Deterministic Order Selection

Order Selection

Problem: Given an unsorted list of n numbers, determine the kth number in the list if the list *were* sorted.

Example: $7, 11, 5, 8, 32, 9, 26, 2$ with $k = 5$

Answer: 9

Monkey: $\Theta(n \log n)$

Lower bound: $\Omega(n)$

Applications: determining statistical quartiles, making quicksort faster by choosing the pivot to be the median of the list

Randomized Selection

Assume that we are allowed to make random choices.

Example: $7, 11, 5, 8, 32, 9, 26, 2$ with $k = 5$

Choose a random element, partition around that element, eliminate the irrelevant part of the list, and repeat.

7, 5, 8, 9, 2, ~~11, 32, 26~~ with $k = 5$

~~5, 2, 7~~, 8, 9 with $k = 5$

8, 9 with $k = 2$

Done!

With randomness, we roughly expect

$$T(n) = T(\tfrac{n}{2}) + \Theta(n) \Rightarrow T(n) = \Theta(n)$$

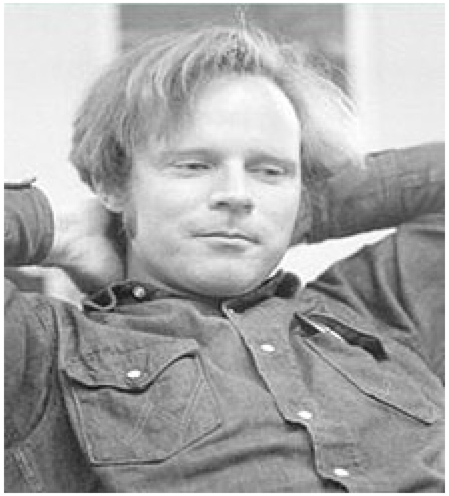
Can we do this without choosing random numbers?

Manuel Blum

A cryptography expert, attended MIT, bachelor's degree (1959), master's degree (1961), Ph.D. in Mathematics (1964) under Marvin Minsky. Professor of CS at UC Berkeley (until 1999), then to Carnegie Mellon (resigned in 2018 in protest of sexism with wife Lenore Blum and son Avrim Blum, both also professors of CS). Among his PhD advisees are Leonard Adleman, Shafi Goldwasser, Russell Impagliazzo, Silvio Micali, Michael Sipser, Umesh Vazirani and Vijay Vazirani. He won the Turing Award in 1995.

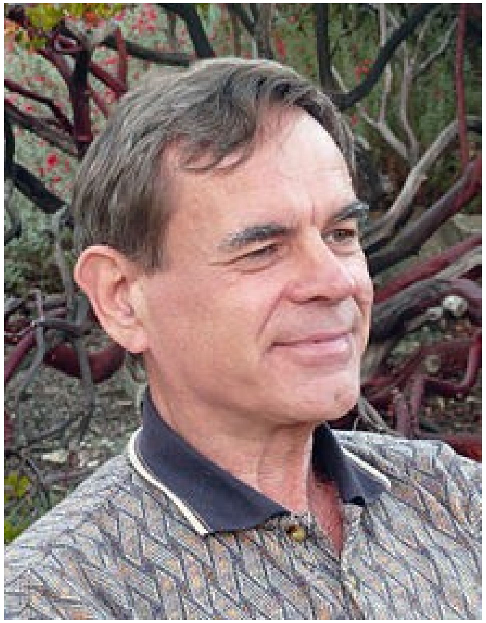
Bob Floyd

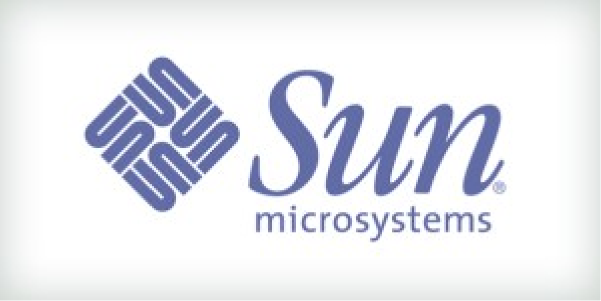
Floyd/Warshall algorithm (independently of Stephen Warshall), which efficiently finds all shortest paths in a graph. Finished school at age 14. At the University of Chicago, Bachelor's degree in 1953 (when still only 17), second Bachelor's degree in physics in 1958. Appointed an associate professor at Carnegie Mellon University at 27, full professor at Stanford University six years later. Floyd worked closely with Donald Knuth, in particular as the major reviewer for Knuth's seminal book The Art of Computer Programming, and is the person most cited in that work. All this *without a Ph.D.* He received the Turing Award in 1978.



