Algorithms
Professor John Reif

ALG 2.2
Search Algorithms

(a) Binary Search: average case
(b) Binary Search with Errors (homework)
(c) Interpolation Search
(d) Unbounded Search

Main Reading Selections:
CLR, Chapter 13
Auxillary Reading Selections:
AHU-Design, 4.1 and 4.5
AHU-Data, Sections 5.1 and 5.1
BB, Sections 4.3 and 8.4.3
Handout: "An Almost Optimal Algorithm for Unbounded Searching"

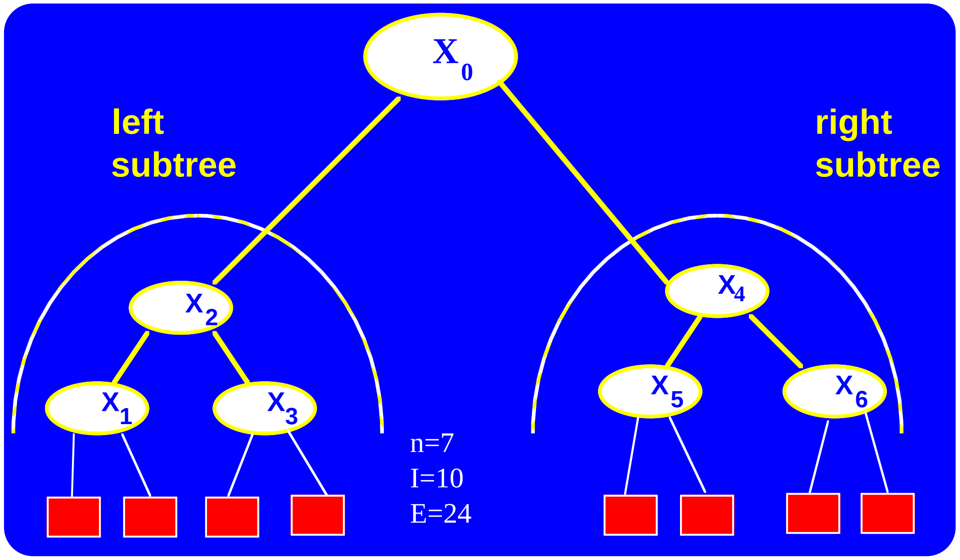
Binary Search Trees

(in sorted Table of) keys $k_0, \ldots, k_{n-1}$

Binary Search Tree property:
at each node x

key (x) > key(y) $\forall$ y nodes on left subtree of x

key (x) < key(z) $\forall$ z nodes on right subtree of x



Assume

(1) keys inserted into tree in *random order*

(2) Search with *all keys equally likely*

length = # of edges

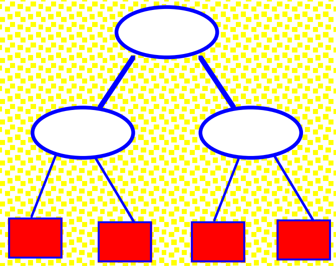
n + 1 = number of leaves

internal path length I

= sum of lengths of all internal paths of length > 1
(from root to nonleaves)

external path length E

= sum of lengths of all external paths
(from root to leaves)
= I+2n



n = 3
I = 2
E = 8

successful search:

expected #comparisons $\bar{C}_n = 1 + (I/n)$

$$= \left[\sum_{i=0}^{n-1} (\bar{C}'_i + 1) \right] / n$$

unsuccessful search:

expected #comparisons

$$\bar{C}'_n = E/(n+1) = (I+2n)/(n+1)$$

$$= (n\,\bar{C}_n + n)/(n+1)$$

$$= \left[\sum_{i=0}^{n-1} (\bar{C}'_i + 2) \right] / (n+1)$$

$$= \sum_{i=1}^{n} \frac{2}{(i+1)} \approx 2\ln(n)$$

$$= 1.386 \log n$$

Model of *Random Input over Reals*

input

Set S of n keys each independently *randomly chosen* over real interval [L,U] for $0 < L < U$

operations

- *comparison operations*
- $\lfloor\ \rfloor, \lceil\ \rceil$ *operations*

results

(1) *sort* in 0(n) expected time

(2) *selection* in 0(loglogn) expected time

input

set of n keys, S randomly chosen over [L,U]

