Part II

Background

Geometry of BST [Demaine, Harmon, Iacono, Kane, Pătrașcu, SODA'09]

Geometry of BST [Demaine, Harmon, Iacono, Kane, Pătrașcu, SODA'09]

access sequence X
e.g. 4, 5, 6, 1, 2, 3

Geometry of BST [Demaine, Harmon, Iacono, Kane, Pătrașcu, SODA'09]

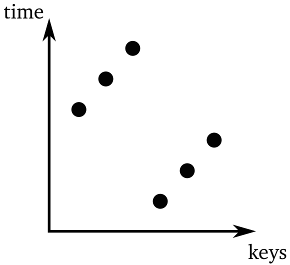
access sequence X
e.g. 4, 5, 6, 1, 2, 3

$\rightarrow$ point set X

Geometry of BST [Demaine, Harmon, Iacono, Kane, Pătrașcu, SODA'09]

access sequence X
e.g. 4, 5, 6, 1, 2, 3

$\rightarrow$ point set X

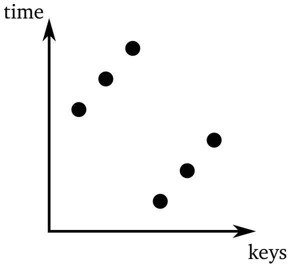


Geometry of BST [Demaine, Harmon, Iacono, Kane, Pătrașcu, SODA'09]

access sequence X
e.g. 4, 5, 6, 1, 2, 3

$\rightarrow$ point set X

BST algorithm serving X



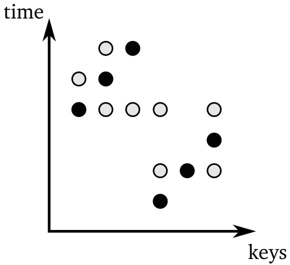
Geometry of BST [Demaine, Harmon, Iacono, Kane, Pătrașcu, SODA'09]

access sequence X
e.g. 4, 5, 6, 1, 2, 3

$\rightarrow$ point set X

BST algorithm serving X

$\rightarrow$ point set $Y \supseteq X$



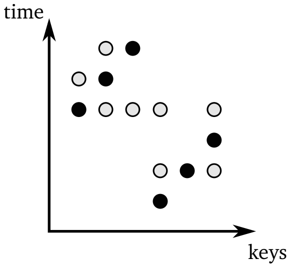
Geometry of BST [Demaine, Harmon, Iacono, Kane, Pătrașcu, SODA'09]

access sequence X
e.g. 4, 5, 6, 1, 2, 3

$\rightarrow$ point set X

BST algorithm serving X

$\rightarrow$ point set $Y \supseteq X$
$\uparrow$
nodes touched by access and rotations at each time



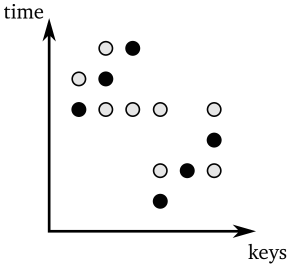
Geometry of BST [Demaine, Harmon, Iacono, Kane, Pătrașcu, SODA'09]

access sequence X
e.g. 4, 5, 6, 1, 2, 3

$\rightarrow$ point set X

BST algorithm serving X

$\rightarrow$ point set $Y \supseteq X$
$\uparrow$
nodes touched by access and rotations at each time



Y is a BST execution of X $\iff$ Y is a satisfied superset of X

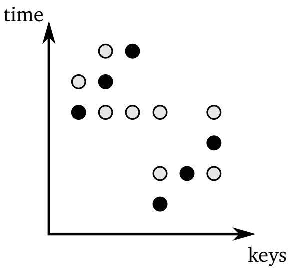
Geometry of BST [Demaine, Harmon, Iacono, Kane, Pătrașcu, SODA'09]

access sequence X
e.g. 4, 5, 6, 1, 2, 3

$\rightarrow$ point set X

BST algorithm serving X

$\rightarrow$ point set $Y \supseteq X$
$\uparrow$
nodes touched by access and rotations at each time



Y is a BST execution of X $\iff$ Y is a satisfied superset of X

Geometry of BST [Demaine, Harmon, Iacono, Kane, Pătrașcu, SODA'09]

