lecture 20

randomized algorithms

expected run time analysis

- quick select

  https://class.coursera.org/algo-004/lecture/37 ← Roughgarden

- quick sort

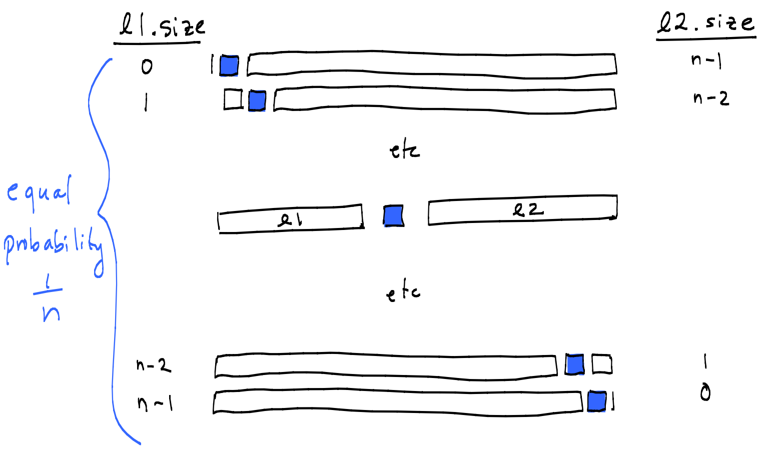
  (Kleinberg & Tardos)

Recall selection problem from lecture 18

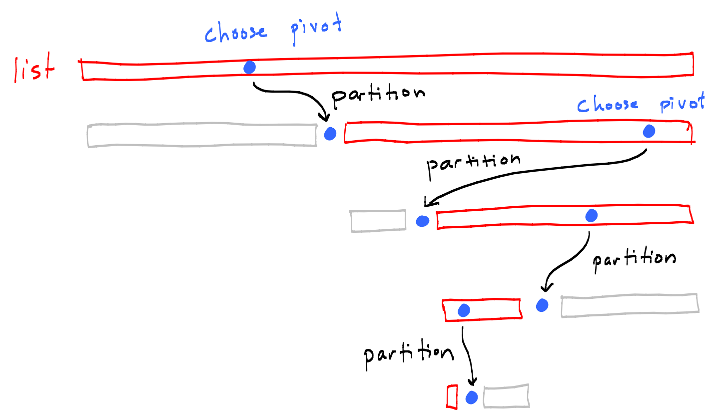
```
select (list, i) {
   choose pivot
   partition the list around pivot
   l1 is the list of elements < pivot
   l2 is   "    "   "    "    > pivot
   if ( i == l1.size )
      return pivot
   else if ( i < l1.size)
      return select ( l1, i )
   else
      return select ( l2, i -(l1.size + 1))
}
```

What if we choose pivot p uniformly at random i.e. equal probability for each list position?

Suppose the list has n elements.



Consider recursive calls select(list , i)



We look at how list.size decreases.

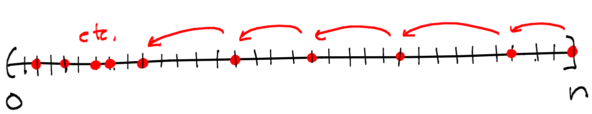
- worst case :

  list.size decreases by 1

- best case : (assuming ith element not found)

  list.size decreases to 1

If we choose the pivot randomly, then list.size will decrease randomly.



Q: What is the probability that list.size is decreased by some factor

e.g. list.size ← list.size * $\frac{3}{4}$

on any call of select (list, i)?

There are two issues here:

1. whether a good pivot is chosen

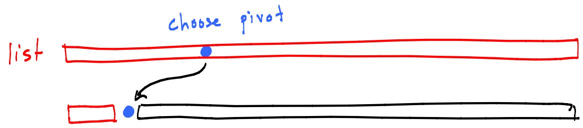
   i.e. how balanced is the l1, l2 split.

2. whether the i-th element is in l1 or l2

It is relatively difficult to disentangle these.

For example, it is possible to choose a bad pivot ($\ell 1$ and $\ell 2$ are unbalanced) and yet still shrink list.size by alot.

e.g. Select (list , 0)



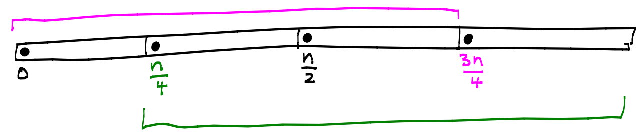
We got lucky here ($i = 0$).

