Generalization
What if n is large?

20

2 Trees

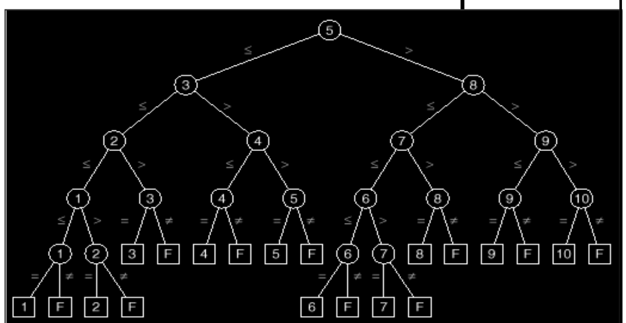
DESIGN &
ALALYSIS OF
ALGORITHM

- Definition: 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 $t \geq 0$.

ALG00S, javed@kent.edu

Analysis of Binary1 Search

- Both successful and unsuccessful search terminates at leaves.
- Number of leaves 2n.
- They are in the last two levels.
- The height is also the smallest integer t for which:
  - $2^{t} \geq 2n$
  - $2^{t-1} \geq n$
  - t-1=log n
  - t=log n+1

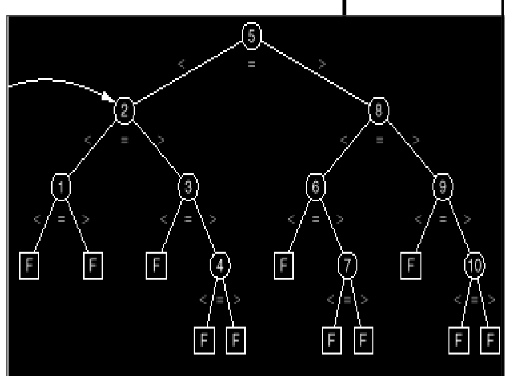


Analysis of Binary 2 Search

DESIGN &
ALALYSIS OF
ALGORITHM

<u>Unsuccessful Search:</u>

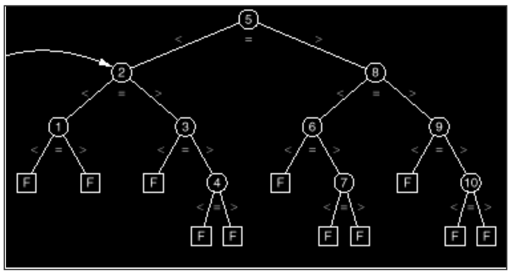
- The tree is full at top
- All F are leaf nodes.
- Leafs are in last two levels.
- Number of leafs = n+1
  - h≈ log (n+1)
- 2 comparison per node
- number of comparisons
  - 2*log (n+1)



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

$$E = I + 2q$$

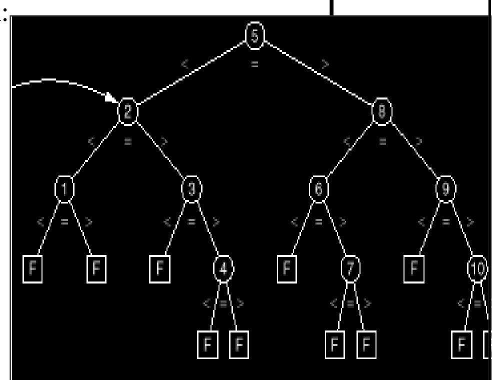


DESIGN & ALALYSIS OF ALGORITHM

Proof of Theorem 6.3

- For a tree with only root: I=0, E=0, q=0
- Lets v be an immediate parent to any two leaf nodes:
- Lets now delete these two children:
- new E = E - 2(k+1) +k
- new I = I-k
- new q = q-1

- Does the relationship still holds?
- E -k-2 = I +2q - k-2
- With reorganization:
- E - 2(k+1)+k = (I-k) + 2(q-1)
- new E = new I + 2 * new q
- That's proof by induction!



DESIGN & ALALYSIS OF ALGORITHM

Analysis of Binary 2 Search

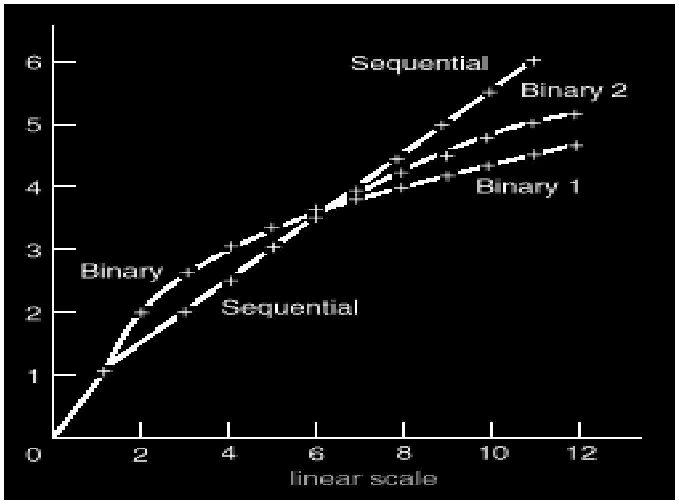
<u>Successful Search:</u>

- Number of leaves (n+1)
  - 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 nodes in internal paths:
  - I/n +1
- Each node except the last one has two comparisons: number of comparisons:
  - 2*(I/n +1) -1 = 2* [(n+1)/n * log (n+1) -2 +1] -1
  - 2 (n+1)/n * log (n+1) -3 ≈ 2 log n -3

