上海科技大学
ShanghaiTech University

CS240 Algorithm Design and Analysis

Lecture 21

Randomized Algorithms (Cont.)

Quan Li

Fall 2023

2023.12.21

Global Minimum Cut

A problem for which the best-known randomized algorithm is faster than the best-known deterministic algorithm

Global Minimum Cut

Global min cut. Given a connected, undirected graph G = (V, E), find a cut (A, B) of minimum cardinality.

Applications. Partitioning items in a database, identify clusters of related documents, network reliability, network design, circuit design, TSP solvers.

Network flow solution.

- Replace every edge (u, v) with two antiparallel edges (u, v) and (v, u).
- Pick some vertex s and compute min s-v cut separating s from each other vertex $v \in V$.

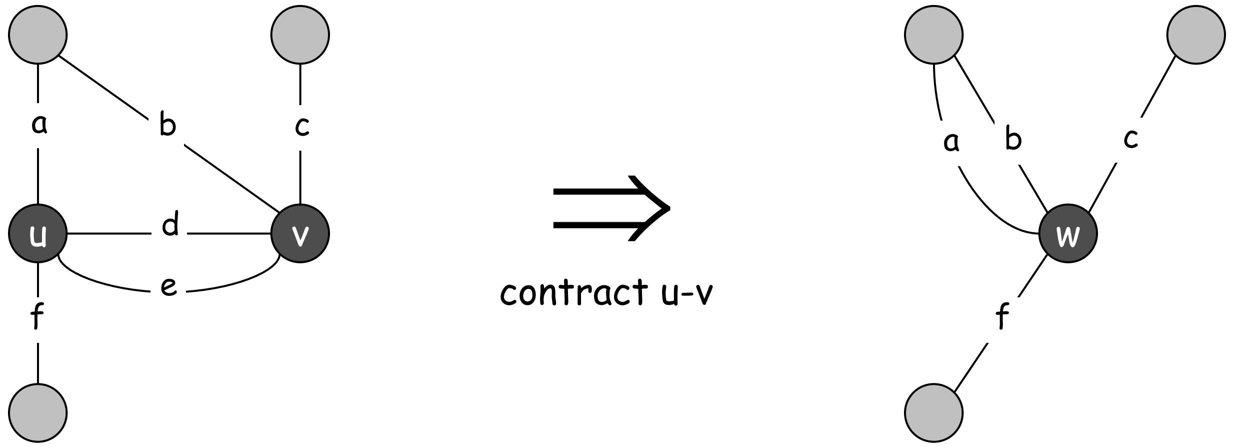
False intuition. Global min-cut is harder than min s-t cut.

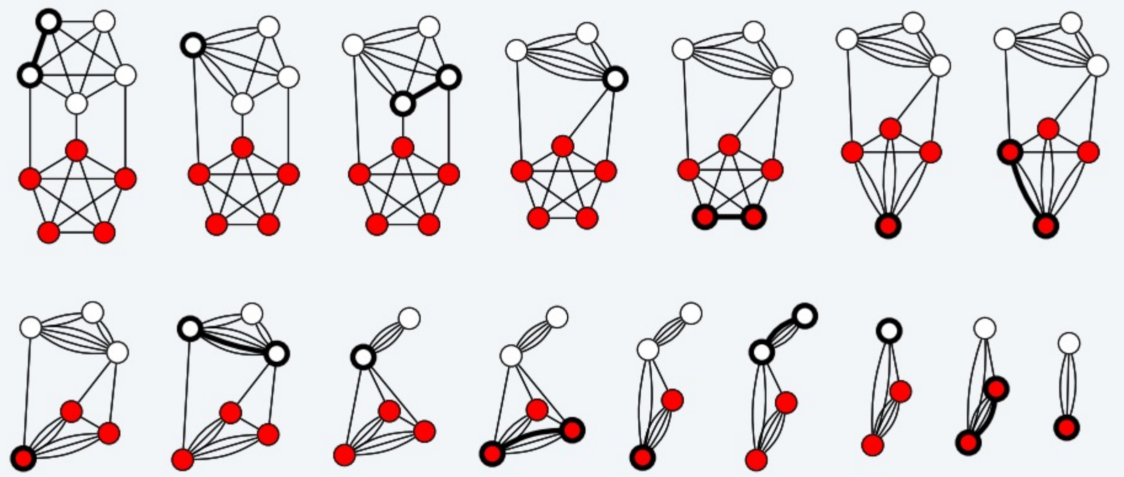
立志成才 报国裕民

Contraction Algorithm

Contraction algorithm. [Karger 1995]

- Pick an edge e = (u, v) uniformly at random.
- Contract edge e.
  - replace u and v by a single new supernode w
  - preserve edges, updating endpoints of u and v to w
  - keep parallel edges, but delete self-loops
- Repeat until graph has just two supernodes v_1 and v_2.
- Return the cut (between the two supernodes).





