Data Structure

Lecture#20: Searching (Chapter 9)

U Kang
Seoul National University

In This Lecture

- Motivation of searching
- Main idea and cost of jump search, interpolation search, and quadratic binary search
- Using lists ordered by frequency for searching

Searching Ordered Arrays (1)

- If elements are not sorted, than linear search is the best we can do.

- Assume that the elements are in the ascending order.

- Is linear search still optimal? Why not?

Searching Ordered Arrays (2)

- Binary search is better than linear search for searching an ordered array
  - Best method if we have no information on the array

- But we will find a method better than binary search when we have additional knowledges including data distribution and usage patterns

Searching Ordered Arrays (3)

- Idea 1: use linear search, but test if the element is greater than K (= the key we are looking for).

- Observation: If we look at **L**[5] and find that K is bigger, then we rule out **L**[1] to **L**[4] as well.

- More is Better: If K > **L**[n], then we know in one test that K is not in **L**.
  - What is wrong here?

Jump Search

- Jump Search
  - Check every k'th element (**L**[k], **L**[$2k$], ...).
  - If K is greater, then go on.
  - If K is less, then use linear search on the k elements.

- What is the right amount to jump?

Analysis of Jump Search

n: input size
3-way comparison: check ≤, =, and ≥

- If $(m\text{-}1)k < n \le mk$, then the total cost is at most $m + k - 1$ 3-way comparisons.

- Cost $T(n,k) = m + k - 1 = \left\lceil \frac{n}{k} \right\rceil + k - 1$

- What is the best k to choose?

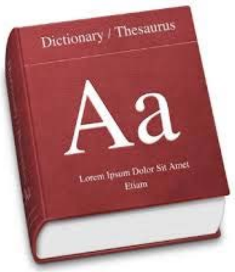
$$min_{1 \le k \le n} \left\lceil \frac{n}{k} \right\rceil + k - 1$$

Jump Search Analysis (cont)

- When is $T(n,k) = \left\lceil \frac{n}{k} \right\rceil + k - 1$ minimized?
- Take the derivative and solve for $\mathbf{T}'(k) = 0$ to find the minimum.
  - The minimum is found when $k = \sqrt{n}$
  - In that case, $T(n,k) \approx 2\sqrt{n}$

Interpolation Search

- (Also known as Dictionary Search)



- Given a sorted array, can we search an item K faster than the binary search?
  - Yes!
  - Search **L** at a position p that is proportional to the value K.

$$p = \frac{K - L[1]}{L[n] - L[1]}$$

  - E.g., if we search for the word ‘yearning’ in a dictionary, we would prefer to start search from near the end, not from the middle
  - Repeat as necessary to recalculate p for future searches.

Quadratic Binary Search (QBS)

- A variation on dictionary search

- Compute p and examine $L[\lceil pn \rceil]$

- If $\mathrm{K} = L[\lceil pn \rceil]$, the search is complete.
- If $\mathrm{K} < L[\lceil pn \rceil]$ then do a jump search: sequentially probe $L[\lceil pn - i\sqrt{n} \rceil]$, i = 1, 2, 3, …

  until we reach a value less than or equal to K.
- Similarly for $\mathrm{K} > L[\lceil pn \rceil]$

Quadratic Binary Search (cont)

- We are now within $\sqrt{n}$ positions of K.
- ASSUME (for now) that the jump search takes a constant number of comparisons.
- We have a sublist of size $\sqrt{n}$.
- Repeat the process recursively.
- What is the cost?

QBS Cost

- QBS cost is $\Theta(\log \log n)$ if the number of probes on jump search is constant.
  - (Proof)
  - The probe gap : $\sqrt{n} \Rightarrow \sqrt{\sqrt{n}} \Rightarrow \sqrt{\sqrt{\sqrt{n}}} \ldots.$
  - In other words, $n^{1/2} \Rightarrow n^{1/4} \Rightarrow n^{1/8} \ldots$
  - Since n = $2^{\log n}$, the probe gap: $2^{logn/2} \Rightarrow 2^{logn/4} \ldots \Rightarrow 2^1$
  - That means we need Θ (log log n) steps, and jump search in each step takes constant time

QBS Probe Count

- We can show that on uniformly distributed data, the average number of probes required is at most 2.4 (constant).
  - Proof: appendix

- Is this better than binary search?
  - Theoretically, yes (in the average case for the uniformly distributed data).

