Today we will look at three linear-time algorithms for the **selection** problem, where we are given a list of n items and a number k and are asked for the k'th smallest item in a particular ordering. All three are **comparison-based** algorithms, in that the only operation allowed on the items is to see which of two comes first in the ordering. Selection is also called **median-finding** because the hardest case comes when $k = n/2$ and we are asked for the middle element in the ordering.

Of course sorting the elements gives a solution to the selection problem in $O(n \log n)$ time, but we hope to do better. It's easy to show that if $k = 1$ or $k = n$, then $n-1$ comparisons are necessary and sufficient. Two classic exercises extend this result – in each case the upper bound is easy and the lower bound challenging:

- If n is even, $3n/2 - 2$ comparisons are necessary and sufficient to solve *both* the $k = 1$ and $k = n$ problems.
- If $n = 2^k$, then $n+\log n-2$ comparisons are necessary and sufficient to solve the $k = 2$ problem. (Solving $k = 2$ means that you also know the answer to $k = 1$).

Our first selection algorithm is a simple adaptation of Quicksort, which we'll call **Quickselect**. Given the n elements, we choose a pivot uniformly at random, split the elements into those greater than and those less than the pivot, and recurse on *one of* the two sets. Which set of course depends on k and on the size of the sets – if there are t elements less than the pivot, we recurse on the smaller-than set if $k \leq t$, simply return the pivot if $k = t+1$, and recurse on the greater-than set if $k > t+1$.

If we are very unlucky in our pivots, we could wind up making $\binom{n}{2}$ comparisons (for example, suppose $k = 1$ and our random pivot is always the greatest remaining element). But on the average we will do much better than that – let's compute an upper bound on the average-case number of comparisons.

A nice simplifying assumption is to assume that k always winds up in the larger of the two sets. This is a conservative assumption, because we will always do better if k is in the smaller set or if it turns out to be the pivot. And since we will get a linear upper bound and we know there is a linear lower bound on the number of comparisons, the assumption isn't too conservative.

If we split the input using a uniformly chosen random pivot, what is the distribution of the size of the larger set? There are two choices, 1 and n, that make the larger set size $n-1$. There are two choices, 2 and $n-1$, that make it size $n-2$, two that make it $n-3$, and so forth until finally there is one choice making it $(n-1)/2$ (if n is odd) or two making it $n/2$ (if n is even).

This gives us a recurrence for the expected time:

$$\bar{T}(n) = \frac{2}{n} \sum_{i=n/2}^{n-1} \bar{T}(i) + O(n).$$

This is the recurrence for even n – the one for odd n is similar.

It's not hard to prove $\bar{T}(n) = O(n)$ from this recurrence. If we write the final $O(n)$ as "dn", we can look for a constant c such that we can prove $\bar{T}(n) \leq cn$:

$$\begin{aligned}
\bar{T}(n) &\leq \frac{2}{n} \sum_{i=n/2}^{n-1} \bar{T}(i) + dn \\
&\leq \frac{2}{n} \sum_{i=n/2}^{n-1} ci + dn \\
&\leq \frac{2}{n}(n/2)c(3n/4) + dn \\
&= (3c/4 + d)n
\end{aligned}$$

and making $c = 4d$ allows us to complete an inductive proof.

Measuring in terms of the number of comparisons, "d" above will be about 1, giving us an upper bound of about $4n$ on the number of comparisons. Accounting for the possibility of k falling in the smaller of the two sets would presumably give us a somewhat smaller constant.

It's a natural theoretical question whether selection can be carried out in *deterministic* linear time. We'll present the first solution, from 1973, not because it is of any practical use but because it gives an interesting analysis.

The algorithm proceeds as follows:

- Divide the n items into groups of five.
- Sort each group and find *its* median.
- Recursively find the median of these medians, m
- Using m as a pivot as in Quickselect, divide the set in two.
- Recursively select the element we want from one of these two sets.