algorithm

BUCKET-SORT(S):

begin

for i=1 *to* n *do* B[i] ← empty list

for i = 1 *to* n *do* add x_i to $B\left[\left\lfloor \frac{n(x_i-L)}{(U-L)} \right\rfloor + 1\right]$

for i=1 to n do sort (B[i])

output B[1] · B[2] ··· B[n]

end

Theorem

The *expected time* $\bar{T}$ of BUCKET-SORT *is 0(n)*

proof

|B[i]| is *upper bounded by a Binomial* variable with parameters $n, p = \frac{1}{n}$

Hence $\exists\, c > 1 \;\; \forall i,j \quad \text{Prob}\{ |B[i]| > j \} < c^{-j}$

$$\text{So } \bar{T} \le n \sum_{j=0}^{n} c^{-j} (j \log j) = O(n)$$

note

generalizes to case keys have distribution F

Random Search Table

$X = (x_0 < x_1 < \ldots < x_n < x_{n+1})$

where $x_1, \ldots, x_n$ random reals chosen independently from real interval (x_0, x_{n+1})

Selection Problem

input key Y

problem find index k^* s.t. $X_{k^*} = Y$

note k^* has *Binomial distribution* with parameters $n, p = (Y - X_0)/(X_{n+1} - X_0)$

Algorithm
INTERPOLATION-SEARCH (X,Y)

[1] ***initialize*** $k \leftarrow \lceil np \rceil$ ***comment*** $k = \lceil E(k^*) \rceil$

[2] **if** $X_k = Y$ ***then return*** k

[3] ***if*** $X_k < Y$ **then**

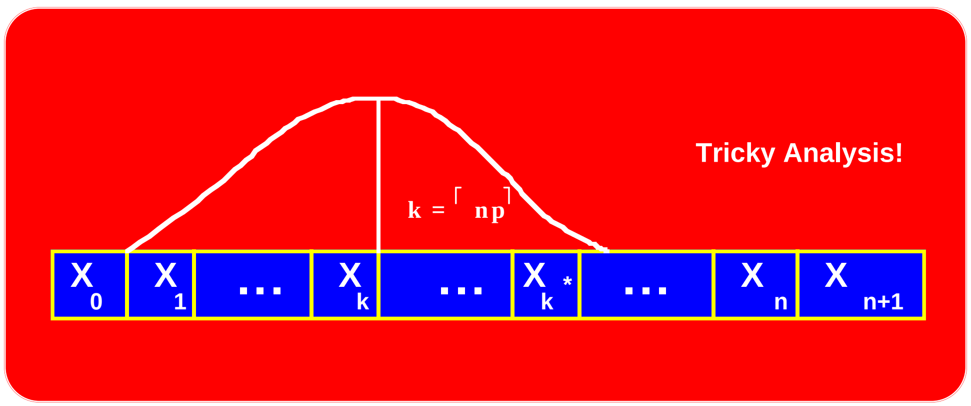
output INTERPOLATION-SEARCH (X',Y)

where $X' = (X_k, \ldots, X_{n+1})$

[4] ***else*** $X_k > Y$ and

output INTERPOLATION-SEARCH (X'',Y)

where $X'' = (X_0, \ldots, X_k)$



Random Table

$X = (X_0, X_1, \ldots, X_n, X_{n+1})$

Algorithm

pseudo interpolation search (X,Y)

[0] $k \leftarrow \lceil pn \rceil$ where $p = (Y - X_0)/(X_{n+1} - X_0)$

[1] if $Y = X_k$ ***then*** return k

[2] ***if*** $Y > X_k$ ***then***

for $k' = k, k+\sqrt{n}, k+2\sqrt{n}, \ldots$

if $Y < X_{\lceil k'+\sqrt{n} \rceil}$ ***then*** exit with

output pseudo interpolation search (X',Y)

