18.00 |
| 65: | ------------------------------ | ---------------------0001-10-1x-- | 0 | -1.00 | -0.00 | 18.00 |
| 66: | ------------------------------ | ----------------------1-0-0--n--x | 0 | -2.00 | -0.19 | 17.00 |
| 67: | ----------------------------n--- | -------------------1-11u0----n-x | 0 | -2.00 | -0.42 | 15.00 |
| 68: | ----------------------------n-- | ----------------------00-u---x--u | 0 | -1.00 | -0.00 | 13.00 |
| 69: | ------------------------------ | ----------------------0-1--n-xx- | 0 | -3.00 | -0.36 | 12.00 |
| 70: | --------------------------n---- | ---------------------1u0---0n-xx | 0 | -3.00 | -0.42 | 9.00 |
| 71: | ----------------------------n--- | ---------------------0-u0-0xu-ux | 0 | -3.00 | -0.42 | 6.00 |
| 72: | --------------------------u---- | ---------------------0-n--u-xx-u | 0 | -3.00 | -0.36 | 3.00 |
| 73: | ------------------------u------ | -------------------1n0----nxxx- | 0 | -4.00 | -0.91 | 0.00 |
| 74: | --------------------------n---- | ----------------------u---x--u-- | 0 | -1.00 | -0.00 | 0.00 |
| 75: | ------------------------------ | | | | | |

**Table 3. Characteristics used for the second block of the 75-step collision**

| $i$ | $\nabla A_i$ | $\nabla W_i$ | $F_W$ | $P_u(i)$ | $P_c(i)$ | $N_s(i)$ |
|---|---|---|---|---|---|---|
| -4: | 01001110010010001100011000001n000 | | | | | |
| -3: | 10110110011011111000100010unn011 | | | | | |
| -2: | 00000110011100111011010110u10010 | | | | | |
| -1: | 01000111000011100110101011101n1000 | | | | | |
| 0: | 11100000000001001100101111110101 | uuuu000001-----0------110nu1u1u1 | 8 | 0.00 | 0.00 | 0.00 |
| 1: | 000n1000101----0-1--0-10010n0nn0 | n00011----1----------0100u101010 | 4 | -0.42 | -0.42 | 3.69 |
| 2: | n1101110101101111-1---1n1n0010u00 | u10u1111111----------1-00unu1001 | 6 | -0.96 | -0.83 | 7.28 |
| 3: | u0un110001010--1-n-n-0-0uun1n1n1 | u0nu100----------------0u0n00uu | 2 | 0.00 | 0.00 | 12.31 |
| 4: | 101n0n0110nu1uu1-01u-uun011101nn | 11un111----------------1101u101 | 2 | 0.00 | 0.00 | 14.31 |
| 5: | uuuuuuuu1n0101u10uu1-1-n0u00u10n | 111111100001------------1u0001nu | 3 | -2.00 | -2.00 | 16.31 |
| 6: | 10001000u1101n00-n-uu0-1110nn01n | un1n111111--------------nnn00u0 | 2 | -1.00 | -1.00 | 17.31 |
| 7: | n110101unn1000u01--10110011011un | u1un1111011--------------10n1110 | 4 | 0.00 | 0.00 | 18.31 |
| 8: | 010000100100--1uu--0000100001000 | nnuu010-------1-----0--0-uu101u1 | 3 | -0.05 | 0.00 | 22.31 |
| 9: | 0011100001000-0-n01-1u001n1n011n | uu011011----------------n1111n0 | 8 | -0.02 | 0.00 | 25.27 |
| 10: | n010000010---11-u----1uuu01nn1u1 | 000101--0--1-------------0nu0100 | 1 | -1.10 | -1.09 | 33.24 |
| 11: | 10111000011101010n0100--n00n1uu0n | unu1100000000000-------11n0111n1 | 5 | -3.64 | -2.98 | 33.15 |
| 12: | 11001011111111111-0----111001001 | 00n1100----------------1nu11100 | 9 | 0.00 | 0.00 | 34.50 |
| 13: | 1001001000100-----0--0--11111n01 | nnn1010----------------n01001u | 6 | 0.00 | 0.00 | 43.50 |
| 14: | n10010111111---------1-0-0101000 | 00n101-------1-----------u0n1100 | 9 | -7.22 | 0.00 | 49.50 |