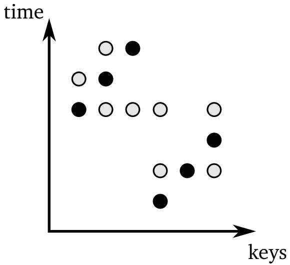
access sequence X
e.g. 4, 5, 6, 1, 2, 3

$\rightarrow$ point set X

BST algorithm serving X

$\rightarrow$ point set $\underset{\substack{\uparrow}}{Y} \supseteq X$

nodes touched by access and rotations at each time



Y is a BST execution of X $\iff$ Y is a satisfied superset of X

$\downarrow$

no $a, b \in Y$ form an empty rectangle

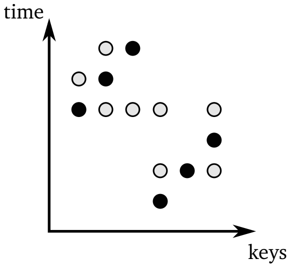
Geometry of BST [Demaine, Harmon, Iacono, Kane, Pătrașcu, SODA'09]

access sequence X
e.g. 4, 5, 6, 1, 2, 3

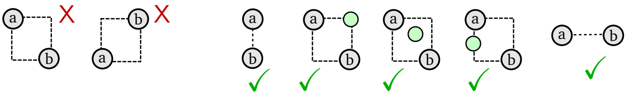
$\rightarrow$ point set X

BST algorithm serving X

$\rightarrow$ point set $Y \supseteq X$
$\uparrow$
nodes touched by access and rotations at each time



Y is a BST execution of X $\iff$ Y is a satisfied superset of X
$\downarrow$
no $a, b \in Y$ form an empty rectangle



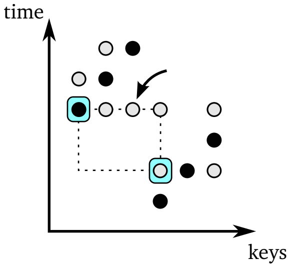
Geometry of BST [Demaine, Harmon, Iacono, Kane, Pătrașcu, SODA'09]

access sequence X
e.g. 4, 5, 6, 1, 2, 3

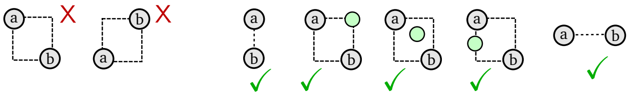
$\rightarrow$ point set X

BST algorithm serving X

$\rightarrow$ point set $Y \supseteq X$
$\uparrow$
nodes touched by access and rotations at each time



Y is a BST execution of X $\iff$ Y is a satisfied superset of X
$\downarrow$
no $a, b \in Y$ form an empty rectangle



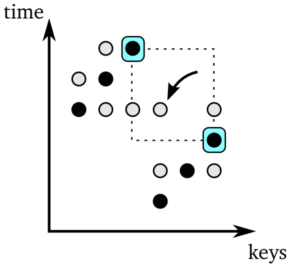
Geometry of BST [Demaine, Harmon, Iacono, Kane, Pătrașcu, SODA'09]

access sequence X
e.g. 4, 5, 6, 1, 2, 3

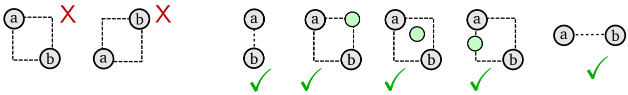
$\rightarrow$ point set X

BST algorithm serving X

$\rightarrow$ point set $Y \supseteq X$
$\uparrow$
nodes touched by access and rotations at each time



Y **is a BST execution of** X $\iff$ Y **is a satisfied superset of** X
$\downarrow$
no $a, b \in Y$ form an empty rectangle



GREEDY [Lucas '88; Munro '00; Demaine et al. '09]

GREEDY [Lucas '88; Munro '00; Demaine et al. '09]

GREEDY:
a natural offline BST algorithm.

GREEDY [Lucas '88; Munro '00; Demaine et al. '09]

GREEDY:
a natural offline BST algorithm.

In geometric view GREEDY
becomes:

GREEDY [Lucas '88; Munro '00; Demaine et al. '09]

GREEDY:
a natural offline BST algorithm.