where $X' = \left(X_{k'}, \ldots, X_{k'+\sqrt{n}} \right)$

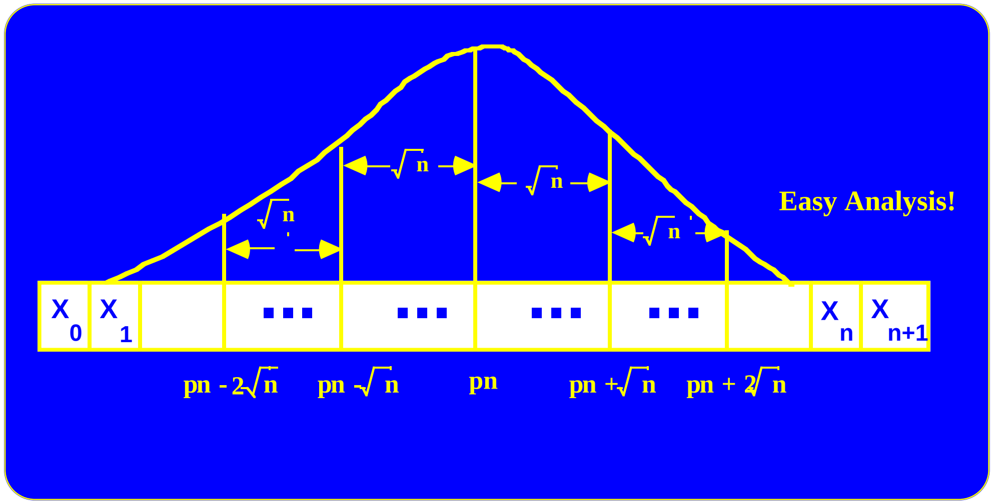
[3] ***else if*** $Y < X_k$ ***then***

for $k' = k, k-\sqrt{n}, k-2\sqrt{n}, \ldots$

if $Y > X_{\lceil k'-\sqrt{n} \rceil}$ ***then*** exit with

output pseudo interpolation search (X'',Y)

where $X'' = \left(X_{k'-\sqrt{n}}, \ldots, X_{k'} \right)$



Probabilistic Analysis of Psuedo Interpolation Search

k^* is *Binomial with mean* pn
variance $\sigma^2 = p(1-p)n$

so $\frac{k^* - \lceil pn \rceil}{\sigma}$ *approximates normal* as $n \to \infty$

Hence $\text{Prob}\left(\frac{k^* - \lceil pn \rceil}{\sigma} \geq Z\right) \leq \Psi(Z)/Z$

where $\Psi(Z) = \frac{e^{-\frac{Z^2}{2}}}{\sqrt{2\pi}}$

So Prob($\geq$ i probes used in given call)

$< \text{Prob}(|k^* - \lceil pn \rceil| > (i-2)\sqrt{n})$

$\leq \Psi(Z_i)/Z_i$

where $Z_i = \frac{(i-2)\sqrt{n}}{\sigma} = \frac{(i-2)}{\sqrt{p(1-p)}} \geq 2(i-2)$

since $p(1-p) \leq \frac{1}{4}$

Lemma $\bar{C} \leq 2.03$ where

$\bar{C}$ = ***expected number of probes*** **in given call**

proof $\bar{C} = \sum_{i>1} i \ \text{Prob}(i \text{ probes used})$

$= \sum_{i>1} \text{Prob}(\geq i \text{ probes used})$

$\leq 2 + \sum_{i\geq 3} \Psi(Z_i)/Z_i \leq 2.03$

Theorem

Pseudo Interpolation Search

has expected time $\bar{T} \leq \bar{C} \log\log n$

proof

$\bar{T}(n) \leq \bar{C} + \bar{T}(\sqrt{n})$

$\leq \bar{C} \log\log n$

Probabilistic Analysis of Interpolation Search

Lemma

$$\text{Prob}(|k^* - \lceil pn \rceil| \geq O(\sqrt{n\log n})) \leq \frac{1}{n^{\alpha}}$$

where α is constant

proof

Since k^* is *Binomial with parameters* p,n

$$\text{Prob}(|k^* - pn| \geq Z\sigma) \leq \frac{2\, e^{-z^2/2}}{Z\sqrt{2\pi}} \leq \frac{1}{n^{\alpha}}$$

