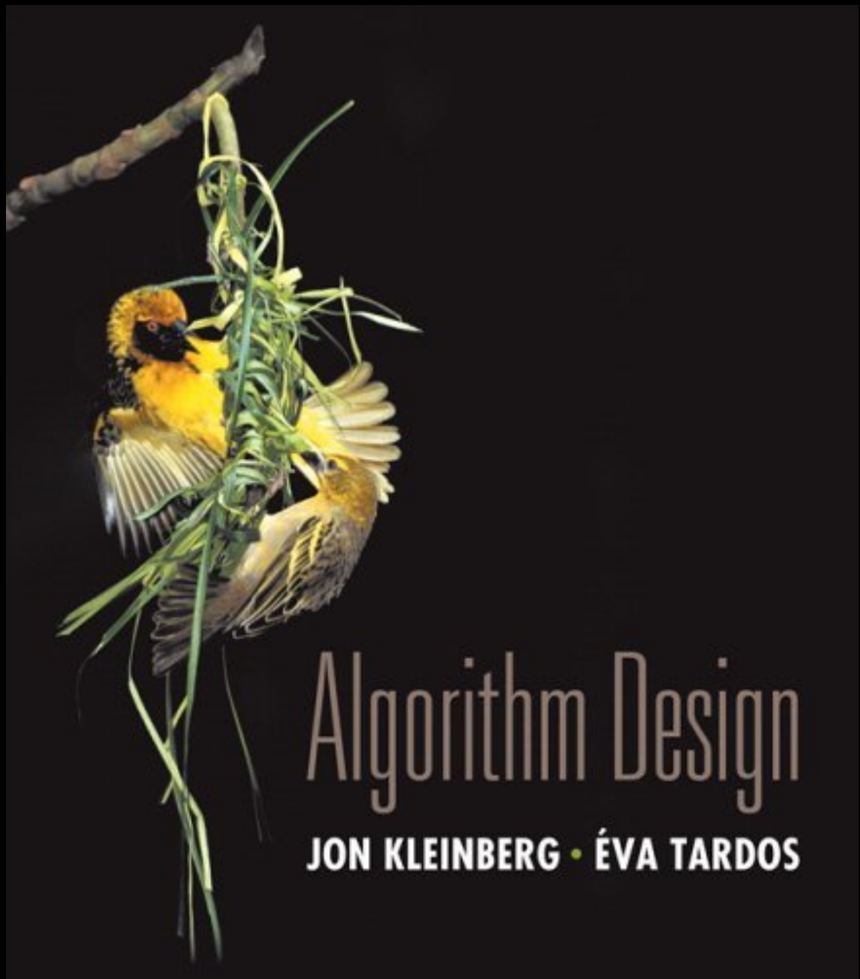


Chapter 13

Randomized Algorithms

Part 1

Slides by Kevin Wayne, modified by Rasmus Pagh.

Randomization

Algorithmic design patterns.

- Greedy.
- Divide-and-conquer.
- Dynamic programming.
- Network flow.
- Randomization.

in practice, access to a pseudo-random number generator

Randomization. Allow fair coin flip in unit time.
Generate random number in [0;1] in unit time.

Why randomize? Can lead to simplest, fastest, or only known algorithm for a particular problem.

13.1 Contention Resolution

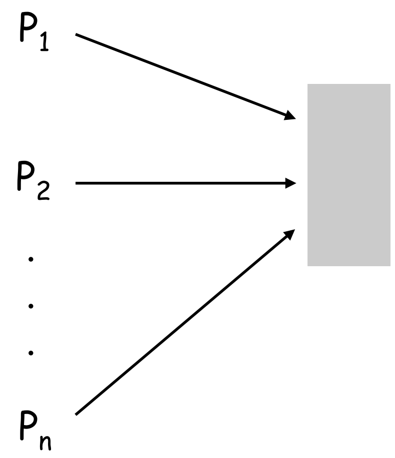
Contention Resolution in a Distributed System

Contention resolution. Given n processes $P_1, \ldots, P_n$, each competing for access to a shared database. If two or more processes access the database simultaneously, all processes are locked out. Devise protocol to ensure all processes get through on a regular basis.

Restriction. Processes can't communicate, but have knowledge of n.

Challenge. Need symmetry-breaking paradigm.