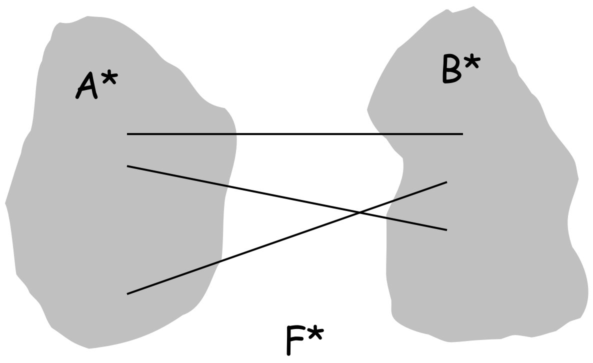
Reference: Thore Husfeldt

Contraction Algorithm

Claim. The contraction algorithm returns a min cut with prob $\geq 2/n^2$.

Pf. Consider a global min-cut (A^*, B^*) of G. Let F^* be edges with one endpoint in A^* and the other in B^*. Let $k = |F^*|$ = size of min cut.

- In first step, algorithm contracts an edge in F^* with prob $k / |E|$.
- Every node has degree $\geq k$ since otherwise (A^*, B^*) would not be min-cut. $\Rightarrow$ $|E| \geq \frac{1}{2}kn$.
- Thus, algorithm contracts an edge in F^* with probability $\leq 2/n$.



Contraction Algorithm

Claim. The contraction algorithm returns a min cut with prob $\geq 2/n^2$.

Pf. Consider a global min-cut (A*, B*) of G. Let F* be edges with one endpoint in A* and the other in B*. Let k = |F*| = size of min cut.

- Let G' be graph after j iterations. There are n' = n-j supernodes.
- Suppose no edge in F* has been contracted. The min-cut in G' is still k.
- Since value of min-cut is k, $|E'| \geq \frac{1}{2}kn'$ -> $k/|E'|$ <= $2/n'$
- Thus, algorithm contracts an edge in F* with probability $\leq 2/n'$.

- Let E_j = event that an edge in F* is not contracted in iteration j.

$$
\begin{aligned}
\Pr[E_1 \cap E_2 \cdots \cap E_{n-2}] &= \Pr[E_1] \times \Pr[E_2 \mid E_1] \times \cdots \times \Pr[E_{n-2} \mid E_1 \cap E_2 \cdots \cap E_{n-3}] \\
&\geq \left(1-\tfrac{2}{n}\right)\left(1-\tfrac{2}{n-1}\right)\cdots\left(1-\tfrac{2}{4}\right)\left(1-\tfrac{2}{3}\right) \\
&= \left(\tfrac{n-2}{n}\right)\left(\tfrac{n-3}{n-1}\right)\cdots\left(\tfrac{2}{4}\right)\left(\tfrac{1}{3}\right) \\
&= \tfrac{2}{n(n-1)} \\
&\geq \tfrac{2}{n^2}
\end{aligned}
$$

Contraction Algorithm

Amplification. To amplify the probability of success, run the contraction algorithm many times.

Claim. If we repeat the contraction algorithm n^2 times with independent random choices and return the best cut found, then the algorithm finds the min-cut with constant probability.

Pf. By independence, the probability of failure is at most

$$\left(1-\frac{2}{n^2}\right)^{n^2} = \left[\left(1-\frac{2}{n^2}\right)^{\frac{1}{2}n^2}\right]^2 \le \left(e^{-1}\right)^2 = \frac{1}{e^2}$$

Global Min Cut: Context

Remark. Overall running time is slow since we perform $O(V^2)$ iterations, and each takes $O(E \log V)$ time (we always merge the vertex with smaller degree into the other).

Best known. [Karger 2000] $O(E \log^3 V)$.

faster than best known deterministic global min cut algorithm

Random Variables and Expectations

A Quick Review of Probability Theory

Expectation. Given a discrete random variables X, its expectation $E[X]$ is defined as:

$$E[X] = \sum i \cdot \Pr[X = i]$$

Q: Roll a 6-sided dice. What is the expected value?

A: ?

Q: Roll two dice. What is the expected maximum value?

A: ?

Expectation: Two Properties

Indicator random variables. If X only takes 0 or 1, $E[X] = \Pr[X = 1]$.

