A Lecture on Divide-and-Conquer Algorithms and the Select Problem

Olivia Shi, Yutong Zhang

February 27, 2018

This is the augmented transcript of a lecture given by Luc Devroye on January 18th, 2018 for the Honours Algorithms and Data Structures class (COMP 252, McGill University). The subject was Euclid's algorithm, ham-sandwich and pancake theorems, half space counting and the linear time selection problem.

Euclid's Algorithm

Euclid's algorithm efficiently computes the greatest common divisor (GCD) of two nonnegative numbers m, n ($m < n$), the largest number that divides both of them without leaving a remainder.

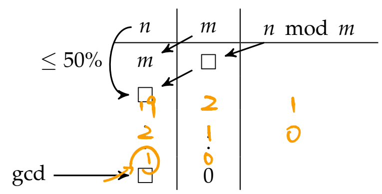


Figure 1: Illustration of Euclid's method. The number written two rows below n is at most 50% of m — we say that n is "halved".

GCD(n, m)
if $m == 0$
  return ~~$m = 0$~~ n
else
  return GCD$(m, n \bmod m)$

two step

Denote T_n as the worst case "time" for any GCD(k, l) with $n \geq k \geq l > 0$. We have the complexity $T_n \leq 2 \log_2 n$, since n gets halved every second iteration, and a number n can get halved at most $\log_2 n$ times.

Exercise 1. Using the bit model, show that the complexity of GCD(n, m) is $O(\log_2 n \cdot \log_2 m)$.

Ham-Sandwich Theorem (3D)

The Ham-Sandwich Theorem[1] describes the following: take a sandwich made of a slice of ham and two slices of bread. As long as one's knife is long enough, one can cut all three pieces in half in only one pass. The precise mathematical statement of the theorem, generalized to n dimensions, is that given n compact sets in $\mathbb{R}^n$, there is a hyperplane that bisects each set so that the two halves of both sets have equal measure.

[1] Libgober [2008]

Can we cut a hamburger exactly in half so that each half has exactly 50% of the cheese, 50% of the meat, and 50% of the bread? Yes.

RED AND BLACK POINTS

In 2D, this theorem is known as the pancake theorem. It implies that we can perfectly bisect n red points and n black points in the plane, so that each side has $n/2$ red and $n/2$ black points.

Application: We can divide m points into four sets of $n/4$ points each by two lines (Figure 2). First draw a horizontal line that divides the points in half. Color all points below the line red, and above the line black. By the pancake theorem, we can find a second line that evenly divides red and black, yielding $n/4$ points in each of the four sets of the partition.

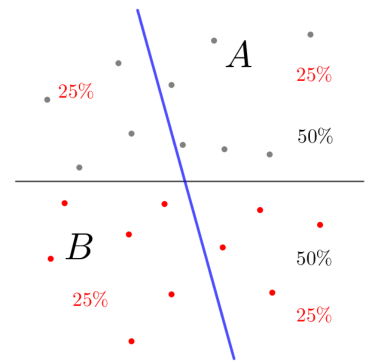


Figure 2: Illustration of the application.

Half Space Counting Problem

Let $x_1, \cdots, x_n$ be drawn from $\mathbb{R}^2$. Make a data structure such that one can "efficiently" answer queries that take as input a line given by the user, and outputs the number of points on one side of the line.

DATA STRUCTURE (DIVIDE-AND-CONQUER) $\textsc{Count}(\ell, S)$ is a divide-and-conquer algorithm for computing the number of points of S on one side of ℓ, say the side that contains the origin. Assume that we partitioned our space recursively using the 25%-trick suggested by the pancake theorem.

we return the count to the set S.

```
COUNT(ℓ, S)
if |S| ≤ 10
     do it manually
else
     determine the three sets cut by ℓ, say A, B, C
     if the fourth set is on the good side of ℓ
          return |S|/4 + COUNT(ℓ, A) + COUNT(ℓ, B) + COUNT(ℓ, C)
     else
          return COUNT(ℓ, A) + COUNT(ℓ, B) + COUNT(ℓ, C)
```

We can indeed determine A, B, C in constant time. In the RAM model, $T_n = 1 + 3T_{n/4}$, which yields $T_n = \Theta\left(n^{\log_4 3}\right)$ by the master theorem.

