Minimize average access time

• Items have weights: Item $i$ has weight $w_i$.

• Let $W = \sum w_i$ be the total weight of the items.

• Want the search to heavy items to be faster.

• If $p_i = w_i/W$ represents the access frequency to item $i$ then the average access time is

$$\sum p_i \cdot d_i$$

where $d_i$ is the depth of item $i$. 
There is a lower bound

\[ \sum p_i d_i \geq \sum p_i \log_b \left( \frac{1}{p_i} \right) \]

for every tree with maximum degree \( b \)

So we will be looking for trees for which \( d_i = O(\log (W/w_i)) \)

In particular if all weights are equal the regular search trees which we have studied, will do the job.
Approximation (Mehlhorn)
Approximation (Mehlhorn)
Keep the element in the part with which its intersection is larger.
A subproblem with one element makes a leaf
A subproblem with one element makes a leaf
A subproblem with one element makes a leaf
A subproblem with two elements makes a splits to two leaves.
Skip trivial splits
Skip trivial splits
Skip trivial splits
Analysis

An internal node at level $i$ corresponds to an interval of length $1/2^i$.

The sum of the weights of the pieces corresponding to an internal node is no larger than the length of its interval.
Analysis

Look at a leaf of weight \( p \) at depth \( d \)

At least half of \( p \) belong to the subproblem corresponding to the parent of \( p \) (which is an internal node) so we have

\[
\frac{p}{2} \leq \frac{1}{2^{d-1}}
\]

\[
d - 2 \leq \log\left(\frac{1}{p}\right)
\]

\[
d \leq \log\left(\frac{1}{p}\right) + 2
\]
Implementation

Maintain the elements is an array
Compute prefix sums of the probabilities
Then we can find where to split a problem using binary search
\( \Rightarrow O(n \log(n)) \) time
Dynamic Versions

• Biased 2-b trees, (Bent, Sleator, Tarjan 1980)
• D-trees (Mehlhorn)
• We will see splay trees that also achieve this, in an amortized sense