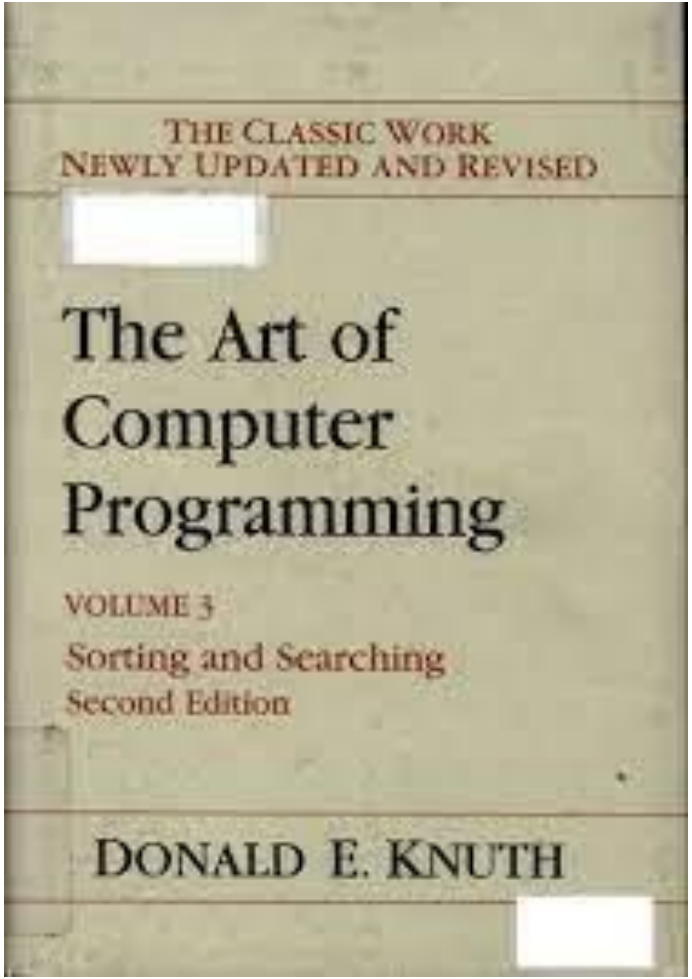
DB101 – Sorting

Goetz Graefe – Madison, Wis.

25% of all CPU cycles…

- Index creation
- Index maintenance
- Index lookup (join)
- Duplicate removal
- Grouping, rollup, cube
- Joining, merging, matching
- Intersection, difference
- Compression…



Von Neumann's First Computer Program

DONALD E. KNUTH

Stanford University, Stanford, California*

ACM Computing Surveys 1970

An analysis of the two earliest sets of instruction codes planned for stored program computers, and the earliest extant program for such a computer, gives insight into the thoughts of John von Neumann, the man who designed the instruction sets and wrote the program, and shows how several important aspects of computing have evolved. The paper is based on previously unpublished documents from the files of Herman H. Goldstine.

Key words and phrases: electronic computers, computer history, stored program computers, machine organization and architecture, sorting, latency time, ENIAC, EDVAC, order code, programming techniques

Computerworld 1968

First Patent Is Issued for Software, Full Implications Are Not Yet Known

WASHINGTON, D.C. – The first patent for a software computer program has been issued by the Patent Office to Martin Goetz, Applied Data Research vice president. This represents the first landmark after the withdrawal earlier this year of the proposed "Section 106" designed to outlaw patenting of computer programs.

The objections to the patenting of programs have been on two basic levels. Theoretically the main objective was that "programs were just mathematics and

Details of first patent appear on page 2.

you cannot patent math." On a more practical level, it has been observed that there was no way of searching for prior art, as the computer community has not yet developed any effective way of classifying programs. This practical problem of searching has a secondary, but definitely real, impact due to the fact that the Patent Office is currently short of funds and presumably would not welcome a flock of computer patent applications.

Inventing in a Garret

It appears possible that such a rush may develop as a result of the precedent set by the software patent. Before this, patents involved logic design of actual hardware which few programmers were able to do and which even fewer could arrange to have built if the Patent Office wanted to see the system in operation.

Under the new procedure, it appears that a programmer working by himself, or in the traditional garret, may be able to get a patent and so exclude the hardware manufacturers from using his technique without his agreement. He still has to show novelty, and the traditional flash of genius, but the almost mandatory cooperation of one of the industry giants is no longer necessary.

Infringement Possibilities

In the meantime the current position is that any installation which is currently using the sorting technique proposed by Martin Goetz is legally infringing his rights – no matter where they got the program, whether from a manufacturer or from one of their own programmers.

