Average - case lower bound on searching for x in a sorted array E with n elements by means of decisions

Theorem.

The average number of decisions (in particual, comparisons of keys) that any algorithm to search an ordered array must perform is at least

$$\left(1 + \frac{q}{n}\right) (\text{Log2}[n + 1] + \epsilon[n + 1]) - q \approx$$

$$\approx \text{Log2}[n + 1] + \epsilon[n + 1] - q.$$

where q is the probability of successful search.

Proof.

Input : array E of n elements;
key x.

Output : and index i s.t.
x = E[i]
if found (successful search);
otherwise (unsuccessful search)
a pair of idicies $-\infty : 0$ if $x < E[0]$
or
a pair of idices $i : i + 1$ if $E[i] < x < E[i + 1]$
or
a pair of idices $n - 1 : \infty$ if $x > E[n - 1]$

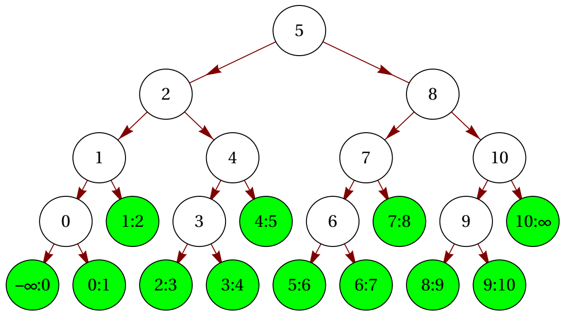
Asserted even probability distribution :

All successful search outcomes are equally likely
and
all unsuccessful search outcomes are equally likely.

The result is proven for any algorithm that searches via decision tree;
in such a general case, the number of decisions
(for instance, the number of comparisons of keys) is the measure of the running time.

Example decision tree T for searching (reused from Binary Search);
nodes represent indicies (success) or pairs of indicies (failure) in array E :

```
TreePlot[{5 → 2, 5 → 8, 2 → 1, 2 → 4, 8 → 7, 8 → 10, 0 → "-∞:0", 1 → 0,
  3 → "2:3", 4 → 3, 6 → "5:6", 7 → 6, 10 → 9, 10 → "10:∞", 0 → "0:1", 1 → "1:2",
  3 → "3:4", 4 → "4:5", 6 → "6:7", 7 → "7:8", 9 → "8:9", 9 → "9:10"},
 VertexLabeling → True, DirectedEdges → True, VertexRenderingFunction →
  ({White, EdgeForm[Black], Disk[#, .3], Black, Text[#2, #1]} &), AspectRatio → 0.55]
```



External nodes are green. These are non - decision nodes that represent the final outcomes of unsuccessful searches.
All others are decision nodes or the final outcomes of successful searches.

Since the avarage number of decisions comparisons of keys in any search is given by the formula :

$T_{avg}(n) = q \times T_{avg}^{succ}(n) + (1 - q) \times T_{avg}^{fail}(n)$

and q (the probability of successful search is fixed), any search that is average - case optimal when successful and average - case optimal when unsuccessful is average - case optimal.

1. Successful search

p - the length of a path to an internal node in T

p + 1 - the number of comparisons done along a path to an internal node in T

N - the number of decision nodes in the decision tree (N = 11 on the above picture)