for $\sigma^2 = p(1-p)n$ and $Z = O(\sqrt{\log n})$

Theorem

The expected number of comparisons of Interpolation Search is

$$\overline{T}(n) \leq \log\log n + c_1(\log\log\log n)^2$$

proof

$$\overline{T}(n) \leq 1 + \left[(1 - \frac{1}{n^{\alpha}})\, \overline{T}(O(\sqrt{n \log n})) + \frac{n}{n^{\alpha}} \right]$$

$$\leq 1 + \log\log(\sqrt{n\log n}) + c_1 \log\log\log(\sqrt{n\log n})^2 + o(1)$$

$$\leq 1 + \log\left(\frac{1}{2}\log n\right) + c_1(\log\log\log n)^2$$

$$\leq \log\log n + c_1(\log\log\log n)^2 \quad \text{since } \log 2 = 1$$

Unbounded Search

input table X[1], X[2], ...

where for j = 1,2,...

$$X[j] = \begin{cases} 0 & j < n \\ 1 & j \geq n \end{cases}$$

unbounded Search Problem

find n such that X[n-1] = 0 and X[n]=1

Cost for algorithm A:

$C_A(n)$=m if algorithm A uses
m evaluations to determine that
n is the solution to the
unbounded search problem

Applications

(1) *Table Look-up* in an ordered, *infinite table*

(2) *binary encoding of integers*

if S_n represents integer n,

then S_n is *not* a prefix of any S_j $n \neq j$

$\{S_1, S_2, ...\}$ called a prefix set

idea: use $S_n = \left(b_1, b_2, \ldots, b_{C_A(n)}\right)$

where $b_m = 1$

if the m'th evaluation of X is 1

in algorithm A for unbounded search

Unary Search Algorithm

Algorithm B_0

try X[1], X[2] ,...., until X[n] = 1

Cost $C_{B_0}(n) = n$

Binary Search Algorithm

Algorithm B_1

1st stage try $X[2^i - 1]$ for $i = 1, 2, \ldots, m$

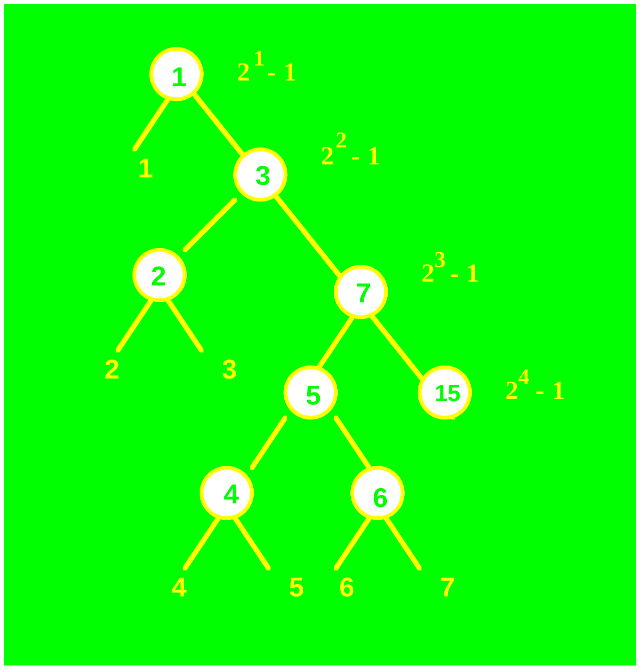
until $X[2^m - 1] = 1$

(*cost* $m = \lfloor \log n \rfloor + 1$ where $2^{m-1} \le n \le 2^m - 1$)

2nd stage binary search over 2^{m-1} elements

cost $\log(2^{m-1}) = m - 1 = \lfloor \log n \rfloor$

Total Cost $C_{B_1}(n) = 2\lfloor \log n \rfloor + 1$



Double Binary Search

Algorithm B_2

1st stage try $X\left[2^{(2^1-1)}-1\right], \ldots, X\left[2^{(2^{m_1}-1)}\right] = 1$