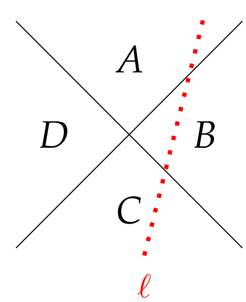


Figure 3: Illustration of the $\textsc{Count}(\ell, S)$ algorithm.

Linear time selection: finding the k^{th} smallest element

We present a selection algorithm invented by Blum, Floyd, Pratt, Rivest and Tarjan[2] for finding the k^{th} smallest number in a list or array; such a number is called the k^{th} order statistic. This includes the cases of finding the minimum, maximum, and median elements.

[2] Blum et al. [1973]

In particular, we want to find the k^{th} smallest number in an unordered set $S = \{x_1, \cdots, x_n\}$, and can use a comparison oracle. By sorting we would have time complexity $O(n \log_2 n)$.

We will give an $O(n)$ complexity solution, called $\textsc{Select}(k, S)$, where S is the collection of elements, and $1 \le k \le |S|$.

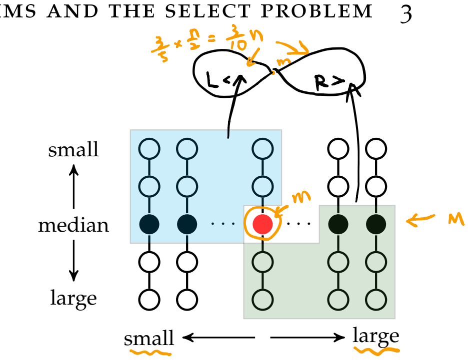


Figure 4: Medians of the median.

$\frac{3n}{10} \le |L|, |R| \le \frac{7n}{10}$

$\textsc{Select}(k, S)$

1 **if** $|S| \le 5$
2 $\quad\quad \textsc{Sort}(S)$
3 $\quad\quad$ **return** the k^{th} smallest element of S
4 **else**
5 $\quad$ (1) group all elements in groups of 5, and find the median in each group and call the set of medians M $\quad \frac{n}{5}\times 6 \quad$ 5 数选 M, 6 步 (pair)
$\quad\quad$ // This costs 6 comparisons.
6 $\quad$ (2) let $m = \textsc{Select}(|M|/2, M)$ $\quad$ Find m, medium of M. $\quad T_{n/5}$
$\quad\quad$ // m is the median of the medians
7 $\quad$ (3) compare all elements of S with m, forming the sets L and R of smaller and larger elements $\quad \frac{4n}{5}$ ~~n~~ -1 $\quad \frac{3}{10}\times 2$ 已定了.
8 $\quad$ (4) **if** $|L| == k - 1$
9 $\quad\quad\quad$ **return** m
10 $\quad\quad$ **elseif** $|L| \ge k$ $\quad$ doesn't count time complex $\quad$ worse case
11 $\quad\quad\quad$ **return** $\textsc{Select}(k, L)$ $\quad T_{|L|} \le T_{\frac{7n}{10}}$
12 $\quad\quad$ **else** $\quad 0 \le T_{\frac{7n}{10}}$
13 $\quad\quad\quad$ **return** $\textsc{Select}(k - |L| - 1, R)$ $\quad T_{|R|} \le T_{\frac{7n}{10}}$

$$T_n \le \begin{cases} T_{n/5} + T_{\frac{7n}{10}} + \frac{11}{5}n, & n > 5 \\ 7 & n \le 5 \end{cases}$$

The bound on the cost for every critical step of the $\textsc{Select}(k, S)$ algorithm is listed in the table below:

Line number	*Bound on the cost*
1 - 3	≤ 7
5	$\le 6n/5$
6	$= T_{n/5}$
7	$= n$
11	$= T_{\lvert L\rvert} \le T_{7n/10}$
13	$= T_{\lvert R\rvert} \le T_{7n/10}$

A POSSIBLE IMPROVEMENT. As shown in Figure 5, L and R each have at least $3|S|/10$ and at most $7|S|/10$ elements. Hence, to identify L and R, if we program correctly, we only require at most $4n/10$ new comparisons. Therefore, the recurrence that calculates the complexity can be reduced to $T_n \le 8n/5 + T_{n/5} + T_{7n/10}$. We will prove by induction that $T_n \le Cn$ for all $n \in \mathbb{N}$.