ipl (T) - the sum of lengths of all paths to internal nodes in T

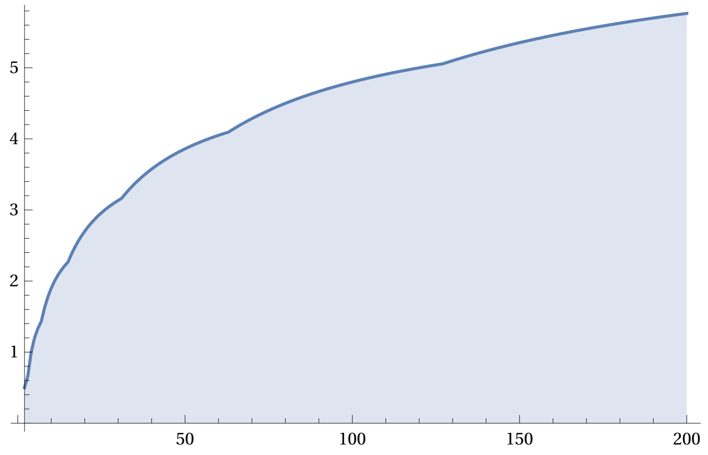
ipl (T) + N - the sum of numbers of comparisons done along all paths to internal nodes in T

The minimal internal path length in any binary tree T with N nodes is $ipl_{min}(N)$.

Fact $\frac{ipl_{min}(N)}{N}$ is a growing function.

Exercise : Prove it !

```
DiscretePlot[∑_{i=1}^n ⌊Log2[i]⌋ / n, {n, 2, 200}]
```



We have :

$$T^{succ}_{avg}(N) = \frac{ipl\,(T)}{N} + 1 \geq \frac{1}{N} ipl_{min}(N) + 1 \geq \frac{1}{n} ipl_{min}(n) + 1 =$$

$$\frac{1}{n}\left(\sum_{i=1}^{n} \lfloor Log2[i] \rfloor\right) + 1 = \frac{1}{n}((n+1)(Log2[n+1] + \epsilon[n+1]) - 2n) + 1 =$$

$$\frac{n+1}{n}(Log2[n+1] + \epsilon[n+1]) - 1 \approx$$

$$\left[\text{since } \frac{n+1}{n} = 1 + \frac{1}{n} \approx 1 \text{ for large } n, \right.$$

$$\left. \text{for instance, } \frac{n+1}{n} < 1.003 \text{ for } n \geq 334 \right]$$

$$\approx Log2[n+1] + \epsilon[n+1] - 1.$$

2. Unsuccessful search

p - the length of a path to an external node in T

p - the number of comparisons done along a path to an external node in T

N + 1 - the number of non - decision nodes in the decision tree (N + 1 = 12 on the above picture)

epl (T) - the sum of lengths of all paths to external nodes in T

epl (T) - the sum of numbers of comparisons done along all paths to internal nodes in T

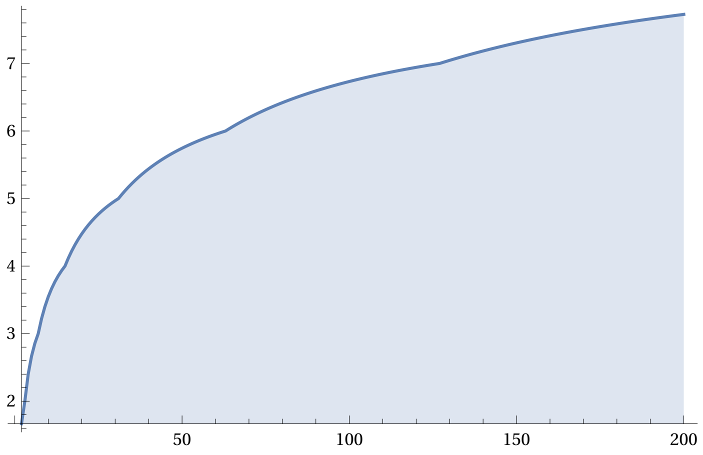
The minimal external path length in
any binary tree T with N nodes is $epl_{min}(N)$.

Fact $\frac{epl_{min}(N)}{N+1}$ is a growing function.