The non-recursive parts of this algorithm pretty clearly take linear time. The interesting analysis will be of the recursive parts.

- Divide the n items into groups of five.
- Sort each group and find *its* median.
- Recursively find the median of these medians, m
- Using m as a pivot as in Quickselect, divide the set in two.
- Recursively select the element we want from one of these two sets.

If $T(n)$ is the worst-case time to select from n elements for any k, it will take us only $T(n/5)$ time to select the median of the set-of-five medians once we have them all.

The key point here is that this element m is guaranteed to be a pretty good pivot. Let's look at this carefully. About half the elements are in groups of five whose median is less than m. At least *three* element in each of these groups, then, the median and the two smaller ones, are smaller than m – a total of 30% of the elements. Similarly, half the elements are in groups whose median is greater than m, and at least three-fifths of these (30% of the total) are greater than m.

- Divide the n items into groups of five.
- Sort each group and find *its* median.
- Recursively find the median of these medians, m
- Using m as a pivot as in Quickselect, divide the set in two.
- Recursively select the element we want from one of these two sets.

The last recursive call to Quickselect, then, applies to a set whose size is at most $0.7n$ and thus takes at most $T(0.7n)$ comparisons. Thus we have the following recurrence:

$$T(n) \leq T(0.2n) + T(0.7n) + O(n),$$

which has solution $T(n) = O(n)$. More specifically, if the final $O(n)$ of the recurrence is replaced by cn, we can prove by induction that $T(n) = 10cn$ is a solution to the recurrence.

The constant from this recurrence is actually quite bad, because "c" in the discussion above is about 3 (the sorts of five take about $2n$ and the split about n) and the final answer is $10cn$. But there have since been more complicated deterministic algorithms with $3n$ comparisons.

Our third median algorithm, called **LazySelect** by Motwani and Raghavan, actually has an average case performance of $1.5n + o(n)$ comparisons. It has been proved that no deterministic algorithm can succeed with fewer than $2n$ comparisons in the worst case, so here is a case where a randomized algorithm is provably better.

LazySelect finds the k'th smallest element from a totally ordered set S, of size n, as follows:

- Choose $n^{3/4}$ elements uniformly and independently, with replacement, from S.
- Sort this multiset R completely.
- Let $t = kn^{-1/4}$. Let a be the $t - \sqrt{n}$'th smallest element of R, and let b be the $t + \sqrt{n}$'th smallest.
- Compare each element against a and/or b to find the elements that fall between them, and how many fall before a and after b,
- If the k'th smallest element is guaranteed to fall between a and b, and there are no more than $4n^{3/4}$ elements between them, sort those elements and find the target.
- If the conditions above are not true, repeat the entire algorithm with independent choices.

First let's show that the expected number of comparisons used by this algorithm is $1.5n + o(n)$ in the case that it doesn't give up and repeat. The two sorts of $O(n^{3/4})$ elements, using our favorite deterministic $O(n \log n)$ algorithm, each take $O(n^{3/4} \log n)$ which is $o(n)$. If we compare each element of S to both a and b, this is $2n$ comparisons. But we know that either nearly half the elements are less than a or nearly half are greater than b, and in the no-repeat case we can tell which, based on k. Suppose a and b are both in the lower half of S – we compare each element against b first, and against a only if it is less than a. This will sort the elements into three groups with $1.5n + o(n)$ comparisons.

If the algorithm does not repeat, it is correct. All we need to do to show the randomized algorithm correct, then, is to show that it has at least a constant probability of success. Then, with $O(1)$ expected repetitions, we will get a confirmed correct answer. In fact we'll show that the probability of failure on a single run of the algorithm is much smaller, only $O(n^{1/4})$.