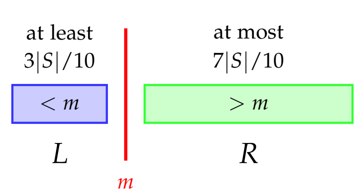
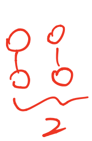


Figure 5: Illustration of L and R.

Proof. Base case: If $n \le 5$, then $C \ge 7$ is a safe choice.

Induction hypothesis: Assume $T_k \le Ck$ for all $k < n$.



代入C

Inductive step: $T_n \leq 8n/5 + T_{n/5} + T_{7n/10} \leq 8n/5 + C \cdot 9n/10$. This should be $\leq Cn$, so we have the requirement $8n/5 \leq C \cdot n/10$, or $C \geq 16$. We have thus shown that $T_n \leq 16n$ for all $n \in \mathbb{N}$. □

MEDIAN-OF-3 RECURRENCE. When we replace the median-of-5 step by a median-of-3 step, then one can see that $T_n = \Theta(n) + T_{n/3} + T_{2n/3}$. This equation is analyzed via a recursion tree. Note that in each level the work adds up to n. We have $\left(\frac{2}{3}\right)^k n = 1$, where k is the height of the recursion tree, so $k = \log_2 n / \log_2(3/2)$. Therefore, $T_n = \Theta(n \log_2 n)$.

worse case

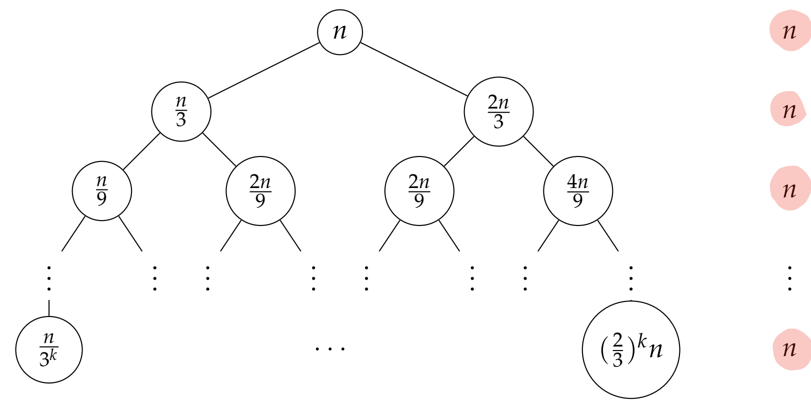


Figure 6: Recursion tree.

Observe that the recursion tree is not balanced. Nevertheless, the work at each level is precisely n.

References

M. Blum, R. W. Floyd, V. R. Pratt, R. L. Rivest, and R. E. Tarjan. Time bounds for selection. *Journal of Computer and System Sciences*, pages 448–461, Aug 1973. DOI: 10.1016/S0022-0000(73)80033-9. URL `http://people.csail.mit.edu/rivest/pubs/BFPRT73.pdf`.

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. The MIT Press, 3rd edition, 2009. ISBN 978-0-262-53305-8.

Brian Libgober. The Borsuk-Ulam and ham-sandwich theorems. May 2008. URL `http://www.math.uchicago.edu/~may/VIGRE/VIGRE2008/REUPapers/Libgober.pdf`.

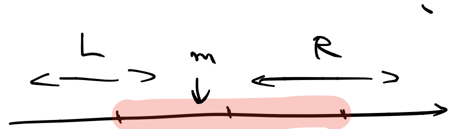
"Find"

T_n random. L m R
random

$T_n = n - 1 + \begin{cases} T_{|L|} \\ T_{|R|} \\ 0 \end{cases}$

$T_{|L|} \leq T_{\frac{3n}{4}}$ half the time

$\leq T_n$ half the time.