Linearity of expectation. Given two random variables X and Y (not necessarily independent),

$$E[X+Y] = E[X] + E[Y].$$

Remark: $E[XY] = E[X]E[Y]$ only when X and Y are independent.

Example. Shuffle a deck of n cards; turn them over one at a time; try to guess each card. Suppose you can't remember what's been turned over already, and just guess a card from full deck uniformly at random.

Q. What's the expected number of correct guesses?

A. (surprisingly effortless using linearity of expectation)

- Let $X_i = 1$ if i^{th} guess is correct and 0 otherwise.
- Let X = number of correct guesses $= X_1 + \cdots + X_n$.
- $E[X_i] = \Pr[X_i = 1] = 1/n$.
- $E[X] = E[X_1] + \cdots + E[X_n] = 1/n + \cdots + 1/n = 1.$

Guessing Cards with Memory

Guessing with memory. Guess a card uniformly at random from cards not yet seen.

Q. What's the expected number of correct guesses?

A.

- Let $X_i = 1$ if i^{th} guess is correct and 0 otherwise.
- Let X = number of correct guesses $= X_1 + \cdots + X_n$.
- $E[X_i] = \Pr[X_i = 1] = 1/(n - i + 1)$.
- $E[X] = E[X_1] + \cdots + E[X_n] = \frac{1}{n} + \cdots + \frac{1}{2} + \frac{1}{1} = \Theta(\log n)$.

立志成才报国裕民

The Birthday Paradox

Problem: Suppose there are $n = 365$ days in a year, and in a room of k people, each person's birthday falls in any one of the n days with equal probability. How large should k be for us to expect two people with the same birthday?

Analysis:

- Define $X_{ij} = 1$ if person i and person j have the same birthday, and 0 otherwise.
- We know $E\left[X_{ij}\right] = \Pr\left[X_{ij} = 1\right] = 1/n$.
- Let $X = \sum_{1 \le i < j \le k} X_{ij}$ be the number of pairs of people having the same birthday.
- We have
$$E[X] = E\left[\sum_{1 \le i < j \le k} X_{ij}\right] = \binom{k}{2}\frac{1}{n} = \frac{k(k-1)}{2n}$$
- So, when $\frac{k(k-1)}{2n} \ge \frac{(k-1)^2}{2n} \ge 1$, or $k \ge \sqrt{2n} + 1 \approx 28$, we expect to see at least one pair of people having the same birthday.

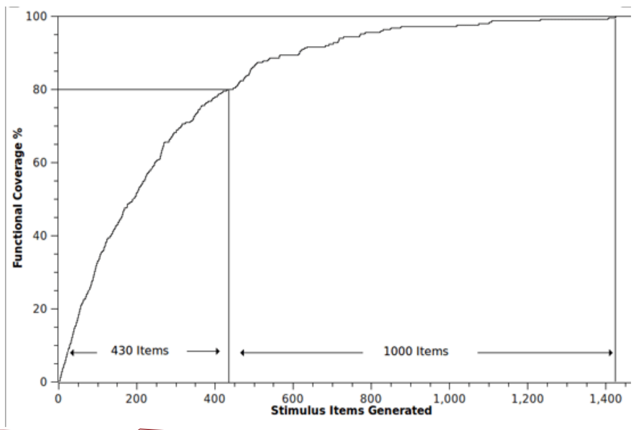
Coupon Collector

Coupon collector. Each box of cereal contains a coupon. There are n different types of coupons. Assuming a box contains each type of coupon equally likely, how many boxes do you need to open to have at least one coupon of each type?

Solution.

- Stage i = time between i and $i+1$ distinct coupons.
- Let X_i = number of steps you spend in stage i.
- Let X = number of steps in total $= X_0 + X_1 + \cdots + X_{n-1}$.

$$E[X] = \sum_{i=0}^{n-1} E[X_i] = \sum_{i=0}^{n-1} \frac{n}{n-i} = n \sum_{i=1}^{n} \frac{1}{i} = \Theta(n \log n)$$



prob of success $= (n-i)/n$
$\Rightarrow$ expected waiting time $= n/(n-i)$

MAX 3-SAT

An extremely simple randomized approximation algorithm

Maximum 3-Satisfiability

上海科技大学 ShanghaiTech University

exactly 3 distinct literals per clause

MAX-3SAT. Given 3-SAT formula, find a truth assignment that satisfies as many clauses as possible.

$$\begin{aligned}
C_1 &= x_2 \vee \overline{x_3} \vee \overline{x_4} \\
C_2 &= x_2 \vee x_3 \vee \overline{x_4} \\
C_3 &= \overline{x_1} \vee x_2 \vee x_4 \\
C_4 &= \overline{x_1} \vee \overline{x_2} \vee x_3 \\
C_5 &= x_1 \vee \overline{x_2} \vee \overline{x_4}
\end{aligned}$$

Remark. NP-hard search problem.

