Parallel And Sequential Data Structures and Algorithms

Randomized Algorithms I (Quickselect)

Learning Objectives

- Learn how to apply expectation and **high probability bounds** to analyze work/span of randomized algorithms
- Understand the **Order Statistics** problem and develop an efficient randomized algorithm for it
- Analyze the expected work and span of **QuickSelect** via probability tools developed in earlier lectures

Analyzing Algorithms

Skittles Game

Problem (Skittles Game): The Skittles game is played with a fair coin and a pile of n Skittles. It is a single-player game played in rounds. Initially there are $s = n$ Skittles. Each round consists of the player flipping the coin once. If it comes up heads, then the player eats $s/2$ (rounded up) of the Skittles and there are $s/2$ (rounded down) remaining. If it comes up tails, the player proceeds to the next round without eating any Skittles. The game ends when there are no Skittles remaining.

We are interested in the random variable $R(n)$ which is the number of rounds the game lasts.

Skittles Game Bound Proof

Theorem (Skittles Game Bound): The Skittles game will end in $5\log_2 n$ rounds with high probability. In other words, $R(n) \in O(\log n)$ w.h.p.

First, let's modify the game slightly:

- Let the number of skittles we eat when we get heads to be $s/2$ rather than $\lceil s/2 \rceil$.
- The game ends when the number of skittles we have goes below 1.

It turns out that the games (using the same coinflips) end in the same number of rounds.

(You can prove by induction that $y_r \leq x_r < 1 + y_r$, where $(y_r, x_r) =$ the number of skittles at round r of the original game and the modified game, respectively.)

Skittles Game Bound Proof

Theorem (Skittles Game Bound): The Skittles game will end in $5\log_2 n$ rounds with high probability. In other words, $R(n) \in O(\log n)$ w.h.p.

Proof: Let $X_r =$ (a RV) be the number of Skittles remaining after r rounds.
Let $s_r =$ the actual number that occur in a specific run of the algorithm.

- On any round, we have a $1/2$ probability of flipping heads, which reduces the size by $1/2$. Otherwise, the size is unchanged

- Thus, $X_{r+1} = \begin{cases} \frac{1}{2}s_r & \text{prob. } \frac{1}{2} \\ s_r & \text{prob. } \frac{1}{2} \end{cases} \Rightarrow \mathbb{E}[X_{r+1} | X_r = s_r] = \frac{1}{2}\left(\frac{1}{2}s_r\right) + \frac{1}{2}s_r = \frac{3}{4}s_r$

Expecting the Expected

$$X_{r+1} = \begin{cases} \frac{1}{2}s_r & \text{prob. } \frac{1}{2} \\ s_r & \text{prob. } \frac{1}{2} \end{cases} \implies \mathbb{E}[X_{r+1}|X_r = s_r] = \frac{3}{4}s_r \implies \mathbb{E}[X_{r+1}] = \frac{3}{4}\mathbb{E}[X_r]$$

This step depends on something called **The Law of Total Expectation**, which says this: For any two random variables Y and X the following holds:

$$\mathbb{E}[Y] = \sum_x \mathbb{E}[Y|X = x] \Pr(X = x)$$

The notation $\mathbb{E}[Y|X = x]$ refers to conditional expectation... It's the expectation of Y given that $X = x$. You don't need to understand this for this class.

Skittles Game Bound Proof (Cont.)

Theorem (Skittles Game Bound): The Skittles game will end in $5\log_2 n$ rounds with high probability. In other words, $R(n) \in O(\log n)$ w.h.p.

Proof:

- We now unroll the recurrence to get $\mathbb{E}[X_r] = n\left(\frac{3}{4}\right)^r$

- Additionally, $\Pr[R(n) > r] = \Pr[X_r \geq 1] \leq \frac{\mathbb{E}[X_r]}{1} = n\left(\frac{3}{4}\right)^r$

Markov's Inequality

Skittles Game Bound Proof (Cont.)

Theorem (Skittles Game Bound): The Skittles game will end in $5\log_2 n$ rounds with high probability. In other words, $R(n) \in O(\log n)$ w.h.p.