When we pick elements for R, each has a probability of k/n of being less than or equal to our target element, the k'th smallest in S. Thus the expected number of such elements in R is $n^{3/4}(k/n) = kn^{-1/4} = t$. The only way that the target will *fail* to be between a and b is if this number is smaller than $t - \sqrt{n}$ or greater than $t + \sqrt{n}$. We need to know how probable this might be.

The other way the algorithm could fail is if there are too many elements between a and b. But the only way this could happen is either the number less than a, or the number less than b, is sufficiently far from the expected number. We'll put an upper bound on this probability as well.

A **binomial** random variable is 1 with some probability p and 0 otherwise. We noted before that its mean is p. It's distance from the mean is thus $1 - p$ with probability p and p with probability $1-p$, so its variance (the expected value of $(X - \mu)^2$) is $p(1-p)^2 + (1-p)p^2 = p(1-p)$, a constant depending on p that is maximized at $1/4$ when $p = 1/2$.

If we have n **independent, identically distributed (i.i.d.)** binomial random variables, each with probability p, it is clear that their sum, as a random variable, has mean pn. Because variances of independent random variables add, the variance of the sum is $p(1-p)n$, at most $n/4$. (In fact pairwise independence of a set of random variables is sufficient to imply that their variances add.)

Let's apply this to the median algorithm. The probability that one element of S, chosen uniformly at random, is less than or equal to the k'th element is $p = k/n$. Our R is made from $n^{3/4}$ independent choices of this kind. The number of elements of R that are less than or equal to the k'th element of S is thus a random variable with mean $pn^{3/4} = kn^{-1/4} = t$ and variance $p(1-p)n^{3/4} \leq n^{3/4}/4$.

Recall that in the last lecture we proved:

Markov's Inequality: If X is a non-negative random variable with mean μ, the probability that $X \geq \alpha$ is at most μ/α.

Chebyshev's Inequality: If X is a random variable with mean μ and variance σ^2, then the probability that $|X - \mu| \geq k\sigma$ is at most $1/k^2$.

So what is the chance that the number of elements in R less than or equal to the k'th element of S, which is expected to be t and has variance at most $n^{3/4}/4$, is more than $\sqrt{n}$ away from t? The standard deviation of the random variable is $n^{3/8}/2$, and by Chebyshev the chance of being more than $2n^{1/8}$ standard deviations away from the mean is at most $1/(2n^{1/8})^2 = n^{-1/4}/4 = O(n^{-1/4})$.

The target element of S will be between a and b unless this happens, so the only other way the run of the algorithm could fail is if there are too many elements of S between a and b – more than twice the expected number of $2n^{3/4}$. We can bound this probability by a similar argument to that above.

For example, let u be element number $k-2n^{3/4}$ of S, and let v be element number $k+2n^{3/4}$. The range between a and b can be too big only if $a < u$ or $v < b$. We can bound either of these probabilities. For $a < u$ to be true, the number of elements of R that are less then u must be less than $t - \sqrt{n}$. But this number is expected to be $t - 2\sqrt{n}$, and as we have seen the chance that it is $\sqrt{n}$ away from its mean is $O(n^{1/4})$. Similarly there is only an $O(n^{1/4})$ chance that b is greater than v.

The event that a single run of the algorithm fails to find the target element, then, is the union of three events each of which has probability $O(n^{-1/4})$ and thus has probability $O(n^{-1/4})$ itself. The expected number of runs until we find a solution is $1 + O(n^{-1/4})$, and thus the expected total number of comparisons, like the expected number for one run, is $1.5n + o(n)$. As we mentioned above, this is *faster* than a proven lower bound on deterministic algorithms for selection.

We conclude this lecture with a look at another way to show lower bounds on the probability that *certain* random variables are far away from their mean. The **Chernoff bounds**, as we state them here, apply to binomial random variables (sums of i.i.d. 0-1 variables with probability p of being 1).