Simple idea. Flip a coin, and set each variable true with probability ½, independently for each variable.

立志成才报国裕民

Maximum 3-Satisfiability: Analysis

Claim. Given a 3-SAT formula with k clauses, the expected number of clauses satisfied by a random assignment is 7k/8.

Pf. Consider random variable $$Z_j = \begin{cases} 1 & \text{if clause } C_j \text{ is satisfied} \\ 0 & \text{otherwise.} \end{cases}$$

- Let Z = total number of clauses satisfied.

$$\begin{aligned} E[Z] &= \sum_{j=1}^{k} E[Z_j] \\ &= \sum_{j=1}^{k} \Pr[\text{clause } C_j \text{ is satisfied}] \\ &= \tfrac{7}{8} k \end{aligned}$$

linearity of expectation

Maximum 3-Satisfiability: Analysis

Lemma. The probability that a random assignment satisfies $\geq 7k/8$ clauses is at least $1/(8k)$.

Pf. Let p_j be probability that exactly j clauses are satisfied.

We start by writing

$$\frac{7}{8}k = \sum_{j=0}^{k} jp_j = \sum_{j<7k/8} jp_j + \sum_{j\geq 7k/8} jp_j$$
$$\leq \sum_{j<7k/8} k'p_j + \sum_{j\geq 7k/8} kp_j$$
$$= k'(1-p) + kp \leq k' + kp$$

Hence, $kp \geq \frac{7}{8}k - k'$

But $\frac{7}{8}k - k' \geq 1/8$ (k′ is the largest natural number that is strictly smaller than $\frac{7}{8}$k)

So

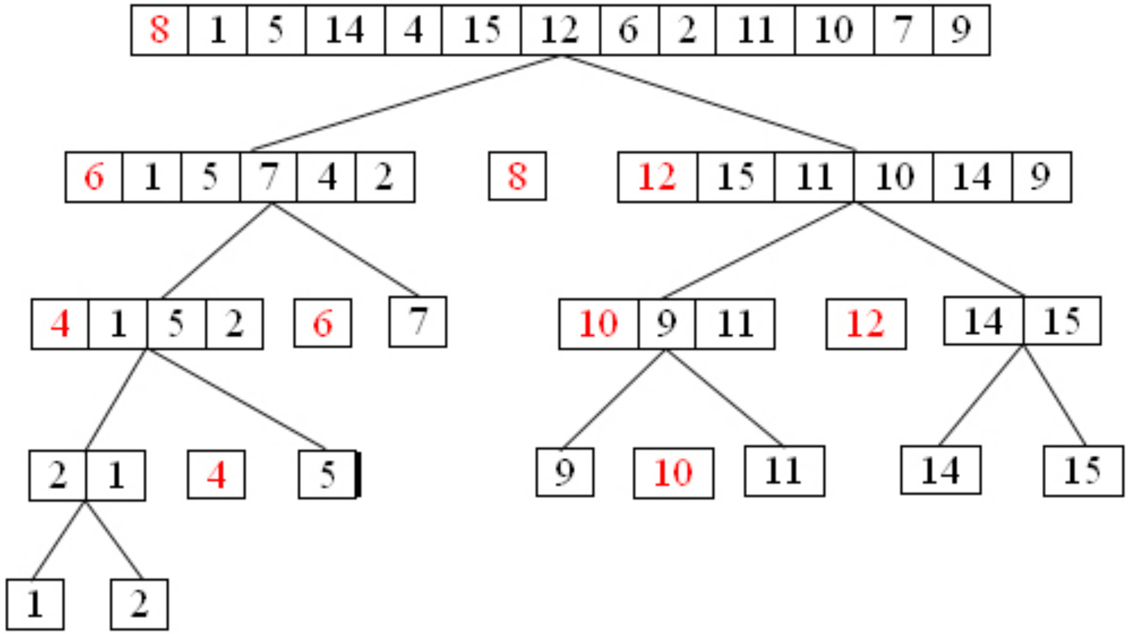
$$p \geq \frac{\frac{7}{8}k - k'}{k} \geq \frac{1}{8k}.$$