Vaughan Pratt

Completed a Ph.D. thesis at Stanford University in 20 months under the supervision of advisor Donald Knuth. KMP pattern-matching. Assistant Professor at MIT (1972 to 1976), Associate Professor (1976 to 1982), on sabbatical from MIT to Stanford (1980 to 1981), was appointed a full professor at Stanford in 1981. Contributed to the founding and early operation of Sun Microsystems, consultant for its first year, then, taking a leave of absence from Stanford, becoming Director of Research, and finally returning to Stanford in 1985. He also designed the Sun logo, which features four interleaved copies of the word ”sun”; it is an ambigram.



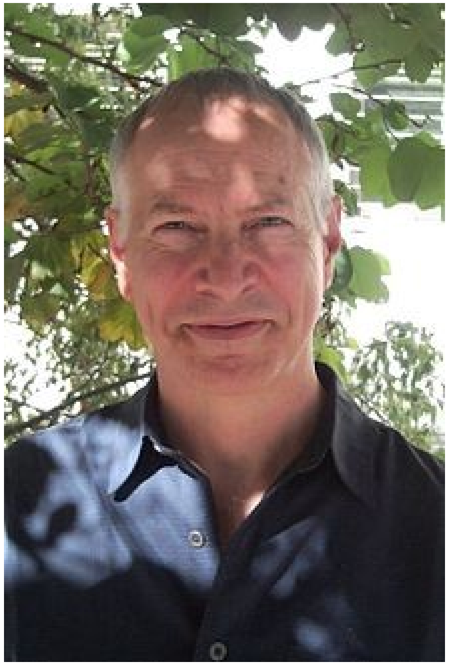


Ron Rivest

Cryptographer, one of the inventors of the RSA algorithm (along with Adi Shamir and Len Adleman). Inventor of the symmetric key encryption algorithms RC2, RC4, RC5, and co-inventor of RC6. The "RC" stands for "Rivest Cipher", or alternatively, "Ron's Code." (RC3 was broken at RSA Security during development; similarly, RC1 was never published.) He also authored the MD2, MD4, MD5 and MD6 cryptographic hash functions. In 2006, he published his invention of the ThreeBallot voting system, an innovative voting system that incorporates the ability for the voter to discern that their vote was counted while still protecting their voter privacy. Immediately placed ThreeBallot in the public domain. Shared the Turing Award with Adi Shamir and Len Adleman in 2002.

Bob Tarjan

Bachelor's degree in mathematics from the California Institute of Technology in 1969. At Stanford University, he received his Master's degree in computer science in 1971 and a Ph.D. in computer science (with a minor in mathematics) in 1972. At Stanford, he was supervised by Robert Floyd and Donald Knuth. He has published more than 228 refereed journal articles and book chapters. Tarjan is known for his pioneering work on graph theory algorithms and data structures. Discovered a large fraction of the material in this class and CECS 428. Tarjan received the Turing Award jointly with John Hopcroft in 1986.



Determinstic Selection

SELECT(L, k)

If $(|L| \leq 100)$ sort L and return the element in the kth position.

Partition L into groups of five elements each (roughly $n/5$ subsets total) and identify the median m_i of each group.