To simplify the probability calculation, we pose a slightly different question.

Q: What is the probability that $\ell 1$.size and $\ell 2$.size are both less than $\frac{3}{4}n$?

A:

$\ell 1.\text{size} < \frac{3}{4}n \iff$ pivot position $< \frac{3}{4}n$

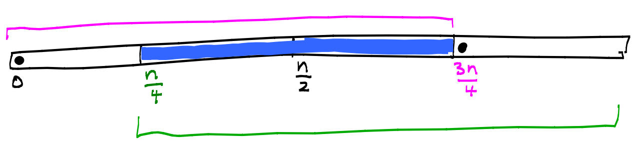


$\ell 2.\text{size} < \frac{3n}{4} \iff \frac{n}{4} \leq$ pivot position

Both of these conditions are met

$\iff \frac{n}{4} \leq$ pivot position $< \frac{3}{4}n$

We call this a good choice of pivot.



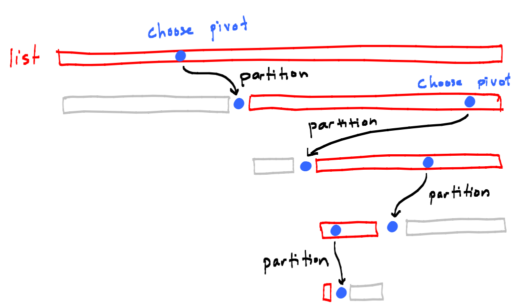
This condition holds with probability $\frac{1}{2}$

(which is why we chose $\frac{3}{4}$ as the factor).

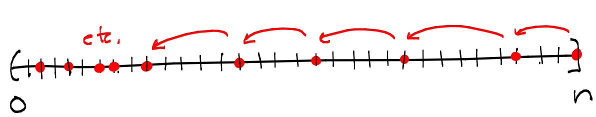
Summary so far:

With probability $p > \frac{1}{2}$, a randomly selected pivot will result in the list in the next recursive select call being at most $\frac{3}{4}$ as large as the current list.

(Why $p > \frac{1}{2}$ and not $p = \frac{1}{2}$? Because a bad pivot might still yield a small list in the next select call.)

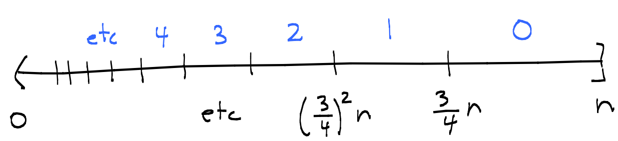


What more can we say about the decrease in list.size ?



Partition the real line interval $(0, n]$ into disjoint intervals:

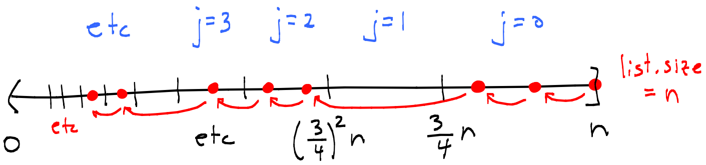
$$\text{interval } j \equiv \left(\left(\tfrac{3}{4}\right)^{j+1} n \, , \left(\tfrac{3}{4}\right)^{j} n \right]$$



ASIDE: We only need $j \leq \log_{\frac{4}{3}} n$, since we will only care about interval sizes ≥ 1.

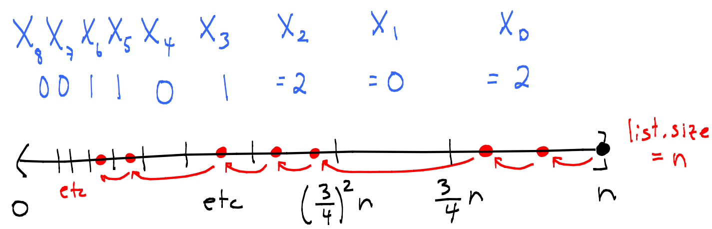
We say the algorithm is in phase j when it makes a select(list, i) call with list.size in interval j.

i.e. it starts in phase $j = 0$.