Quicksort

Quicksort

Recall the Quicksort algorithm

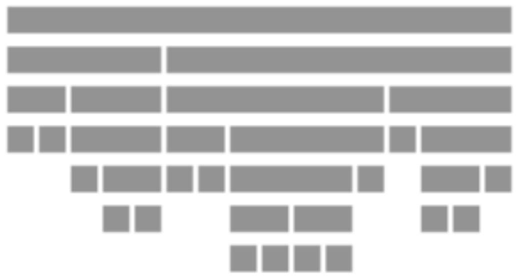
- Pick a pivot element s
- Partition the elements into two sets, those less than s and those more than s
- Recursively Quicksort the two sets



Complexity of Quicksort

- Let T(n) be the time to Quicksort n numbers.
- T(n) is small in practice.
- But in the worst case, $T(n)=O(n^2)$.
  - Occurs with very uneven splits, i.e. the rank of the pivot is very small or large.
  - Ex If pivot is smallest element, then $T(n)=T(1)+T(n-1)+n-1$. This solves to $T(n)=O(n^2)$.
    - $T(1)$ and $T(n-1)$ to recursively sort each side, $n-1$ to partition the elements wrt the pivot.

- As long as the pivot is near the middle, Quicksort takes $O(n \log n)$ time.
  - Ex If the pivot is always in the middle half, $[n/4, 3n/4]$, then $T(n) \leq T(n/4)+T(3n/4)+n-1$, which solves to $O(n \log n)$.



Pivot selection is crucial

Running time.

- [Best case.] Select the median element as the pivot: quicksort runs in $\Theta(n \log n)$ time.
- [Worst case.] Select the smallest (or the largest) element as the pivot: quicksort runs in $\Theta(n^2)$ time.

Q: How to find the median element?

A: Sort?

A: Randomly choose an element as the pivot!

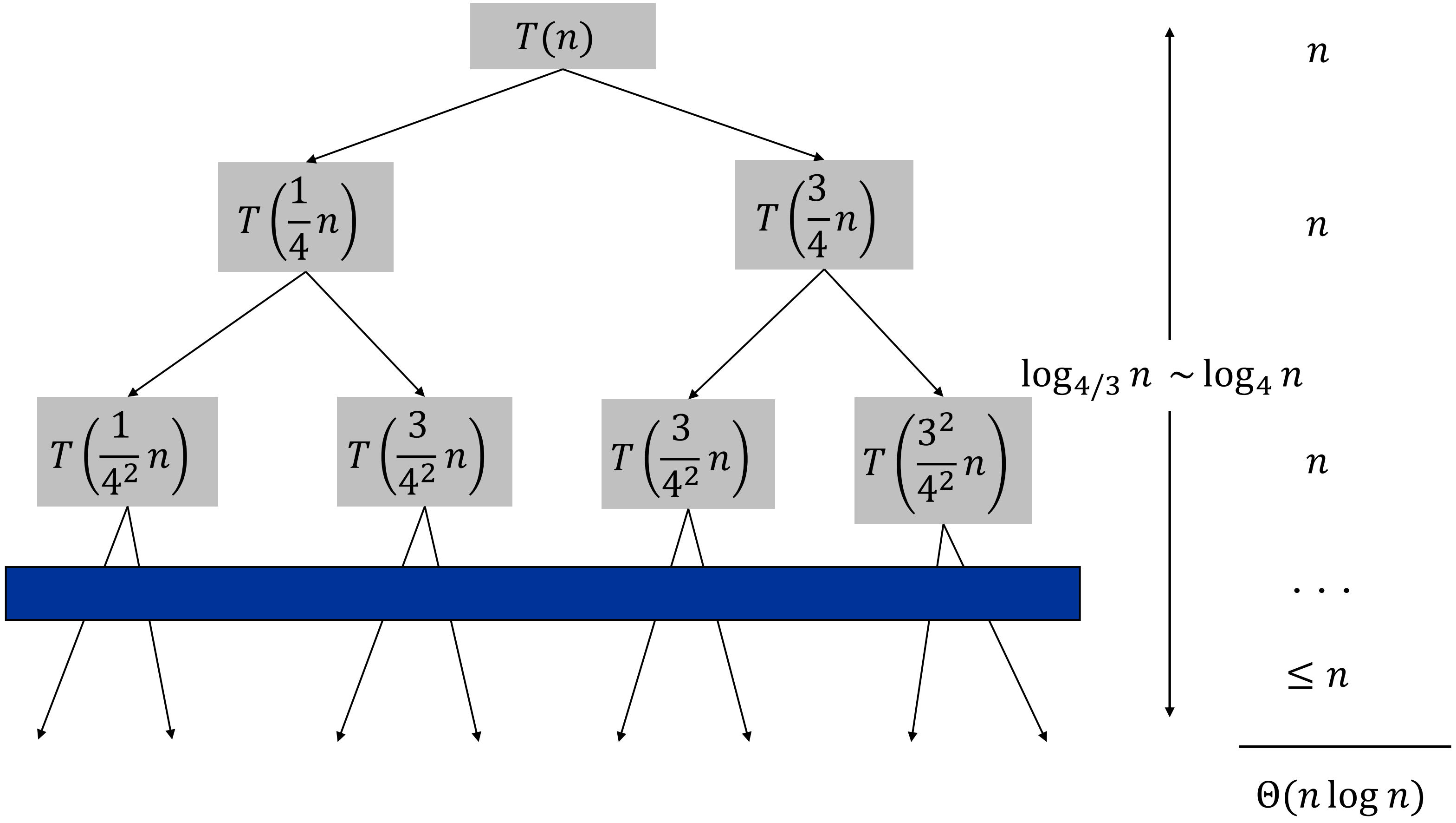
Intuition: A randomly selected pivot "typically" partitions the array as 25% vs 75%, so we have the recurrence

