**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 $x$
  - $\forall y$ nodes on left subtree of $x$: $\text{key}(x) > \text{key}(y)$
  - $\forall z$ nodes on right subtree of $x$: $\text{key}(x) < \text{key}(z)$

---

**Diagram:**

- $n=7$
- $l=10$
- $E=24$
**Assume**

1. keys inserted into tree in *random order*
2. Search with *all keys equally likely*

**length** = # of edges

\[ n + 1 = \text{number of leaves} \]

**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)

\[ = I + 2n \]

---

**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

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

\[ = (n \bar{C}_n + n) / (n+1) \]

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

\[ = \sum_{i=1}^{n} \frac{2}{(i+1)} = 2 \ln(n) \]

\[ = 1.386 \ln n \]
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
- $\sum \cdots \sum$ operations

**results**
(1) sort in $O(n)$ expected time
(2) selection in $O(\log \log n)$ expected time

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

**algorithm**
BUCKET-SORT($S$):

begin
for $i=1$ to $n$ do $B[i] \leftarrow$ empty list

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

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

end
**Theorem**

The expected time $\overline{T}$ of BUCKET-SORT is $O(n)$

**proof**

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

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

So $\overline{T} \leq n \sum_{j=0}^{n} c^{-j} (j \log j) = O(n)$

**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 \( k \leftarrow \left \lfloor np \right \rfloor \) \( \text{comment } k = \left \lfloor E(k^*) \right \rfloor \)
[2] if \( X_k = Y \) \( \text{then return } k \)
[3] if \( X_k < Y \) \( \text{then output } \) INTERPOLATION-SEARCH (X',Y) \( \text{where } X' = (X_k, \ldots, X_{n+1}) \)
[4] else \( X_k > Y \) \( \text{and output } \) INTERPOLATION-SEARCH (X'',Y) \( \text{where } X'' = (X_0, \ldots, X_k) \)

Tricky Analysis!

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

Algorithm
pseudo interpolation search (X,Y)

[0] \( k \leftarrow \left \lfloor pn \right \rfloor \) \( \text{where } p = \frac{(Y - X_0)}{(X_{n+1} - X_0)} \)
[1] if \( Y = X_k \) \( \text{then return } k \)
[2] if \( Y > X_k \) \( \text{then} \) for \( k' = k, k+n, k+2n, \ldots \) \( \text{if } Y < X_{k'+n} \text{ then exit with} \)
\( \text{output pseudo interpolation search (X',Y)} \) \( \text{where } X' = (X_k', \ldots, X_{k'+n}) \)
[3] else if \( Y < X_k \) \( \text{then} \) for \( k' = k, k-n, k-2n, \ldots \) \( \text{if } Y > X_{k'-n} \text{ then exit with} \)
\( \text{output pseudo interpolation search (X'',Y)} \) \( \text{where } X'' = (X_{k'-n}, \ldots, X_{k'}) \)
Easy Analysis!

\[ k^* \text{ is Binomial with mean } \mu = pn \]
\[ \text{variance } \sigma^2 = p(1-p)n \]

so

\[ k^* - \frac{pn}{\sigma} \text{ approximates normal as } n \to \infty \]

Hence

\[ \text{Prob} \left( \frac{k^* - \frac{pn}{\sigma}}{\sigma} \geq Z \right) \leq \Psi(Z) / Z \]

where

\[ \Psi(Z) = \frac{e^{-\frac{Z^2}{2}}}{\sqrt{2\pi n}} \]
So \( \text{Prob}( \geq i \text{ probes used in given call}) \) 
\[
< \text{Prob}( |k^* - \lfloor pn \rfloor| > (i-2)\sqrt{n}) 
\]
\[
\leq \frac{\Psi(Z_i)}{Z_i} 
\]
where 
\[
Z_i = \frac{(i-2)\sqrt{n}}{\sigma} = \frac{(i-2)}{\sqrt{p(1-p)}} \geq 2(i-2) 
\]