Let random variable X_j be the number of times that select is called recursively when list.size in interval j.

i.e. algorithm is in phase j.



Let $t(n)$ be the time taken by select for randomly chosen pivots.

$$t(n) \le \sum_{j=0}^{\infty} X_j \cdot \underbrace{c \left(\frac{3}{4}\right)^j n}_{\text{time needed to partition a list of size } \left(\frac{3}{4}\right)^j n}$$

overkill

Now take expected values and use linearity of expectation.

$$\mathcal{E}\, t(n) \le \sum_{j=0}^{\infty} \underbrace{\mathcal{E}\, X_j}_{\text{how to calculate?}} \cdot c \left(\frac{3}{4}\right)^j$$

With probability $p > \frac{1}{2}$, a randomly selected pivot will result in the next select call being on a list that is at most $\frac{3}{4}$ as large,

and hence in a different phase

i.e. list.size in a different interval. !

$\mathcal{E}\, X_j$ = expected number of recursive select calls in phase j

$<$ expected number of times you flip a coin $(p = \frac{1}{2})$ until you get heads

i.e. heads $\Rightarrow$ jump to next phase

$= 2$ (from last lecture)

$$\mathcal{E}\; t(n) \le \sum_{j=0}^{\infty} \mathcal{E}\, X_j \cdot c\left(\tfrac{3}{4}\right)^j n$$

$$< 2cn \sum_{j=0}^{\infty} \left(\tfrac{3}{4}\right)^j$$

$$= 2cn \frac{1}{1-\frac{3}{4}}$$

$$= 8cn$$

Thus, $\mathcal{E}\, t(n)$ is $O(n)$.

Summary of Main Idea

- Each recursive call to select shrinks the list by a constant factor ($\frac{1}{4}$) with probability $p > \frac{1}{2}$. Thus, the expected number of recursive calls to shrink by that constant factor is < 2.
- Partitioning a list takes linear time i.e. $c *$ list. size
- Thus, expected time is at most $2cn(1 + r + r^2 + \ldots)$ where $r = \frac{3}{4}$, which is $O(n)$.

lecture 20

randomized algorithms

expected run time analysis

- quick select

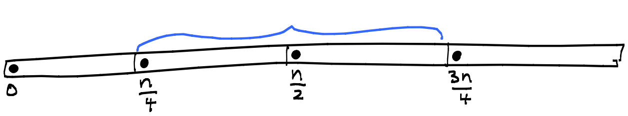
https://class.coursera.org/algo-004/lecture/37

- quick sort
  (I used Kleinberg & Tardos)

```
quicksort (list) {
  if (list. size() <= 1)
     return list
  else { choose random pivot (uniform)
         l1 ← elements less than pivot
         l2 ← elements greater than pivot
         l1 ← quicksort (l1)
         l2 ← quicksort (l2)
         return concatenate (l1, pivot , l2)
       }
}
```

To simplify the analysis, lets suppose we make recursive calls only when have found a good pivot, namely l1. size and l2. size are both at least $n/4$ (or equivalently, both at most $\frac{3n}{4}$).

As we saw earlier, a good pivot is chosen with probability $p = \frac{1}{2}$.

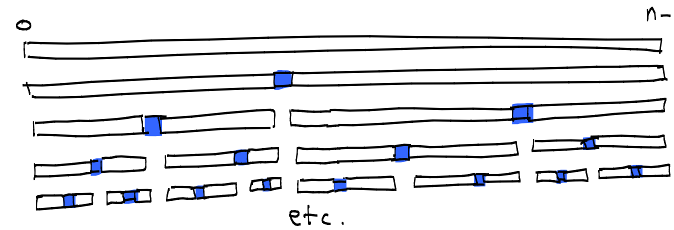


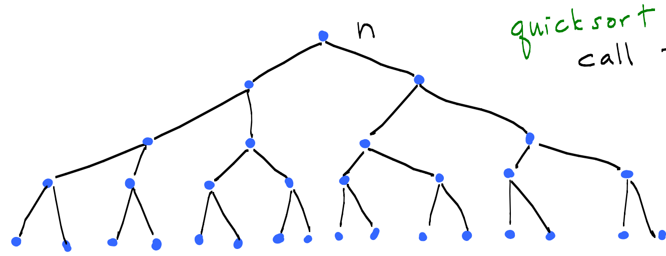

```
quicksort (list) {
  if (list. size() <= 1)
     return list
  else while true {
         choose pivot
         l1 ← elements less than pivot
         l2 ← elements greater than pivot
         if pivot was good {   // prob = 1/2
             l1 ← quicksort (l1)
             l2 ← quicksort (l2)
             return concatenate (l1, pivot , l2)
         }
       }
  }
}
```

Q: What is the probability that the body of the while loop is executed infinitely many times

i.e. infinite loop?

