Design and Analysis of Algorithms

Lecture 7: Searching Algorithms Generalization

Binary Search

- In the general case we have n nodes not 10

2 Trees

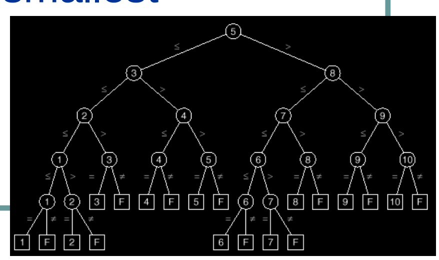
- Def: As 2-tree is a tree in which every vertex except the leaves has exactly two children.
- LEMMA 6.1 The number of vertices on each level of a 2-tree is at most twice the number on the level immediately above.
- LEMMA 6.2 In a 2-tree, the number of vertices on level *t is at most* 2^t *for* $t \geq 0$.

Analysis of Binary1Search - Iterative

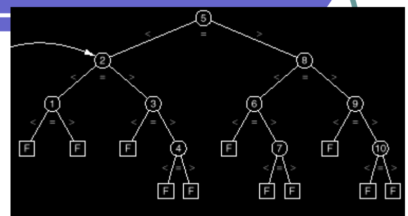
- Both successful and unsuccessful search terminates at leaves.
- Number of leaves 2n.
- Leaves occupy last two levels.
- The height is also the smallest

integer t for which $2^t \geq 2n$

$t \geq lg(n)+1$



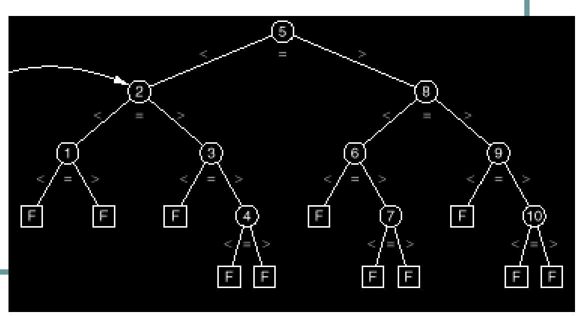
Analysis of Binary2Search - Iterative



- Unsuccessful Search
  - The tree is full at top
  - All failures are leaves
  - Leaves are at the last two levels
  - Number of leaves = n+1
    - $2^h \geq n+1$
    - $h \geq \log(n+1)$
  - *2 comparisons per node*
  - *2*log(n+1) number of comparisons*

Theorem 6.3

- THEOREM 6.3 Denote the external path length of a 2-tree by *E, the internal path length by I , and let q* be the number of vertices that are not leaves.
- Then *E= I+ 2q*



Proof by Induction $E = I + 2q$

- If only root I=E=q=0
- Let v be immediate parent of two leaves
- Delete two of the children
- Show that if $E_{old} = I_{old} + 2q_{old}$ *will produce* $E_{new} = I_{new} + 2q_{new}$
- $E_{old} = E_{new} + 2(k+1) - k = E_{new} + k + 2$
- $I_{old} = I_{new} + k$
- $q_{old} = q_{new} + 1$

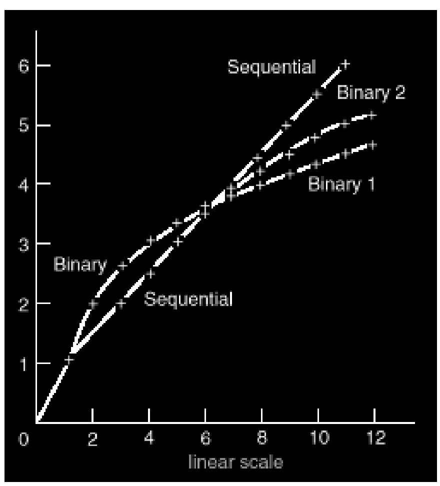
Analysis of Binary2Search - Iterative

- Successful Search
  - Therefore E=(n+1)log(n+1)
  - Number of internal nodes n
  - From Theorem 6.3 (I=E-2q)
    - I=(n+1)log(n+1)-2n
  - Average internal path length: I/n
  - Average number of vertices traversed (I/n)+1
  - 2*((I/n)+1) -1 is the average number of comparisons done
  - Opening the brackets we will get $\frac{2(n+1)}{n}\lg(n+1)-3$