$$T(n) = T\left(\frac{1}{4}n\right) + T\left(\frac{3}{4}n\right) + n$$

which solves to $T(n) = \Theta(n \log n)$. (See next page.)

Solve the recurrence

$$T(n) = T\left(\frac{1}{4}n\right) + T\left(\frac{3}{4}n\right) + n$$



Randomized Quicksort

- Quicksort is only slow if we keep picking very small or large pivots.
- Let's pick the pivot at random. Intuitively, we shouldn't be unlucky and always pick small or large pivots.
- Pick a random pivot element s.
- Partition the elements into two sets, those less than s and those more than s.
- Recursively RQuicksort the two sets.

1. Complexity of RQuicksort

- Let $R(n)$ be the expected time to RQuicksort n numbers.
- With probability $1/n$, the pivot has rank 1 (is smallest element), in which case
$$R(n) = R(1) + R(n-1) + n - 1$$
- With probability $1/n$, the pivot has rank 2, and $R(n) = R(2) + R(n-2) + n - 1$
- ...
- With probability $1/n$, the pivot has rank k, and $R(n) = R(k) + R(n-k) + n - 1$
- Putting these together, we have

$$R(n) = \frac{1}{n} * \left(R(1) + R(n-1) + R(2) + R(n-2) + \cdots + R(n-1) + R(1) + (n-1) * \ (n-1)\right)$$

$$\leq \frac{2}{n} \sum_{k=1}^{n-1} R(k) + \Theta(n)$$

1. Complexity of RQuicksort

- We solve the recurrence for R(n) using the substitution method. We guess $R(n) \le an\log n + b$ for some constants a, b>0 to be determined.
- We first need the following lemma.

$$\sum_{k=1}^{n-1} klogk \le \frac{1}{2}n^2logn - \frac{1}{8}n^2$$

- Proof:

$$\sum_{k=1}^{n-1} klogk = \sum_{k=1}^{\left\lceil\frac{n}{2}\right\rceil-1} klogk + \sum_{k=\lceil n/2\rceil}^{n-1} klogk$$

$$\le (logn-1)\sum_{k=1}^{\left\lceil\frac{n}{2}\right\rceil-1} k + logn\sum_{k=\left\lceil\frac{n}{2}\right\rceil}^{n-1} k$$

$$= logn\sum_{k=1}^{n-1} k - \sum_{k=1}^{\lceil n/2\rceil-1} k$$

$$\le \frac{1}{2}n(n-1)logn - \frac{1}{2}\left(\frac{n}{2}-1\right)\frac{n}{2}$$

$$\le \frac{1}{2}n^2logn - \frac{1}{8}n^2$$

1. Complexity of RQuicksort

- Now we can solve for R(n).

$$R(n) = \frac{2}{n}\sum_{k=1}^{n-1} R(k) + \Theta(n)$$

$$\leq \frac{2}{n}\sum_{k=1}^{n-1} (aklogk + b) + \Theta(n)$$

$$= \frac{2a}{n}\sum_{k=1}^{n-1} klogk + \frac{2b(n-1)}{n} + \Theta(n)$$

$$\leq \frac{2a}{n}\left(\frac{1}{2}n^2logn - \frac{1}{8}n^2\right) + \frac{2b}{n}(n-1) + \Theta(n)$$

