CS 240 – Data Structures and Data Management

Module 6E: Dictionaries for special keys - Enriched

T. Biedl É. Schost O. Veksler

Based on lecture notes by many previous cs240 instructors

David R. Cheriton School of Computer Science, University of Waterloo

Winter 2021

version *2021−02−10 12:29*

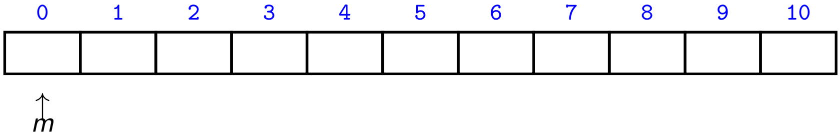
Improving Interpolation Search

- Had: Average-case run-time of *interpolation-search* is $O(\log\log n)$.
- This is very complicated to prove!

$$\left(\begin{array}{l} \blacktriangleright \text{ Study error, i.e., distance between index of } k \text{ and where we probed.} \\ \blacktriangleright \text{ Argue that error is in } O(\sqrt{n}) \text{ in first round.} \\ \blacktriangleright \text{ Argue that error is in } O(\frac{1}{2^i} n) \text{ after } i \text{ rounds.} \\ \blacktriangleright \text{ Study the martingale formed by the errors in the rounds.} \\ \blacktriangleright \text{ Argue that its expected length is } O(\log\log n). \end{array}\right)$$

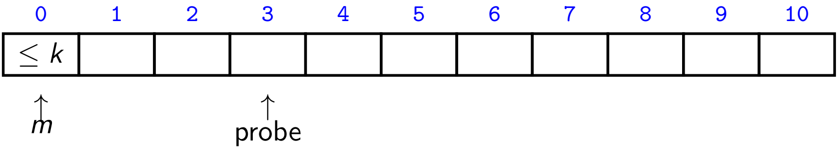
- Instead: Define a variant of *interpolatation-search*
  - Better worst-case run-time.
  - Easier to analyze.
- Idea: *Force* the sub-array to have size $\sqrt{n}$
- To do so, search for suitable sub-array with probes.
- Crucial question: how many probes are needed?

Improving Interpolation Search



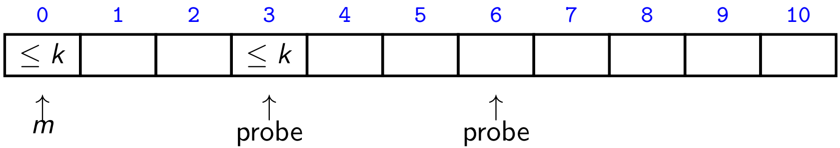
- First compare at m as before.

Improving Interpolation Search



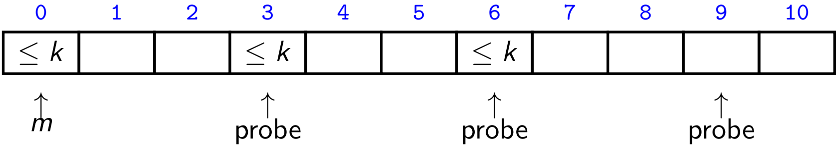
- First compare at m as before.
- If $A[m] \leq k$, probe rightward.
- Probes always go $\lfloor \sqrt{N} \rfloor$ indices rightward
  (where $N = r - \ell$ is the size of the sub-array where k could be)

Improving Interpolation Search



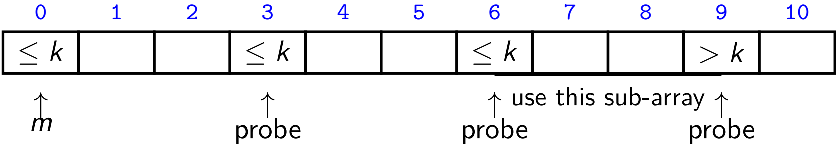
- First compare at m as before.
- If $A[m] \leq k$, probe rightward.
- Probes always go $\lfloor \sqrt{N} \rfloor$ indices rightward (where $N = r - \ell$ is the size of the sub-array where k could be)
- Continue probing until $> k$ or out-of-bounds

Improving Interpolation Search



- First compare at m as before.
- If $A[m] \leq k$, probe rightward.
- Probes always go $\lfloor \sqrt{N} \rfloor$ indices rightward (where $N = r - \ell$ is the size of the sub-array where k could be)
- Continue probing until $> k$ or out-of-bounds

Improving Interpolation Search



- First compare at m as before.
- If $A[m] \leq k$, probe rightward.
- Probes always go $\lfloor \sqrt{N} \rfloor$ indices rightward (where $N = r - \ell$ is the size of the sub-array where k could be)
- Continue probing until $> k$ or out-of-bounds
- Observe: $\#$ probes $\leq \frac{N}{\lfloor \sqrt{N} \rfloor} \leq \sqrt{N} + 1$.

Improving Interpolation Search

Interpolation-search-modified(A, n, k)
A: sorted array of size n, k: key

if $(k < A[0]$ or $k > A[n-1])$ **return** "not found"
if $(k = A[n-1])$ **return** "found at index $n-1$"
$\ell \leftarrow 0$, $r \leftarrow n-1$ // have $A[\ell] \leq k < A[r]$
while $(N \leftarrow (r-\ell) \geq 2)$
    $m \leftarrow \ell + \frac{k - A[\ell])}{A[r]-A[\ell]} \cdot (r-\ell)$
    if $(A[m] \leq k)$ // probe rightward
        $\ell \leftarrow m, m_r \leftarrow \min\{r, m + \lfloor\sqrt{N}\rfloor\}$
        while $(m_r < r$ and $A[m_r] < k)$
            $\ell \leftarrow m_r$, $m_r \leftarrow \min\{r, m + \lfloor\sqrt{N}\rfloor\}$
        $r \leftarrow m_r$ // found suitable sub-array
    else
        $\vdots$ // symmetrically probe leftward
if $(k = A[\ell])$ **return** "found at index ℓ"
else return "not found"

Analysis of *interpolation-search-improved*

- $T(n) \leq T(\sqrt{n}) + O(\#\text{probes})$
- $\#$ probes $\leq \sqrt{n} + 1$.
- The worst-case run-time satisfies

$$T^{\text{worst}}(n) \leq T^{\text{worst}}(\sqrt{n}) + c \cdot (\sqrt{n} + 1)$$

- Show: $T^{\text{worst}}(n) \leq \frac{5}{4} c\sqrt{n}$ for $n \geq 16$

- Therefore worst-case run-time is $O(\sqrt{n})$.

Analysis of *interpolation-search-improved*

- What is the number of probes on average?
- Rephrase: If numbers are chosen uniformly at random, what is the expected number of probes?
- **Can show:** Expected number of probes is in $O(1)$.
- The average-case run-time satisfies

$$T^{\text{avg}}(n) \leq T^{\text{avg}}(\sqrt{n}) + c$$

- Show: $T^{\text{avg}}(n) \leq c\lceil \log\log n \rceil$ for $n \geq 4$.

- Therefore the average-case run-time is $O(\log\log n)$.