Chapter 6
Randomization

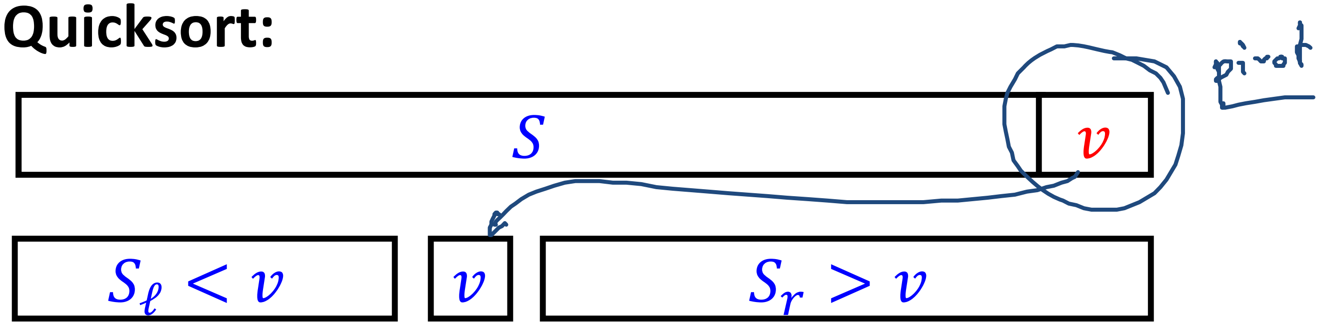
Algorithm Theory
WS 2013/14

contention res
primality test

Fabian Kuhn

Randomized Quicksort

Quicksort:



function Quick (S: sequence): sequence;

{returns the sorted sequence S}

begin

  if $\#S \leq 1$ then **return** S

  else { choose pivot element v in S;

    partition S into S_ℓ with elements $< v$,

    and S_r with elements $> v$

    return Quick(S_ℓ) v Quick(S_r)

end;

running time can quadratic

choose pivot at random

Randomized Quicksort Analysis

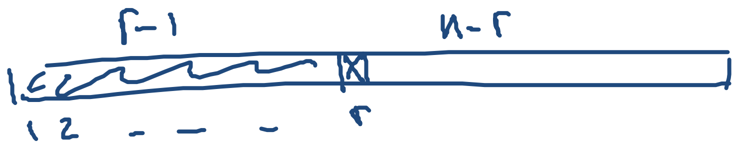
Randomized Quicksort: pick uniform random element as pivot

Running Time of sorting n elements:

- Let's just count the number of comparisons
- In the partitioning step, all $n-1$ non-pivot elements have to be compared to the pivot
- **Number of comparisons:**

$$\boldsymbol{n-1} \; + \; \textbf{\#comparisons in recursive calls}$$

- **If rank of pivot is r:**
  recursive calls with $r-1$ and $n-r$ elements



Randomized Quicksort Analysis

Random variables: $C = n-1 + C_\ell + C_r$ $\mathbb{E}[C]$

- C: total number of comparisons (for a given array of length n)
- R: rank of first pivot $\mathbb{P}(R=r) = \frac{1}{n}$
- C_ℓ, C_r: number of comparisons for the 2 recursive calls

$\mathbb{E}[C] = \mathbb{E}[n-1+C_\ell+C_r]$

$$\mathbb{E}[C] = n - 1 + \mathbb{E}[C_\ell] + \mathbb{E}[C_r]$$

$\mathbb{E}[X+Y] = \mathbb{E}$

Law of Total Expectation:

$$\mathbb{E}[C] = \sum_{r=1}^{n} \mathbb{P}(R=r) \cdot \mathbb{E}[C|R=r]$$

$$= \sum_{r=1}^{n} \mathbb{P}(R=r) \cdot \left(n - 1 + \mathbb{E}[C_\ell|R=r] + \mathbb{E}[C_r|R=r]\right)$$

Randomized Quicksort Analysis

We have seen that:

$$\mathbb{E}[C] = \sum_{r=1}^{n} \mathbb{P}(R=r) \cdot (n - 1 + \mathbb{E}[C_\ell | R = r] + \mathbb{E}[C_r | R = r])$$

T(n) T(r-1) T(n-r)

Define:

- $\boldsymbol{T(n)}$: expected number of comparisons when sorting n elements

$$\mathbb{E}[C] = T(n)$$
$$\mathbb{E}[C_\ell | R = r] = T(r-1)$$
$$\mathbb{E}[C_r | R = r] = T(n-r)$$

