Randomized Algorithms, Lecture 1

Jacob Holm (jaho@di.ku.dk)

April 23rd 2019

Good afternoon. My name is Jacob Holm, and I will be your lecturer and course responsible for this course in Randomized Algorithms.

I have taken over the role from professor Mikkel Thorup who designed this course, and this is my first time as course responsible so I hope you will help me to do a good job.

You can do this by telling me if there is anything missing or wrong in the course pages on Absalon, and by asking questions during class if there is anything that is not clear.

Remember, if it is not clear to you, then it is probably also unclear to at least one other person in the room.

You can help more than just yourself by asking for clarification.

Why Randomized Algorithms?

- Faster.
- Simpler code.
- Sometimes only option, e.g. Big Data, Machine Learning, Security, etc.

Why Randomized Algorithms?

- Faster.
- Simpler code.
- Sometimes only option, e.g. Big Data, Machine Learning, Security, etc.

Why Randomized Algorithms?

- Faster.
- Simpler code.
- Sometimes only option, e.g. Big Data, Machine Learning, Security, etc.

Why Randomized Algorithms?

- Faster.
- Simpler code.
- Sometimes only option, e.g. Big Data, Machine Learning, Security, etc.

Why Randomized Algorithms?

- Faster.
- Simpler code.
- Sometimes only option, e.g. Big Data, Machine Learning, Security, etc.

Why Randomized Algorithms?

- Faster, but weaker guarantees.
- Simpler code.
- Sometimes only option, e.g. Big Data, Machine Learning, Security, etc.

Why Randomized Algorithms?

- Faster, but weaker guarantees.
- Simpler code, but harder to analyze.
- Sometimes only option, e.g. Big Data, Machine Learning, Security, etc.

Why Randomized Algorithms?

- Faster, but weaker guarantees.
- Simpler code, but harder to analyze.
- Sometimes only option, e.g. Big Data, Machine Learning, Security, etc.

Therefore this course!

Todays Lecture

- Quicksort
  - Linearity of expectation
  - Expectation of indicator variable
- Min-Cut
  - Conditional probabilities
  - Time/error probability tradeoff
- Las Vegas vs Monte Carlo
- Binary planar partitions
  - Probabilistic method

Basic Quicksort [Hoare]

Q: Does anyone see what essential part is missing from this description?

```
1: function QS(S)
     ▷ Assumes all elements in S are distinct.
2:     if |S| ≤ 1 then
3:         return S
4:     else
5:         Pick pivot x ∈ S
6:         L ← {y ∈ S | y < x}
7:         R ← {y ∈ S | y > x}
8:         return QS(L)+[x]+QS(R)
```

(Line 6–7 uses $|L| + |R| = |S| - 1$ comparisons)

Basic Quicksort [Hoare]

Q: Does anyone see what essential part is missing from this description?

1: **function** $\mathrm{QS}(S)$
 ▷ Assumes all elements in S are distinct.
2: **if** $|S| \leq 1$ **then**
3: **return** S
4: **else**
5: Pick pivot $x \in S$, (How?)
6: $L \leftarrow \{y \in S \mid y < x\}$
7: $R \leftarrow \{y \in S \mid y > x\}$
8: **return** $\mathrm{QS}(L) + [x] + \mathrm{QS}(R)$

(Line 6–7 uses $|L| + |R| = |S| - 1$ comparisons)

Quicksort Example 1

Sorting $S = [70, 12, 34, 47, 9, 72, 60, 7]$.

Total #comparisons:

Quicksort Example 1

Sorting $S = [70, 12, 34, 47, 9, 72, 60, 7]$.

$[12, 34, 9, 7]$ 47 $[70, 72, 60]$

Total #comparisons:

Quicksort Example 1

Sorting $S = [70, 12, 34, 47, 9, 72, 60, 7]$.

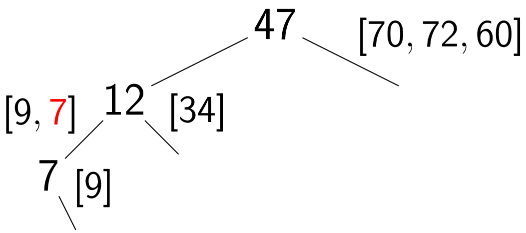
$[12, 34, 9, 7]$ $\quad 47 \quad$ $[70, 72, 60]$

$[9, 7]$ $\quad 12 \quad$ $[34]$

Total #comparisons:

Quicksort Example 1

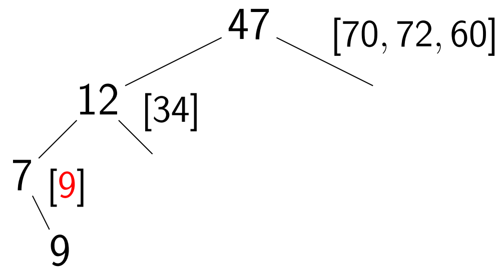
Sorting $S = [70, 12, 34, 47, 9, 72, 60, 7]$.



Total #comparisons:

Quicksort Example 1

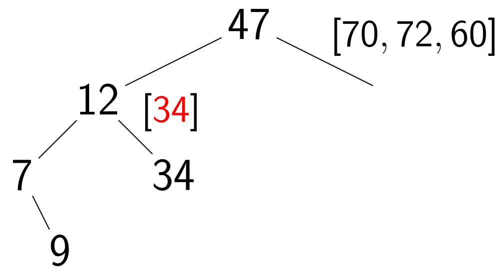
Sorting $S = [70, 12, 34, 47, 9, 72, 60, 7]$.