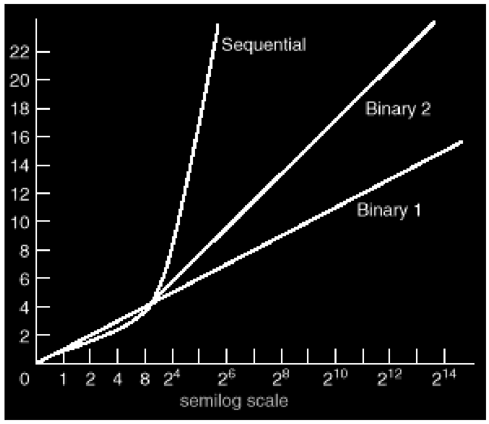
Comparison of Methods

	Successful search	Unsuccessful search
Binary1Search	**lg(n)+1**	**lg(n)+1**
Binary2Search	**2lg(n)-3**	**2lg(n)**

Graphs - Linear



Graphs – Semilogarithmic



- Is there even a better algorithm?

Lemma 6.5

LEMMA 6.5 Let T be a 2-tree with k leaves. Then the height h of T satisfies $h \geq \lceil \lg k \rceil$ and the external path length $E(T)$ satisfies $E(T) \geq k \lg k$. The minimum values for h and $E(T)$ occur when all the leaves of T are on the same level or on two adjacent levels.

Lemma 6.5

LEMMA 6.5 Let T be a 2-tree with k leaves. Then the height h of T satisfies $h \geq \lceil \lg k \rceil$ and the external path length $E(T)$ satisfies $E(T) \geq k \lg k$. The minimum values for h and $E(T)$ occur when all the leaves of T are on the same level or on two adjacent levels.

- E(T')=E(T)-2r +(r-1) -s +2 (s-1) =E(T)-r+s+1<E(T)
- All leaves are either on the same level or two adjacent levels.
- Lemma 2 will still hold,
  - If all the k leaves are in last level h then $k \leq 2^h$
  - If there are some leaves on the upper level this fact makes inequality $k \leq 2^h$ even stronger
  - Always $k \leq 2^h$ or h≥ ┌log(k)┐
- It is possible to show that $E(T) \geq k(\log(k)+1+e-2^e)$ for all $0 \leq e<1$ where $0 \leq (1+e-2^e)<0.0861$
- Thus the minimum external path length is at least to ┌klogk ┐

Lowest Bound on Search

Theorem 6.6 Suppose that an algorithm uses comparisons of keys to search for a target in a list. If there are k possible outcomes, then the algorithm must make at least $\lceil \lg k \rceil$ comparisons of keys in its worst case and at least $\lg k$ in its average case.

- Irrespective of the search method, any comparison based search will have 2n+1 outcomes.
- Each comparison will result in a two way fork. Thus, the comparison tree will always be a 2-tree.
- Applying the Theorem 6.6 to the Binary Search where n positive and n+1 negative outcomes the number of comparisons is bounded as $\lceil \log(2n+1) \rceil \geq \lceil \log(2n) \rceil = \lceil \log(n) \rceil + 1$

Conclusion

Corollary 6.7 Binary1Search is optimal in the class of all algorithms that search an ordered list by making comparisons of keys. In both the average and worst cases, Binary1Search achieves the optimal bound.

- Other ways:
  - – Keys are all integers 1-n.
- Interpolation Search:
  - If keys are uniformly distributed log(log(n))
  - for n = 1,000,000 Binary1Search will require 21 comparisons.
  - Interpolation search will require about 4.32 comparison.