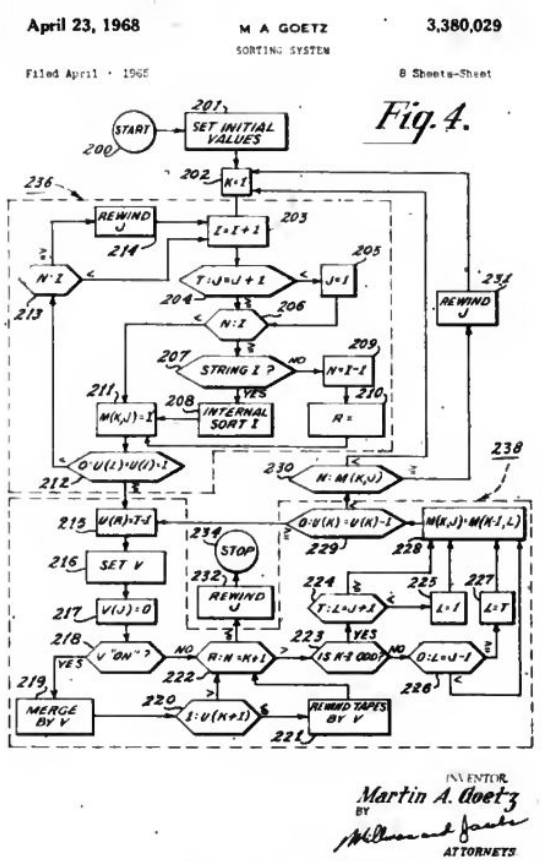
The infringement appears to start when the program is loaded (because that effectively is a manufacturing process) and continues when the program is used. Whether the program is obtained free, or is paid for

7090 or User-Designed Codes Work on System

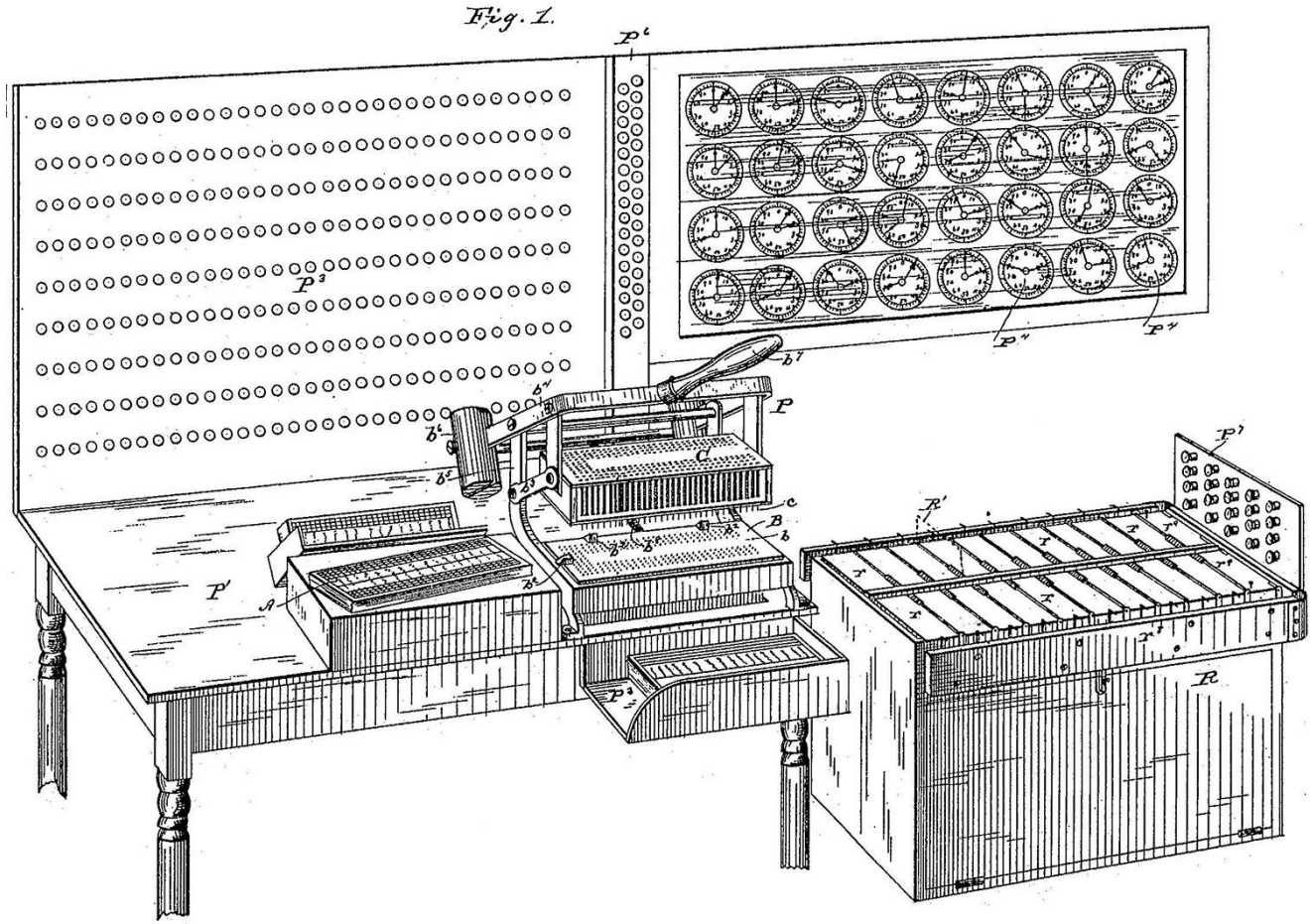
LOS ANGELES, Calif. – A new computer system was introduced here which, far from forbidding the user to change its microprogramming, encourages him to do so. The IC-6000E introduced by Standard Computer Corp. is particularly for universities and other educational institutions and comes with a number of different machine codes. These machine codes, which currently include those of the IBM 7090, 7044, and 1130 systems and a specialized Fortran system, are supported through a microprogramming system called Miniflow. The same Miniflow system allows other machine codes to be