$E\{T\}$ expected time. avg

$E T_n \leq \frac{1}{2}\left(E T_n + E T_{\frac{3n}{4}}\right) + n.$

$2 E T_n \leq E T_n + E T_{\frac{3n}{4}} + 2n.$

worst case n^2

$E T_n \leq E T_{\frac{3n}{4}} + 2n \leq Cn$

$\frac{3n}{4} C + 2n \leq Cn$

$C \geq 8.$

T_n = max time taken on any problem of size n or less.

$T_n \uparrow$ in n.

$$T_n \le \begin{cases} T_{\lceil n/5 \rceil} + T_{\lceil 7n/10 \rceil} + \frac{11}{5} n, & n > 5 \\ 7, & n \le 5 \end{cases}$$

Will show: "$T_n \le Cn$" $\begin{cases} \text{for some } C \\ \text{for all } n \end{cases}$

INDUCTION

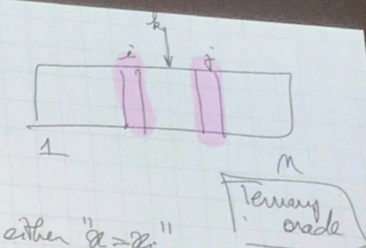
$n \le 5$: $T_n \le 7 \overset{\text{should be}}{\le} Cn \quad \forall\, 1 \le n \le 5$ — $C \ge 7$ needed

$n > 5$, assume truth up to $n-1$.

$$T_n \le T_{n/5} + T_{7n/10} + \frac{11n}{5} \le Cn\left(\frac{1}{5} + \frac{7}{10}\right) + \frac{11n}{5}$$

$$= \boxed{\frac{9C\cancel{n}}{10} + \frac{11\cancel{n}}{5} \overset{\text{should be}}{\le} C\cancel{n}} \longrightarrow \boxed{C \ge 22}$$

BIN SEARCH $(x; i, j)$ $\quad i=1,\ j=n$



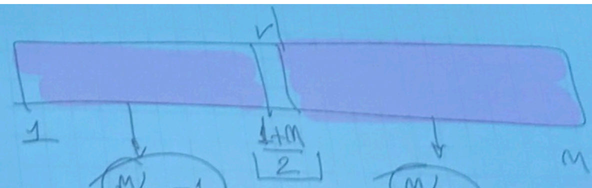
Cases:

- $i > j$: return "x not present".
- $i = j$: Ternary Oracle (x, x_i). return either "$x = x_i$" or "x not present".
- $i < j$: $k \leftarrow \left\lfloor \frac{i+j}{2} \right\rfloor$

  Ternary oracle (x, x_k)

  Cases:
  - $x < x_k$: return BIN SEARCH$(x; i, k-1)$
  - $x = x_k$: return "$x = x_k$".
  - $x > x_k$: return BIN SEARCH$(x; k+1, j)$

到最后index之下 之后再分析 base case



n even $\quad \frac{n}{2}-1 \qquad \frac{n}{2}$ ← worst case

n odd $\quad \frac{1+n}{2}-1 = \frac{n-1}{2} \qquad \frac{n-1}{2}$ ← worst case

$$\boxed{T_n \le 1 + T_{\lfloor n/2 \rfloor}, \quad T_1 = 1, \quad T_0 = 0.}$$

Show that: $T_n \le \lceil \log_2(n+1) \rceil$.

Induction proof.

$n=0, n=1$: ✓

assume true up to $n-1$. $(n \ge 2)$

$$T_n \le 1 + T_{\lfloor n/2 \rfloor}$$

n even $\quad = 1 + T_{n/2} \le 1 + \lceil \log_2(\underbrace{\tfrac{n}{2}+1}_{\frac{n+2}{2}}) \rceil$

$$= 1 + \lceil \log_2(n+2) - \log_2 2 \rceil$$

$$= \lceil \log_2(n+2) \rceil$$

should be

$$= \le \lceil \log_2(n+1) \rceil.$$

$\lceil \log_2 23 \rceil$

$= \lceil \log_2 24 \rceil$