Search Algorithms

Analysis of Algorithms

Prepared by
John Reif, Ph.D.
Search Algorithms

a) Binary Search: average case
b) Interpolation Search
c) Unbounded Search (Advanced material)
Readings

• Reading Selection:
  • CLR, Chapter 12
Binary Search Trees

- (in sorted Table of) keys $k_0, \ldots, k_{n-1}$
- Binary Search Tree property:
  at each node $x$
  
  $\text{key}(x) > \text{key}(y)$ \quad \# \; y \; \text{nodes on left subtree of } x
  
  $\text{key}(x) < \text{key}(z)$ \quad \# \; z \; \text{nodes on right subtree of } x
Binary Search Trees (cont’d)
Binary Search Trees (cont’d)

- **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)
Binary Search Trees (cont’d)

• **External path length E**
  
  \[ E = \text{sum of lengths of all external paths} \]
  
  \[ \text{(from root to leaves)} = I + 2n \]
Successful Search

- Expected # comparisons

\[
\overline{C}_n = 1 + \frac{I}{n} \\
= \left[ \sum_{i=0}^{n-1} (\overline{C}'_i + 1) \right] / n
\]
Unsuccessful Search

- Expected # comparisons

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

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

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

\[
= \sum_{i=0}^{n} \frac{2}{(i+1)} \approx 2 \ln(n)
\]

\[
= 1.386 \log n
\]
Model of Random Real Inputs over an Interval

• **Input**
  Set S of n keys each independently randomly chosen over real interval |L,U| for 0 < L < U

• **Operations**
  • Comparison operations
  • +, -, *, % operations:
    • ⌈ r ⌈ = largest integer below or equal to r
    • ⌊ r ⌋ = smallest integer above of equal to r
Sorting and Selection with Random Real Inputs

• Results
  1) Sorting in $O(n)$ expected time

  2) Selection in $O(\log\log n)$ expected time
Bucket Sorting with Random Inputs

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

**Algorithm**

BUCKEY-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\cdot L)}{(U-L)} + 1\right]

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


end
Bucket Sorting with Random Inputs (cont’d)

- **Theorem** The expected time $T$ of BUCKET-SORT is $O(n)$

  Proof

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

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

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

- Generalizes to case keys have nonuniform distribution
Random Search Table

- **Table** $X = (x_0 < x_1 < ... < x_n < x_{n+1})$
  where $x_1, ..., x_n$ random reals chosen independently from interval $(x_0, x_{n+1})$

- **Selection Problem**
  
  **Input** key Y

  **Problem** find index $k^*$ with $X_{k^*} = Y$

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

PSEUDO INTERPOLATION-SEARCH (X, Y)

Random Table

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

Algorithm

pseudo interpolation search (X, Y)

\[ [0] \ k \leftarrow \left\lceil \frac{p}{n} \right\rceil \text{ where } p = \frac{(Y-X_0)}{(X_{n+1}-X_0)} \]

\[ [1] \text{ if } Y = X_k \text{ then return } k \]
Algorithm PSEUDO-INTERPOLATION-SEARCH (X,Y) (cont’d)

[2] If $Y > X_k$ then

\[
\text{for } k' = k, k + \sqrt{n}, k + 2\sqrt{n}, \ldots
\]

\[
\text{if } Y < X_{\lfloor k' + \sqrt{n} \rfloor} \text{ then exit with }
\]

\[
\text{output pseudo interpolation search } (X', Y)
\]

\[
X' = (X_{k'}, \ldots, X_{k' + \sqrt{n}})
\]

where
Algorithm PSEUDO INTERPOLATION-SEARCH (X,Y) (cont’d)

[3] Else if $Y < X_k$ then

\[
\text{for } k' = k, k-\sqrt{n}, k-2\sqrt{n}, \ldots
\]

\[
\text{if } Y > X_k' \left[ k', \sqrt{n} \right] \text{ then exit with }
\]

output pseudo interpolation search $(X', Y)$

\[
X'' = (X_{k'-\sqrt{n}}, \ldots, X_{k'})
\]

where
Probability Distribution of Pseudo Interpolation Search
Probabilistic Analysis of Pseudo 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 $n$ grows

- Hence
  \[
  \Pr\left(\frac{k^* - \lceil pn \rceil}{\sigma} \geq Z\right) \leq \Psi(Z)/Z
  \]
  \[
  \text{where } \Psi(Z) = \frac{-Z^2}{\sqrt{2\pi}}
  \]
Probabilistic Analysis of Pseudo Interpolation Search (cont’d)

- So \( \text{Prob}( \leq i \text{ probes used in given call}) \)

\[
< \text{Prob}(|k^* - \lfloor pn \rfloor| > (i - 2)\sqrt{n}) \leq \Psi(Z_i)/Z_i
\]

- where

\[
Z_i = \frac{(i - 2)\sqrt{n}}{\sigma} = \frac{(i - 2)}{\sqrt{p(i-p)}} \geq 2(i - 2)
\]

because \( p(1-p) \leq \frac{1}{4} \)
Probabilistic Analysis of Pseudo Interpolation Search (cont’d)