| 15: | 111101-101111----------u-1001101 | nu1000-01----------------u10100u | 10 | -0.21 | 0.00 | 51.29 |
| 16: | n-0010-1-----------------1000n01 | nnn1111----------------0nnn01n0 | 0 | -0.07 | -0.03 | 61.07 |
| 17: | 0-1----------------------11--nu0 | u0u1011----------------101100un | 0 | -3.00 | -2.68 | 61.00 |
| 18: | n-0-0------------------------0-- | un0010----------------111u00u0 | 0 | -1.00 | -0.83 | 58.00 |
| 19: | --0--------------------------0 | 1nn000----------------00n11111u | 0 | -1.00 | -0.96 | 57.00 |
| 20: | n------------------------------ | uu0111----------------10un00u1 | 0 | 0.00 | -0.00 | 56.00 |
| 21: | ------------------------------ | 00n11----------------1101010u | 0 | -1.00 | -1.00 | 56.00 |
| 22: | ------------------------------ | nn01100----------------000011n0 | 0 | -2.00 | -2.00 | 55.00 |
| 23: | ----------------------------n- | nnn101----------------0u1100n1 | 0 | -2.00 | -2.00 | 53.00 |
| 24: | ----------------------------n- | 0un0-0----------------0u1001n0 | 0 | -1.00 | -1.00 | 51.00 |
| 25: | ------------------------------ | u00110----------------001111u1 | 0 | -1.00 | -1.00 | 50.00 |
| 26: | ------------------------------ | 00100-0----------------01010111 | 0 | 0.00 | 0.00 | 49.00 |
| 27: | ------------------------------ | 110-1----------------10101010 | 0 | 0.00 | -0.00 | 49.00 |
| 28: | ------------------------------ | u00100----------------11011000 | 0 | 0.00 | 0.00 | 49.00 |
| 29: | ------------------------------ | 1001-1----------------111100n0 | 0 | -1.00 | -1.00 | 49.00 |
| 30: | ----------------------------n- | 10-1-0----------------1u011101 | 0 | 0.00 | -0.00 | 48.00 |
| 31: | ------------------------------ | 10100----------------11100100 | 0 | -2.00 | -2.00 | 48.00 |
| 32: | ----------------------------n- | u00-0----------------101u000100 | 0 | 0.00 | -0.00 | 46.00 |
| 33: | ------------------------------ | n-1-1----------------1011011110 | 0 | -2.00 | -2.00 | 46.00 |
| 34: | ----------------------------n- | 1010----------------11u000101 | 0 | 0.00 | 0.00 | 44.00 |
| 35: | ------------------------------ | u0-00----------------10000100u | 0 | -2.00 | -2.00 | 44.00 |
| 36: | ----------------------------n | -1-1----------------10un10010 | 0 | 0.00 | 0.00 | 42.00 |
| 37: | ------------------------------ | n11----------------1001000un | 0 | -1.00 | -1.00 | 42.00 |
| 38: | ------------------------------ | 1x0----------------1011110n0 | 0 | -2.00 | -1.00 | 41.00 |
| 39: | ----------------------------n- | ux1--0----------------11u11000- | 0 | -1.00 | 0.00 | 39.00 |
| 40: | ------------------------------ | nn---0----------------1101010u1 | 0 | -1.00 | -1.00 | 38.00 |
| 41: | ------------------------------ | x10-0----------------001100111 | 0 | -1.00 | -1.00 | 37.00 |
| 42: | ------------------------------ | x1--1----------------1011101-1 | 0 | -1.00 | -1.00 | 36.00 |
| 43: | ------------------------------ | u----0----------------1111100n1 | 0 | -1.00 | -1.00 | 35.00 |
| 44: | ----------------------------n- | 0----------------000u10000- | 0 | 0.00 | -0.00 | 34.00 |