Recursion:

$$\boldsymbol{T(n) = \sum_{r=1}^{n} \frac{1}{n} \cdot \big(n - 1 + T(r-1) + T(n-r)\big)}$$

$$\boldsymbol{T(0) = T(1) = 0}$$

Randomized Quicksort Analysis

Theorem: The expected number of comparisons when sorting n elements using randomized quicksort is $T(n) \le 2n \ln n$. $(n \geq 1)$

Proof: $T(1) \le 2 \cdot 1 \cdot \ln(1) = 0$

$$T(n) = \sum_{r=1}^{n} \frac{1}{n} \cdot \big(n - 1 + T(r-1) + T(n-r)\big), \qquad T(0) = 0$$

repl. r-1 by i

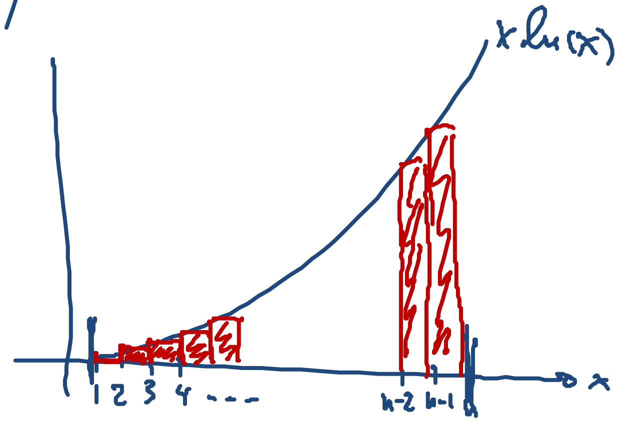
$$= n - 1 + \frac{1}{n} \cdot \sum_{i=0}^{n-1} \big(T(i) + T(n-i-1) \big)$$

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

$T(i) \le 2 \cdot i \cdot \ln(i)$ (by I.H.)

$$\le n - 1 + \frac{4}{n} \sum_{i=1}^{n-1} i \cdot \ln(i) \le n - 1 + \frac{4}{n} \cdot \int_1^n x \cdot \ln(x)\, dx$$

mon. incr. with i



Randomized Quicksort Analysis

Theorem: The expected number of comparisons when sorting n elements using randomized quicksort is $T(n) \leq 2n \ln n$.

Proof:

$$T(n) \leq n - 1 + \frac{4}{n} \cdot \int_1^n x \ln x \, dx$$

$$\int x \ln x \, dx = \frac{x^2 \ln x}{2} - \frac{x^2}{4}$$

$$T(n) \leq n-1 + \frac{4}{n}\left(\frac{n^2 \ln(n)}{2} - \frac{n^2}{4} + \frac{1}{4}\right)$$

$$= n-1 + 2n\ln(n) - n + \frac{1}{n}$$

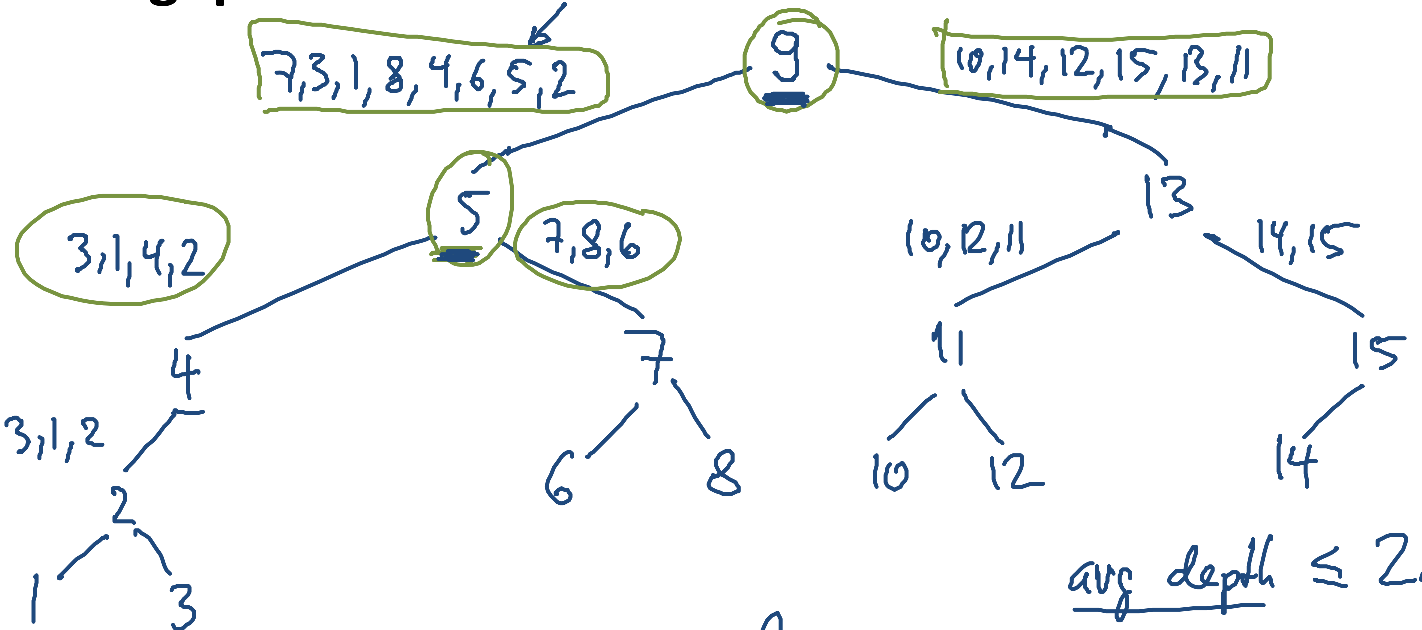
$$= 2n\ln(n) + \underbrace{\frac{1}{n} - 1}_{\leq 0} \leq 2n\ln(n)$$