Operating details of the IC-6000E appear on page 7.

sciences, linguistics, electrical engineering, etc., can create new computer systems at will and study their applicability to the particular type of problem they are considering. Changing the machine code from one version to another takes less than a minute and can be done without any engineering support. It is expected that particular researchers will create machine code operations that currently require macros and that as a result, they will get a higher efficiency for their



Flow chart from first software patent.



Hollerith patent 1888

1880: 50M

1887: counted

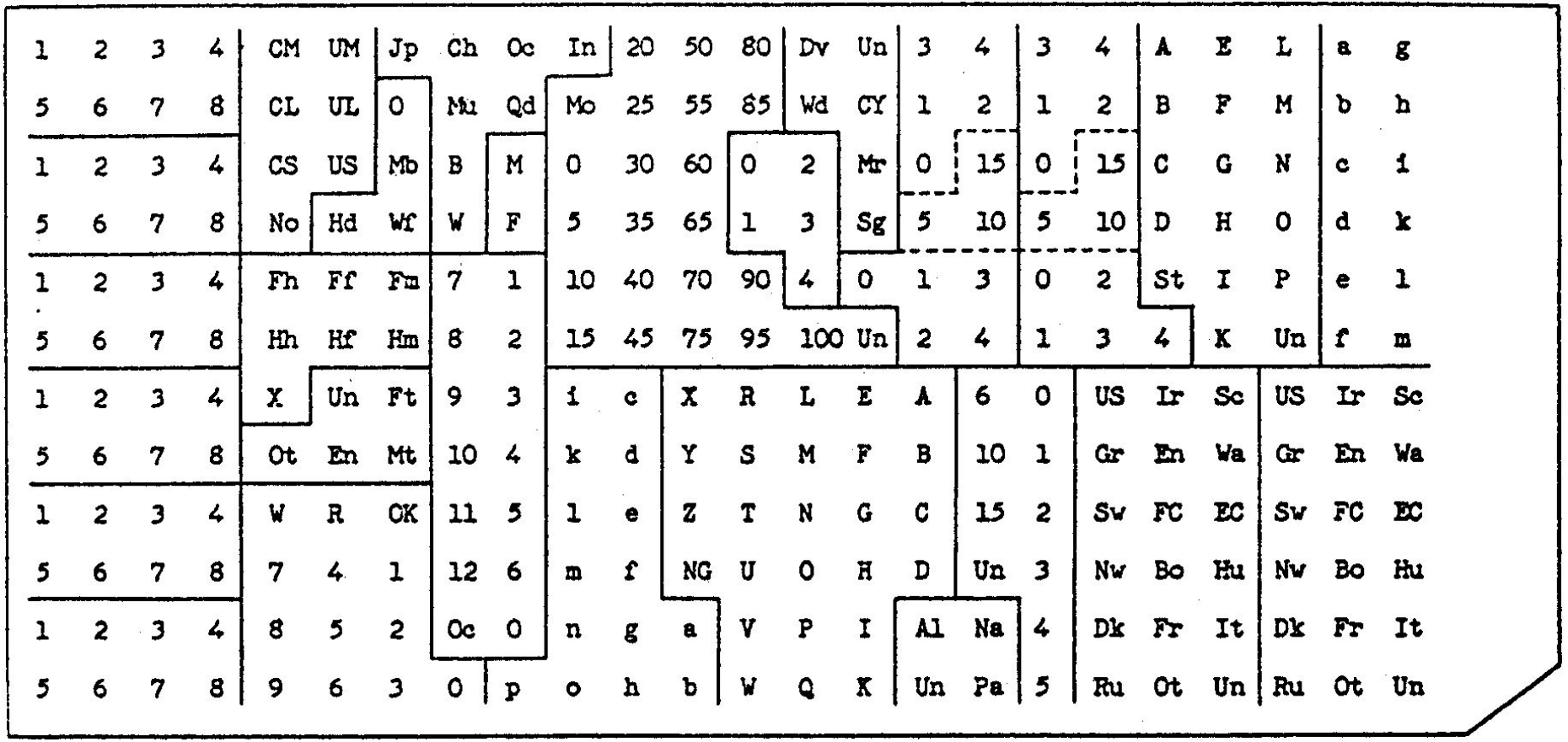
1888: prize

1890: 63M

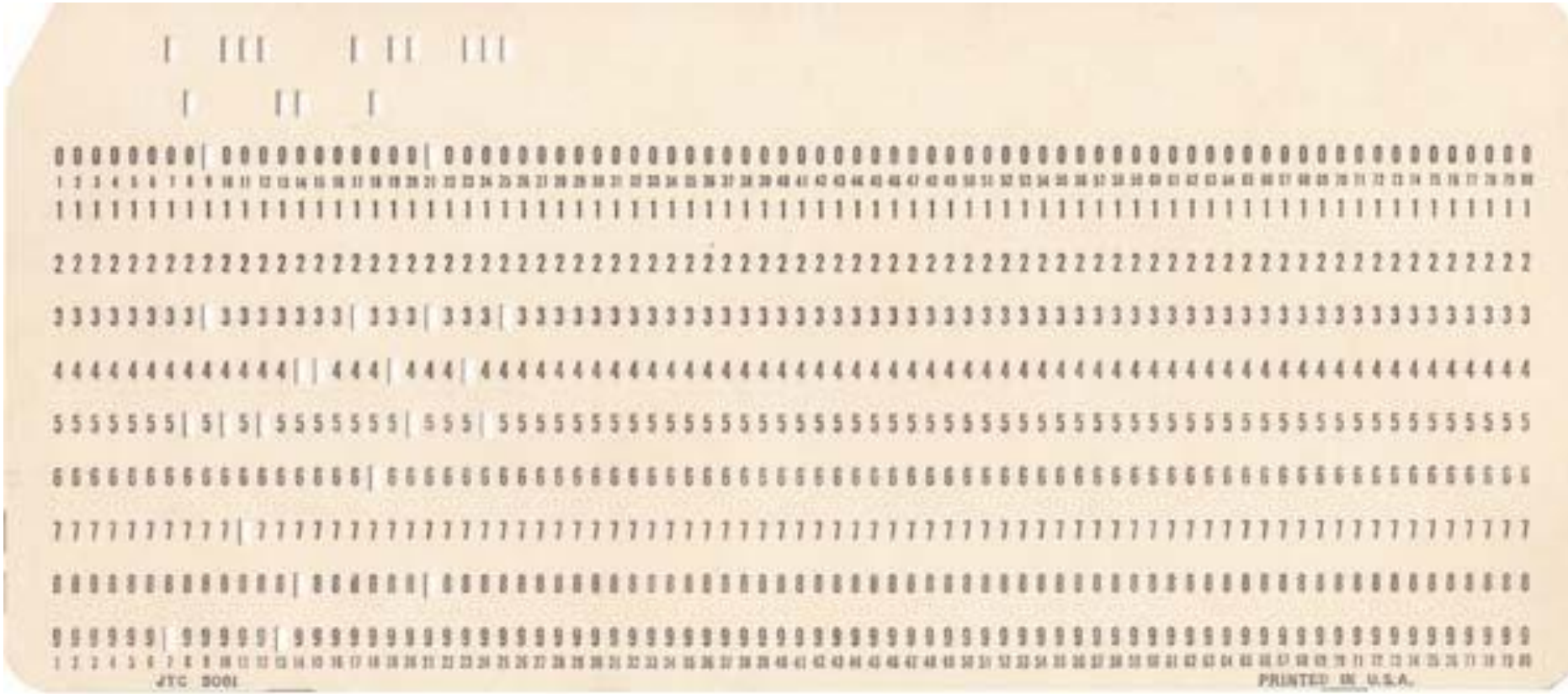
?: counted

2023: 333M

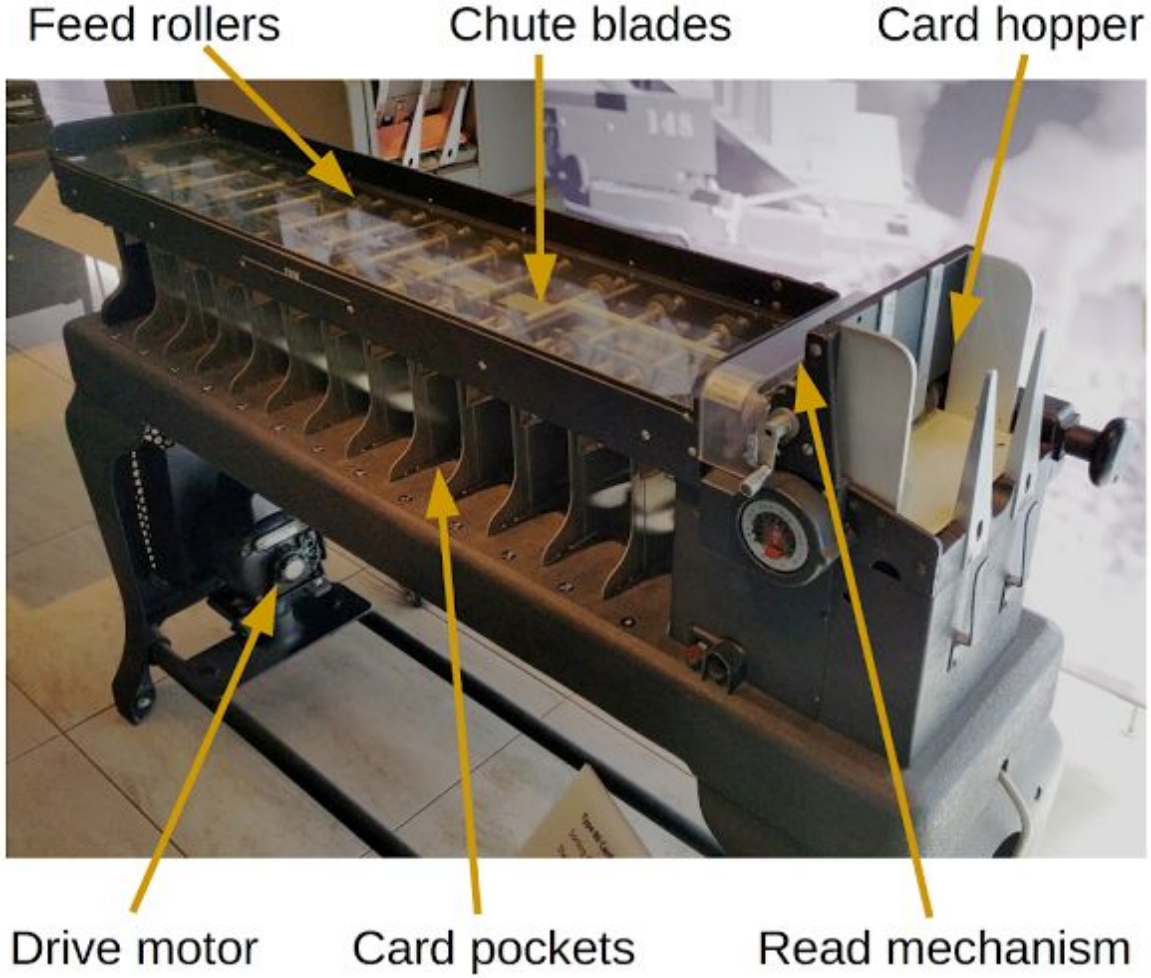
US 1890 Census Hollerith counters + sorter


1 2 3 4 CM UM Jp Ch Oc In 20 50 80 Dv Un 3 4 3 4 A E L a g
5 6 7 8 CL UL 0 Mu Qd Mo 25 55 85 Wd CY 1 2 1 2 B F M b h
1 2 3 4 CS US Mb B M 0 30 60 0 2 Mr 0 15 0 15 C G N c i
5 6 7 8 No Hd Wf W F 5 35 65 1 3 Sg 5 10 5 10 D H O d k
1 2 3 4 Fh Ff Fm 7 1 10 40 70 90 4 0 1 3 0 2 St I P e l
5 6 7 8 Hh Hf Hm 8 2 15 45 75 95 100 Un 2 4 1 3 4 K Un f m
1 2 3 4 X Un Ft 9 3 i c X R L E A 6 0 US Ir Sc US Ir Sc
5 6 7 8 Ot En Mt 10 4 k d Y S M F B 10 1 Gr En Wa Gr En Wa
1 2 3 4 W R CK 11 5 l e Z T N G C 15 2 Sw FC EC Sw FC EC
5 6 7 8 7 4 1 12 6 m f NG U O H D Un 3 Nw Bo Hu Nw Bo Hu
1 2 3 4 8 5 2 Oc 0 n g a V P I Al Na 4 Dk Fr It Dk Fr It
5 6 7 8 9 6 3 0 p o h b W Q K Un Pa 5 Ru Ot Un Ru Ot Un

IBM punch card 1928 … 1978 …?