Protocol. Each process requests access to the database at time t with probability $p = 1/n$.



Contention Resolution: Randomized Protocol

Questions.

What is the probability that process i successfully accesses the database in a given round?

What is the probability that process i fails to access the database in each of tn rounds, for some $t>0$?

How many rounds will pass before all processes have successfully accessed the database?

Contention Resolution: Randomized Protocol

Protocol. Each process requests access to the database at time t with probability $p = 1/n$.

Claim. Let $S[i, t]$ = event that process i succeeds in accessing the database at time t. Then $1/(e \cdot n) \le \Pr[S(i, t)] \le 1/(2n)$.

Pf. By independence, $\Pr[S(i, t)] = p\,(1-p)^{n-1}$.

process i requests access

none of remaining n-1 processes request access

- Setting $p = 1/n$, we have $\Pr[S(i, t)] = 1/n\, \underbrace{(1 - 1/n)^{n-1}}_{\text{between } 1/e \text{ and } 1/2}$. ▪

value that maximizes $\Pr[S(i, t)]$

Useful facts from calculus. As n increases from 2, the function:
- $(1 - 1/n)^n$ converges monotonically from 1/4 up to 1/e
- $(1 - 1/n)^{n-1}$ converges monotonically from 1/2 down to 1/e.

Contention Resolution: Randomized Protocol

Claim. The probability that process i fails to access the database in en rounds is at most 1/e. After e·n(c ln n) rounds, the probability is at most n^{-c}.

Pf. Let F[i, t] = event that process i fails to access database in rounds 1 through t. By independence and previous claim, we have
Pr[F(i, t)] $\leq$ (1 - 1/(en)) ^{t}.

- Choose t = $\lceil e \cdot n \rceil$: $\Pr[F(i,t)] \leq \left(1-\frac{1}{en}\right)^{\lceil en \rceil} \leq \left(1-\frac{1}{en}\right)^{en} \leq \frac{1}{e}$
- Choose t = $\lceil e \cdot n \rceil \lceil c \ln n \rceil$: $\Pr[F(i,t)] \leq \left(\frac{1}{e}\right)^{c \ln n} = n^{-c}$

Contention Resolution: Randomized Protocol

Claim. The probability that all processes succeed within $2e \cdot n \ln n$ rounds is at least $1 - 1/n$.

Pf. Let $F[t]$ = event that at least one of the n processes fails to access database in any of the rounds 1 through t.

$$\Pr[F[t]] = \Pr\left[\bigcup_{i=1}^{n} F[i,t]\right] \underset{\text{union bound}}{\leq} \sum_{i=1}^{n} \Pr[F[i,t]] \underset{\text{previous slide}}{\leq} n\left(1-\tfrac{1}{en}\right)^t$$

- Choosing $t = 2 \lceil en \rceil \lceil c \ln n \rceil$ yields $\Pr[F[t]] \leq n \cdot n^{-2} = 1/n$. ▪

Union bound. Given events $E_1, \ldots, E_n$, $\Pr\left[\bigcup_{i=1}^{n} E_i\right] \leq \sum_{i=1}^{n} \Pr[E_i]$

13.2 Global Minimum Cut

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.

Contraction Algorithm

Contraction algorithm. [Karger 1995]

- Pick an edge e = (u, v) uniformly at random.
- Contract edge e.
  - replace u and v by single new super-node w
  - preserve edges, updating endpoints of u and v to w
  - keep parallel edges, but delete self-loops
- Repeat until graph has just two nodes v_1 and v_2.
- Return the cut: All nodes that were contracted to form v_1.

Amplification. To amplify success prob., run the algorithm many times.