In geometric view GREEDY
becomes:

a natural **online** algorithm.

GREEDY [Lucas '88; Munro '00; Demaine et al. '09]

GREEDY:
a natural offline BST algorithm.

In geometric view GREEDY
becomes:

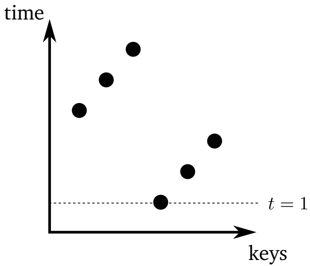
a natural **online** algorithm.
(a simple geometric sweepline)

GREEDY [Lucas '88; Munro '00; Demaine et al. '09]

GREEDY:
a natural offline BST algorithm.

In geometric view GREEDY becomes:

a natural **online** algorithm.
(a simple geometric sweepline)

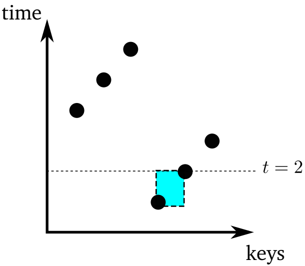


GREEDY [Lucas '88; Munro '00; Demaine et al. '09]

GREEDY:
a natural offline BST algorithm.

In geometric view GREEDY becomes:

a natural **online** algorithm.
(a simple geometric sweepline)

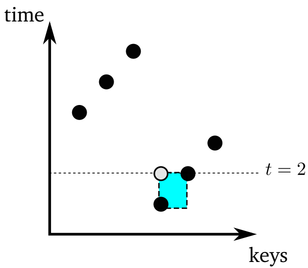


GREEDY [Lucas '88; Munro '00; Demaine et al. '09]

GREEDY:
a natural offline BST algorithm.

In geometric view GREEDY becomes:

a natural **online** algorithm.
(a simple geometric sweepline)

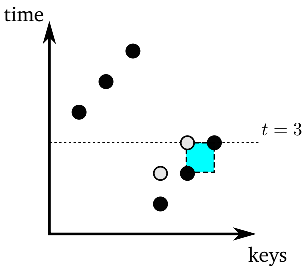


GREEDY [Lucas '88; Munro '00; Demaine et al. '09]

GREEDY:
a natural offline BST algorithm.

In geometric view GREEDY becomes:

a natural **online** algorithm.
(a simple geometric sweepline)

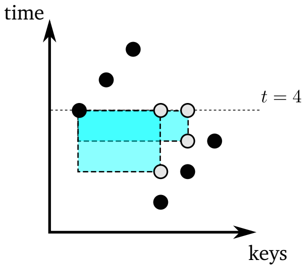


Greedy [Lucas '88; Munro '00; Demaine et al. '09]

Greedy:
a natural offline BST algorithm.

In geometric view Greedy becomes:

a natural **online** algorithm.
(a simple geometric sweepline)

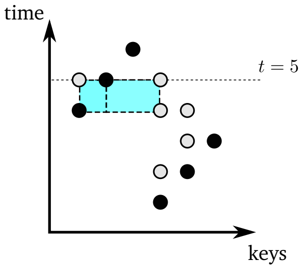


Greedy [Lucas '88; Munro '00; Demaine et al. '09]

Greedy:
a natural offline BST algorithm.

In geometric view Greedy becomes:

a natural **online** algorithm.
(a simple geometric sweepline)

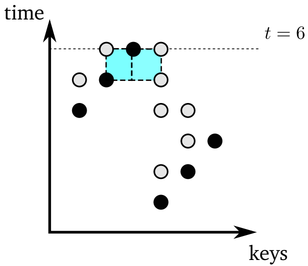


Greedy [Lucas '88; Munro '00; Demaine et al. '09]

Greedy:
a natural offline BST algorithm.

In geometric view Greedy becomes:

a natural **online** algorithm.
(a simple geometric sweepline)

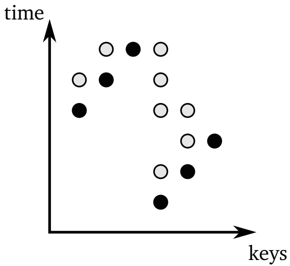


GREEDY [Lucas '88; Munro '00; Demaine et al. '09]

GREEDY:
a natural offline BST algorithm.