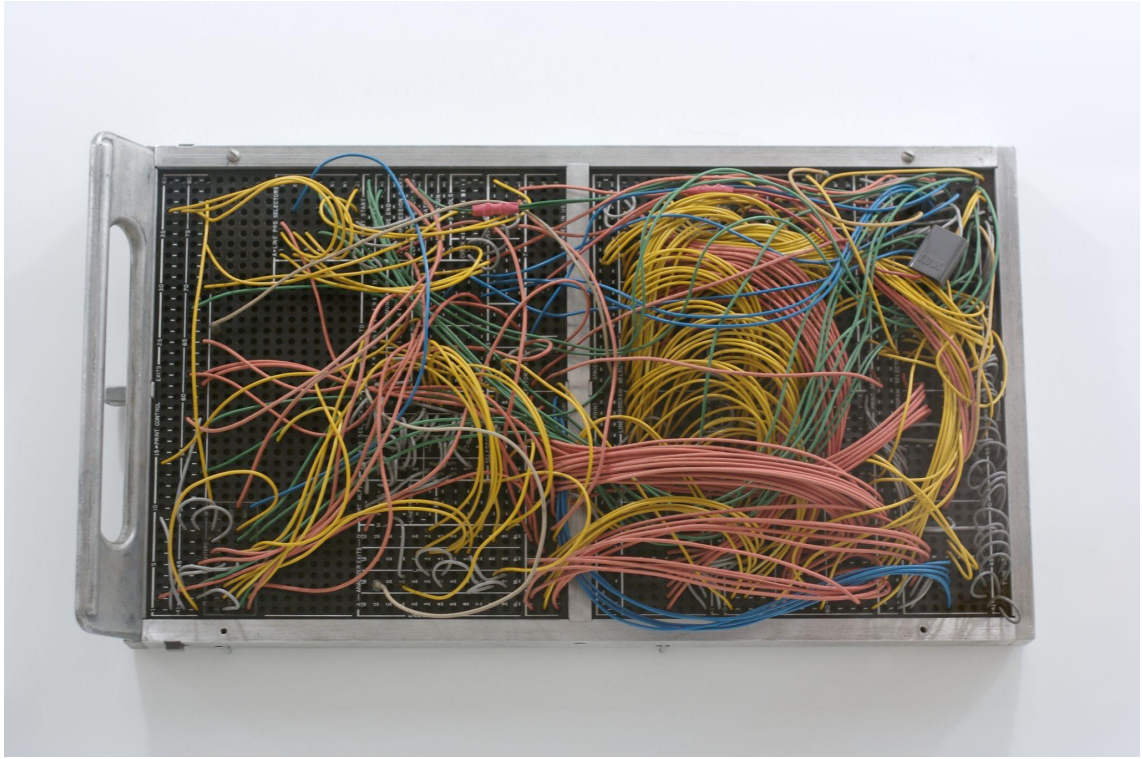


IBM card sorter 1955



IBM 1955 card sorter – wiring

IBM 1956: RAMAC disk drive



Sorting – milestones by decade

1940s: von Neumann's first program [CSUR 1970]

1950s: Friend, Isaac & Singleton [JACM 3(3) 1956]

1960s: ACM sorting workshop 1962 [CACM 6(5) 1963], quicksort, Cobol 'sort', 1st software patent, "The art of…"

1970s: b-trees, relational data model + SQL queries, interesting orderings, offset-value coding

1980s: IBM's CFC & UPT instructions, five-minute rule

1990s: AlphaSort, "A measure of…", "group by… rollup"

Sorting on Electronic Computer Systems*

EDWARD HARRY FRIEND

New York Life Insurance Co., New York, N. Y.

JACM 3(3) 1956

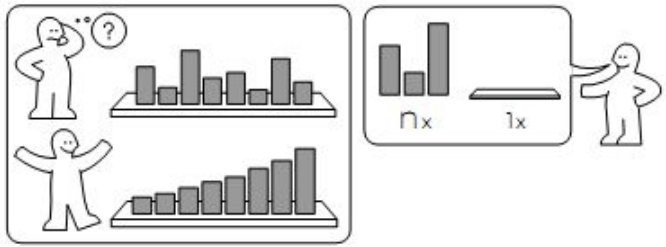
INTRODUCTION

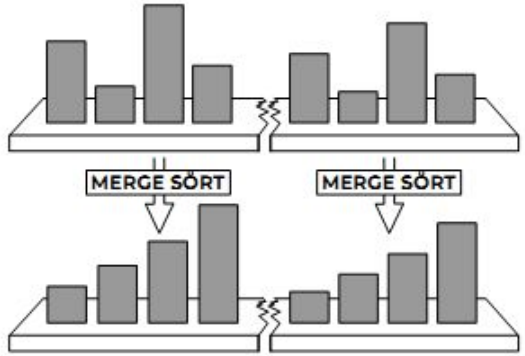
The efficient utilization of an electronic computer system for the sorting of large amounts of data is an important step toward helping electronic equipment reduce the man hours required to process scientific investigations as well as business transactions.

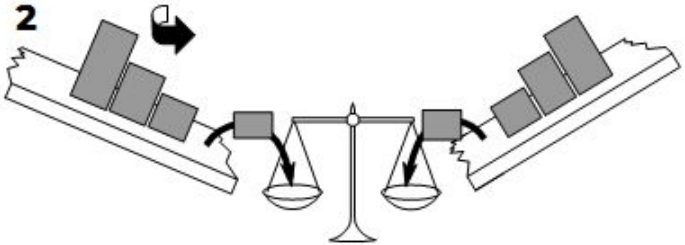
Sorting is one of the basic operations which is indispensable to most business and scientific data processing procedures. Firstly, it makes possible the collation, inside a limited capacity high speed memory, of two or more input (usually magnetic tape) files with common control fields, thereby permitting the opera-

