CS6033 Lecture 6 Slides/Notes

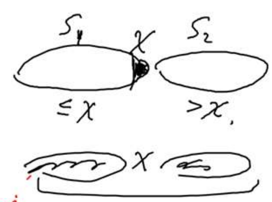
Divide and Conquer: Randomized QuickSort, Randomized QuickSelect, Comparison-Based Sorting Lower Bound, Linear-Time Sorting, Closest Pair in 2D (Notes, Ch 7, Secs. 9.2, 8.1 – 8.3, Notes)

By Prof. Yi-Jen Chiang
CSE Dept., Tandon School of Engineering
New York University

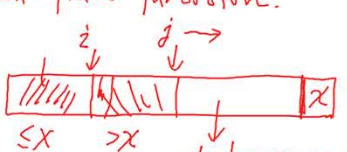
1

Randomized QuickSort:

1. Randomly pick one item x as the pivot (each of the n items has the same probability of $\frac{1}{n}$ to be picked) $O(1)$
2. Use x to partition the items into 2 sets: $O(n)$
3. Recursively sort S_1, recursively sort S_2.



In-place Partition.



e.g. 10 5 3 2 8 12 6 $\boxed{7}$

5 10 3

5 3 10 2

5 3 2 10 8 12 6

5 3 2 6 8 12 10 7

5 3 2 6 7 12 10 8

Done

2

$T(n) = \underline{T(|S_1|)} + \underline{T(|S_2|)} + cn$. But $|S_1|, |S_2|$ may differ at different recursions.

$\Rightarrow$ NOT easy to get a definite recurrence.

Recursion Tree

$T(n)$

$T(a)$ $\quad$ $T(b)$ $\quad$ $+ cn$

$(a+b=n)$

$(\bar{c}+d=a)$ $\quad$ $(e+f=b)$

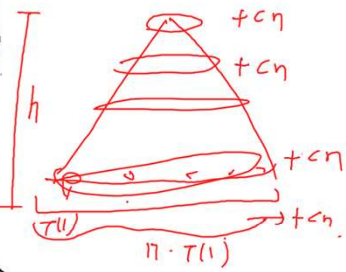
$T(\bar{c})$ $\quad$ $T(d)$ $\quad$ $T(e)$ $\quad$ $T(f)$ $\quad$ $\to + ca + cb$

$+ ca$ $\quad$ $+ cb$ $\quad$ $= + cn$

$+c\bar{c}$ $\quad$ $+cd$ $\quad$ $+ce$ $\quad$ $+cf$ $\quad$ $\to + c(\bar{c}+d+e+f)$

$= + cn$

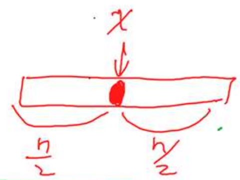
$\to + cn$



Each level cost $= cn$

$T(n) = O(n \cdot h)$

Expected length of any path from root to leaf?



$n \cdot \left(\frac{1}{2}\right)^k = 1$

$\Rightarrow k = \log_2 n$

(From next slide)

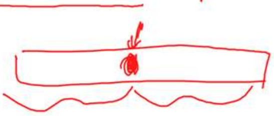
$\to E[h] = O(\log n)$

$E[T(n)] = O(n \cdot E[h]) = O(n \log n)$ ※

3

(M0):

"Good Event": pivot x is right in the middle.

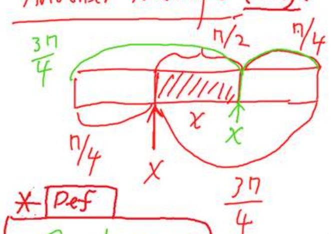


$\Pr[\text{good event}] = p = \frac{1}{n}$. $\quad$ $\boxed{\text{Bernoulli Trial}}$

Expected # trials to get a good event: $\frac{1}{p} = n$

Bad!!

Another Attempt (M1):



$\Pr[x \in \boxed{////}] = \frac{n/2}{n} = \frac{1}{2} = p$.