Comparison

n	log n (1)	log log n (2)	Diff
16	4	2	2
256	8	3	2.7
64k	16	4	4
2^{32}	32	5	6.4

Diff $= \frac{(1)}{(2)}$

n	log n	2.4 log log n	Diff
16	3	4.8	worse
256	7	7.2	same
64k	15	9.6	1.6
2^{32}	31	12	2.6

Summary (Until This Point)

- Searching ordered list
  - Jump search: $2\sqrt{n}$
  - Binary search: log n
  - QBS: log log n

- Can we search an item faster than the above methods, if items are arranged in a special way?
  - Yes. Main idea: people search only ‘frequent’ items in most cases (details follow)

Lists Ordered by Frequency

- Order lists by (expected) frequency of occurrence.
  - Perform sequential search
- Cost to access first record: 1
- Cost to access second record: 2
- …
- Expected search cost:

P_i: relative frequency of i th record

$$\overline{C}_n = 1p_1 + 2p_2 + \ldots + np_n$$

Examples (1)

(1) All records have equal frequency.

$$\overline{C}_n = \sum_{i=1}^{n} i / n = (n+1)/2$$

Examples (2)

(2) Geometric Frequency

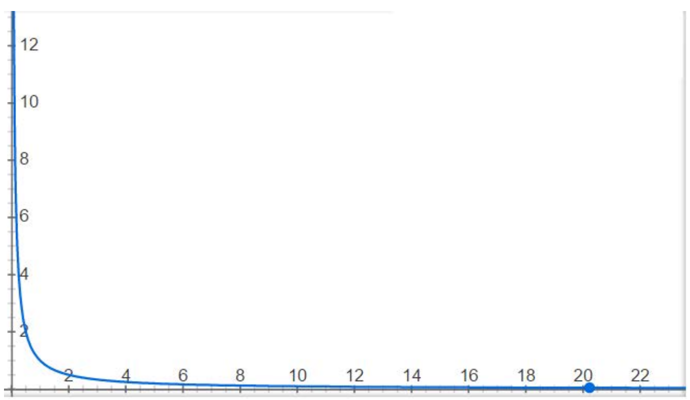
$$p_i = \begin{cases} 1/2^i & \text{if } 1 \le i \le n-1 \\ 1/2^{n-1} & \text{if } i = n \end{cases}$$

$$\overline{C}_n \approx \sum_{i=1}^{n} (i/2^i) \approx 2$$

Zipf Distributions

- Zipf distribution: $P(x) \propto x^{-1}$
  - Distribution for frequency of word usage in natural languages.
  - Distribution for populations of cities, etc.

$$\overline{C_n} = \sum_{i=1}^{n} \frac{i}{iH_n} = \frac{n}{H_n} \approx \frac{n}{\log_e n}$$



Zipf Distributions

- 80/20 rule:
  - 80% of accesses are to 20% of the records.
  - For distributions following 80/20 rule,

$$\overline{C}_n \approx 0.122n$$

What you need to know

- Searching: very important task with many usages

- Main idea of jump search, interpolation search, quadratic binary search
  - QBS: better than binary search when we have knowledges about data distribution

- Main idea of using lists ordered by frequency
  - Better than binary search/QBS in skewed usage patterns

Questions?

Appendix

QBS Probe Proof (1)

Details

- On uniformly distributed data, the average number of probes in a step of QBS is at most 2.4.
  - (Proof)
  - Let P_j = probability of needing exactly j probes
  - Since we need to do at least 2 probes, the average number of probes in a step of QBS is given by
    - $2 + \sum_{j=3}^{\sqrt{n}} (j-2)\, P(need\ exactly\ j\ probes)$
    - $= 2 + [1P_3 + 2P_4 + \cdots + (\sqrt{n}-2)P_{\sqrt{n}}]$
    - $= 2 + (1 - P_1 - P_2) + (1 - P_1 - P_2 - P_3) \ldots + P_{\sqrt{n}}$
    - $= 2 + \sum_{j=3}^{\sqrt{n}} P(need\ at\ least\ j\ probes)$

QBS Probe Proof (2)

Details

- (Proof cont.)
  - Let Q_j = P(need *at least* j probes)
  - Let X be a random variable denoting the number of items in L[1]..L[n] smaller than *K*.
  - The probability that each L[i] is smaller than K is $p = \frac{K-L[1]}{L[n]-L[1]}$, assuming uniform distribution.
  - Since each L[i] is independent, X is a binomial distribution with $\mu = pn$ and $\sigma^2 = p(1-p)n$
  - $Q_j \leq Prob(|X-\mu| \geq (j-2)\sqrt{n}) \leq \frac{\sigma^2}{\left((j-2)\sqrt{n}\right)^2} = \frac{p(1-p)}{(j-2)^2}$

Chebyshev inequality:

$$\boldsymbol{P(|X-\mu| \geq c) \leq \frac{\sigma^2}{c^2}}$$

QBS Probe Proof (3)

Details

- (Proof cont.)
  - Average number of probes in a step of QBS
  - $= 2 + \sum_{j=3}^{\sqrt{n}} P(need\ at\ least\ j\ probes)$
  - $= 2 + \sum_{j=3}^{\sqrt{n}} Q_j$
  - $\leq 2 + \sum_{j=3}^{\sqrt{n}} \frac{p(1-p)}{(j-2)^2}$
  - $\leq 2 + \sum_{j=3}^{\sqrt{n}} \frac{1}{4(j-2)^2}$
  - $< 2 + \frac{1}{4}\sum_{j=1}^{\infty} \frac{1}{j^2} = 2 + \frac{1}{4}\frac{\pi^2}{6} \approx 2.4$