As we said, the expected value of the sum of n trials with probability p is np. The Chernoff bound says that the probability that this sum is a constant fraction of the mean away from the mean. Specifically, for any positive δ the probability of being greater than $(1+\delta)np$ is:

$$\left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right)^{np}.$$

If p and δ stay constant, this probability goes down exponentially with n. (Remember that e^x is about $1+x$, making the term inside the parentheses less than 1.)

We won't prove this bound here – you can look in Motwani-Raghavan if you're interested. There are two variants of the Chernoff bound that are particularly useful if $\delta \leq 1$:

- The probability of being at most $(1 - \delta)np$ is at most $e^{-\delta^2 np/2}$.
- The probability of being at most $(1 + \delta)np$ is at most $e^{-\delta^2 np/3}$.

Let's use this to analyze the chance that a baseball team's season record will be significantly different from what the team's true ability would predict. The Kansas City Royals were the worst team in major league baseball in 2005, winning only 56 out of 162 games, about 35%. Let's suppose that each of the Royals' games was an independent binomial trial, with probability $p = 0.35$ of winning. What is the probability that the Royals would win at least *half* their 162 games?

Markov's Inequality tells us that the chance of winning $10/7$ as many games as expected can be no more than $7/10$, a not very useful bound. Since the variance of n such trials is $n(0.35)(0.65)$ or about $0.22n$, the standard deviation is about $0.47\sqrt{n}$, or about 6 with $n = 162$. The winning season would require the sum to be about $162(0.5 - 0.35)/6$ or about 4 standard deviations higher than the mean. Chebyshev's Inequality tells us that the chance of this is at most $1/16$ or about 6%.

What does the Chernoff bound tell us here? We want to bound the probability that the sum is at least $10/7$ times the mean, or $1 + \delta$ times the mean with $\delta = 3/7$. The last version of the bound says that this probability is at most $e^{-\delta^2 np/3}$, or $e^{-(3/7)^2(162)(0.35)/3}$ or about $e^{-3.47}$, which is about 3%.

In this example the Chernoff bound was only twice as strong as the Chebyshev bounds. But this is highly dependent on the constants involved. In the Adler notes he does the same calculation with a less realistic $p = 0.25$ and gets about 2% for Chebyshev and about 10^{-6} for Chernoff.

We should briefly mention the **normal approximation to the binomial**. By a general result called the **Central Limit Theorem**, any sum of n i.i.d random variables becomes more and more closely approximated by a **normal** random variable as n increases. The normal distribution with a particular mean and variance has the familiar bell shape. You can look up the probability, for any positive k, that a normal random variable will be more than k standard deviations away from its mean. (For example, with $k = 2$ it is about 95%.) As k increases, this probability decays exponentially with k.

In our baseball example, the chance of a normal random variable being more than 3.47 standard deviations larger than its mean is about 0.025% or 1 in 4000.

Our final example is the **coupon collector's problem**. As in the last lecture, we have m balls each tossed independently and uniformly into n bins. Here we want to know how large m must be so that with high probability, each bin has at least one ball.

Let X_i be the number of balls in bin i. Each X_i is a binomial random variable with $p = 1/n$ and m trials. The expected number of balls is $mp = m/n$, and we want to bound the probability of there being at most 0 balls in the bin. This can be analyzed with the second version of the Chernoff bound with $\delta = 1$:

$$\Pr(X_i \leq (1-\delta)mp) \leq e^{-\delta^2 mp/2}.$$

This simplifies to $e^{-mp/2} = e^{-m/2n}$. If we let m be $4n \ln n$, then this becomes $e^{-2\ln n} = \frac{1}{n^2}$. This is the probability that bin i is empty after m balls. The probability that at least one of the n bins is empty is no more than n times this, or $1/n$.

So $m = 4n \ln n$ guarantees only a $1/n$ chance of an empty bin. A more sophisticated analysis shows that with $m = n \ln n + o(n \ln n)$, there is a small chance of an empty bin, and with $m = n \ln n - o(n \ln n)$ it is very likely that there will be an empty bin.