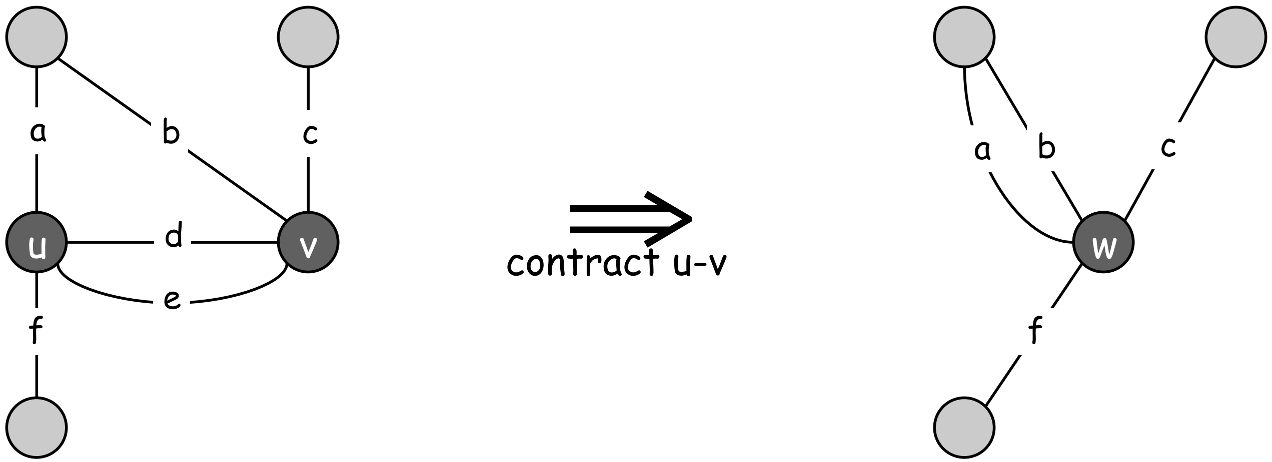
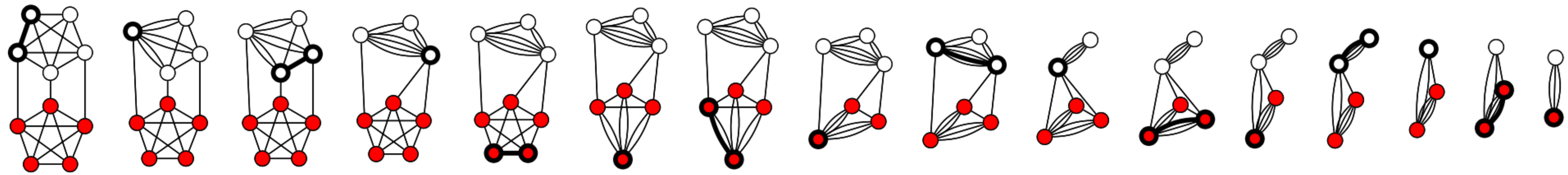
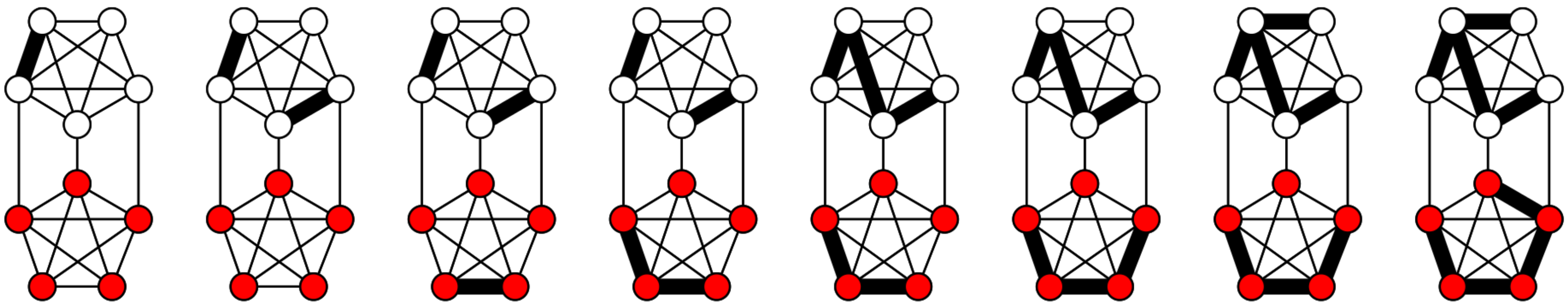


Illustration of Karger's algorithm

Figures by Thore Husfeldt





Contraction Algorithm Analysis

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

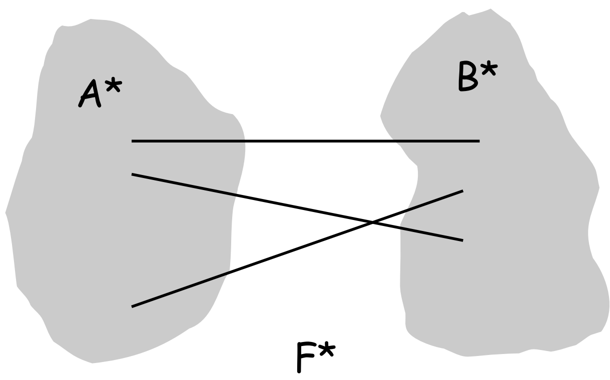
Claim. If we repeat the contraction algorithm $n^2 \ln n$ times with independent random choices, the probability of failing to find the global min-cut is at most $1/n^2$.