$$\leq anlogn - \frac{a}{4}n + 2b + \Theta(n)$$

$$= anlogn + b + \left(\Theta(n) + b - \frac{a}{4}n\right)$$

$$\leq anlogn + b$$

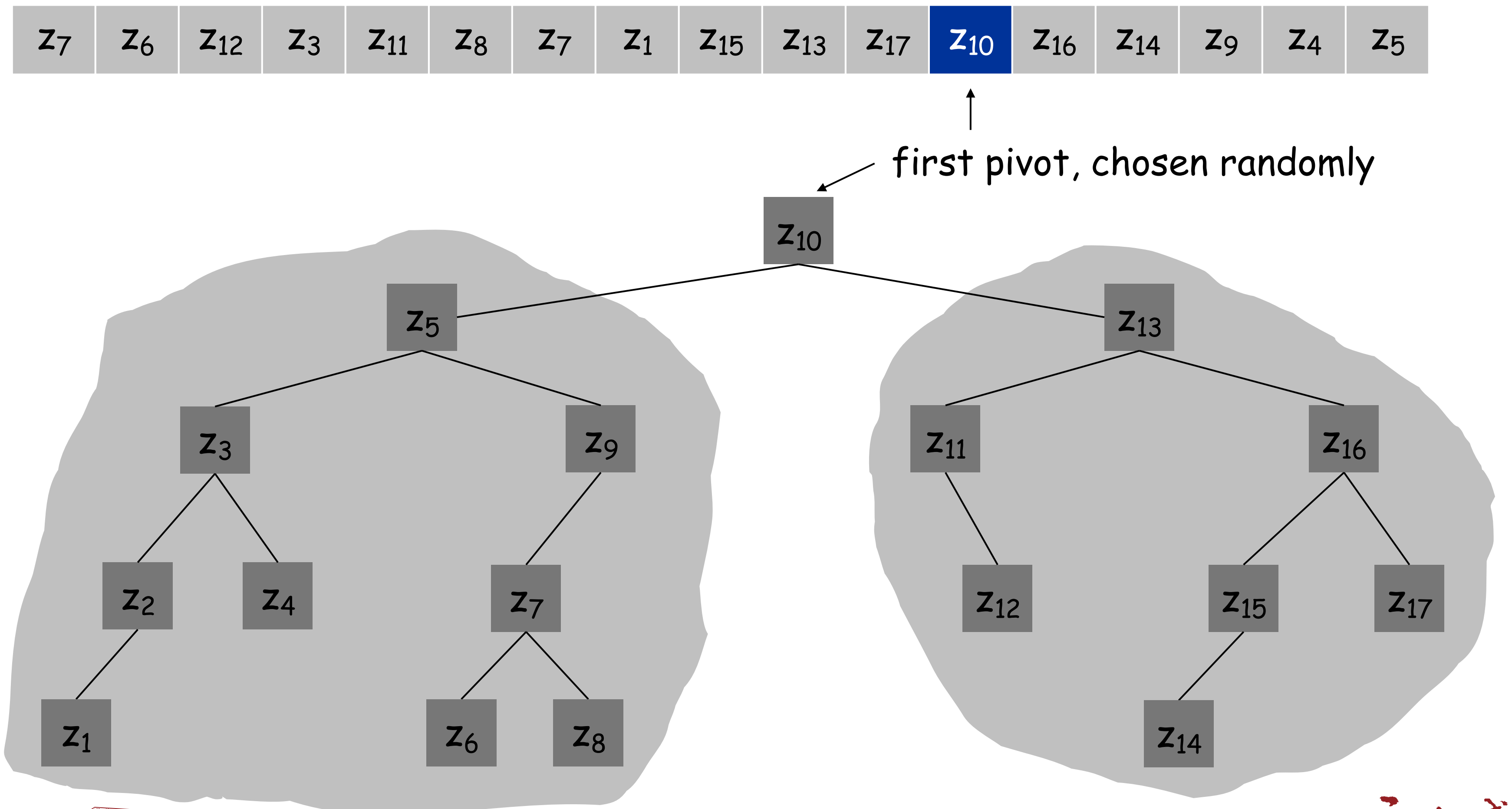
By choosing a so that $\frac{a}{4}n > \Theta(n) + b$

2. Analysis of quicksort: the binary tree representation

Assumption: All elements are distinct

Note: Running time = Θ(# comparisons)

Relabel the elements from small to large as $z_1, z_2, \ldots, z_n$



2. Analysis of quicksort (Cont.)

Theorem. Expected # of comparisons is $\Theta(n \log n)$.

Pf.