Expected # Trials to have $x \in \boxed{///}$: $\frac{1}{p} = 2$

Subproblem size $\leq \frac{3n}{4}$. $\quad$ $n \cdot \left(\frac{3}{4}\right)^k = 1 \Rightarrow \boxed{k = \log_{\frac{4}{3}} n}$

After k good events, the subproblem size $= 1$.

$n \to 1$

In a root to leaf path, there can be $\leq k$ good events.

※ Def

$\boxed{\text{Good Event} = x \in \boxed{////}}$

Expected # Trials to get k heads/Good events $= \frac{k}{p} = \boxed{2k}$

$=$ Expected length root $\to$ leaf $= E[h] = 2k = O(\log n)$ (go to previous slide bottom part)

4

* A good exercise for solving a recurrence by recursion trees

Assume: Each good event always splits current set S into 2 subsets of sizes $\frac{1}{4}|S|, \frac{3}{4}|S|$.

$$T(n) = b \cdot cn + T\left(\frac{n}{4}\right) + T\left(\frac{3n}{4}\right)$$

Note: This assumption is NOT true

b trials to get a good event.

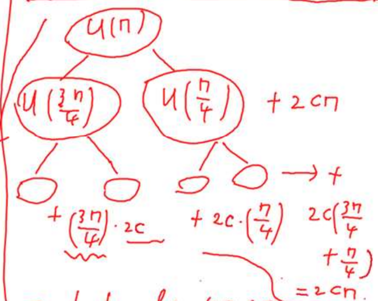
$$E\left[T(n)\right] = E\left[b \cdot cn\right] + E\left[T\left(\frac{n}{4}\right)\right] + E\left[T\left(\frac{3n}{4}\right)\right]$$

$$\boxed{E\left[b \cdot cn\right] = E[b] \cdot cn. \quad E[b] = \text{expected \# trials to get a good event} = \frac{1}{p} = 2.}$$

Define: $E\left[T(n)\right] = U(n)$. $E\left[T\left(\frac{n}{4}\right)\right] = U\left(\frac{n}{4}\right)$ $E\left[T\left(\frac{3n}{4}\right)\right] = U\left(\frac{3n}{4}\right)$

$U(n)$: expected run-time for problem of size n.

$$U(n) = U\left(\frac{3n}{4}\right) + U\left(\frac{n}{4}\right) + 2cn$$



Each level: $+2cn$.

Root-to-leaf path: Each time we go down to a child subproblem size is $\leq \frac{3}{4} \times$ (original size)

$$n \cdot \left(\frac{3}{4}\right)^k = 1 \Rightarrow k = \log_{\frac{4}{3}} n.$$

Length of any root-to-leaf path $\leq k = O(\log n)$.

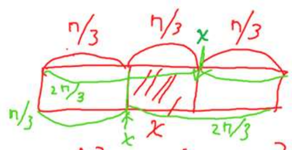
$h \leq k = O(\log n)$.

$$U(n) = h \cdot (2cn) \leq k \cdot 2cn = 2cn \log_{\frac{4}{3}} n = O(n \log n)$$ \#

(M2)

Another way:

Define Good event:



$X \in$ ▨: $\Pr\{\text{Good event}\} = \Pr\{X \in \text{▨}\} = p = \frac{n/3}{n} = \frac{1}{3}$.

Expected # trials to get a good event $= \frac{1}{p} = 3$

In any root-to-leaf path, there can be at most k good events where $n \cdot \left(\frac{2}{3}\right)^k = 1$. $\boxed{k = \log_{\frac{3}{2}} n.}$

$\boxed{\text{cf. } 2cn \log_{\frac{4}{3}} n.}$

Expected # trials to get k good events

$$= E[h] = \frac{k}{p} = 3 \log_{\frac{3}{2}} n$$

$$E\left[T(n)\right] = (cn) \cdot E[h] = cn \cdot 3 \log_{\frac{3}{2}} n = 3cn \log_{\frac{3}{2}} n = O(n \log n)$$ \#

Question:

$\log_{\frac{4}{3}} n = O(\log n)$. Why?

Ans:

$$\log_a n = \frac{\log n}{\log a}$$

(const > 1)

$$= \left(\frac{1}{\log a}\right) \log n$$

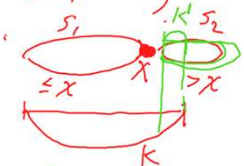
$$= \text{const} \cdot \log n$$

$$= O(\log n)$$

Randomized Quick Select.

