The function $ACKP$ is called the Ackermann function. (There are several similar formulations.) The extreme growth rate of $ACKP$ is illustrated in Figure 3. Note that

$$ACKP(6) = I^4P3 = I^3P4 = (I^2P)^41 = I^2P(I^2P(4))$$

Set $M = I^2P4$, a tower of twos of height 2^{16}. Then $ACKP(6)$ is a tower of twos of height a tower of twos of height ... of height one, where the statement repeats " tower of twos" M times.

Application of Theorem 1 to the $ACKP$-search gives a convergent and divergent series that are extremely close. While the ratio of their terms approaches infinity it is at most $2^6 = 64$ for the n-th term for all $n \leq ACKP(6)$.

References.

J. Bentley, A. Yao, An Almost Optimal Algorithm for Unbounded Searching, Information Processing Letters *5*, (1976), 82-87

D. Knuth, Supernatural Numbers, *in* The Mathematical Gardner (D. Klarner, ed.), Wadsworth, pp 310-325

beginning with n, until the value becomes less than or equal one. We can write

$$S(n) = \sum \lceil \lg^i n \rceil - 1$$

where the sum is over $1 \leq i \leq \lg^* n$.

Corollary. Let

$$L(n) = \sum \lceil \lg^i n \rceil \text{ and } U(n) = \sum \lfloor \lg^i n \rfloor,$$

both sums over all $i \geq 1$ with $\lg^i n > 0$. Then

$$\sum 2^{-L(n)} \text{ diverges while } \sum 2^{-U(n)} \text{ converges}$$

Proof. Apply Theorem 1 to the IP-search. For convenience of presentation we have used the equality $\lceil \lg^i n \rceil - 1 = \lfloor \lg^i n \rfloor$. This fails only when n is a power of two and those terms are too sparse to affect the convergence.

Ackermania. For any strictly increasing f with $f1 = 2, f2 = 4$ we have induced an If-search from an f-search. As If has the same properties we may now induce an $I^2 f$-search and continue. In particular, we induce from our basic P-search an $I^t P$-search for each $t \geq 1$ and, applying Theorem 1, these give us examples "closer and closer to the edge of convergence".

Lets take $I^2 P$ as an example. Set $F(n) = \sum \lceil \lg^i n \rceil - 1$, summed over $1 \leq i \leq \lg^* n$. Then we may write

$$S(n) = F(n) + F(\lg^* n) + F(\lg^* \lg^* n) + \ldots$$

where the sum continues until the argument is less than one. The function $B(n)$ is then the number of times $\lg^*$ is applied, beginning with n, until the result becomes one.

We are able to diagonalize once again. Given any such f define $ACKf$ by $ACKf(1) = 2, ACKf(2) = 4$ and

$$ACKf(t) = (I^{t-2} f)(3), t \geq 3$$

Now we induce an $ACKf$-search. Clearly if $ACKf1 < x \leq ACKf2$ we ask if $x \leq 3$ and then stop. Suppose now $t \geq 3$ and $ACKf(t-1) < x \leq ACKf(t)$. Unravelling the definitions $ACKf(t-1) = (I^{t-3} f)(3)$ and $ACKf(t) = (I^{t-2} f)(3) = (I(I^{t-3} f))(3) = (I^{t-3} f)^3 1 = (I^{t-3} f)(4)$ since, by induction, $I^{t-3} f1 = 2$ and $I^{t-3} f2 = 4$. In this case the $ACKf$-search is given by the $I^{t-3} f$-search.

Suppose that $A(n) \le S(n)+c$ for all but finitely many n, say for all $n \ge n_0$. Then

$$\sum 2^{-A(n)} \ge \sum_{n \ge n_0} 2^{-S(n)-c} = \infty$$

But any method that determines n in $A(n)$ steps can be written as a B-tree and so, by the Lemma, any finite sum and therefore any infinite sum of the terms $2^{-A(n)}$ is at most one. Thus no such method can exist. $\Box$

By choosing rapidly growing functions f Theorem 1 gives series which lie on the edge of convergence. In particular the P-search, P^2-search and P^3-search defined earlier give (ignoring constants) the convergent and divergent series of Preamble 2.

Beyond Infinity. For any f-search we induce an f^t-search by induction as follows. Given $f^t(i-1) < x \le f^t(i)$ set $y = f^{-(t-1)}x$. Run an f-search to find y and then, by induction, an f^{t-1}-search to find x. When $f = P$ this duplicated the $P^2, P^3, \ldots$ searches described earlier.

Assume now that $f : N \to N$ is strictly increasing and that $f1 = 2, f2 = 4$. Define a function If by $If(i) = f^i 1$. Note, e.g., $(If)(1) = f(1) = 2$, $(If)(2) = f(f(1)) = 4$, $(If)(3) = f(f(f(1))) = f(4)$.) We induce an If-search as follows:

Given $If(i-1) < x \le If(i)$ with $i > 2$
That is, $f^{i-2}(f(1)) < x \le f^{i-2}(f(f(1)))$
That is, $f^{i-2}(2) < x \le f^{i-2}(4)$
Ask " Is $x \le f^{i-2}3$?"
With either answer run an f^{i-2}-search to find x.
(For $i = 2$, $2 < x \le 4$, simply ask if $x \le 3$.)

IP is usually called the tower function, $IP(i)$ is an exponential tower of i twos. Note that IP grows more rapidly than any P^i. We think of IP as a diagonalization, though verticalization may be more accurate.