| 45: | ------------------------------ | 0--1----------------010000-u- | 0 | -1.00 | -1.00 | 34.00 |
| 46: | ------------------------------ | x--11----------------100100000 | 0 | -1.00 | -1.00 | 33.00 |
| 47: | ------------------------------ | n-0-1----------------1010101-0 | 0 | -1.00 | -1.00 | 32.00 |
| 48: | ------------------------------ | x-1----------------000101-0-0 | 0 | 0.00 | -0.00 | 31.00 |
| 49: | ------------------------------ | ---0----------------01001000n- | 0 | -1.00 | -0.42 | 31.00 |
| 50: | ----------------------------n- | -----1----------------010u101-0- | 0 | 0.00 | -0.00 | 30.00 |
| 51: | ------------------------------ | -0----------------11000-0-0- | 0 | -2.00 | -1.42 | 30.00 |
| 52: | ----------------------------n- | x11----------------101u0101-- | 0 | -1.00 | -1.00 | 28.00 |
| 53: | ------------------------------ | u----------------100111-0u- | 0 | -2.00 | -2.00 | 27.00 |
| 54: | ------------------------------ | 0----------------0111-0-1-- | 0 | -1.00 | -1.00 | 25.00 |
| 55: | ------------------------------ | x0----------------0000101--- | 0 | -1.00 | -1.00 | 24.00 |
| 56: | ------------------------------ | x--1----------------01111-1--- | 0 | 0.00 | 0.00 | 23.00 |
| 57: | ------------------------------ | ----------------101-1-0--1 | 0 | 0.00 | -0.00 | 23.00 |
| 58: | ------------------------------ | 1----------------001010---0 | 0 | 0.00 | -0.00 | 23.00 |
| 59: | ------------------------------ | 0-0----------------1100-10-0- | 0 | 0.00 | 0.00 | 23.00 |
| 60: | ------------------------------ | ----------------10-1-0--0- | 0 | 0.00 | 0.00 | 23.00 |
| 61: | ------------------------------ | ----------------10000----0 | 0 | 0.00 | -0.00 | 23.00 |
| 62: | ------------------------------ | -1----------------110-0----- | 0 | 0.00 | 0.00 | 23.00 |
| 63: | ------------------------------ | ----------------0-0-1--n-- | 0 | -1.00 | -0.19 | 23.00 |
| 64: | ----------------------------n-- | ----------------1-101u0---01- | 0 | 0.00 | 0.00 | 22.00 |
| 65: | ------------------------------ | ----------------111-01-0x-- | 0 | -1.00 | 0.00 | 22.00 |
| 66: | ------------------------------ | ----------------1-0-1--n--x | 0 | -2.00 | -0.19 | 21.00 |
| 67: | --------------------------n--- | ----------------111u1----n-x | 0 | -2.00 | -0.42 | 19.00 |
| 68: | --------------------------n-- | ----------------110-u---x--u | 0 | -1.00 | -0.00 | 17.00 |
| 69: | ------------------------------ | ----------------0-0--u-xx- | 0 | -3.00 | -0.36 | 16.00 |
| 70: | -------------------------u---- | ----------------0n1---0u-xx | 0 | -3.00 | -0.42 | 13.00 |
| 71: | -------------------------u--- | ----------------1-n0-1xn-ux | 0 | -3.00 | -0.42 | 10.00 |
| 72: | --------------------------n--- | ----------------0-u--n-xx-u | 0 | -3.00 | -0.36 | 7.00 |
| 73: | ------------------------n----- | ----------------u1----uxxx- | 0 | -4.00 | -0.91 | 4.00 |
| 74: | -------------------------u---- | ----------------n---x--n-- | 0 | -1.00 | -0.00 | 1.00 |
| 75: | ------------------------------ | | | | | |