Total #comparisons:

Quicksort Example 1

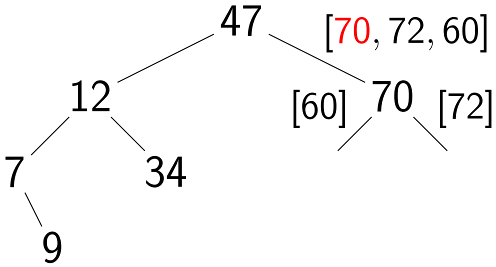
Sorting $S = [70, 12, 34, 47, 9, 72, 60, 7]$.



Total #comparisons:

Quicksort Example 1

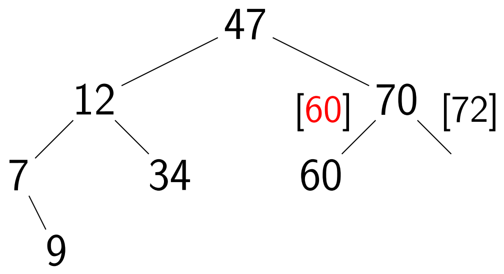
Sorting $S = [70, 12, 34, 47, 9, 72, 60, 7]$.



Total #comparisons:

Quicksort Example 1

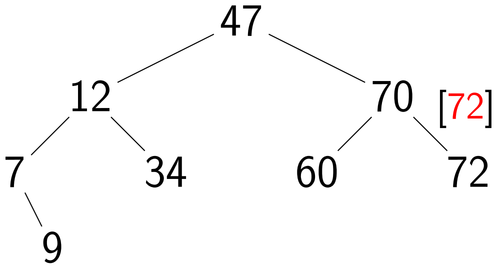
Sorting $S = [70, 12, 34, 47, 9, 72, 60, 7]$.



Total #comparisons:

Quicksort Example 1

Sorting $S = [70, 12, 34, 47, 9, 72, 60, 7]$.

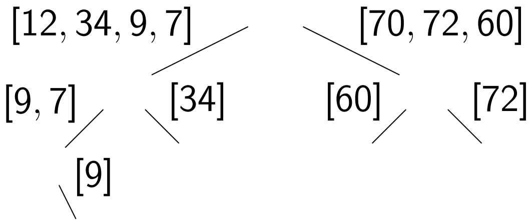


Total #comparisons:

As you can see this looks rather like a balanced binary search tree, so you might expect the number of comparisons to be small. Something like $n \log n$.

Quicksort Example 1

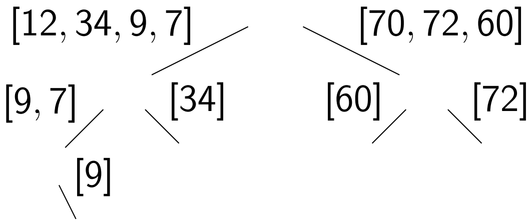
Sorting $S = [70, 12, 34, 47, 9, 72, 60, 7]$.



Total #comparisons:

Quicksort Example 1

Sorting $S = [70, 12, 34, 47, 9, 72, 60, 7]$.



Total #comparisons: 13

Quicksort Example 2

Sorting $S = [70, 12, 34, 47, 9, 72, 60, 7]$.

Total #comparisons:

Quicksort Example 2

Sorting $S = [70, 12, 34, 47, 9, 72, 60, 7]$.

7 $[70, 12, 34, 47, 9, 72, 60]$

Total #comparisons:

Quicksort Example 2

Sorting $S = [70, 12, 34, 47, 9, 72, 60, 7]$.

7 $[70, 12, 34, 47, 9, 72, 60]$

9 $[70, 12, 34, 47, 72, 60]$

Total #comparisons:

Quicksort Example 2

Sorting $S = [70, 12, 34, 47, 9, 72, 60, 7]$.

7

9 $[70, 12, 34, 47, 72, 60]$

12 $[70, 34, 47, 72, 60]$

Total #comparisons:

Quicksort Example 2

Sorting $S = [70, 12, 34, 47, 9, 72, 60, 7]$.

7

9

12 $[70, 34, 47, 72, 60]$

34 $[70, 47, 72, 60]$

Total #comparisons:

Quicksort Example 2

Sorting $S = [70, 12, 34, 47, 9, 72, 60, 7]$.

7

9

12

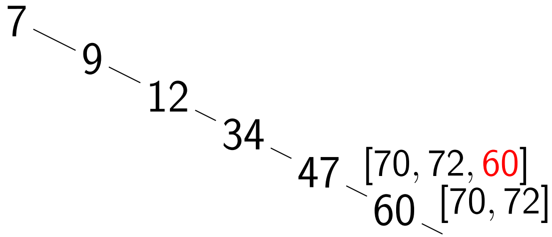
34 $[70, 47, 72, 60]$

47 $[70, 72, 60]$

Total #comparisons:

Quicksort Example 2

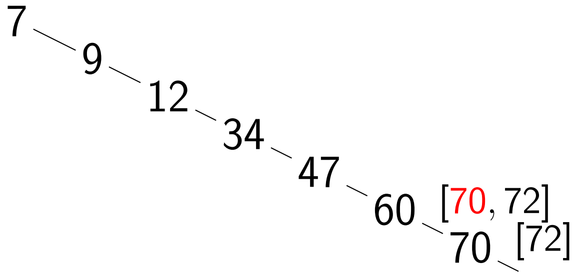
Sorting $S = [70, 12, 34, 47, 9, 72, 60, 7]$.



Total #comparisons:

Quicksort Example 2

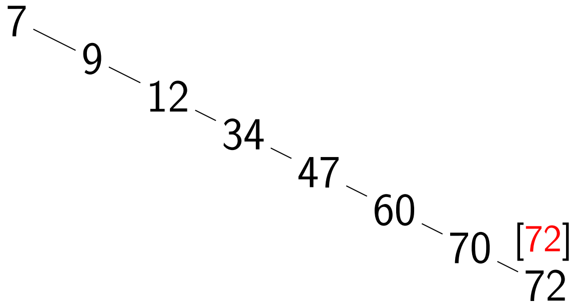
Sorting $S = [70, 12, 34, 47, 9, 72, 60, 7]$.