where $m_1 = \lfloor \log n \rfloor + 1$

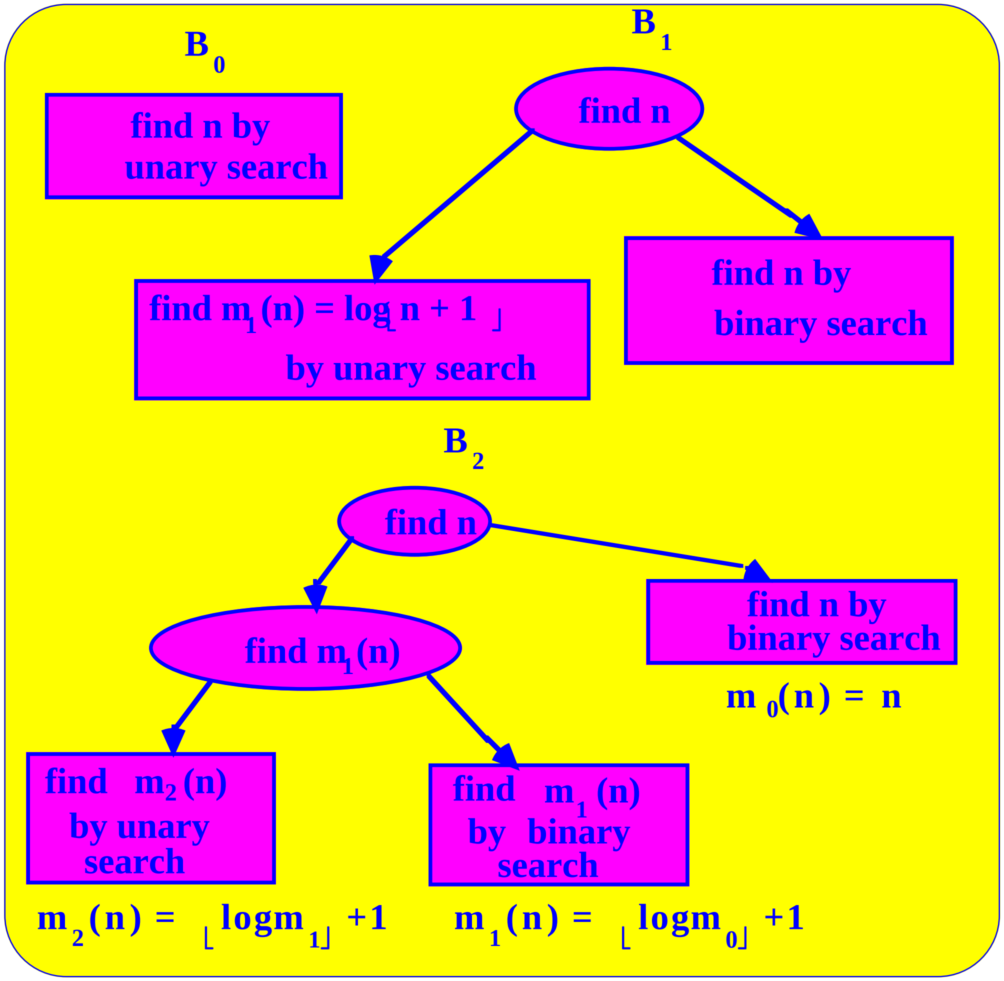
(cost is $C_{B_1}(m_1) = 2\lfloor \log m_1 \rfloor + 1$)