Proof:

- To match this bound with the definition of high probability, we **want to find a value of r** that satisfies $n\left(\frac{3}{4}\right)^r < \frac{1}{n^k}$

- Taking logs, and rearranging gives us

$$r > (k+1)\log_{\frac{4}{3}} n = (k+1)\frac{\log_2 n}{\log_2(4/3)} \approx (k+1)2.409\log_2 n$$

Skittles Game Bound Proof (Cont.)

Theorem (Skittles Game Bound): The Skittles game will end in $5\log_2 n$ rounds with high probability. In other words, $R(n) \in O(\log n)$ w.h.p.

Proof:

Since $2k \geq k + 1$ and $5 \geq 2 \cdot 2.409$, we can choose $r = k \cdot 5 \log_2 n$.

Hence, $\Pr[R(n) > k \cdot 5 \log_2 n] < \frac{1}{n^k}$ so

$R(n) \leq 5 \, \mathrm{k} \log_2 n$ w.h.p. and $R(n) \in O(\log n)$ w.h.p. ■

Skittles Game Lemma

Lemma (Skittles Lemma): Let $X_0 = n$ and let X_{r+1} be computed by a random process from X_r that guarantees that:

- $0 \leq X_{r+1} \leq X_r$,
- $\mathbb{E}[X_{r+1}] \leq \alpha \mathbb{E}[X_r]$ where $0 < \alpha < 1$ is a constant.

Let $R(n)$ be a random variable which is the first round r when $X_r < 1$. Then $R(n)$ is $O(\log n)$ w.h.p.

(Specifically, $R(n) \leq \beta \log_2 n$ w.h.p., where $\beta = -2/\log_2 \alpha$.)

Order Statistics

Order Statistics

Definition (Rank): The **rank-k** element of a sequence is the kth element when the sequence is sorted in ascending order (zero-indexing). For example, 7 is the rank-5 element of `[7,1,1,9,5,6,7,6]`.

Problem (Order Statistics): The **order statistics** problem takes a sequence of integers S and an integer k as input. Given these, our algorithm is supposed to return the rank-k element of S.

Inefficient Approach

- We could sort the sequence and return the k^{th} element

```
fun rank_k(S : sequence<T>, k : int) -> T:
   return sort(S)[k]
```

- This costs $O(|S| \log |S|)$ work and $O(\log^2 |S|)$ span
- We sort the entire sequence just to get a single value out of it!

How can we make this faster...?

Randomization!

QuickSelect

QuickSelect (S, k)

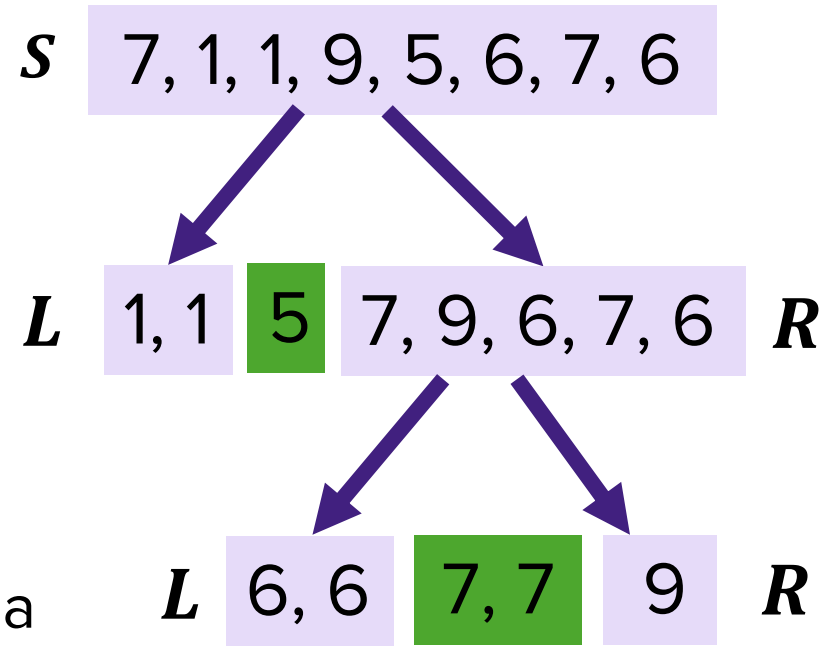
- Pick a uniformly random pivot p in S
- Split S into two sequences (L, R) around p
  - L contains all values smaller than p, R contains all values larger
- Recurse on the side that contains the element we want
  - If $\mathrm{k} < |L|$, the element we want must be in L
  - If $\mathrm{k} \geq |\mathrm{S}| - |\mathrm{R}|$, the element we want must be in R
  - Otherwise, our pivot is the element we want