Total #comparisons:

Quicksort Example 2

Sorting $S = [70, 12, 34, 47, 9, 72, 60, 7]$.



Total #comparisons:

As you can see this looks like an extremely unbalanced binary search tree, so you might expect the number of comparisons to be large. Something like n^2.

Quicksort Example 2

Sorting $S = [70, 12, 34, 47, 9, 72, 60, 7]$.

$[70, 12, 34, 47, 9, 72, 60]$

$[70, 12, 34, 47, 72, 60]$

$[70, 34, 47, 72, 60]$

$[70, 47, 72, 60]$

$[70, 72, 60]$

$[70, 72]$

$[72]$

Total #comparisons:

Quicksort Example 2

Sorting $S = [70, 12, 34, 47, 9, 72, 60, 7]$.

$[70, 12, 34, 47, 9, 72, 60]$

$[70, 12, 34, 47, 72, 60]$

$[70, 34, 47, 72, 60]$

$[70, 47, 72, 60]$

$[70, 72, 60]$

$[70, 72]$

$[72]$

Total #comparisons: 28

Randomized Quicksort

1: **function** $\textsc{RandQS}(S)$
$\triangleright$ Assumes all elements in S are distinct.
2: **if** $|S| \leq 1$ **then**
3: **return** S
4: **else**
5: Pick pivot $x \in S$, uniformly at random
6: $L \leftarrow \{y \in S \mid y < x\}$
7: $R \leftarrow \{y \in S \mid y > x\}$
8: **return** $\textsc{RandQS}(L)+[x]+\textsc{RandQS}(R)$

(Line 6–7 uses $|L| + |R| = |S| - 1$ comparisons)

Randomized Quicksort, Analysis

Q: What is the expected number of comparisons?

Randomized Quicksort, Analysis

Let $[S_{(1)}, \ldots, S_{(n)}] := \textsc{RandQS}(S)$.
For $i < j$ let $X_{ij} \in \{0, 1\}$ be the number of times that $S_{(i)}$ and $S_{(j)}$ are compared.

$$\mathbb{E}[\#\text{comparisons}] = \mathbb{E}\left[\sum_{i<j} X_{ij}\right] = \sum_{i<j} \mathbb{E}[X_{ij}]$$

Uses *linearity of expectation*:

$$\mathbb{E}[A + B] = \mathbb{E}[A] + \mathbb{E}[B]$$

Note that the $\sum_{i<j}$ is really a shorthand for $\sum_{1 \le i < j \le n}$, or even more explicit $\sum_{i=1}^{n-1} \sum_{j=i+1}^{n}$.

Randomized Quicksort, Analysis

Let $[S_{(1)}, \ldots, S_{(n)}] := \textsc{RandQS}(S)$.
For $i < j$ let $X_{ij} \in \{0, 1\}$ be the number of times that $S_{(i)}$ and $S_{(j)}$ are compared.

$$\mathbb{E}[\#\text{comparisons}] = \mathbb{E}\left[\sum_{i<j} X_{ij}\right] = \sum_{i<j} \mathbb{E}[X_{ij}]$$

Uses *linearity of expectation*:

$$\mathbb{E}[A + B] = \mathbb{E}[A] + \mathbb{E}[B]$$

Note that the $\sum_{i<j}$ is really a shorthand for $\sum_{1 \le i < j \le n}$, or even more explicit $\sum_{i=1}^{n-1} \sum_{j=i+1}^{n}$.

Randomized Quicksort, Analysis

Since $X_{ij} \in \{0, 1\}$, it is an *indicator variable* for the event that $S_{(i)}$ and $S_{(j)}$ are compared. Let p_{ij} be the probability of this event. Then

$$\mathbb{E}[X_{ij}] = 0 \cdot (1 - p_{ij}) + 1 \cdot p_{ij} = p_{ij}$$

Thus *the expectation of an indicator variable equals the probability of the indicated event*.

$$\mathbb{E}[\#\text{comparisons}] = \sum_{i<j} \mathbb{E}[X_{ij}] = \sum_{i<j} p_{ij}$$

Randomized Quicksort, Analysis

Lemma

$S_{(i)}$ and $S_{(j)}$ are compared iff $S_{(i)}$ or $S_{(j)}$ is first of $S_{(i)}, \ldots, S_{(j)}$ to be chosen as pivot.

Randomized Quicksort, Analysis

Proof.

Each recursive call returns $[S_{(a)}, \ldots, S_{(b)}]$.

Let $x = S_{(c)}$ be the pivot.

Suppose $a \leq i < j \leq b$. $\begin{array}{|c|c|c|c|c|c|c|}\hline a & \cdots & i & \cdots & j & \cdots & b \\ \hline \end{array}$

$c < i$ or $c > j$: $S_{(i)}$ and $S_{(j)}$ not compared now, but together in recursion.

$i < c < j$: $S_{(i)}$ and $S_{(j)}$ never compared.

$c = i$ or $c = j$: $S_{(i)}$ and $S_{(j)}$ compared once.