- **Lemma**  \( \bar{C} \approx 2.03 \) where 
  \( \bar{C} = \text{expected number of probes} \) in given call

**proof**

\[
\bar{C} = \sum_{i>1} i \cdot \text{Prob}(i \text{ probes used}) \\
= \sum_{i>1} \text{Prob}(\geq i \text{ probes used}) \\
\leq 2 + \sum_{i \geq 3} \Psi(Z_i)/Z_i \leq 2.03
\]
Probabilistic Analysis of Pseudo Interpolation Search (cont’d)

- Theorem

Pseudo Interpolation Search

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

**proof**

\[
\bar{T}(n) \leq \bar{C} + \bar{T}(\sqrt{n}) \\
\leq \bar{C} \log\log n
\]
Algorithm
INTERPOLATION-SEARCH (X,Y)

1) Initialize \( k = \lceil np \rceil \)
   
   comment \( k = \lceil E(k^*) \rceil \)

1) If \( X_k = Y \) then return \( k \)
2) If \( X_k < Y \) then
   output INTERPOLATION-SEARCH
   \( (X',Y) \) where \( X' = (x_k, \ldots, x_{n+1}) \)
3) Else \( X_k > Y \) and
   output INTERPOLATION-SEARCH
   \( (X'',Y) \) where \( X'' = (x_0, \ldots, x_k) \)
Probability Distribution of INTERPOLATION-SEARCH (X,Y)

k = \left\lfloor np \right\rfloor

Tricky Analysis!
Advanced Material: Probabilistic Analysis of Interpolation Search

- **Lemma**

\[
\text{Prob}(|k^* - \lfloor pn \rfloor| \geq 0(\sqrt{n \log n})) \leq \frac{1}{n^a}
\]

where \(a\) is a constant

- **Proof**

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

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

for \(\sigma^2 = p(1-p)n\) and \(Z = 0(\sqrt{\log n})\)
Probabilistic Analysis of Interpolation Search (cont’ d)

- Theorem
  - The expected number of comparisons of Interpolation Search is

\[ \bar{T}(n) \leq \log\log n + c_1 (\log\log\log n)^2 \]
Probabilistic Analysis of Interpolation Search (cont’d)

- Proof

\[
\bar{T}(n) \leq 1 + \left[ (1 - \frac{1}{n^a}) \bar{T} \left( O(\sqrt{n \log n} \right) + \frac{n}{n^a} \right] \\
\leq 1 + \log \log \sqrt{n \log n} + c_1 \log \log \log \sqrt{n \log n}^2 + o(1) \\
\leq 1 + \log \left( \frac{1}{2} \log n \right) + c_1 (\log \log \log n)^2 \\
\leq \log \log n + c_1 (\log \log \log n)^2 \text{ since } \log 2 = 1
\]
Advance Material: 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 of Unbounded Search

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 $S_j$ for $n > j$
   - $\{S_1, S_2, ...\}$ called a prefix set

- **Idea**: use $S_n \left( b_1, b_2, ..., b_{c_A(n)} \right)$
  - where $b_m = 1$
  - if the $m$’th evaluation of $X$ is 1
  - in algorithm $A$ for unbounded search
Unary Unbounded Search Algorithm

• Algorithm $B_0$
  • Try $X[1], X[2], \ldots$, until $X[n] = 1$
  • Cost $C_{B_0}(n) = n$
Binary Unbounded Search Algorithm

- Algorithm $B_1$

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

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

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

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

Total Cost $C_{B_1}(n) = 2 \lfloor \log n \rfloor + 1$
Binary Unbounded Search Algorithm (cont’d)
Double Unbounded Search Search

- Algorithm $B_2$

1st stage  
try $X[2^{(2^1 - 1)} - 1]$, ..., $X[2^{(2^{m_1} - 1)}] = 1$

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 is $C_{B_0}(n) = m - 1 = \lfloor \log n \rfloor$

Total Cost  
$C_{B_2}(n) = C_{B_1}(m_1) + C_{B_0}(n)$

$= 2(\lfloor \log (\lfloor \log n + 1 \rfloor + 1) \rfloor + 1) + \lfloor \log n \rfloor$
Double Unbounded Search Algorithm (cont’d)

\[ B_0 \]
- find \( n \) by unary search

\[ B_1 \]
- find \( n \)
- find \( m_1(n) = \log_2 n + 1 \) by unary search

\[ B_2 \]
- find \( n \)
- find \( m_1(n) \)
- find \( m_2(n) \) by unary search

\[ m_2(n) = \lfloor \log_{m_1} n \rfloor + 1 \]

\[ m_1(n) = \lfloor \log_{m_0} n \rfloor + 1 \]

- find \( n \) by binary search

\[ m_0(n) = n \]
Ultimate Unbounded Search Algorithm

\[ B_k \]

**find** \( m_0(n) = n \)

**find** \( m_1(n) \)

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

**find** \( m_{j-1}(n) \)

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

**find** \( m_k(n) \) by unary search

**find** \( m_{k-1}(n) \) by binary search

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 \( L^i(n) = m_i(n) - 1 \))
Search Algorithms

Analysis of Algorithms

Prepared by
John Reif, Ph.D.