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 edge in F^* w. probability $k / |E|$.
- 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 ≥ $2/n^2$.

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, $|E'| \geq \frac{1}{2}kn'$.
- Thus, algorithm contracts an edge in F^* with probability ≤ $2/n'$.

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

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

Contraction Algorithm

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

Claim. If we repeat the contraction algorithm $n^2 \ln n$ times with independent random choices, the probability of failing to find the global min-cut is at most $1/n^2$.

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

$$\left(1 - \frac{2}{n^2}\right)^{n^2 \ln n} = \left[\left(1 - \frac{2}{n^2}\right)^{\frac{1}{2}n^2}\right]^{2\ln n} \underset{\uparrow}{\leq} \left(e^{-1}\right)^{2\ln n} = \frac{1}{n^2}$$

$(1 - 1/x)^x \leq 1/e$

Global Min Cut: Context

Remark. Overall running time is slow since we perform $\Theta(n^2 \log n)$ iterations and each takes $\Omega(n)$ time.

Improvement. [Karger-Stein 1996] $O(n^2 \log^3 n)$.

- Early iterations are less risky than later ones: probability of contracting an edge in min cut hits 50% when $n / \sqrt{2}$ nodes remain.
- Run contraction algorithm until $n / \sqrt{2}$ nodes remain.
- Recursively run contraction algorithm twice on resulting graph, and return best of two cuts.

Best known. [Karger 2000] $O(m \log^3 n)$.

faster than best known max flow algorithm or deterministic global min cut algorithm

13.3 Linearity of Expectation

Expectation

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

$$E[X] = \sum_{j=0}^{\infty} j \Pr[X = j]$$

Example: Waiting for a first success.

Coin is heads with probability p and tails with probability 1-p.

How many independent flips X until first heads?

Expectation: Two Properties

Useful property. If X is a 0/1 random variable, E[X] = Pr[X = 1].

Pf.

$$E[X] = \sum_{j=0}^{\infty} j \cdot \Pr[X = j] = \sum_{j=0}^{1} j \cdot \Pr[X = j] = \Pr[X = 1]$$

not necessarily independent

Linearity of expectation. Given two random variables X and Y defined over the same probability space, E[X + Y] = E[X] + E[Y].

Decouples a complex calculation into simpler pieces.

Guessing Cards

Game. Shuffle a deck of n cards; turn them over one at a time; try to guess each card.

Memoryless guessing. No psychic abilities; can't even remember what's been turned over already. Guess a card from full deck uniformly at random.

Claim. The expected number of correct "memoryless" guesses is 1.

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

Claim. Expected number of correct guesses "with memory" is $\Theta(\log n)$.

Coupon Collector

Coupon collector. Each box of cereal contains a coupon. There are n different types of coupons. Assuming all boxes are equally likely to contain each coupon, how many boxes before you have ≥ 1 coupon of each type?

Claim. The expected number of steps is $\Theta(n \log n)$.

13.5 Randomized Divide-and-Conquer

Quicksort

Recall Quicksort.

```
RandomizedQuicksort(S) {
   if |S| = 0 return

   choose a splitter aᵢ ∈ S uniformly at random
   foreach (a ∈ S) {
      if      (a < aᵢ) put a in S⁻
      else if (a > aᵢ) put a in S⁺
   }
   RandomizedQuicksort(S⁻)
   output aᵢ
   RandomizedQuicksort(S⁺)
}
```

Quicksort

Running time.