Given n unsorted items and an integer $k \in [1, n]$, find the k-th smallest item among the n items.

Alg:
1. Randomly pick an item x as the pivot. (each item has the same prob $\frac{1}{n}$ to be picked)
2. Use x to partition the n items:

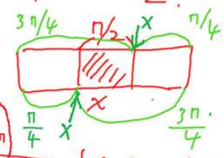


3. Recurse on either S_1 or S_2.

$K' = K - |S_1| - 1$

Use K' when recursing on S_2

Analysis: Define Good Event: $x \in$ [hatched]



$T(n)$: Running time

size of subproblem $\le \frac{3}{4}n$

$$T(n) \le (cn)\cdot b + T\left(\frac{3n}{4}\right)$$

b: # trials to get a good event

Cf. Randomized QuickSort: Not easy to get a definite recurrence.

7

$$T(n) \le b \cdot cn + T\left(\frac{3n}{4}\right)$$

$$E[T(n)] \le E[b] \cdot cn + E\left[T\left(\frac{3n}{4}\right)\right]$$

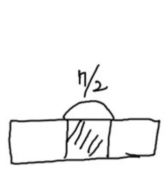
Define: $U(n) = E[T(n)]$

expected running time for problem size n.

$E[b]$ = expected # trials to get a good event

where $= \frac{1}{p} = 2$.

$$p = \Pr[x \in \text{[hatched]}] = \frac{n/2}{n} = \frac{1}{2}$$



$$U(n) \le 2cn + U\left(\frac{3n}{4}\right)$$

$$U(n) \le U\left(\frac{3n}{4}\right) + 2cn$$

$$\le \left[U\left(\frac{3}{4}\cdot\frac{3n}{4}\right) + 2c\left(\frac{3n}{4}\right)\right] + 2cn$$

$$= U\left(\left(\frac{3}{4}\right)^2 n\right) + 2cn\left(1 + \frac{3}{4}\right)$$

$$\le \left[U\left(\frac{3}{4}\cdot\left(\frac{3}{4}\right)^2 n\right) + 2c\left(\left(\frac{3}{4}\right)^2 n\right)\right] + 2cn\left(1 + \frac{3}{4}\right)$$

$$= U\left(\left(\frac{3}{4}\right)^3 n\right) + 2cn\left(1 + \frac{3}{4} + \left(\frac{3}{4}\right)^2\right)$$

$$\le \cdots \le U\left(\left(\frac{3}{4}\right)^i n\right) + 2cn\left(1 + \frac{3}{4} + \cdots + \left(\frac{3}{4}\right)^{i-1}\right)$$

$$\le \text{const.} + 2cn\left(\frac{1}{1 - 3/4}\right) = \text{const} + 2cn \cdot \frac{1}{1/4}$$

$$= \text{const.} + 8cn = O(n)$$

8

Comparison-Based Sorting Lower Bound

* Lower Bound: In the worst-case, any algorithm needs to use t time to solve the problem then that problem has a run-time lower bound of t.
* Comparison-Based Sorting Lower Bound:

  Only comparisons are allowed to operate on the keys.

Decision Tree Model.

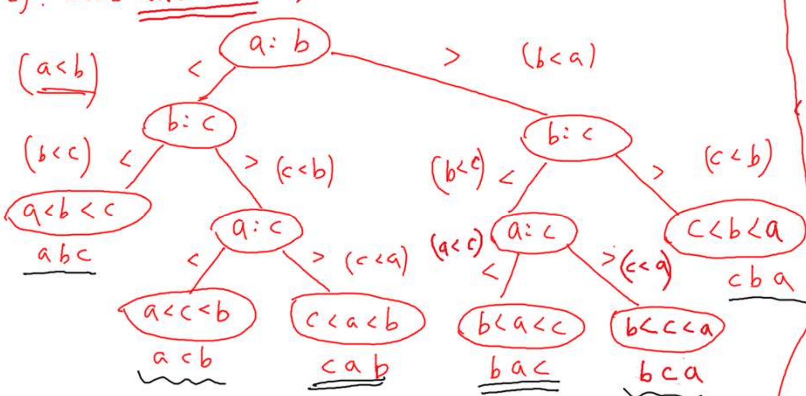
Eg. Sort items a, b, c.