In geometric view GREEDY becomes:

a natural **online** algorithm.
(a simple geometric sweepline)



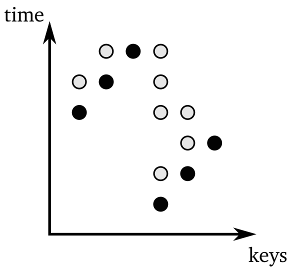
Task: Bound the cost of GREEDY

GREEDY [Lucas '88; Munro '00; Demaine et al. '09]

GREEDY:
a natural offline BST algorithm.

In geometric view GREEDY
becomes:

a natural **online** algorithm.
(a simple geometric sweepline)



Task: Bound the cost of GREEDY
$\downarrow$
$\approx$ # of points in the GREEDY execution

Forbidden Submatrix Theory

Forbidden Submatrix Theory

A useful tool since the 50s.

Forbidden Submatrix Theory

A useful tool since the 50s.

[Zarankiewicz 1951] [Kővári, Sós, Turán '55] [Bollobás, Erdős '78] [Hart, Sharir '86] [Bienstock, Győri, '91] [Füredi, Hajnal '92] [Marcus, Tardos '04] [Pettie '10]

Forbidden Submatrix Theory

A useful tool since the 50s.

[Zarankiewicz 1951] [Kővári, Sós, Turán '55] [Bollobás, Erdős '78] [Hart, Sharir '86] [Bienstock, Győri, '91] [Füredi, Hajnal '92] [Marcus, Tardos '04] [Pettie '10]

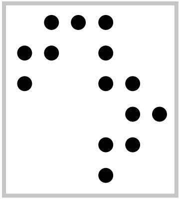
Studies patterns in $0/1$-matrices.

Forbidden Submatrix Theory

A useful tool since the 50s.

[Zarankiewicz 1951] [Kővári, Sós, Turán '55] [Bollobás, Erdős '78] [Hart, Sharir '86] [Bienstock, Győri, '91] [Füredi, Hajnal '92] [Marcus, Tardos '04] [Pettie '10]

Studies patterns in ~~0/1-matrices~~
points on a grid

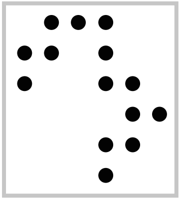


Forbidden Submatrix Theory

A useful tool since the 50s.

[Zarankiewicz 1951] [Kővári, Sós, Turán '55] [Bollobás, Erdős '78] [Hart, Sharir '86] [Bienstock, Győri, '91] [Füredi, Hajnal '92] [Marcus, Tardos '04] [Pettie '10]

Studies patterns in ~~0/1-matrices~~
points on a grid

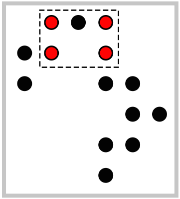


Forbidden Submatrix Theory

A useful tool since the 50s.

[Zarankiewicz 1951] [Kővári, Sós, Turán '55] [Bollobás, Erdős '78] [Hart, Sharir '86] [Bienstock, Győri, '91] [Füredi, Hajnal '92] [Marcus, Tardos '04] [Pettie '10]

Studies patterns in ~~0/1-matrices~~
points on a grid

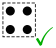


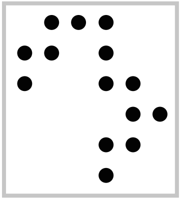
Forbidden Submatrix Theory

A useful tool since the 50s.

[Zarankiewicz 1951] [Kővári, Sós, Turán '55] [Bollobás, Erdős '78] [Hart, Sharir '86] [Bienstock, Győri, '91] [Füredi, Hajnal '92] [Marcus, Tardos '04] [Pettie '10]

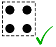
Studies patterns in ~~0/1-matrices~~
points on a grid



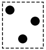


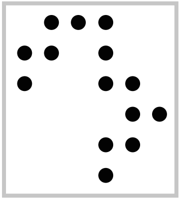
Forbidden Submatrix Theory

A useful tool since the 50s.