also possible to show that $T(n) = O(n \log n)$ with high prob. $1 - \frac{1}{n^c}$

Alternative Analysis

Array to sort: [7 , 3 , 1 , 10 , 14 , 8 , 12 , 9 , 4 , 6 , 5 , 15 , 2 , 13 , 11]

Viewing quicksort run as a tree:



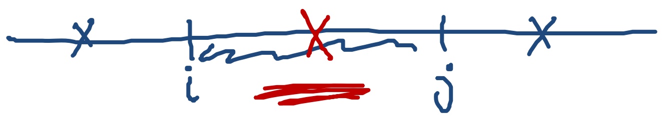
$$\text{avg depth} \leq 2\ln(n)$$

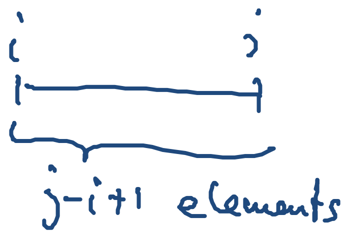
$$\log_2(n) = \frac{\ln(n)}{\ln(2)}$$

Comparisons

- Comparisons are only between pivot and non-pivot elements
- Every element can only be the pivot once:
  → every 2 elements can only be compared once!

- W.l.o.g., assume that the elements to sort are $1,2, \dots, n$
- Elements i and j are compared if and only if either i or j is a pivot before any element $h: i < h < j$ is chosen as pivot
  – i.e., iff i is an ancestor of j or j is an ancestor of i





$$\mathbb{P}(\text{comparison betw. } i \text{ and } j) = \frac{2}{j - i + 1}$$

Counting Comparisons

Random variable for every pair of elements (i, j):

$$X_{ij} = \begin{cases} 1, & \text{if there is a comparison between } i \text{ and } j \\ 0, & \text{otherwise} \end{cases}$$

$$\mathbb{P}(X_{ij} = 1) = \frac{2}{j-i+1} \rightarrow \mathbb{E}[X_{ij}] = \frac{2}{j-i+1}$$

Number of comparisons: X

$$X = \sum_{i<j} X_{ij}$$

- What is $\mathbb{E}[X]$?

Randomized Quicksort Analysis

Theorem: The expected number of comparisons when sorting n elements using randomized quicksort is $T(n) \leq 2n \ln n$.

Proof:

$$\mathbb{E}[X_{i,j}] = \frac{2}{j-i+1}$$

- **Linearity of expectation:**
  For all random variables $X_1, \dots, X_n$ and all $a_1, \dots, a_n \in \mathbb{R}$,

$$\mathbb{E}\left[\sum_i^n a_i X_i\right] = \sum_i^n a_i \mathbb{E}[X_i].$$

$$X = \sum_{i<j} X_{ij}$$

$$\mathbb{E}[X] = \mathbb{E}\left[\sum_{i<j} X_{ij}\right] = \sum_{i<j} \mathbb{E}[X_{ij}] = \sum_{i<j} \frac{2}{j-i+1} = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1}$$

Randomized Quicksort Analysis

Theorem: The expected number of comparisons when sorting n elements using randomized quicksort is $T(n) \leq 2n \ln n$.