(next slide)

9

Decision Tree

Eg. Sort distinct keys a, b, c



Leaves are all sorting outcomes.

$\#$ leaves $= \#$ permutations of n distinct items $= n!$

Internal nodes: comparisons

Running time of any sorting alg: length of some root-to-leaf path

Worst-case run time = length of longest such path = $\boxed{\text{tree height}} \geq \log(\#\text{ leaves}) = \log(n!) = \Theta(n \log n)$

Lower bound is $\Omega(n \log n)$

10

Linear-Time Sorting:

(NOT comparison-based)

Integer Sorting: (counting sort, bucket sort)

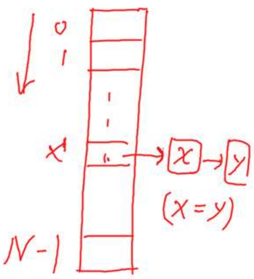
n integers.

Find the min item, max item.

$N = \max - \min$.

Each item x: $x' \leftarrow x - \min$ $(x' \geq 0)$

Use a bucket of N slots.



Put each item x into slot x'

In cn time, place all items to the slots.

From slot 0 to slot $N-1$ read out items from non-empty slots

$\Rightarrow$ items are sorted. $\boxed{O(n+N) \text{ time}}$

Using chained lists: bucket sort

counters: counting sort.

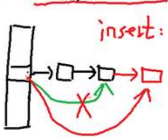
(* **Def**: stable sorting

For any two items a, b that compare as equal, their relative order (eg. a is before b) is preserved after sorting)

11

* We can make bucket sorting stable:

For each list, maintain a pointer to the last item in the list, so that we can insert an item to the end of list in $O(1)$ time.



Read out items from list: in this order

$\Rightarrow$ order is preserved.

eg.

845	342	837	342
837	845	342	348
348	837	845	457
457	457	348	837
342	348	457	845
	stable sorted	stable sorted	stable sorted.

Done

Radix Sort: n items, each has d digits.

Each digit has N possible values. (eg. base 10, $N=10$)

Alg: Perform stable sorting on 1 digit in each pass, from the least to the most significant digit.

* Each pass: $O(n+N)$ time (via stable bucket sorting)

Total: d passes

$\Rightarrow O(d(n+N))$ time ($= O(dn)$ time if N = const)

12

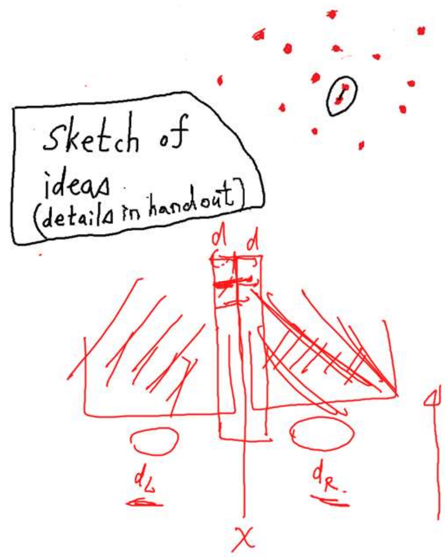
* A nice example of divide-and-conquer algorithm:

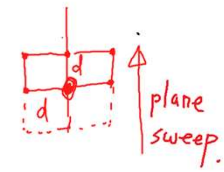
Closest Pair Problem in 2D

(Handout: textbook 3rd ed. Sec. 33.4)

Given a set of n points in 2D, find a pair of points in the set whose Euclidean distance is the smallest

(such pair is called the closest pair, and their distance is called the closest-pair distance)





$d \leftarrow \min\{d_L, d_R\}$

$T(n) = 2T(\frac{n}{2}) + cn$

$T(n) = O(n \log_2 n)$

* Handout: Closest Pair --- ``CLRS-3rd-Ed-Closest-Pair.pdf''

13