[Zarankiewicz 1951] [Kővári, Sós, Turán '55] [Bollobás, Erdős '78] [Hart, Sharir '86] [Bienstock, Győri, '91] [Füredi, Hajnal '92] [Marcus, Tardos '04] [Pettie '10]





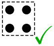
Studies patterns in ~~0/1-matrices~~
points on a grid

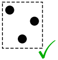
Forbidden Submatrix Theory

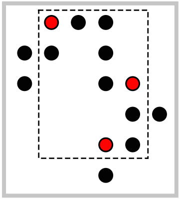
A useful tool since the 50s.

[Zarankiewicz 1951] [Kővári, Sós, Turán '55] [Bollobás, Erdős '78] [Hart, Sharir '86] [Bienstock, Győri, '91] [Füredi, Hajnal '92] [Marcus, Tardos '04] [Pettie '10]

Studies patterns in ~~0/1-matrices~~ points on a grid





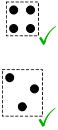


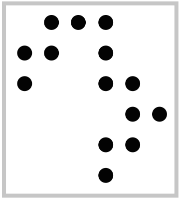
Forbidden Submatrix Theory

A useful tool since the 50s.

[Zarankiewicz 1951] [Kővári, Sós, Turán '55] [Bollobás, Erdős '78] [Hart, Sharir '86] [Bienstock, Győri, '91] [Füredi, Hajnal '92] [Marcus, Tardos '04] [Pettie '10]

Studies patterns in ~~0/1-matrices~~
points on a grid



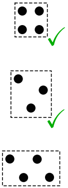


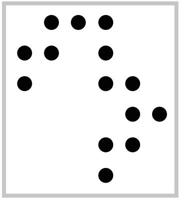
Forbidden Submatrix Theory

A useful tool since the 50s.

[Zarankiewicz 1951] [Kővári, Sós, Turán '55] [Bollobás, Erdős '78] [Hart, Sharir '86] [Bienstock, Győri, '91] [Füredi, Hajnal '92] [Marcus, Tardos '04] [Pettie '10]

Studies patterns in ~~0/1-matrices~~ points on a grid

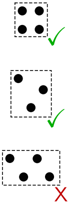


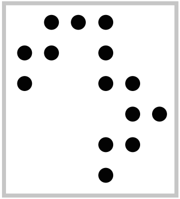


Forbidden Submatrix Theory

A useful tool since the 50s.

[Zarankiewicz 1951] [Kővári, Sós, Turán '55] [Bollobás, Erdős '78] [Hart, Sharir '86] [Bienstock, Győri, '91] [Füredi, Hajnal '92] [Marcus, Tardos '04] [Pettie '10]





Studies patterns in ~~0/1-matrices~~ points on a grid

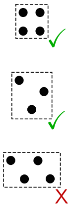
How many points can we have while avoiding some pattern?

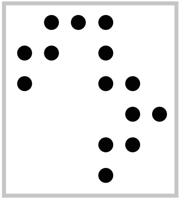
Forbidden Submatrix Theory

A useful tool since the 50s.

[Zarankiewicz 1951] [Kővári, Sós, Turán '55] [Bollobás, Erdős '78] [Hart, Sharir '86] [Bienstock, Győri, '91] [Füredi, Hajnal '92] [Marcus, Tardos '04] [Pettie '10]

Studies patterns in ~~0/1-matrices~~ points on a grid





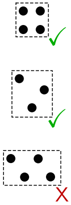
Theorems of the form:

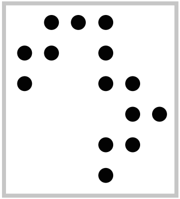
Forbidden Submatrix Theory

A useful tool since the 50s.

[Zarankiewicz 1951] [Kővári, Sós, Turán '55] [Bollobás, Erdős '78] [Hart, Sharir '86] [Bienstock, Győri, '91] [Füredi, Hajnal '92] [Marcus, Tardos '04] [Pettie '10]

Studies patterns in ~~0/1-matrices~~
points on a grid





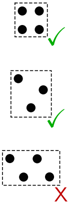
Theorems of the form:

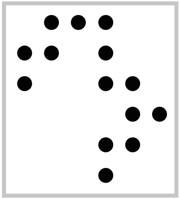
M is a set of points on the n-by-n grid **avoiding** pattern P

Forbidden Submatrix Theory

A useful tool since the 50s.