A: $\left(\frac{1}{2}\right)^i \to 0$ as $i \to \infty$.





quicksort call tree

Let y_j be the number of subproblems of quicksort such that

list.size is in $\left(\left(\frac{3}{4}\right)^{j+1} n, \left(\frac{3}{4}\right)^{j} n \right]$.

We will call these problems of type j and soon we will calculate $\mathcal{E}\, y_j$.

total work for quicksort

$$\leq \sum_j \left(\begin{array}{l} \text{work for each} \\ \text{subproblem of type } j \end{array} \right) \cdot y_j$$

Now take expected value

$\mathcal{E}$(total work for quicksort) = ?

A few observations...

Each node in the quicksort call tree has two children whose sizes are at most $\frac{3}{4}$ as large. Thus,

- the height of the quicksort call tree is $\log_{\frac{4}{3}} n$.

- parent and child contribute to different y_j i.e. they are different type

- Two nodes in the quicksort call tree have overlapping (not disjoint) lists if and only if one is an ancestor of the other. (parent-child, grand parent-grandchild, etc). Thus, all subproblems of type j are disjoint.

[MODIFIED April 10]

Q: How big is y_j ?

A: Since the size of each subproblem of type j at least $\left(\frac{3}{4}\right)^{j+1} n$ and since subproblems of type j don't overlap, we have $y_j * \left(\frac{3}{4}\right)^{j+1} n \leq n$. (see next slide)

Thus, $y_j \leq \left(\frac{4}{3}\right)^{j+1}$.

ASIDE : [added April 10]

We have y_j subproblems of type j and let's say that they are of sizes $\ell_1\, \ell_2 \ldots \ell_{y_j}$. Then $\sum_{i=1}^{y_j} \ell_i \leq n$.

But $\left(\frac{3}{4}\right)^{j+1} n \leq \ell_i$ for each $i = 1, \ldots y_j$.

Substituting ℓ_i gives:

$$\left(\frac{3}{4}\right)^{j+1} n \cdot y_j \leq n .$$

$\mathcal{E}$ (total work for quicksort)

$$\leq \sum_{j=0}^{\log_{\frac{4}{3}} n} \underbrace{\mathcal{E}(\text{work for each subproblem of type } j)}_{2c\left(\frac{3}{4}\right)^{j} n} \; \underbrace{y_j}_{\left(\frac{4}{3}\right)^{j+1}}$$

recall we reject pivots that were not good partitions

$$= \frac{4}{3} \cdot 2cn \cdot \log_{\frac{4}{3}} n$$

Summary of main idea

- quicksort recursively divides into subproblems; we can upper bound the number of problems of a given size. This upper bound on number grows as the problem size shrinks; the two effects cancel, leaving a total linear work (proportional to n) for each problem size, and the number of problem sizes is $\log n$.

Announcements

- next week is the last 2 lectures (The following week I will hold open office hours 9 - 5)

Final Exam

COMP 251 Algorithms and Data Structures
Tues. April 15, 2014 9 AM

Examiner: Michael Langer
Associate Examiner: Joseph Vibihal

LASTNAME: ____________ FIRSTNAME: ________________ ID: ________________

Instructions:

- This is a closed book exam.
- * → You may use up to five double sided CRIB sheets.
- No electronic devices are allowed.
- If your answer does not fit on a page, then use the reverse side and indicate that you have done so..

question	points	score
1	4	
2	4	
3	3	
4	4	
5	4	
6	4	
7	3	
8	4	
9	3	
10	3	
11	4	
12	4	
13	3	
14	3	
TOTAL	50	

midterm 1 (15)

midterm 2 (15)

after midterm 2 (20)