So decision only made when $i \leq c \leq j$. $\square$

Randomized Quicksort, Analysis

Thus

$$p_{ij} = \frac{2}{j+1-i}$$

And

$$\mathbb{E}[\#\text{comparisons}] = \sum_{i<j} p_{ij} = \sum_{i<j} \frac{2}{j+1-i}$$

Randomized Quicksort, Analysis

$$\mathbb{E}[\#\text{comparisons}] = \sum_{i<j} \frac{2}{j+1-i}$$

$$= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j+1-i}$$

$$= \sum_{i=1}^{n-1} \sum_{k=2}^{n+1-i} \frac{2}{k}$$

$$< \sum_{i=1}^{n} \sum_{k=1}^{n} \frac{2}{k} = \sum_{i=1}^{n} 2H_n = 2nH_n$$

Randomized Quicksort, Analysis

$$
\begin{aligned}
\mathbb{E}[\#\text{comparisons}] &= \sum_{i<j} \frac{2}{j+1-i} \\
&= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j+1-i} \\
&= \sum_{i=1}^{n-1} \sum_{k=2}^{n+1-i} \frac{2}{k} \\
&< \sum_{i=1}^{n} \sum_{k=1}^{n} \frac{2}{k} = \sum_{i=1}^{n} 2H_n = 2nH_n
\end{aligned}
$$

Randomized Quicksort, Analysis

$$
\begin{aligned}
\mathbb{E}[\#\text{comparisons}] &= \sum_{i<j} \frac{2}{j+1-i} \\
&= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j+1-i} \\
&= \sum_{i=1}^{n-1} \sum_{k=2}^{n+1-i} \frac{2}{k} \\
&< \sum_{i=1}^{n} \sum_{k=1}^{n} \frac{2}{k} = \sum_{i=1}^{n} 2H_n = 2nH_n
\end{aligned}
$$

Randomized Quicksort, Analysis

$$
\begin{aligned}
\mathbb{E}[\#\text{comparisons}] &= \sum_{i<j} \frac{2}{j+1-i} \\
&= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j+1-i} \\
&= \sum_{i=1}^{n-1} \sum_{k=2}^{n+1-i} \frac{2}{k} \\
&< \sum_{i=1}^{n} \sum_{k=1}^{n} \frac{2}{k} = \sum_{i=1}^{n} 2H_n = 2nH_n
\end{aligned}
$$

Randomized Quicksort, Analysis

$$
\begin{aligned}
\mathbb{E}[\#\text{comparisons}] &= \sum_{i<j} \frac{2}{j+1-i} \\
&= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j+1-i} \\
&= \sum_{i=1}^{n-1} \sum_{k=2}^{n+1-i} \frac{2}{k} \\
&< \sum_{i=1}^{n} \sum_{k=1}^{n} \frac{2}{k} = \sum_{i=1}^{n} 2H_n = 2nH_n
\end{aligned}
$$

Randomized Quicksort, Analysis

$$
\begin{aligned}
\mathbb{E}[\#\text{comparisons}] &= \sum_{i<j} \frac{2}{j+1-i} \\
&= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j+1-i} \\
&= \sum_{i=1}^{n-1} \sum_{k=2}^{n+1-i} \frac{2}{k} \\
&< \sum_{i=1}^{n} \sum_{k=1}^{n} \frac{2}{k} = \sum_{i=1}^{n} 2H_n = 2{\color{red}n}H_n
\end{aligned}
$$

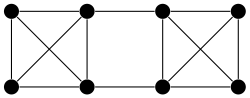
Randomized Quicksort, Summary

When $|S| = n$, the expected number of comparisons done by $\textsc{RandQS}(S)$ is less than $2nH_n \in \mathcal{O}(n \log n)$ *for any input*.

Even stronger (see Problem 4.14), we can show that the number of comparisons is $\mathcal{O}(n \log n)$ *with high probability*.

Min-Cut

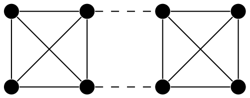
Problem: Given a connected graph $G = (V, E)$



Find smallest $C \subseteq E$ that splits G.
C is called a *min-cut*, and $\lambda(G) := |C|$ is the *edge connectivity* of G.

Min-Cut

Problem: Given a connected graph
$G = (V, E)$

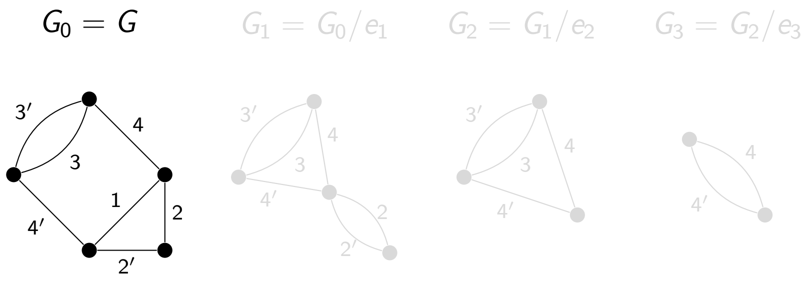


Find smallest $C \subseteq E$ that splits G.
C is called a *min-cut*, and $\lambda(G) := |C|$ is the *edge connectivity* of G.

Randomized Min-Cut [Karger & Stein]

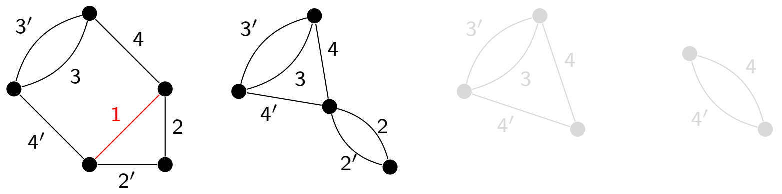
function $\textsc{RandMinCut}(V, E)$
  while $|V| > 2$ **do**
    Pick $e \in E$ uniformly at random.
    Contract e and remove self-loops.
  return E

Randomized Min-Cut, Example



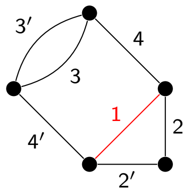
Randomized Min-Cut, Example

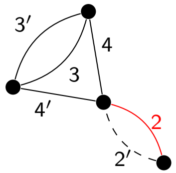
$G_0 = G$ $\quad$ $G_1 = G_0/e_1$ $\quad$ $G_2 = G_1/e_2$ $\quad$ $G_3 = G_2/e_3$