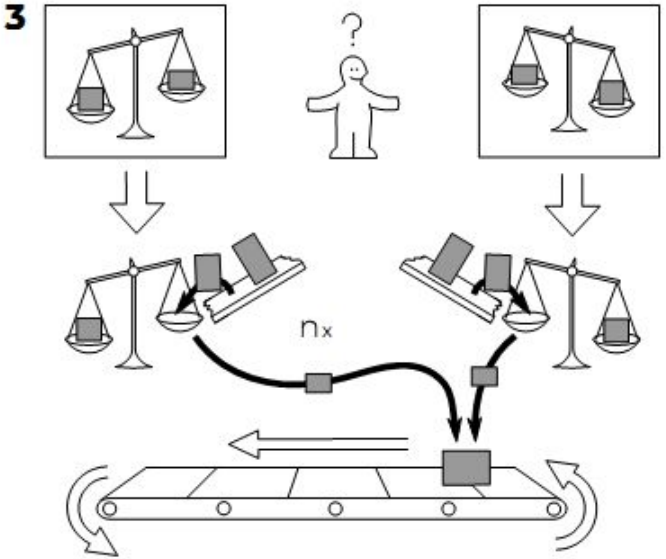
MERGE SÖRT

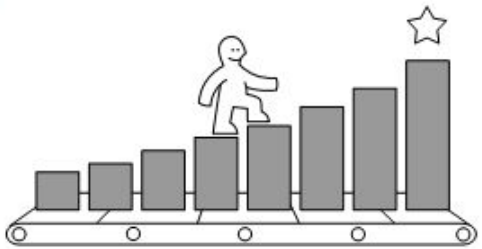
idea-instructions.com/merge-sort/
v1.2, CC by-nc-sa 4.0 IDEA











A bit of theory on sorting

- Provable lower bound ⇔ find the permutation
- N distinct key values ⇒ N! permutations
- Ideally, each comparison disproves half
- $\log_2(N!) \cong N \times \log_2(N/e)$ [Stirling]

⇒ cannot beat O(N log N) complexity (in general case)

⇒ cannot beat $\log_2(N!)$ by clever algorithms

External merge sort – basics

- In-memory run generation
  - Quicksort: run size = memory size M
  - Replacement selection: run size M or ~2M
  - Merge sort of cache-size runs
- Merging runs on external (temporary) storage
  - Fan-in F ≅ M/P − 3, with
    page size P ≅ latency × bandwidth
  - Merge depth $L \cong \log_F (I/M)$
  - Graceful degradation into & beyond one merge

I/O volume in an external merge sort

1. Run generation: none (charge to merging)
2. Merging: L levels, each writes & reads data size I
3. Total I/O volume: $L \times I \times 2 = 2I \times \log_F (I/M)$

Key comparisons in an external merge sort

1. Run generation: I/M runs, each $M \times \log_2 (M/e)$
2. Merging: $L \times I \times \log_2 (F)$ with $L = \log_F (I/M)$
3. Note: $\log_F (I/M) \times \log_2 (F) = \log_2 (I/M)$
4. Entire sort: $I/M \times M \times \log_2(M/e) + I \times \log_2 (I/M)$
5. $= I \times (\log_2(M/e) + \log_2 (I/M))$
6. $= I \times (\log_2(M/e \times I/M)) = I \times \log_2 (I/e)$

Internal and Tape Sorting Using the Replacement-Selection Technique*

CACM 6(5) 1963

Martin A. Goetz
Applied Data Research, Inc., Princeton, N. J.

A general technique for sequencing unsorted records is presented. The technique is shown to be applicable for the first stage of a generalized sort program (the formation of initial strings) as well as for sorting records within a memory storage (an internal sort). It is shown that given N records in memory storage, records are sequenced using $1+\log_2 N$ tests per record, that initial string lengths will average $2N$ for random input records, and that reading, writing and processing can be accomplished simultaneously if the computer permits such overlap.

pared and so on until the record with the smallest key is found. The result of all comparisons are stored in the result words. Records are not moved during the comparison procedure. After a record is selected as the smallest and written out or placed in an output area, a new input record replaces the selected record in its identical physical location in the mass storage area.

A major advantage of this method is that reading, writing and computing operations are performed simultaneously if the computer permits overlap of these functions.

Hardware Assisted Sorting in IBM's DB2 DBMS

Balakrishna R Iyer

IBM Silicon Valley Lab
555 Bailey Avenue
San Jose, CA 95141
USA
balaiyer@us.ibm.com

COMAD 2005

Abstract

For over 20 years, researchers have experimented with hardware innovations to make RDBMS processing more efficient. Efforts to build database machines with special hardware may be contrasted with the introduction of assists for accelerating database processing in general purpose processors. Four CPU intensive DBMS functions are identified in this paper, and all four have been successfully offloaded to assists on processors by IBM's DB2 DBMS. One such

At IBM's research and development labs, we have successfully pursued an alternate approach to leverage hardware technology for achieving RDBMS processing efficiency. We accepted the maxim that processing technology will continue to offer ever increasing efficiencies that would be hard to compete with. We decided to leverage processor advances by identifying several compute intensive functions in RDBMS processing, designed and implemented hardware assists within IBM's general purpose z/Architecture processor, re-architected IBM's DB2 DBMS to offload CPU

Hardware sorting in IBM DB2

- UPT “update tree”: leaf-to-root pass in a tournament tree or tree-of-losers priority queue
- CFC “compare and form codeword”: (byte string) comparison + the loser’s new offset-value code

“*Together, the UPT and CFC instructions do the bulk of sorting in IBM’s commercial DBMS DB2 running on z/Architecture processors.*”