Figure 2 illustrates the functions P, P^j, IP and the IP-search when $x =$ a googol $= 10^{100} \sim 2^{332.19}$. The encircled Binary Search is actually 332 queries required given that $2^{332} < x \le 2^{333}$. There were 5 queries to bound x and $1+3+8+332 = 344$ further queries to determine x so $B(x) = 5$, $S(x) = 344$. The function $B(n) = IP^{-1}(n)$ is generally written $\lg^* n$ (read: log star n) and is the number of times one needs to "press the lg button", Searching and Convergence. Let $f : N \to N$ be a strictly increasing function. By an f-search we mean for each $i > 1$ a search that uniquely determines $x \in (f(i-1), f(i)]$. For $n > f(1)$ let $S(n)$ denote the number of queries used in the search (given the interval n lies in) and let $B(n) = f^{-1}(n)$. By asking $f1, f2, \ldots$ until receiving a Yes answer and then employing an f-search all $n > f1$ are determined by $S(n) + B(n)$ queries. By P-search we mean specifically the Binary Search on $(P(i-1), P(i)]$ taking $i - 1$ queries.

Theorem 1. For any f-search

$$\sum_{n \geq f(1)} 2^{-S(n)} = \infty \text{ while } \sum_{n \geq f(1)} 2^{-[S(n)+B(n)]} = \frac{1}{2}$$

Moreover, for no constant c is there a method to find an arbitrary positive integer which finds n in at most $S(n) + c$ queries for all but finitely many n.

A search on a finite interval I can be represented (in CS-lingo) as a B-tree, a rooted tree in which each nonleaf has outdegree two. Here each leaf represents a unique $n \in I$ and each nonleaf represents a query. Then $S(n)$, as defined above, is the distance from leaf n to the root.

Lemma. In any B-tree

$$\sum 2^{-S(n)} = 1,$$

the sum over all leaves n.

Imagine a particle beginning at the root and taking a random path to the leaves where at each node a fair coin is flipped to determine which direction to take. The particle will reach leaf n with probability precisely $2^{-S(n)}$ and the events "The particle reaches n" are disjoint and cover the probability space so that their probabilities sum to unity. □

Applying the Lemma

$$\sum_{f(i-1) < n \leq f(i)} 2^{-S(n)} = 1$$

and so

$$\sum_{n \geq f(1)} 2^{-S(n)} = \sum_{i=2}^{\infty} \sum_{f(i-1) < n \leq f(i)} 2^{-S(n)} = \sum_{i=2}^{\infty} 1 = \infty$$

while

$$\sum_{n \geq f(1)} 2^{-[S(n)+B(n)]} = \sum_{i=2}^{\infty} \sum_{f(i-1) < n \leq f(i)} 2^{-S(n)-i} = \sum_{i=2}^{\infty} 2^{-i} = \frac{1}{2}$$

On the Edge of Convergence
Joel Spencer

Preamble 1. An unknown integer x betweeen 1 and N can be determined by $\lceil \lg N \rceil$ queries of the form "Is $x \leq a$?" and this is best possible. Suppose the protagonist states "I'm thinking of a positive integer." What is a good strategy to find it.

Preamble 2. The harmonic series $\sum \frac{1}{n}$ diverges but $\sum \frac{1}{n^2}$ converges; $\sum \frac{1}{n \lg n}$ diverges but $\sum \frac{1}{n \lg^2 n}$ converges; $\sum \frac{1}{n \lg \lg n}$ diverges but $\sum \frac{1}{n(\lg \lg n)^2}$ converges. Calculus texts state that these example may be extended forever. Here we go beyond forever, without calculus.

Notations. lg denote logarithm to the base two. $P : N \to N$ denotes the power function, $P(i) = 2^i$. When convenient parentheses are eliminated, $Pi = P(i)$. For any $f : N \to N$, f^t denotes f iterated t times, e.g., $P^3 1 = P(P(P(1))) = 16$. For any strictly increasing f, $f^{-1}(x)$ denotes the least y with $x \leq f(y)$ and f^{-t} denotes $(f^t)^{-1}$. For example, $P^{-1}(x) = \lceil \lg x \rceil$ and $P^{-3}(x) = \lceil \lg \lg \lg x \rceil$. In searching for an unknown integers x, "to ask a" is to ask "Is $x \leq a$?".

Basic Searches. How can we search for an unknown integer? Bentley and Yao [2] gave the fundamental ideas, we note also a wonderful expository paper by Knuth [1]. A basic strategy is to ask $P1, P2, P3, \ldots$ until receiving a Yes answer and then running a Binary Search on $(P(i-1), P(i)]$. The number n is then determined by $P^{-1}n + P^{-1}n - 1 = 2\lceil \lg n \rceil - 1$ queries. To improve this: Set $y = P^{-1}x$ and apply the above method to find y. Then $x \in (P(y-1), P(y)]$, run a Binary Search to find x. Here y is an auxilliary variable - the query "Is $y \leq a$?" is actually asked "Is $x \leq 2^a$?". Figure 1 illustrates the method with 100 the target number. Determination of n takes $\lceil \lg \lg n \rceil + \lceil \lg \lg n \rceil - 1 + \lceil \lg n \rceil - 1$ queries. (More precisely, these searches determine any $x \geq A$ where A depends on the search, here $A = 4$.) A further improvement is given by setting $z = P^{-1}y$, applying the basic strategy to find z, then finding y by Binary Search, and finally x. This method takes $P^{-3}n = \lceil \lg \lg \lg n \rceil$ queries to bound n and a further $\lceil \lg \lg \lg n \rceil - 1 + \lceil \lg \lg n \rceil - 1 + \lceil \lg x \rceil - 1$ queries to determine n.