[Zarankiewicz 1951] [Kővári, Sós, Turán '55] [Bollobás, Erdős '78] [Hart, Sharir '86] [Bienstock, Győri, '91] [Füredi, Hajnal '92] [Marcus, Tardos '04] [Pettie '10]





Studies patterns in ~~0/1-matrices~~
points on a grid

Theorems of the form:

M is a set of points on the n-by-n grid **avoiding** pattern P

$\implies |M| \leq n \cdot \mathbf{f_P(n)}$.

Forbidden Submatrix Theory

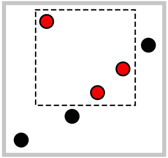
A useful tool since the 50s.

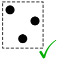
[Zarankiewicz 1951] [Kővári, Sós, Turán '55] [Bollobás, Erdős '78] [Hart, Sharir '86] [Bienstock, Győri, '91] [Füredi, Hajnal '92] [Marcus, Tardos '04] [Pettie '10]

Studies patterns in ~~0/1-matrices~~ points on a grid

Subsumes the pattern-avoidance mentioned earlier:

134562 contains 231





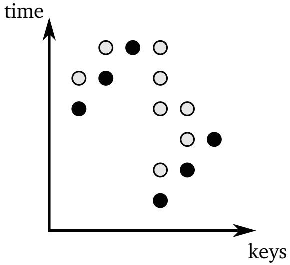
Theorems of the form:

M is a set of points on the n-by-n grid **avoiding** pattern P

$$\implies |M| \leq n \cdot \mathbf{f_P(n)}.$$

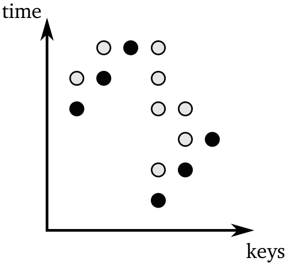
… back to GREEDY

... back to GREEDY



... back to Greedy

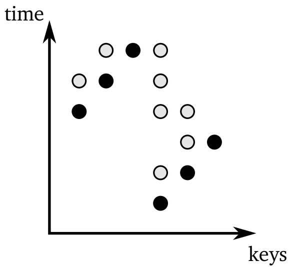
We bound the cost of Greedy using forbidden submatrix theory.



... back to GREEDY

We bound the cost of GREEDY using forbidden submatrix theory.

A first (WRONG) conjecture:

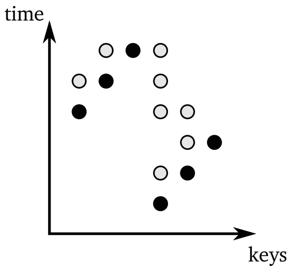


… back to GREEDY

We bound the cost of GREEDY using forbidden submatrix theory.

A first (WRONG) conjecture:

If X avoids P
$\implies$ GREEDY execution avoids P



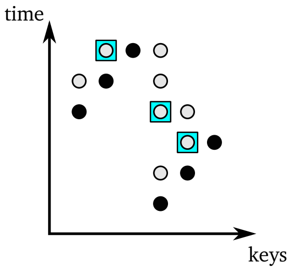
… back to Greedy

We bound the cost of Greedy using forbidden submatrix theory.

A first (WRONG) conjecture:

If X avoids $\begin{pmatrix} \bullet & & \\ & \bullet & \\ & & \bullet \end{pmatrix}$

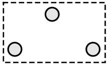
$\implies$ Greedy execution avoids $\begin{pmatrix} \bullet & & \\ & \bullet & \\ & & \bullet \end{pmatrix}$



... back to Greedy

We bound the cost of Greedy using forbidden submatrix theory.

If Greedy execution contains the pattern:

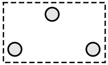


A (correct) **Lemma:**

... back to Greedy

We bound the cost of Greedy using forbidden submatrix theory.

there must be an access point inside

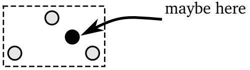


A (correct) **Lemma:**

... back to Greedy

We bound the cost of Greedy using forbidden submatrix theory.

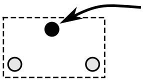
there must be an access point inside



A (correct) **Lemma:**

... back to Greedy

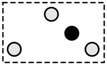
We bound the cost of Greedy using forbidden submatrix theory.