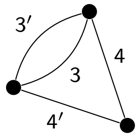


Randomized Min-Cut, Example

$G_0 = G$ $\quad$ $G_1 = G_0/e_1$ $\quad$ $G_2 = G_1/e_2$ $\quad$ $G_3 = G_2/e_3$

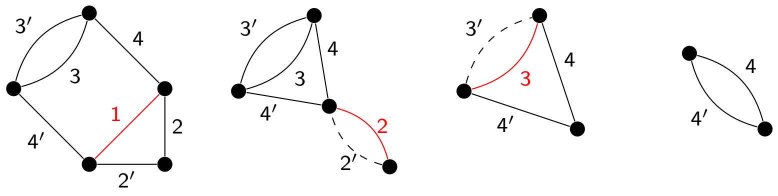






Randomized Min-Cut, Example

$G_0 = G$ $\quad$ $G_1 = G_0/e_1$ $\quad$ $G_2 = G_1/e_2$ $\quad$ $G_3 = G_2/e_3$



Randomized Min-Cut, Analysis

Observation

$\textsc{RandMinCut}(G)$ *may return a cut of size* $> \lambda(G)$.

Lemma

A specific min-cut C is returned iff no edge from C was contracted.

Randomized Min-Cut, Analysis

Theorem

For any min-cut C, the probability that $\textsc{RandMinCut}(G)$ *returns C is* $\geq \frac{2}{n(n-1)}$.

Randomized Min-Cut, Proof

Let $e_1, \ldots, e_{n-2}$ be the contracted edges, let
$G_0 = G$ and $G_i = G_{i-1}/e_i$.
Define $\mathcal{E}_i := [e_i \notin C]$.
C is returned iff $\mathcal{E}_1 \cap \cdots \cap \mathcal{E}_{n-2}$.

Conditional Probabilities

Given events $\mathcal{E}_1, \mathcal{E}_2$, the *conditional probability* of $\mathcal{E}_2$ given $\mathcal{E}_1$ is defined as

$$\Pr[\mathcal{E}_2|\mathcal{E}_1] = \frac{\Pr[\mathcal{E}_1 \cap \mathcal{E}_2]}{\Pr[\mathcal{E}_1]}$$

It follows that

$$\Pr[\mathcal{E}_1 \cap \mathcal{E}_2] = \Pr[\mathcal{E}_1] \cdot \Pr[\mathcal{E}_2|\mathcal{E}_1]$$

And in general for events $\mathcal{E}_1, \ldots, \mathcal{E}_k$

$$\Pr[\cap_{i=1}^k \mathcal{E}_i] = \Pr[\mathcal{E}_1] \cdot \Pr[\mathcal{E}_2|\mathcal{E}_1] \cdots \Pr[\mathcal{E}_k| \cap_{i=1}^{k-1} \mathcal{E}_i]$$

This is easy to prove by induction.

Randomized Min-Cut, Proof

$$
\begin{aligned}
&\Pr[C \text{ returned}] \\
&\quad = \Pr[\mathcal{E}_1 \cap \cdots \cap \mathcal{E}_{n-2}] \\
&\quad = \Pr[\mathcal{E}_1] \cdot \Pr[\mathcal{E}_2 | \mathcal{E}_1] \cdots \Pr[\mathcal{E}_{n-2} | \mathcal{E}_1 \cap \cdots \cap \mathcal{E}_{n-3}] \\
&\quad = \prod_{i=1}^{n-2} p_i \quad \text{where } p_i = \Pr[\mathcal{E}_i | \mathcal{E}_1 \cap \cdots \cap \mathcal{E}_{i-1}]
\end{aligned}
$$

Randomized Min-Cut, Proof

$G_i = (V_i, E_i)$ has $n_i = n - i$ vertices and $\lambda(G_i) \geq |C|$.
So $|E_i| = \frac{1}{2}\sum_{v \in V_i} d(v) \geq \frac{1}{2} n_i |C|$, and

$$\begin{aligned}
1 - p_i &= \Pr[\text{random } e \in E_{i-1} \text{ is in } C \mid \cap_{j=1}^{i-1} \mathcal{E}_j] \\
&= \frac{|C|}{|E_{i-1}|} \leq \frac{|C|}{\frac{1}{2} n_{i-1} |C|} = \frac{2}{n_{i-1}} = \frac{2}{n - (i-1)} \\
p_i &\geq 1 - \frac{2}{n+1-i} = \frac{n-1-i}{n+1-i}
\end{aligned}$$

We use that contractions can never decrease the min-cut.
We use the degree-sum formula for graphs, that $\sum_{v \in V} d(v) = 2|E|$.
We also use the assumption that no edge from C has been contracted yet.

Randomized Min-Cut, Proof

Telescoping product.

$$
\begin{aligned}
&\Pr[C \text{ returned}] \\
&\quad = \prod_{i=1}^{n-2} p_i \quad \text{where } p_i = \Pr[\mathcal{E}_i | \mathcal{E}_1 \cap \cdots \cap \mathcal{E}_{i-1}] \\
&\quad \geq \prod_{i=1}^{n-2} \frac{n-1-i}{n+1-i} \\
&\quad = \frac{n-2}{n} \cdot \frac{n-3}{n-1} \cdot \frac{n-4}{n-2} \cdots \frac{3}{5} \cdot \frac{2}{4} \cdot \frac{1}{3} \\
&\quad = \frac{2}{n(n-1)}
\end{aligned}
$$

Randomized Min-Cut, Proof

Telescoping product.

$$
\begin{aligned}
&\Pr[C \text{ returned}] \\
&\quad = \prod_{i=1}^{n-2} p_i \quad \text{where } p_i = \Pr[\mathcal{E}_i | \mathcal{E}_1 \cap \cdots \cap \mathcal{E}_{i-1}] \\
&\quad \geq \prod_{i=1}^{n-2} \frac{n-1-i}{n+1-i} \\
&\quad = \frac{n-2}{n} \cdot \frac{n-3}{n-1} \cdot \frac{n-4}{n-2} \cdots \frac{3}{5} \cdot \frac{2}{4} \cdot \frac{1}{3} \\
&\quad = \frac{2}{n(n-1)}
\end{aligned}
$$