B. Iyer: *Hardware-assisted sorting*… [COMAD 2005]

Quicksort

ComputerJ 1962

By C. A. R. Hoare

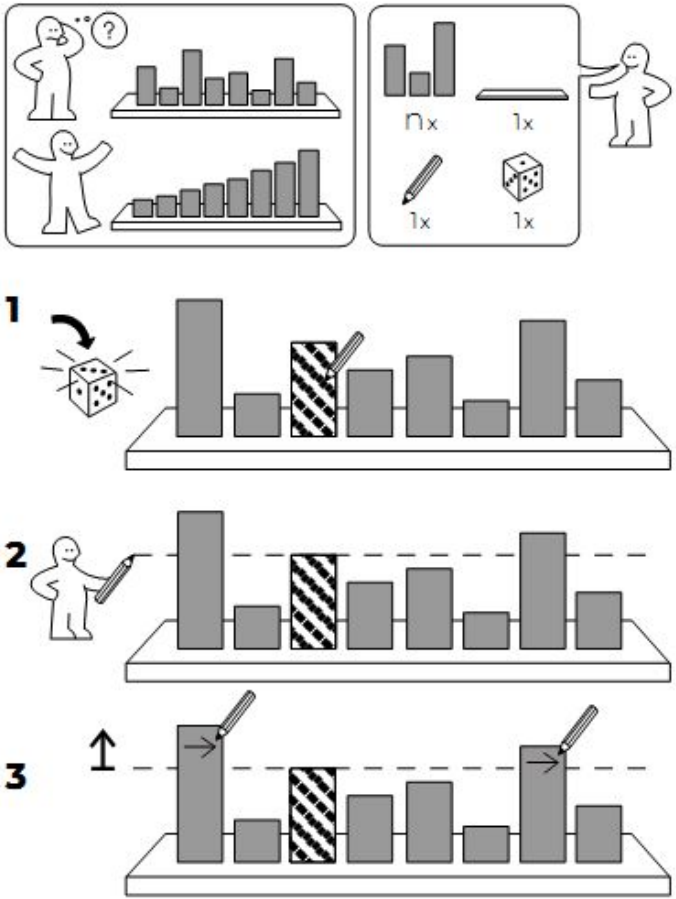
A description is given of a new method of sorting in the random-access store of a computer. The method compares very favourably with other known methods in speed, in economy of storage, and in ease of programming. Certain refinements of the method, which may be useful in the optimization of inner loops, are described in the second part of the paper.

NUMBER OF ITEMS	MERGE SORT	QUICKSORT
500	2 min 8 sec	1 min 21 sec
1,000	4 min 48 sec	3 min 8 sec
1,500	8 min 15 sec*	5 min 6 sec
2,000	11 min 0 sec*	6 min 47 sec

* These figures were computed by formula, since they cannot be achieved on the 405 owing to limited store size.

KVICK SÖRT

idea-instructions.com/quick-sort/
v1.2, CC by-nc-sa 4.0 IDEA



Quicksort

ComputerJ 1962

By C. A. R. Hoare

An awkward situation is liable to arise if the value of the bound is the greatest or the least of all the key values in the segment, or if all the key values are equal. The

The average number of comparisons required by Quicksort is greater than the theoretical minimum by a factor of $2 \log_e 2 \sim 1 \cdot 4$. This factor could be reduced by the expedient of choosing as the bound for each partition the median of a small random sample of the

ALGORITHM 347
AN EFFICIENT ALGORITHM FOR SORTING WITH MINIMAL STORAGE [M1]

RICHARD C. SINGLETON* (Recd. 17 Sept. 1968)

CACM 12(3) 1969

moved from the array while splitting. A more important difference is that the median of the values of $A[i]$, $A[(i+j)\div 2]$, and $A[j]$ is used for t, yielding a better estimate of the median value for the segment than the single element used in the earlier

Sorting by Address Calculation*

JACM 3(3) 1956

E. J. Isaac and R. C. Singleton

Stanford Research Institute, Menlo Park, California

General Description of Method

Sorting in a random access memory is essentially a process of associating the address of the location in which each item is to be placed with the identifying key of the item. The fewer times the items have to be moved from one location to another, the more efficient the sorting process.

The association of memory address to key in the sorted results can be looked upon as a functional relation; for convenience this will be called the "address function." Figure 1 shows a typical plot of key against memory address. The function is discontinuous and is similar to a cumulative histogram.

Sorting by Address Calculation*

JACM 3(3) 1956

If the sorting task exceeds the internal memory capacity of a magnetic tape computer, an internal memory sort can be used as a means of building up sequenced blocks prior to tape sorting by merging. However, fewer tape passes will be required, using a given number of tape units, if the sequence of operations is reversed and the items rough-sorted on tape into groups by key range prior to a final sort in the internal memory of the items within groups. In this method, a sorting function is used to calculate from the key of an item the correct tape on which to place the item on each pass; the result is a series of approximately equal-sized blocks of items grouped by key range. With 4 tape units, the items are successively divided into 3, 9, 27, etc. blocks, continuing until blocks of convenient size for sorting in the internal memory are obtained. These blocks are then selected in proper sequence, sorted in the internal memory, and chained together on a final sorted output tape.