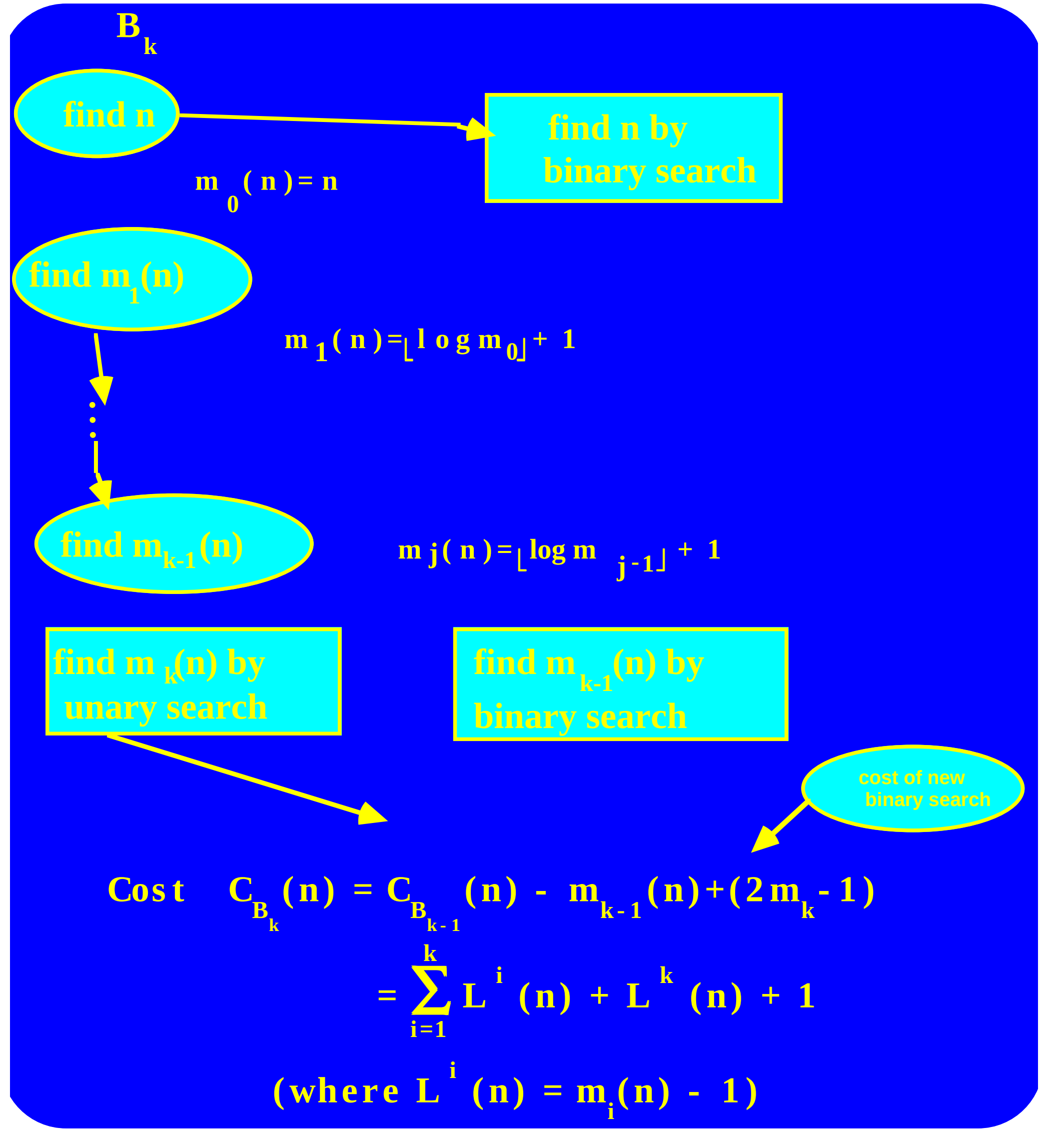
2nd stage same as 2nd stage of B_1 after m was found.

Cost $C_{B_o}(n) = m-1 = \lfloor \log n \rfloor$

Total Cost $C_{B_2}(n) = C_{B_1}(m_1) + C_{B_o}(n)$

$= 2\lfloor \log(\lfloor \log n+1)\rfloor+1) + \lfloor \log n \rfloor$


B0
find n by unary search
B1
find n
find m1(n) = ⌊log n + 1⌋ by unary search
find n by binary search
B2
find n
find n by binary search
find m1(n)
m0(n) = n
find m2(n) by unary search
find m1(n) by binary search
m2(n) = ⌊logm1⌋ +1
m1(n) = ⌊logm0⌋ +1


Bk
find n
find n by binary search
m0(n) = n
find m1(n)
m1(n) = ⌊log m0⌋ + 1
find mk-1(n)
mj(n) = ⌊log mj-1⌋ + 1
find mk(n) by unary search
find mk-1(n) by binary search
cost of new binary search
Cost CBk(n) = CBk-1(n) - mk-1(n) + (2mk - 1)
= Σ_{i=1}^{k} L^i(n) + L^k(n) + 1
(where L^i(n) = mi(n) - 1)

$g(0) = 2$