Proof:

$$\mathbb{E}[X] = 2\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\frac{1}{j-i+1} = 2\sum_{i=1}^{n-1}\sum_{k=2}^{n-i+1}\frac{1}{k}$$

$$= H(n-i+1) - 1 \leq H(n) - 1$$

Harmonic series

$$H(n) = \sum_{i=1}^{n}\frac{1}{i}$$

$$H(n) \leq 1 + \ln(n)$$

$$\leq 2\sum_{i=1}^{n-1}(H(n)-1)$$

$$= 2(n-1)(\underbrace{H(n)-1}_{\leq \ln(n)})$$

□

Types of Randomized Algorithms

Las Vegas Algorithm:

- always a correct solution
- running time is a random variable

- **Example:** randomized quicksort, contention resolution

Monte Carlo Algorithm:

- probabilistic correctness guarantee (**m**ostly **c**orrect)
- fixed (deterministic) running time

- **Example:** primality test

Minimum Cut

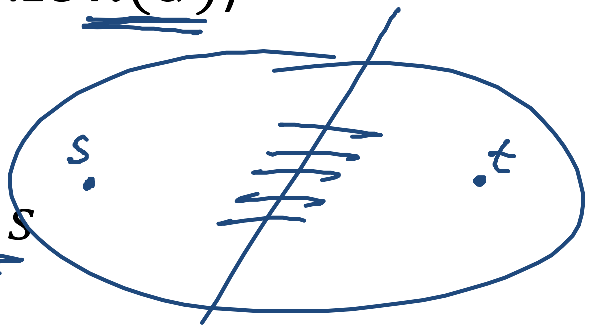
Reminder: Given a graph $G = (V, E)$, a cut is a partition (A, B) of V such that $V = A \cup B$, $A \cap B = \emptyset$, $A, B \neq \emptyset$

Size of the cut (A, B): # of edges crossing the cut

- For weighted graphs, total edge weight crossing the cut

edge connectivity

Goal: Find a cut of minimal size (i.e., of size $\lambda(G)$)

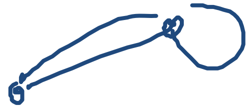


Maximum-flow based algorithm:

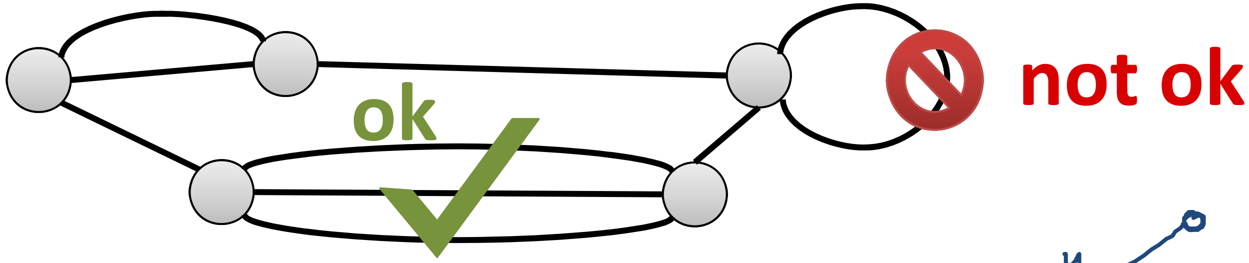
- Fix s, compute min s-t-cut for all $t \neq s$
- $O\big(m \cdot \lambda(G)\big) = O(mn)$ per s-t cut
- Gives an $O\big(mn\lambda(G)\big) = O(mn^2)$-algorithm $O(n^4)$