Randomized Min-Cut, Proof

Telescoping product.

$$
\begin{aligned}
&\Pr[C \text{ returned}] \\
&\quad = \prod_{i=1}^{n-2} p_i \quad \text{where } p_i = \Pr[\mathcal{E}_i | \mathcal{E}_1 \cap \cdots \cap \mathcal{E}_{i-1}] \\
&\quad \geq \prod_{i=1}^{n-2} \frac{n-1-i}{n+1-i} \\
&\quad = \frac{n-2}{n} \cdot \frac{n-3}{n-1} \cdot \frac{n-4}{n-2} \cdots \frac{3}{5} \cdot \frac{2}{4} \cdot \frac{1}{3} \\
&\quad = \frac{2}{n(n-1)}
\end{aligned}
$$

Randomized Min-Cut, Summary

So for min-cut C, $\Pr[C \text{ is returned}] \geq \frac{2}{n(n-1)}$.

Is this good?

How can we improve it?

Randomized Min-Cut, Summary

So for min-cut C, $\Pr[C \text{ is returned}] \geq \frac{2}{n(n-1)}$.

Is this good?

How can we improve it?

Randomized Min-Cut, Summary

So for min-cut C, $\Pr[C \text{ is returned}] \geq \frac{2}{n(n-1)}$.

Is this good?

How can we improve it?

Randomized Min-Cut, Tradeoff

Imagine calling $\textsc{RandMinCut}(G)$ $t\frac{n(n-1)}{2}$ times and taking smallest cut returned.

$$\begin{aligned}\Pr[\text{not a min-cut}] &\leq \left(1 - \frac{2}{n(n-1)}\right)^{t\frac{n(n-1)}{2}} \\ &\leq \left(e^{-\frac{2}{n(n-1)}}\right)^{t\frac{n(n-1)}{2}} \\ &= e^{-t}\end{aligned}$$

(This uses that $1 + x \leq e^x$ for all $x \in \mathbb{R}$).

In each call ro $\textsc{RandMinCut}$, the probability that C is not returned is $1 - \frac{2}{n(n-1)}$.
Each call to $\textsc{RandMinCut}$ is independent.
Thus, the probability that C is not among the cuts returned is the product.
If C is among the cuts considered, the returned cut is minimal.

Choosing e.g. $t = 21$ we reduce the error probability to less than one in a billion.

Las Vegas vs Monte Carlo

What is the main difference between
RandQS and RandMinCut?

Las Vegas vs Monte Carlo

What is the main difference between
RandQS and RandMinCut?

Las Vegas: Always returns correct answer.
#steps used is a random variable.

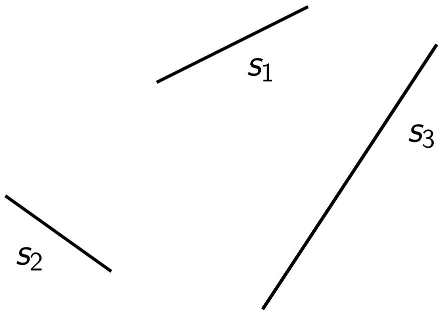
Monte Carlo: Some probability of error.
#steps used may be random or not.

Binary Planar Partitions

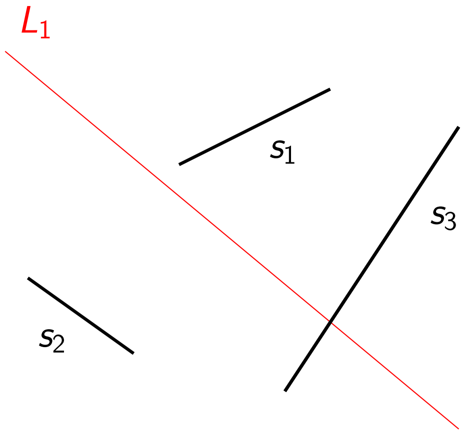
Given non-intersecting line segments $S = \{s_1, \ldots, s_n\}$ in the plane, construct a binary tree where:

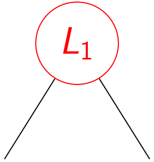
- Each node v has a region $r(v)$ of the plane.
- Each internal node v has a line $\ell(v)$ that intersects $r(v)$ and partitions it into the regions of its two children.
- The root is the whole plane.
- For every leaf v, $r(v)$ intersects at most one s_i.