$M \equiv SELECT(\{m_i\}, n/10)$

Partition L into $L_1 < M, L_2 = M, L_3 > M$ and determine the position of M $(p(M))$. If $k = p(M)$ return M else if $k < p(M)$ return $SELECT(L_1, k)$, else if $k > p(M)$ return $SELECT(L_3, k - |L_1| - |L_2|)$

Example: Splitting the numbers into groups of 5.

$$\begin{pmatrix}
10 & 15 & 9 & 48 & 16 & 7 & 96 & 2 & 41 & 34 & 44 & 90 & 75 & 102 & 103 & 53 & 14 & 12 & 85 & 93 & 73 \\
64 & 23 & 33 & 19 & 63 & 74 & 82 & 91 & 95 & 61 & 38 & 65 & 57 & 6 & 59 & 51 & 78 & 46 & 3 & 13 & 40 \\
45 & 88 & 55 & 8 & 86 & 50 & 77 & 36 & 89 & 4 & 29 & 79 & 76 & 69 & 62 & 104 & 31 & 37 & 87 & 54 & 80 \\
92 & 39 & 52 & 105 & 18 & 67 & 97 & 68 & 70 & 24 & 30 & 26 & 84 & 5 & 22 & 99 & 60 & 66 & 20 & 28 & 81 \\
72 & 1 & 17 & 25 & 101 & 100 & 98 & 42 & 43 & 27 & 35 & 47 & 56 & 21 & 11 & 32 & 49 & 58 & 71 & 83 & 94
\end{pmatrix}$$

Determinstic Selection

SELECT(L, k)

If $(|L| \leq 100)$ sort L and return the element in the kth position.

Partition L into groups of five elements each (roughly $n/5$ subsets total) and identify the median m_i of each group.

$M \equiv SELECT(\{m_i\}, n/10)$

Partition L into $L_1 < M, L_2 = M, L_3 > M$ and determine the position of M $(p(M))$. If $k = p(M)$ return M else if $k < p(M)$ return $SELECT(L_1, k)$, else if $k > p(M)$ return $SELECT(L_3, k - |L_1| - |L_2|)$

Example: Find the medians: sort *only within the group.*

$$\begin{pmatrix}
10 & 1 & 9 & 8 & 16 & 7 & 77 & 2 & 41 & 4 & 29 & 26 & 56 & 5 & 11 & 32 & 14 & 12 & 3 & 13 & 40 \\
45 & 15 & 17 & 19 & 18 & 50 & 82 & 36 & 43 & 24 & 30 & 47 & 57 & 6 & 22 & 51 & 31 & 37 & 20 & 28 & 73 \\
64 & 23 & 33 & 25 & 63 & 67 & 96 & 42 & 70 & 27 & 35 & 65 & 75 & 21 & 59 & 53 & 49 & 46 & 71 & 54 & 80 \\
72 & 39 & 52 & 48 & 86 & 74 & 97 & 68 & 89 & 34 & 38 & 79 & 76 & 69 & 62 & 99 & 60 & 58 & 85 & 83 & 81 \\
92 & 88 & 55 & 105 & 101 & 100 & 98 & 91 & 95 & 61 & 44 & 90 & 84 & 102 & 103 & 104 & 78 & 66 & 87 & 93 & 94
\end{pmatrix}$$

Determinstic Selection

SELECT(L, k)

If $(|L| \leq 100)$ sort L and return the element in the kth position.

Partition L into groups of five elements each (roughly $n/5$ subsets total) and identify the median m_i of each group.

$M \equiv SELECT(\{m_i\}, n/10)$

Partition L into $L_1 < M, L_2 = M, L_3 > M$ and determine the position of M $(p(M))$. If $k = p(M)$ return M else if $k < p(M)$ return $SELECT(L_1, k)$, else if $k > p(M)$ return $SELECT(L_3, k - |L_1| - |L_2|)$

The median of the medians is 54.
Example: Partition groups around median of medians.

$$\begin{pmatrix}
12 & 14 & 32 & 5 & 29 & 4 & 2 & 8 & 9 & 1 & 13 & 10 & 16 & 7 & 77 & 41 & 26 & 56 & 11 & 3 & 40 \\
37 & 31 & 51 & 6 & 30 & 24 & 36 & 19 & 17 & 15 & 28 & 45 & 18 & 50 & 82 & 43 & 47 & 57 & 22 & 20 & 73 \\
46 & 49 & 53 & 21 & 35 & 27 & 42 & 25 & 33 & 23 & 54 & 64 & 63 & 67 & 96 & 70 & 65 & 75 & 59 & 71 & 80 \\
58 & 60 & 99 & 69 & 38 & 34 & 68 & 48 & 52 & 39 & 83 & 72 & 86 & 74 & 97 & 89 & 79 & 76 & 62 & 85 & 81 \\
66 & 78 & 104 & 102 & 44 & 61 & 91 & 105 & 55 & 88 & 93 & 92 & 101 & 100 & 98 & 95 & 90 & 84 & 103 & 87 & 94
\end{pmatrix}$$

Determinstic Selection

SELECT(L, k)

If $(|L| \leq 100)$ sort L and return the element in the kth position.