A (correct) **Lemma:**

... back to GREEDY

We bound the cost of GREEDY using forbidden submatrix theory.



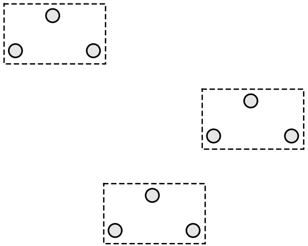
A (correct) **Lemma:**
proof very easy (but skipped).

We call this the **input-revealing** property of GREEDY.

Consequence:

... back to Greedy

We bound the cost of Greedy using forbidden submatrix theory.



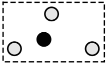
A (correct) **Lemma:**
proof very easy (but skipped).

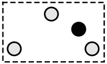
We call this the **input-revealing** property of Greedy.

Consequence:

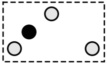
... back to Greedy

We bound the cost of Greedy using forbidden submatrix theory.





A (correct) **Lemma:**
proof very easy (but skipped).

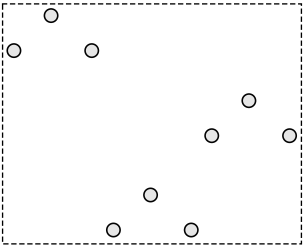


We call this the **input-revealing** property of Greedy.

Consequence:

... back to Greedy

We bound the cost of Greedy using forbidden submatrix theory.



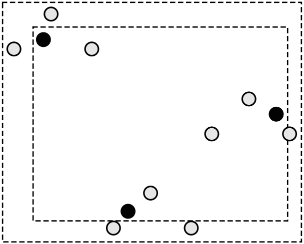
A (correct) **Lemma:**
proof very easy (but skipped).

We call this the **input-revealing** property of Greedy.

Consequence:

... back to GREEDY

We bound the cost of GREEDY using forbidden submatrix theory.



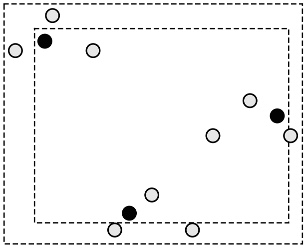
A (correct) **Lemma:**
proof very easy (but skipped).

We call this the **input-revealing** property of GREEDY.

Consequence:

... back to $\textsc{Greedy}$

We bound the cost of $\textsc{Greedy}$ using forbidden submatrix theory.



A (correct) **Lemma:**
proof very easy (but skipped).

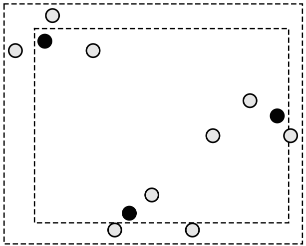
We call this the **input-revealing** property of $\textsc{Greedy}$.

Consequence:

if X avoids $\begin{pmatrix} 1 & & \\ & & 3 \\ & 2 & \end{pmatrix}$

... back to Greedy

We bound the cost of Greedy using forbidden submatrix theory.



A (correct) **Lemma:**
proof very easy (but skipped).

We call this the **input-revealing** property of Greedy.

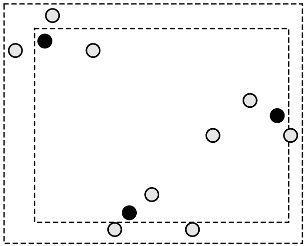
Consequence:

if X avoids $\begin{pmatrix} 1 & & \\ & & 3 \\ & 2 & \end{pmatrix}$

$\implies$ Greedy execution avoids $\begin{pmatrix} & \bullet & & & & & & & \\ \bullet & & \bullet & & & & & & \\ & & & & & & & \bullet & \\ & & & & & & \bullet & & \bullet \\ & & & & \bullet & & & & \\ & & & \bullet & & \bullet & & & \end{pmatrix}$

... back to Greedy

We bound the cost of Greedy using forbidden submatrix theory.



A (correct) **Lemma:**
proof very easy (but skipped).

We call this the **input-revealing** property of Greedy.