Exercise : Prove it !

```
DiscretePlot[(Sum[Floor[Log2[i]], {i, 1, n}] + 2 n) / (n + 1), {n, 2, 200}]
```

$$\text{DiscretePlot}\left[\left(\sum_{i=1}^{n}\lfloor \text{Log2}[i]\rfloor + 2\,n\right) \Big/ (n+1),\ \{n, 2, 200\}\right]$$



We have

$$T_{avg}^{fail}(N) = \frac{\text{epl}(T)}{N+1} \geq (\text{epl}_{min}(N)) / (N+1) \geq (\text{epl}_{min}(n)) / (n+1) =$$

$$(\text{ipl}_{min}(n) + 2\,n) / (n+1) = ((n+1)\ (\text{Log2}[n+1] + \epsilon[n+1])) / (n+1) =$$

$$= \text{Log2}[n+1] + \epsilon[n+1] \approx$$

$$\approx T_{avg}^{succ}(n) + 1.$$

Now,

$$T_{avg}(N) = q \times T_{avg}^{succ}(N) + (1-q) \times T_{avg}^{fail}(N) =$$

$$q\left(\frac{n+1}{n}\ (\text{Log2}[n+1] + \epsilon[n+1]) - 1\right) + (1-q)\ (\text{Log2}[n+1] + \epsilon[n+1]) =$$

$$= \left(q\,\frac{n+1}{n} + (1-q)\right)(\text{Log2}[n+1] + \epsilon[n+1]) - q =$$

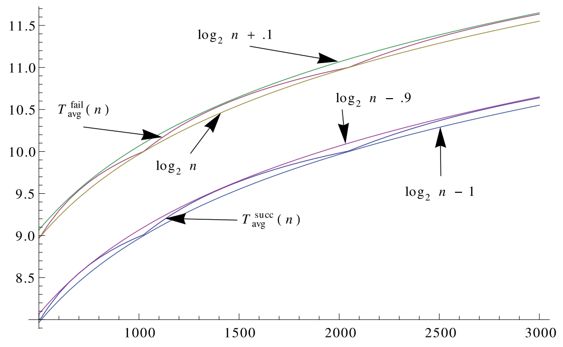
$$= \left(q\left(\frac{n+1}{n} - 1\right) + 1\right)(\text{Log2}[n+1] + \epsilon[n+1]) - q =$$

$$= \left(1 + \frac{q}{n}\right)(\text{Log2}[n+1] + \epsilon[n+1]) - q \approx$$

$$\approx \text{Log2}[n+1] + \epsilon[n+1] - q.$$

End of proof.

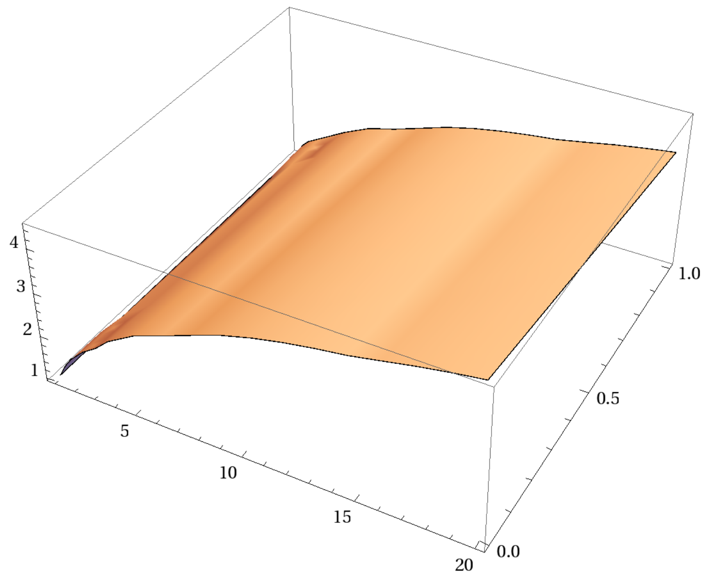
Here are graphs of lower bounds successful and unsuccessful searches :



Here is the graph combined average case, in 3 D with second parameter q (probability of success)

```
Plot3D[(1 + q/N) (Log2[N + 1] + ϵ[N + 1]) - q, {N, 1, 20}, {q, 0, 1}]
```

```
Plot3D::invisop : {} must be a valid 2D coordinate. ≫
```