- Let $X_{ij} = 1$ if z_i is compared with z_j
- # of comparisons is $X = \sum_{i<j} X_{ij}$
- E[# of comparisons] $= \sum_{i<j} E[X_{ij}] = \sum_{i<j} \Pr[z_i$ and z$_j$ are compared$]$

	$j = 2$	3	4	…	n	
$i = 1$	$\frac{1}{2}$	$\frac{1}{3}$	$\frac{1}{4}$	…	$\frac{1}{n}$	$O(\log n)$
2		$\frac{1}{2}$	$\frac{1}{3}$	…	$\frac{1}{n-1}$	$O(\log n)$
3			$\frac{1}{2}$	…	$\frac{1}{n-2}$	$O(\log n)$
…	…	…	…	…	…	…
$n - 1$					$\frac{1}{2}$	$O(\log n)$
						$O(n \log n)$

Q: Can you show this is $\Theta(n \log n)$?

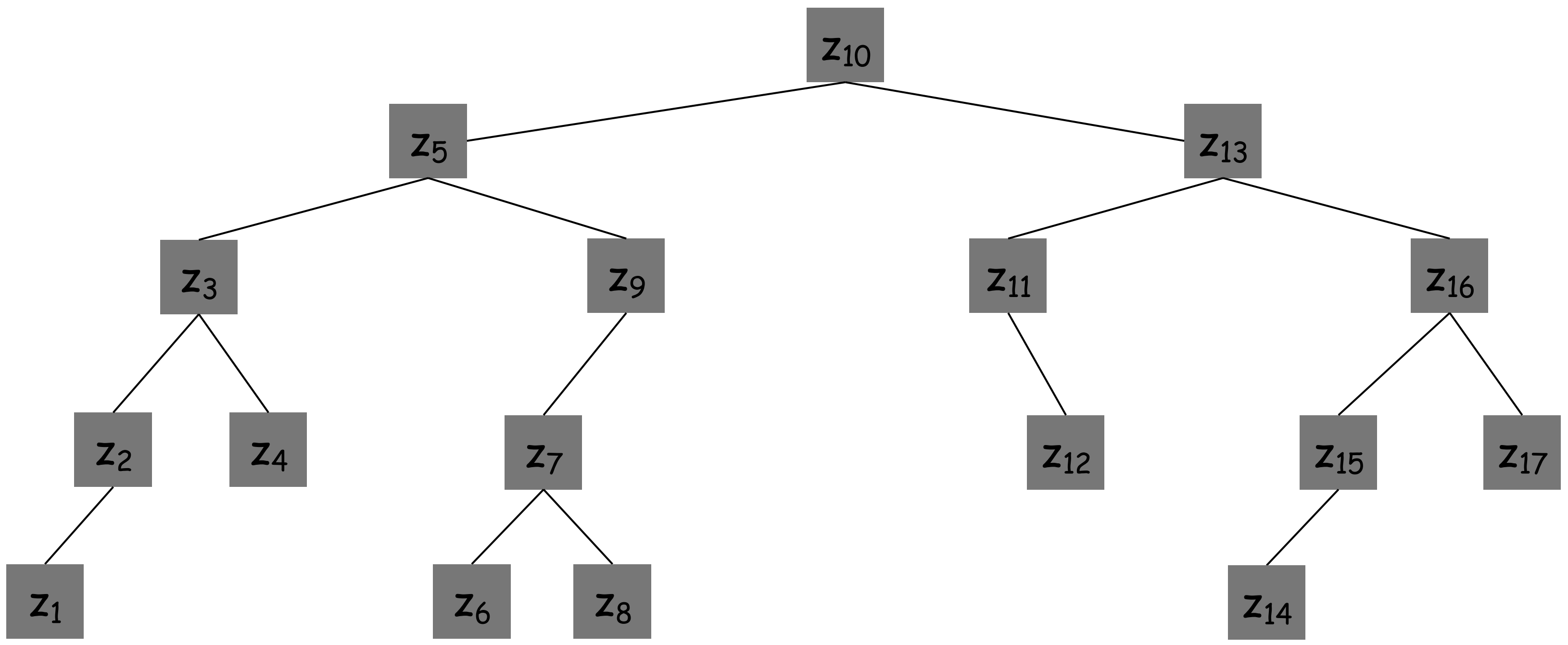
2. Analysis of quicksort

Observation 1: Element only compared with its ancestors and descendants.

- z_2 and z_7 are compared if their lowest common ancestor (lca) is z_2 or z_7.
- z_2 and z_7 are not compared if their lca is z_3, z_4, z_5, or z_6.
- Other elements cannot be the lca of z_2 and z_7

Observation 2: Every element in $\{z_i, \ldots, z_j\}$ is equally likely to be the lca of z_i and z_j

So, $\Pr[z_i \text{ and } z_j \text{ are compared}] = 2 / (j - i + 1)$.



Next Time:
Randomized algorithms (Cont.)

上海科技大学 ShanghaiTech University

立志成才报国裕民