Partition L into groups of five elements each (roughly $n/5$ subsets total) and identify the median m_i of each group.

$M \equiv SELECT(\{m_i\}, n/10)$

Partition L into $L_1 < M, L_2 = M, L_3 > M$ and determine the position of M $(p(M))$. If $k = p(M)$ return M else if $k < p(M)$ return $SELECT(L_1, k)$, else if $k > p(M)$ return $SELECT(L_3, k - |L_1| - |L_2|)$

$$\begin{pmatrix}
12 & 14 & 32 & 5 & 29 & 4 & 2 & 8 & 9 & 1 & 13 & 10 & 16 & 7 & 77 & 41 & 26 & 56 & 11 & 3 & 40 \\
37 & 31 & 51 & 6 & 30 & 24 & 36 & 19 & 17 & 15 & 28 & 45 & 18 & 50 & 82 & 43 & 47 & 57 & 22 & 20 & 73 \\
46 & 49 & 53 & 21 & 35 & 27 & 42 & 25 & 33 & 23 & 54 & 64 & 63 & 67 & 96 & 70 & 65 & 75 & 59 & 71 & 80 \\
58 & 60 & 99 & 69 & 38 & 34 & 68 & 48 & 52 & 39 & 83 & 72 & 86 & 74 & 97 & 89 & 79 & 76 & 62 & 85 & 81 \\
66 & 78 & 104 & 102 & 44 & 61 & 91 & 105 & 55 & 88 & 93 & 92 & 101 & 100 & 98 & 95 & 90 & 84 & 103 & 87 & 94
\end{pmatrix}$$

If 54 happens to be the one we want, we've found it. If the number we want is smaller than 54, throw away the larger numbers. If the number we want is larger than 54, throw away the smaller numbers.

The Cleverness

How many numbers *must* be at least as small as the median of medians?

$$\begin{pmatrix}
12 & 14 & 32 & 5 & 29 & 4 & 2 & 8 & 9 & 1 & 13 & 10 & 16 & 7 & 77 & 41 & 26 & 56 & 11 & 3 & 40 \\
37 & 31 & 51 & 6 & 30 & 24 & 36 & 19 & 17 & 15 & 28 & 45 & 18 & 50 & 82 & 43 & 47 & 57 & 22 & 20 & 73 \\
46 & 49 & 53 & 21 & 35 & 27 & 42 & 25 & 33 & 23 & 54 & 64 & 63 & 67 & 96 & 70 & 65 & 75 & 59 & 71 & 80 \\
58 & 60 & 99 & 69 & 38 & 34 & 68 & 48 & 52 & 39 & 83 & 72 & 86 & 74 & 97 & 89 & 79 & 76 & 62 & 85 & 81 \\
66 & 78 & 104 & 102 & 44 & 61 & 91 & 105 & 55 & 88 & 93 & 92 & 101 & 100 & 98 & 95 & 90 & 84 & 103 & 87 & 94
\end{pmatrix}$$

How many numbers *must* be at least as large as the median of medians?

$$\begin{pmatrix}
12 & 14 & 32 & 5 & 29 & 4 & 2 & 8 & 9 & 1 & 13 & 10 & 16 & 7 & 77 & 41 & 26 & 56 & 11 & 3 & 40 \\
37 & 31 & 51 & 6 & 30 & 24 & 36 & 19 & 17 & 15 & 28 & 45 & 18 & 50 & 82 & 43 & 47 & 57 & 22 & 20 & 73 \\
46 & 49 & 53 & 21 & 35 & 27 & 42 & 25 & 33 & 23 & 54 & 64 & 63 & 67 & 96 & 70 & 65 & 75 & 59 & 71 & 80 \\
58 & 60 & 99 & 69 & 38 & 34 & 68 & 48 & 52 & 39 & 83 & 72 & 86 & 74 & 97 & 89 & 79 & 76 & 62 & 85 & 81 \\
66 & 78 & 104 & 102 & 44 & 61 & 91 & 105 & 55 & 88 & 93 & 92 & 101 & 100 & 98 & 95 & 90 & 84 & 103 & 87 & 94
\end{pmatrix}$$