QuickSelect ([7,1,1,9,5,6,7,6], 5)

Looking for rank 5 in S

Looking for rank 2 in R

Returns rank 0 in the pivot area



QuickSelect

```
fun quickselect(S : sequence<T>, k : int) -> T:
  p = a uniformly random element of S
  L = filter(fn x => (x < p), S)
  R = filter(fn x => (x > p), S)
  if k < |L|: return quickselect(L, k)
  else if k >= |S|-|R|: return quickselect(R, k - (|S|-|R|))
  else: return p
```

Analysis of QuickSelect

Theorem (QuickSelect Recursion Depth): On a sequence of length n, `quickselect` terminates in $5 \log_2 n$ recursive calls with high probability.

Proof:

- Let $X_r =$ length of array on which `quickselect` is being called
  - Initially $r = 0$ and $X_0 = n$
- The algorithm may terminate at this round, but if it does not, then we let X_{r+1} be the size of the recursive call
- Note that L and R from the partitioning step have fewer elements than S because the pivot element is in S but not in L or R. Thus, $X_{r+1} < X_r$

Analysis of QuickSelect

Theorem (QuickSelect Recursion Depth): On a sequence of length n, `quickselect` terminates in $5 \log_2 n$ recursive calls with high probability.

Proof:

- In the worst case, the rank k will be such that the algorithm always does the recursive call into the larger part.
- Let $n' = X_r$, the current size of the array
  - If n' is even then the size of X_{r+1} is chosen uniformly from among $[n'/2, n' - 1]$. The average value of these is $3/4n' - 1/2$
  - If n' is odd then the size of X_{r+1} is chosen uniformly from among $[(n' - 1)/2, n' - 1]$. The average value of these is $3/4(n' - 1)$
- Thus, $\mathbb{E}[X_{r+1}|X_r] < \frac{3}{4}X_r \implies \mathbb{E}[X_{r+1}] < \frac{3}{4}\mathbb{E}[X_r]$.

Analysis of QuickSelect

Theorem (QuickSelect Recursion Depth): On a sequence of length n, `quickselect` terminates in $5 \log_2 n$ recursive calls with high probability.

Proof:

- To summarize: $\mathbb{E}[X_{r+1}] < \frac{3}{4}\mathbb{E}[X_r]$, and $X_{r+1} < X_r$
- So, the Skittles Lemma applies and gives the desired result. ■

Analysis of QuickSelect

Theorem (Cost of QuickSelect): The expected work of `quickselect` on a sequence of length n is $O(n)$. The span is $O(\log^2 n)$ with high probability.

Proof:

- On each round of `quickselect`, we do $O(n)$ work where n is the size of the input sequence for that round
  - Picking a uniformly random element takes constant time
  - Partitioning the sequence requires two filter operations which each take $O(n)$ work
- We can therefore find the expected work of the overall algorithm via the following recurrence

$$W(n) = W(3/4\, n) + O(n) \quad \Longrightarrow \quad O(n)$$

Analysis of QuickSelect

Theorem (Cost of QuickSelect): The expected work of `quickselect` on a sequence of length n is $O(n)$. The span is $O(\log^2 n)$ with high probability.

Proof:

- On each round of `quickselect`, we do $O(\log |S|) \leq O(\log n)$ span where S is the input sequence for that round
  - Picking a uniformly random element takes constant time
  - The filter operations have logarithmic span
- The recursion depth is $O(\log n)$ with high probability, therefore

$$S(n) = O(\log n \cdot \log n) \implies O(\log^2 n)$$

Summary

- The Skittles game gives us an example of how to derive/prove **high probability bounds**
  - Generalizable into the Skittles Lemma
- `QuickSelect` give us an expected $O(n)$ work and expected $O(\log^2 n)$ span algorithm to solving the **Order Statistics** problem
- In general, randomized algorithms can give us solutions that work more **efficiently in expectation** than a corresponding deterministic algorithm