Best-known deterministic algorithm: $O(mn + n^2 \log n) = O(n^3)$

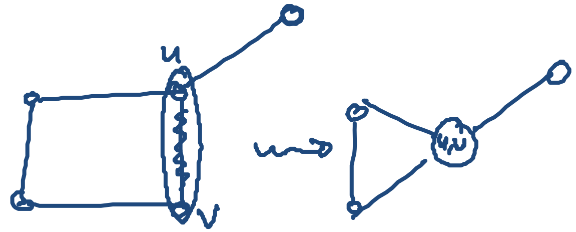
Edge Contractions



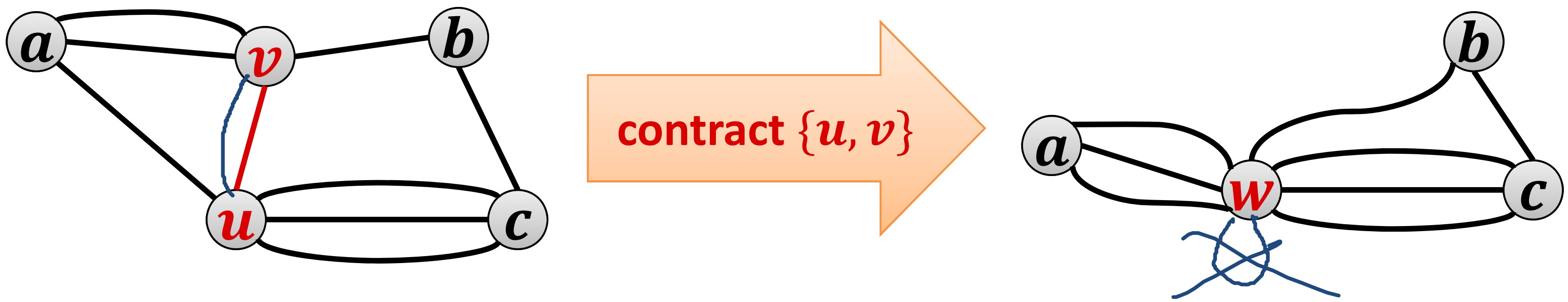
- In the following, we consider multi-graphs that can have multiple edges (but no self-loops)



Contracting edge $\{u, v\}$:



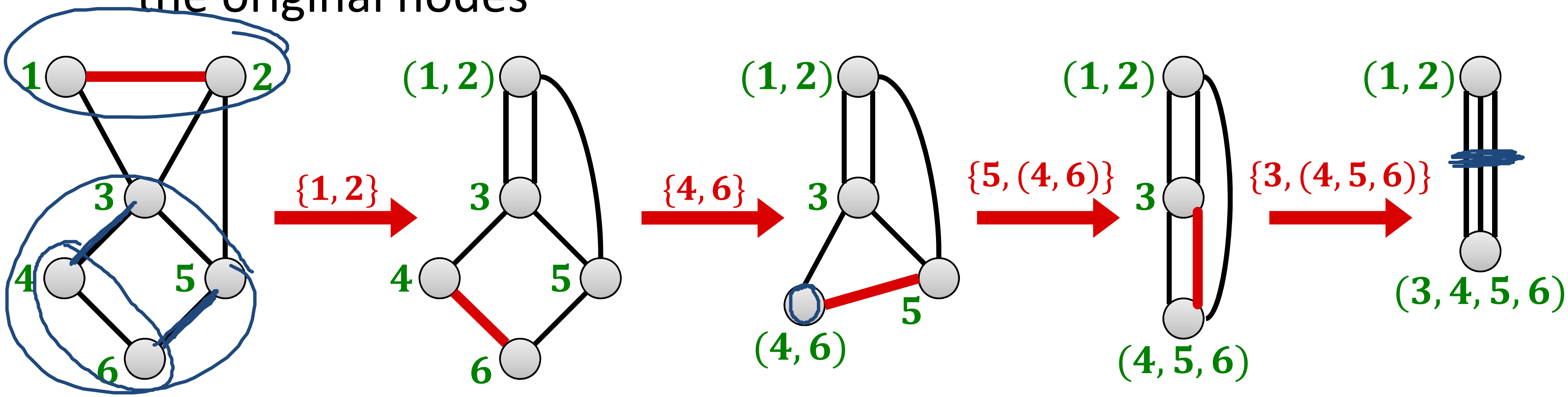
- Replace nodes u, v by new node w
- For all edges $\{u, x\}$ and $\{v, x\}$, add an edge $\{w, x\}$
- Remove self-loops created at node w



Properties of Edge Contractions

Nodes:

- After contracting $\{u, v\}$, the new node represents u and v
- After a series of contractions, each node represents a subset of the original nodes



Cuts:

- Assume in the contracted graph, w represents nodes $S_w \subset V$
- The edges of a node w in a contracted graph are in a one-to-one correspondence with the edges crossing the cut $(S_w, V \setminus S_w)$

Randomized Contraction Algorithm

Algorithm:

while there are >2 nodes **do**

contract a uniformly random edge

return cut induced by the last two remaining nodes

(cut defined by the original node sets represented by the last 2 nodes)

Theorem: The random contraction algorithm returns a minimum cut with probability at least $1/O(n^2)$.

- We will show this next.

Theorem: The random contraction algorithm can be implemented in time $O(n^2)$.

- There are $n-2$ contractions, each can be done in time $O(n)$.
- You will show this in the exercises.