Answer: $\frac{3}{10}$ of the list

Time Analysis

SELECT(L, k)

If $(|L| \leq 100)$ sort L and return the element in the kth position.

Partition L into groups of five elements each (roughly $n/5$ subsets total) and identify the median m_i of each group.

$M \equiv SELECT(\{m_i\}, n/10)$

Partition L into $L_1 < M, L_2 = M, L_3 > M$ and determine the position of M $(p(M))$. If $k = p(M)$ return M else if $k < p(M)$ return $SELECT(L_1, k)$, else if $k > p(M)$ return $SELECT(L_3, k - |L_1| - |L_2|)$

Let $T(n) \equiv$ worst-case work needed to select from a list of n numbers

$T(n) = T(\frac{n}{5}) + T(\frac{7n}{10}) + \Theta(n)$

Time Analysis

$T(n) = T(\frac{n}{5}) + T(\frac{7n}{10}) + \Theta(n) \Rightarrow$
$\exists c > 0$ such that eventually $T(n) \leq T(\frac{n}{5}) + T(\frac{7n}{10}) + cn$

What would it mean if $\exists C > 0$ such that for large n,
$\forall k < n, T(k) \leq Ck \rightarrow T(n) \leq Cn$?

Assume that $\forall k < n, T(k) \leq Ck$

$T(n) \leq T(\frac{n}{5}) + T(\frac{7n}{10}) + cn \leq C\frac{n}{5} + C\frac{7n}{10} + cn \overset{\text{want}}{\leq} Cn \Rightarrow$
$C\frac{9n}{10} + cn \overset{\text{want}}{\leq} Cn \Rightarrow cn \leq C\frac{n}{10} \Rightarrow 10c \overset{\text{want}}{\leq} C$
So the answer is *yes* if we let $C = 10c \Rightarrow T(n) = \Theta(n)$!

Altered Time Analysis

What would the analysis look like if, instead of using groups of 5, we used groups of 3?

$$\begin{pmatrix} 19 & 21 & 3 & 16 & 2 & 6 & 23 & 22 & 5 & 1 & 12 & 33 & 15 & 10 & 17 & 8 & 18 & 11 & 4 & 24 & 9 \\ 26 & 25 & 31 & 32 & 13 & 20 & 27 & 30 & 7 & 35 & 37 & 55 & 44 & 58 & 42 & 40 & 46 & 38 & 43 & 47 & 39 \\ 60 & 34 & 36 & 50 & 29 & 41 & 28 & 45 & 14 & 53 & 52 & 57 & 59 & 62 & 63 & 48 & 61 & 49 & 54 & 51 & 56 \end{pmatrix}$$

$T(n) = T(\frac{n}{3}) + T(\frac{2n}{3}) + \Theta(n) \Rightarrow$

$\exists c > 0$ such that eventually $T(n) \leq T(\frac{n}{3}) + T(\frac{2n}{3}) + cn$

Assume that $\forall k < n, T(k) \leq Ck$

$$T(n) \leq T(\tfrac{n}{3}) + T(\tfrac{2n}{3}) + cn \leq C\tfrac{n}{3} + C\tfrac{2n}{3} + cn \overset{\text{want}}{\leq} Cn \Rightarrow$$

$$cn \overset{\text{want}}{\leq} 0$$

which is false!

Interesting facts

- If you use rows of 3 instead of rows of 5, the algorithm will run correctly *but not in linear time.*
- If you use rows of 5 or larger, the algorithm *will* run in linear time.
- Even though it only takes $n - 1$ comparisons in the worst case to find the minimum and maximum, there exists a $2n$-comparison lower-bound to find the median.