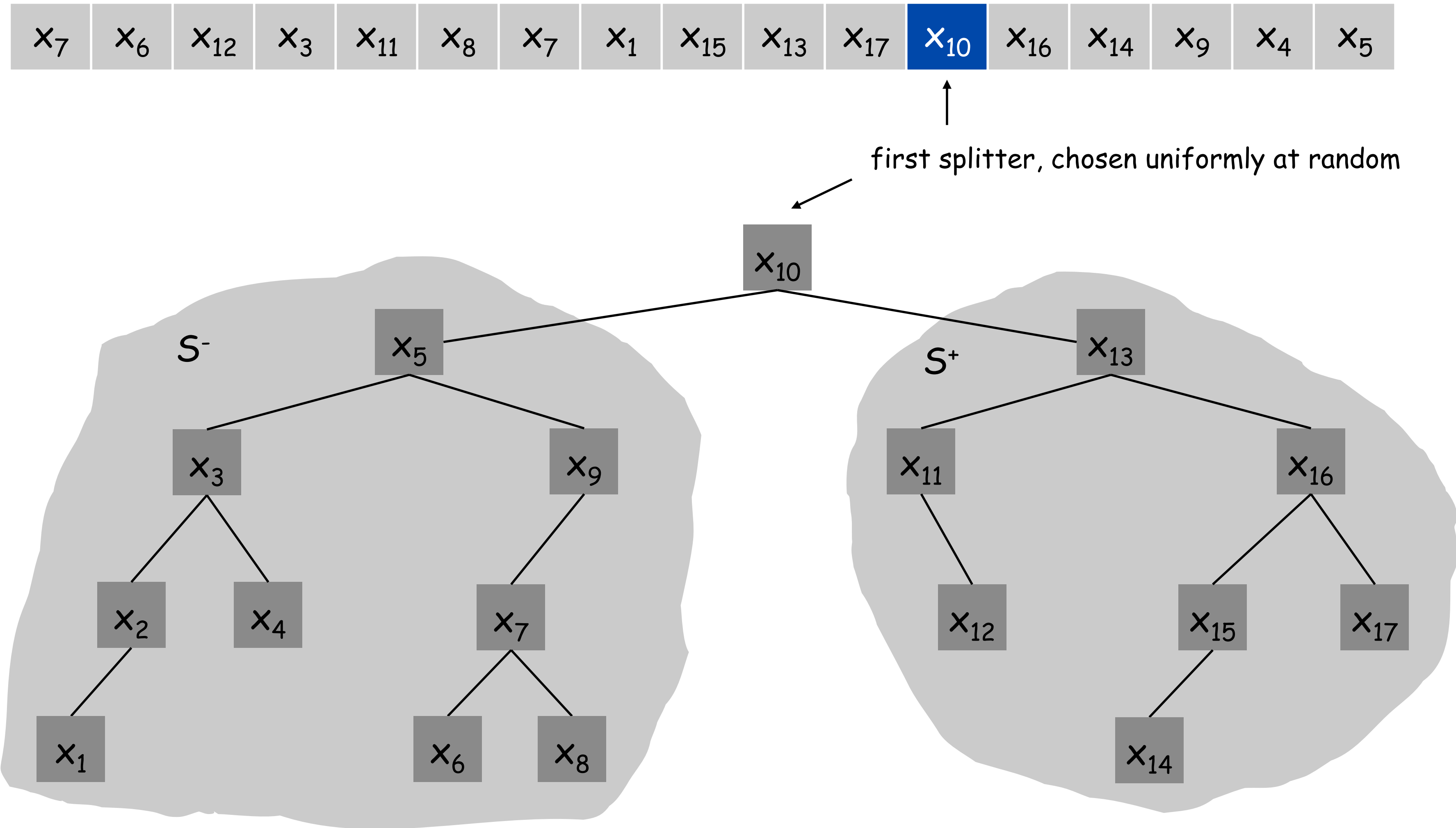
- [Best case.] Always select the median element as the splitter: quicksort makes $\Theta(n \log n)$ comparisons.
- [Worst case.] Always select the smallest element as the splitter: quicksort makes $\Theta(n^2)$ comparisons.

Randomize. Protect against worst case by choosing splitter at random.

Notation. Label elements so that $x_1 < x_2 < \ldots < x_n$.

Quicksort: BST Representation of Splitters

BST representation. Draw recursive BST of splitters.