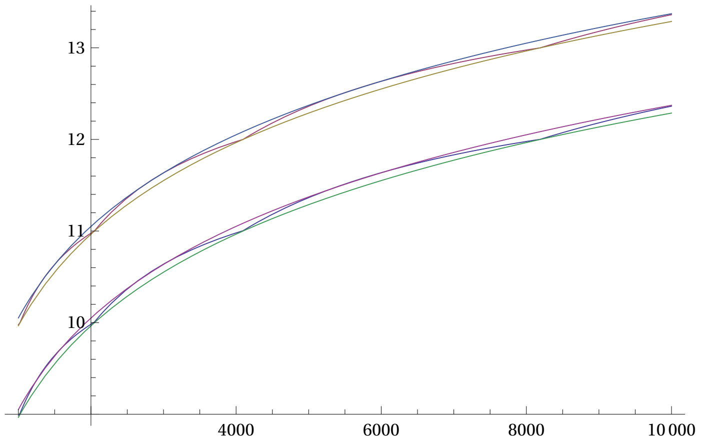
In the next file,
(file : http : // csc.csudh.edu / suchenek / CSC401 / Mathematica / AverageTimeBinSearch.nb) we will show that Binary Search perform exactly that number of comparisons, on the average.

Thus all the formulas and figures above describe the actual average berformance of Binary Search.

The rest of this file is optional for all students.

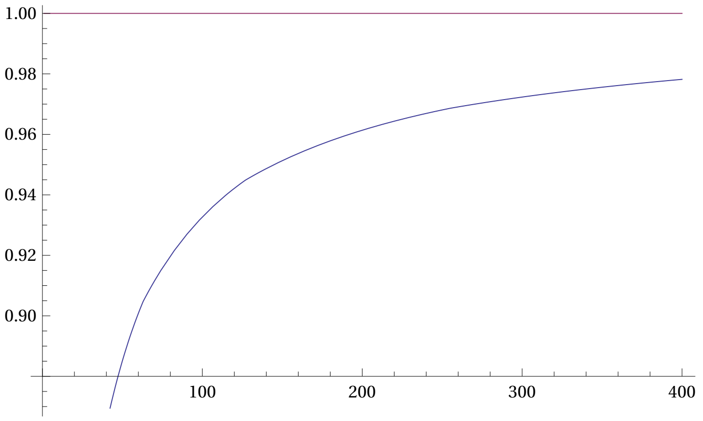
More precise low and high approximations
for N ≥ 1000

```
Plot[Tooltip[{(N + 1)/N (Log2[N + 1] + ϵ[N + 1]) - 1, Log2[N + 1] + ϵ[N + 1],
   Log2[N], Log2[N] - 1, Log2[N] + .085, Log2[N] - .915}], {N, 1000, 10 000}]
```



Here is a graph of the difference $T^{fail}_{avg}(N) - T^{succ}_{avg}(N)$; it converges to 1 as N diverges to ∞.

```
Plot[{Log2[N + 1] + ϵ[N + 1] -
  ((N + 1)/N (Log2[N + 1] + ϵ[N + 1]) - 1), 1}, {N, 1, 400}]
```



Here is the limit in ∞ of the difference :

```
Limit[Log2[N + 1] + ϵ[N + 1] -
  ((N + 1)/N (Log2[N + 1] + ϵ[N + 1]) - 1), N → ∞]
```

```
1
```

Note. Although the difference between the total number of comparisons in unsuccessful search and the total number of comparisons of successfull search in any search algorithm that searches an N - element sorted array by means of decision tree is exactly N, the difference between the respective averages is not exactly 1. This is due to the fact that the successful average is given by the successful total divided by N and the unsuccessful average is given by the unsuccessful total divided by N + 1 . If N is large then the ratio of $\frac{N}{N+1}$ is close to 1, and the difference between these averages is close to 1 as well.