Binary Planar Partitions, Example

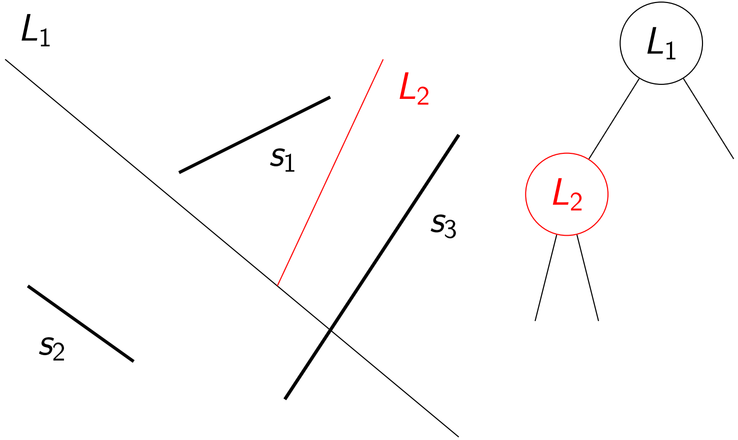


Binary Planar Partitions, Example

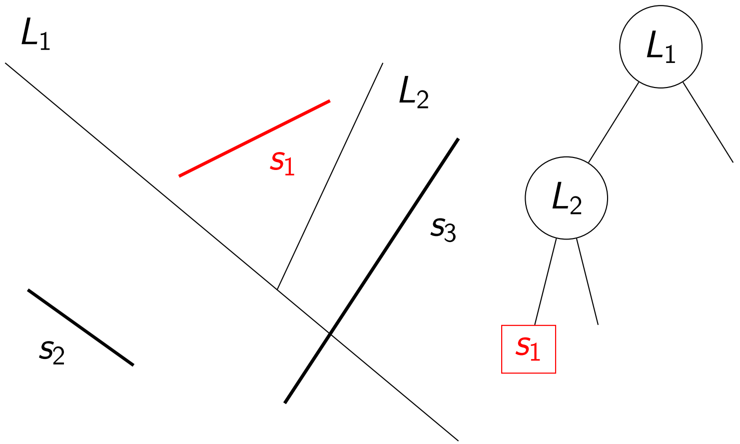




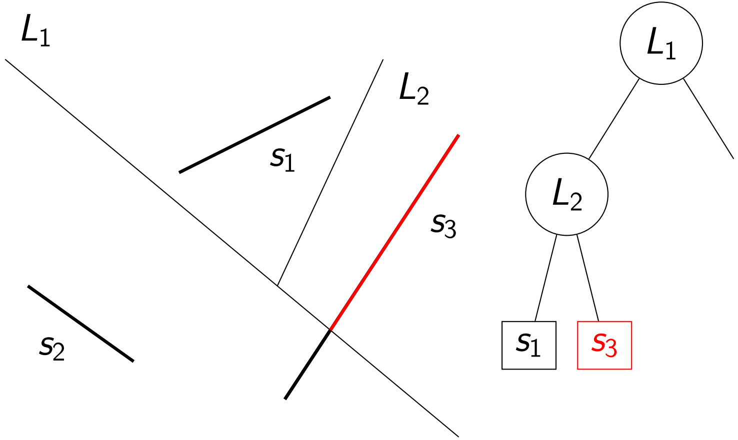
Binary Planar Partitions, Example



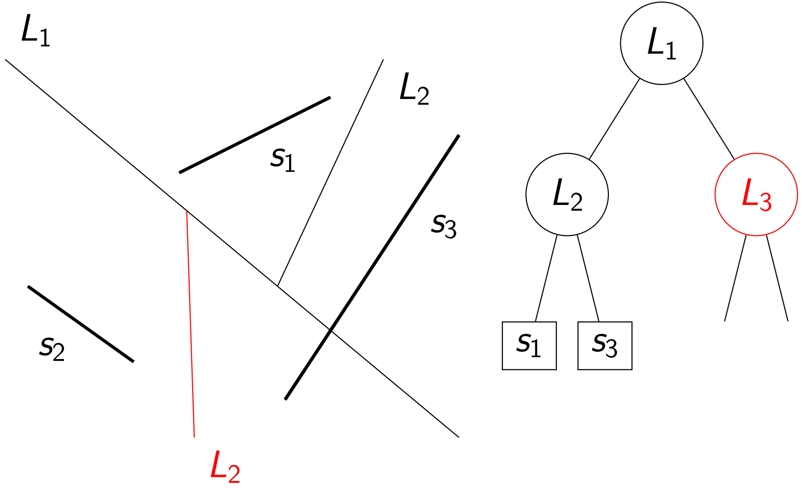
Binary Planar Partitions, Example



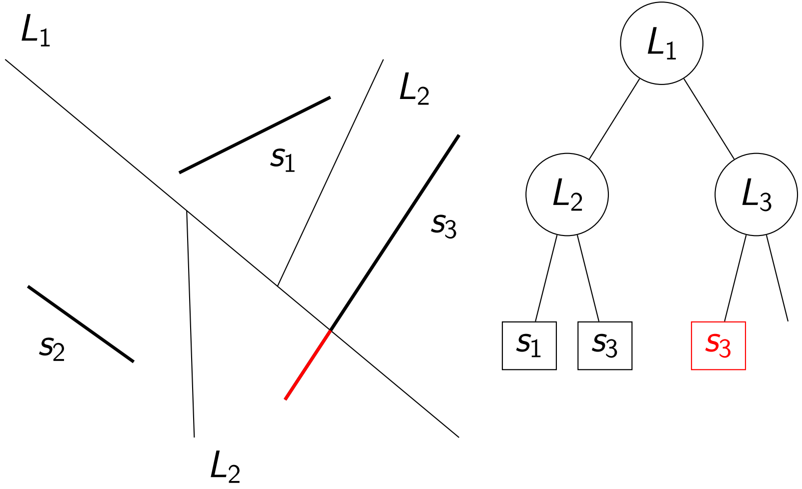
Binary Planar Partitions, Example



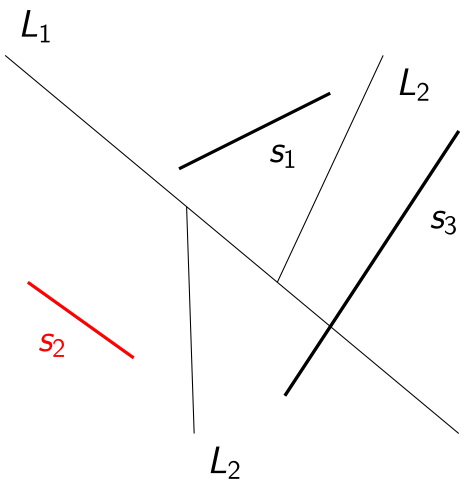
Binary Planar Partitions, Example

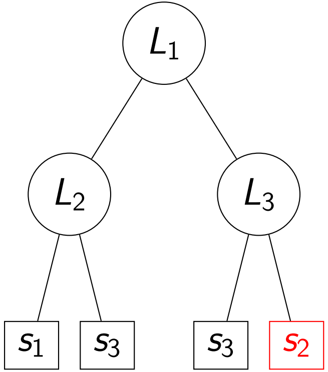


Binary Planar Partitions, Example



Binary Planar Partitions, Example





BPP Applications

Allows efficient queries to find all segments intersecting a given *ray*, sorted by distance. This can be used in a 2D-version of the so-called painters algorithm.
3D version is almost the same, but we won't cover it in this course.