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

\[
\text{Lemma} \quad \overline{C} \leq 2.03 \text{ where} 
\]
\[
\overline{C} = \text{expected number of probes in given call} 
\]

\[
\text{proof} \quad \overline{C} = \sum_{i>1} i \text{ Prob (i probes used)} 
\]
\[
= \sum_{i>1} \text{ Prob (\geq i probes used)} 
\]
\[
\leq 2 + \sum_{i\geq3} \Psi(Z_i)/Z_i \leq 2.03 
\]

\[
\text{Theorem} \quad \text{Pseudo Interpolation Search} 
\]
\[
\text{has expected time} \quad \overline{T} \leq \overline{C} \log\log n 
\]

\[
\text{proof} \quad \overline{T}(n) \leq \overline{C} + \overline{T}(\sqrt{n}) 
\]
\[
\leq \overline{C} \log\log n 
\]
Probabilistic Analysis of Interpolation Search

Lemma

\[ \text{Prob}(|k^* - \lfloor pn \rfloor| \geq \sqrt{n \log n}) \leq \frac{1}{n^\alpha} \]

where \( \alpha \) is constant

\textbf{proof}

Since \( k^* \) is Binomial with parameters \( p, n \)

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

for \( \sigma^2 = p(1-p)n \) and \( Z = 0(\sqrt{\log n}) \)

Theorem

The expected number of comparisons of Interpolation Search is

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

\textbf{proof}

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

\[ \leq 1 + \log \log(\sqrt{n \log n}) + c_1 \log \log \log(\sqrt{n \log n}) + o(1) \]

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

\[ \leq \log n + c_1 (\log \log \log n)^2 \text{ since } \log 2 = 1 \]
Unbounded Search

**Input table** \( X[1], X[2], \ldots \)

where for \( j = 1, 2, \ldots \)

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

**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 \( S_j, n \neq j \)

\( \{ S_1, S_2, \ldots \} \) called a prefix set

**Idea:** use \( S_n = (b_1, b_2, \ldots, b_{C_A(n)}) \)

where \( b_m = 1 \)

if the \( m \)'th evaluation of \( X \) is 1

in algorithm A for unbounded search
**Unary Search Algorithm**

**Algorithm** $B_0$


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

---

**Binary Search Algorithm**

**Algorithm** $B_1$

1st stage

try $X[2^i - 1]$ for $i=1,2,..., m$

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

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

2nd stage

binary search over $2^{m-1}$ elements

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

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

**Algorithm B**₂

1st stage
- try \(X\left[\frac{2^{(2^m)} - 1}{2}ight], \ldots, X\left[\frac{2^{(2^m)} - 1}{2}\right] = 1\)
- where \(m = \lceil \log n \rceil + 1\)
- (cost is \(CB_{B_1}(m) = 2 \lceil \log m \rceil + 1\))

2nd stage
- same as 2nd stage of \(B_1\) after \(m\) was found.
- Cost \(CB_{B_2}(n) = m - 1 = \lceil \log n \rceil\)

**Total Cost**
- \(CB_{B_2}(n) = CB_{B_1}(m) + CB_{B_2}(n)\)
- \(= 2 \lceil \log (\lceil \log n + 1 \rceil + 1) \rceil + \lceil \log n \rceil\)
find n by unary search

\[ m_1(n) = \log n + 1 \]

by unary search

find n by binary search

\[ m_1(n) = \log m_0 + 1 \]

find m (n) by k-1 binary search

\[ m_j(n) = \lceil \log m_{j-1} \rceil + 1 \]

find m_{k-1}(n) by unary search

\[ m_0(n) = n \]

Cost \( C_{B_k}(n) = C_{B_{k-1}}(n) - m_{k-1}(n) + (2m_k - 1) \)

\[ = \sum_{i=1}^{k} L^i(n) + L^k(n) + 1 \]

(where \( L^i(n) = m_i(n) - 1 \))
$g(0) = 2$