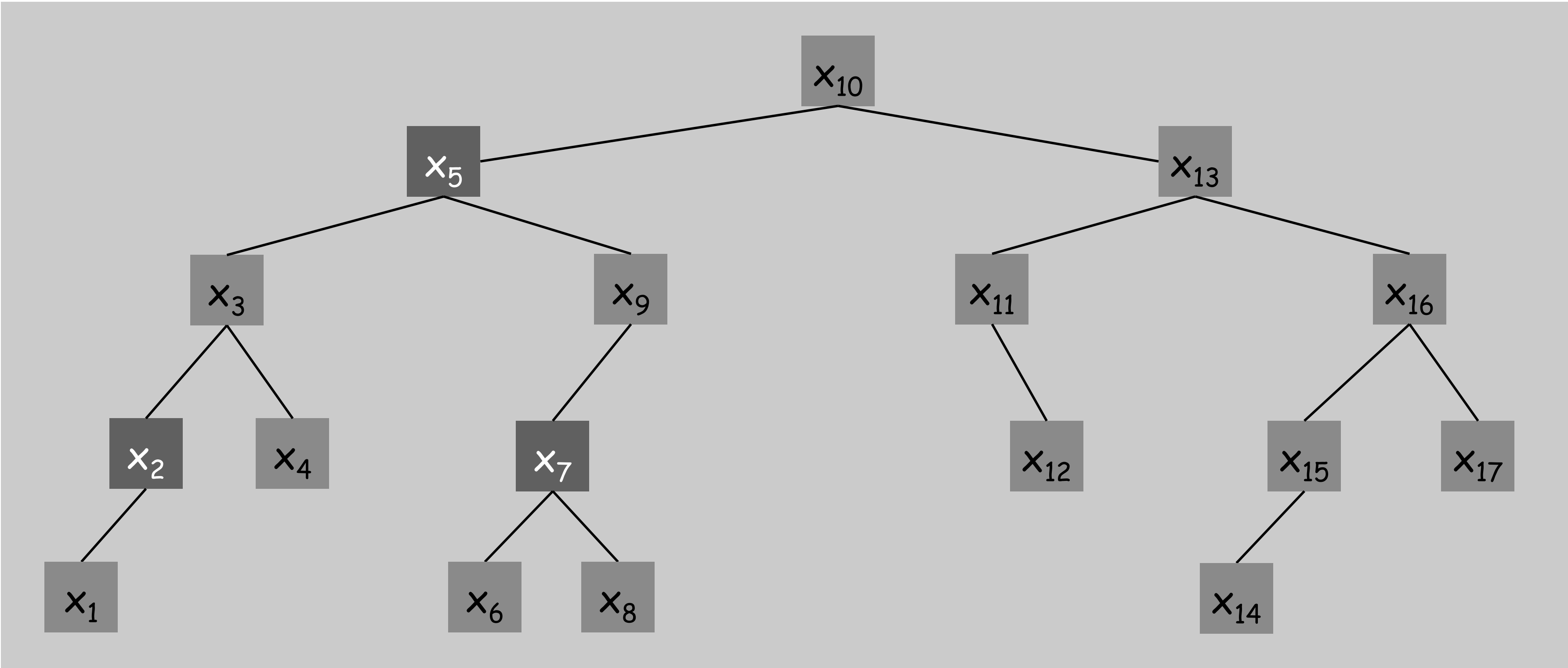
Quicksort: BST Representation of Splitters

Observation. Element only compared with its ancestors and descendants.

- x_2 and x_7 are compared if their lca = x_2 or x_7.
- x_2 and x_7 are not compared if their lca = x_3 or x_4 or x_5 or x_6.

Claim. Pr[x_i and x_j are compared] = $2 / |j - i + 1|$.

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



Quick-select

Partition array so that:

- Entry `a[j]` is in place.
- No larger entry to the left of `j`.
- No smaller entry to the right of `j`.

Repeat in one subarray, depending on `j`; finished when `j` equals `k`.

Proposition. Quick-select takes linear time on average.

```
public static Comparable select(Comparable[] a, int k)
{
    StdRandom.shuffle(a);
    int lo = 0, hi = a.length - 1;
    while (hi > lo)
    {
       int j = partition(a, lo, hi);
       if        (j < k) lo = j + 1;
       else if (j > k) hi = j - 1;
       else             return a[k];
    }
    return a[k];
}
```

if `a[k]` is here set `hi` to `j-1`

if `a[k]` is here set `lo` to `j+1`