Simply traverse the BPP tree top down, and for each node v first traverse the subtree corresponding to the region containing the start of the ray, then the other subtree.

Autopartitions

If every line used in a BPP contains one of the segments, it is called an *autopartition*.

RandAuto

1: **function** RANDAUTO($S = \{s_1, \ldots, s_n\}$)
2: Pick a permutation π of $\{1, \ldots, n\}$ uniformly at random.
3: **while** Some region r is not done **do**
4: Cut r with $\ell(s_i)$ where i is first in π such that s_i intersects r.
5: Let s_i count as intersecting only the least loaded of the two new regions.
6: **return** The resulting autopartition.

Theorem

The expected size of the autopartition produced by RANDAUTO *is* $\mathcal{O}(n \log n)$.

In line 4, each segment intersecting r may be cut into two pieces by s.

RandAuto

1: **function** $\textsc{RandAuto}(S = \{s_1, \ldots, s_n\})$
2: Pick a permutation π of $\{1, \ldots, n\}$ uniformly at random.
3: **while** Some region r is not done **do**
4: Cut r with $\ell(s_i)$ where i is first in π such that s_i intersects r.
5: Let s_i count as intersecting only the least loaded of the two new regions.
6: **return** The resulting autopartition.

Theorem

The expected size of the autopartition produced by $\textsc{RandAuto}$ *is* $\mathcal{O}(n \log n)$.

In line 4, each segment intersecting r may be cut into two pieces by s.

RandAuto Analysis

For distinct segments u and v, define

$$\mathrm{index}(u, v) := \begin{cases} 1+\#\text{segments hit by } \ell(u) \text{ before hitting } v & \text{if } \ell(u) \text{ hits } v \\ \infty & \text{otherwise} \end{cases}$$

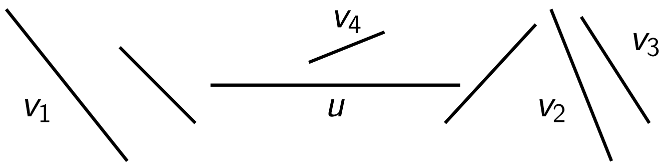
What is $\mathrm{index}(u, v_1), \ldots, \mathrm{index}(u, v_4)$?

RandAuto Analysis

What is index$(u, v_1), \ldots,$index(u, v_4)?

For distinct segments u and v, define

$$\text{index}(u,v) := \begin{cases} 1+\#\text{segments hit by } \ell(u) \text{ before hitting } v & \text{if } \ell(u) \text{ hits } v \\ \infty & \text{otherwise} \end{cases}$$

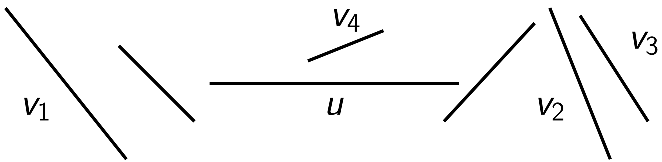


RandAuto Analysis

What is index$(u, v_1), \dots,$index(u, v_4)?

For distinct segments u and v, define

$$\text{index}(u, v) := \begin{cases} 1+\#\text{segments hit by } \ell(u) \text{ before hitting } v & \text{if } \ell(u) \text{ hits } v \\ \infty & \text{otherwise} \end{cases}$$



Here index$(u, v_1) =$ index$(u, v_2) = 2$,

index$(u, v_3) = 3$, and index$(u, v_4) = \infty$

RandAuto Analysis

Let $u \dashv v$ be the event that $\ell(u)$ cuts v.
Let $i = \text{index}(u, v)$ and let $\{u_1, \ldots, u_{i-1}\}$ be the segments hit by u before v.
$u \dashv v$ happens iff u occurs before any of $\{u_1, \ldots, u_{i-1}, v\}$ in π, so $\Pr[u \dashv v] = \frac{1}{i+1}$.
Let C_{uv} indicate that $u \dashv v$, then
$\mathbb{E}[C_{uv}] = \Pr[u \dashv v] = \frac{1}{\text{index}(u,v)+1}$.

We are cheating here by ignoring the case where $\text{index}(u, v) = \infty$, but in that case we still have $\mathbb{E}[C_{uv}] = 0 = 1/\infty$ so we get the correct result.

RandAuto Analysis

The total number of segments in the result is n plus #cuts, so

$$\begin{aligned}
\mathbb{E}[\#segments] &= n + \mathbb{E}\Big[\sum_u \sum_v C_{uv}\Big] \\
&= n + \sum_u \sum_v \mathbb{E}[C_{uv}] \\
&= n + \sum_u \sum_v \frac{1}{\text{index}(u, v) + 1} \\
&\leq n + \sum_u \sum_{i=1}^{n-1} \frac{2}{i + 1} \\
&\leq n + 2nH_n \in \mathcal{O}(n \log n)
\end{aligned}$$

RandAuto Summary

The expected size of the returned autopartition is $\mathcal{O}(n \log n)$ for *any* input.

But then (surprise!) there must *exist* an autopartition of size $\mathcal{O}(n \log n)$ for *every* input.

This is an example of the *probabilistic method*. More on that in Lecture 8 and 9.