反馈与讨论

6月5日

10-3 Searching a sorted compact list

Exercise 10.3-4 asked how we might maintain an n-element list compactly in the first n positions of an array. We shall assume that all keys are distinct and that the compact list is also sorted, that is, $key[i] < key[next[i]]$ for all $i = 1, 2, \ldots, n$ such that $next[i] \neq \text{NIL}$. We will also assume that we have a variable L that contains the index of the first element on the list. Under these assumptions, you will show that we can use the following randomized algorithm to search the list in $O(\sqrt{n})$ expected time.

```
COMPACT-LIST-SEARCH(L, n, k)
 1  i = L
 2  while i ≠ NIL and key[i] < k
 3        j = RANDOM(1, n)
 4        if key[i] < key[j] and key[j] ≤ k
 5              i = j
 6              if key[i] == k
 7                    return i
 8        i = next[i]
 9  if i == NIL or key[i] > k
10        return NIL
11  else return i
```

```
COMPACT-LIST-SEARCH'(L, n, k, t)
 1  i = L
 2  for q = 1 to t
 3        j = RANDOM(1, n)
 4        if key[i] < key[j] and key[j] ≤ k
 5              i = j
 6              if key[i] == k
 7                    return i
 8  while i ≠ NIL and key[i] < k
 9        i = next[i]
10  if i == NIL or key[i] > k
11        return NIL
12  else return i
```

a. Suppose that COMPACT-LIST-SEARCH(L, n, k) takes t iterations of the **while** loop of lines 2–8. Argue that COMPACT-LIST-SEARCH$'(L, n, k, t)$ returns the same answer and that the total number of iterations of both the **for** and **while** loops within COMPACT-LIST-SEARCH$'$ is at least t.

- If the first version COMPACT-LIST-SEARCH found the element on a random skip-ahead in *t* iterations,so will the second version. If not, the last *k* iterations only advanced the pointer until the result was found. Since the second version does not advance in between skip-aheads, it has to perform *k* additional iterations of its **while** loop until the result is found.

b. Argue that the expected running time of COMPACT-LIST-SEARCH$'(L, n, k, t)$ is $O(t + \mathrm{E}[X_t])$.

- Since it either finds the element in *t* skip-aheads, or it has to move forward a number of times, equal to the distance to *Xt*. So the analysis holds.

c. Show that $\mathrm{E}[X_t] \leq \sum_{r=1}^{n} (1 - r/n)^t$. (*Hint:* Use equation (C.25).)

When a random variable X takes on values from the set of natural numbers $\mathbb{N} = \{0, 1, 2, \ldots\}$, we have a nice formula for its expectation:

$$\begin{aligned}
\mathrm{E}[X] &= \sum_{i=0}^{\infty} i \cdot \Pr\{X = i\} \\
&= \sum_{i=0}^{\infty} i (\Pr\{X \geq i\} - \Pr\{X \geq i + 1\}) \\
&= \sum_{i=1}^{\infty} \Pr\{X \geq i\} \ , \qquad \text{(C.25)}
\end{aligned}$$

since each term $\Pr\{X \geq i\}$ is added in i times and subtracted out $i - 1$ times (except $\Pr\{X \geq 0\}$, which is added in 0 times and not subtracted out at all).

$$\Pr\{X_t \geq r\} = \left(\frac{n-r}{n}\right)^t = \left(1-\frac{r}{n}\right)^t$$

$$\mathrm{E}[X_t] = \sum_{r=1}^{\infty} \Pr\{X_t \geq r\} = \sum_{r=1}^{n} \Pr\{X_t \geq r\} = \sum_{r=1}^{n} \left(1-\frac{r}{n}\right)^t$$

d. Show that $\sum_{r=0}^{n-1} r^t \le n^{t+1}/(t+1)$.

Approximation by integrals

When a summation has the form $\sum_{k=m}^{n} f(k)$, where $f(k)$ is a monotonically increasing function, we can approximate it by integrals:

$$\int_{m-1}^{n} f(x)\,dx \le \sum_{k=m}^{n} f(k) \le \int_{m}^{n+1} f(x)\,dx\ . \tag{A.11}$$

We can show (d) by approximating the sum with an integral with (A.11):

$$\sum_{r=0}^{n-1} r^t \le \int_0^n x^t\,dx = \frac{n^{t+1}}{t+1}$$

e. Prove that $\mathrm{E}[X_t] \leq n/(t+1)$.

$$
\begin{aligned}
\mathrm{E}[X_t] &= \sum_{r=1}^{n} \left(1 - \frac{r}{n}\right)^t \\
&= \sum_{r=0}^{n-1} \left(\frac{r}{n}\right)^t \\
&= \frac{1}{n^t} \sum_{r=0}^{n-1} r^t \\
&\leq \frac{1}{n^t} \cdot \frac{n^{t+1}}{t+1} \\
&= \frac{n}{t+1}
\end{aligned}
$$

f. Show that COMPACT-LIST-SEARCH$'(L, n, k, t)$ runs in $O(t + n/t)$ expected time.

$$\mathcal{O}(t + \mathrm{E}[X_t]) = \mathcal{O}(t + n/(t+1)) = \mathcal{O}(t + n/t)$$

g. Conclude that COMPACT-LIST-SEARCH runs in $O(\sqrt{n})$ expected time.

- Since COMPACT-LIST-SEARCH minimizes the running time, we need to find the minimum of $t+n/t$. The first derivative is $1 - n/t^2$ which is zero at $\sqrt{n}$ and this is a local minimum. It's also the minimum in the interval [1,n].
- This makes the expected running time of the first version of the algorithm $\mathcal{O}(\sqrt{n})$

h. Why do we assume that all keys are distinct in COMPACT-LIST-SEARCH? Argue that random skips do not necessarily help asymptotically when the list contains repeated key values.

- As for duplicates, we won't be able to conclude (c) if there are duplicates. The algorithm is able to skip ahead only if the value found by RANDOM is greater than the current.

  if $key[i] < key[j]$ and $key[j] \leq k$

- For example, if we have a list of 0s and we're looking for a 1, the algorithm will still need to iterate to the end of the list, since it will not skip-ahead at all.