Comparison of Methods

	Successful search	*Unsuccessful search*
Binary1Search	$\lg n + 1$	$\lg n + 1$
Binary2Search	$2 \lg n - 3$	$2 \lg n$

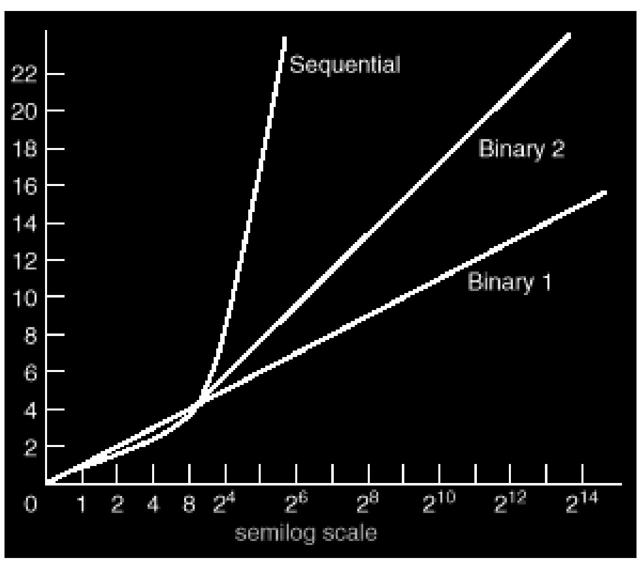
Graphs



DESIGN &
ALALYSIS OF
ALGORITHM

LECT-04, S-28
ALG00S, javed@kent.edu
Javed I. Khan@1999

Logarithmic Plot



DESIGN &
ALALYSIS OF
ALGORITHM

LECT-04, S-29
ALG00S, javed@kent.edu
Javed I. Khan@1999

Can there ever be any better algorithm?

30

Lower Bound on Path Length

DESIGN &
ALALYSIS OF
ALGORITHM

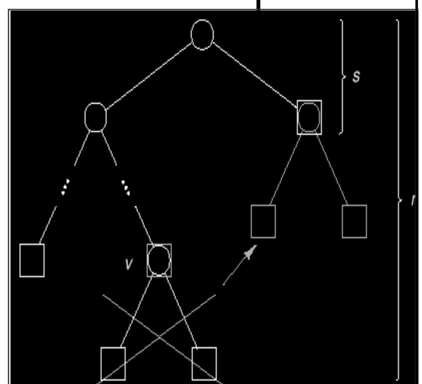
LEMMA 6.5 Let T be a 2-tree with k leaves. Then the height h of T satisfies $h \geq \lceil \lg k \rceil$ and the external path length $E(T)$ satisfies $E(T) \geq k \lg k$. The minimum values for h and $E(T)$ occur when all the leaves of T are on the same level or on two adjacent levels.

Proof

DESIGN & ALALYSIS OF ALGORITHM

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.

- $E(T') = E(T) - 2r + (r-1) - s + 2(s-1) = E(T) - r + s + 1 < E(T)$
- Even after all the compaction when all leaves are just in two adjacent levels.
- Lemma 2 will still hold,
  - If all the k leaves are in last level h then $k \leq 2^h$
  - If some of them in upper level for every hole there is one less leaf.
  - So always $k \leq 2^h$, or $h \geq \lceil \log k \rceil$
- From further analysis it can be shown that $E(T) \geq k(\log k + 1 + e - 2^e)$ for all $o \leq e < 1$
- $(1 + e - 2^e)$ is between 0 and .0861
- Thus the minimum external path length is at least to $\lceil k \log k \rceil$



Lowest Bound on Search

DESIGN & ALALYSIS OF ALGORITHM

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.
- From Lemma 6.5: The external path length
- $E(T) \geq (2n+1) \log (2n+1)$
- The best possible average worst-case:
  - $\geq \lceil \log(2n+1) \rceil \geq \lceil \log 2n \rceil = \lceil \log n \rceil + 1$

Other ways of Searching

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 Binary1 will require 21 comparisons.
  - Interpolation search will require about 4.32 comparison.

Next Class:

- Classification of algorithms’ running time.
- Asymptotics.