Consequence:

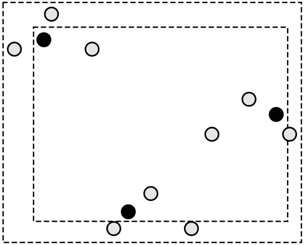
if X avoids $\begin{pmatrix} 1 & & \\ & & 3 \\ & 2 & \end{pmatrix}$

$\implies$ Greedy execution avoids $\begin{pmatrix} & \bullet & & & & & & & \\ \bullet & & \bullet & & & & & & \\ & & & & & & & \bullet & \\ & & & & & & \bullet & & \bullet \\ & & & & \bullet & & & & \\ & & & \bullet & & \bullet & & & \end{pmatrix}$

$\implies$ cost of Greedy on X is at most $n \cdot 2^{\mathrm{poly}(\alpha(n))}$

using [Klazar '00] [Keszegh '09] [Pettie '15]

We bound the cost of $\textsc{Greedy}$ using forbidden submatrix theory.



A (correct) **Lemma:**
proof very easy (but skipped).

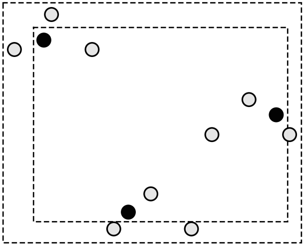
Consequence:

if X avoids P

$\implies$ $\textsc{Greedy}$ execution avoids $P \otimes (\,.^{\bullet}.\,)$

$\implies$ cost of $\textsc{Greedy}$ on X is $n \cdot 2^{\alpha(n)^{O(|P|)}}$

We bound the cost of Greedy using forbidden submatrix theory.



A (correct) **Lemma:**
proof very easy (but skipped).

Consequence:

if X avoids P

$\implies$ Greedy execution avoids $P \otimes (\,{}_{\bullet}\,{}^{\bullet}\,{}_{\bullet}\,)$

$\implies$ cost of Greedy on X is $n \cdot 2^{\alpha(n)^{O(|P|)}}$

$\rightarrow$ for various **special cases** we prove stronger bounds, i.e. $O(n)$

proofs more difficult

A different application of the technique...

A different application of the technique...

Independent Rectangle bound [Demaine et al. '09] [Wilber '89]

A different application of the technique...

Independent Rectangle bound [Demaine et al. '09] [Wilber '89]

$\rightarrow$ Lower bound on the cost of **any** BST algorithm

A different application of the technique...

Independent Rectangle bound [Demaine et al. '09] [Wilber '89]

$\rightarrow$ Lower bound on the cost of **any** BST algorithm

$\rightarrow$ Conjectured to be $\Theta(OPT)$

A different application of the technique...

Independent Rectangle bound [Demaine et al. '09] [Wilber '89]

$\rightarrow$ Lower bound on the cost of **any** BST algorithm

$\rightarrow$ Conjectured to be $\Theta(OPT)$

We show:
If X **avoids** P, then **IR-bound** for X is $O(n)$, for any constant-sized P.

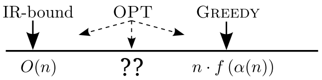
A different application of the technique...

Independent Rectangle bound [Demaine et al. '09] [Wilber '89]

→ Lower bound on the cost of **any** BST algorithm

→ Conjectured to be $\Theta(OPT)$

We show:
If X **avoids** P, then **IR-bound** for X is $O(n)$, for any constant-sized P.



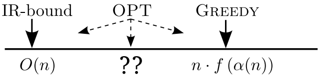
A different application of the technique...

Independent Rectangle bound [Demaine et al. '09] [Wilber '89]

$\rightarrow$ Lower bound on the cost of **any** BST algorithm

$\rightarrow$ Conjectured to be $\Theta(OPT)$

We show:
If X **avoids** P, then **IR-bound** for X is $O(n)$, for any constant-sized P.



Consequence:

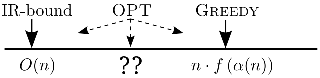
A different application of the technique...

Independent Rectangle bound [Demaine et al. '09] [Wilber '89]

$\rightarrow$ Lower bound on the cost of **any** BST algorithm

$\rightarrow$ Conjectured to be $\Theta(OPT)$

We show:
If X **avoids** P, then **IR-bound** for X is $O(n)$, for any constant-sized P.



Consequence: "something's gotta give ..." ♪ 𝅗𝅥 ♫

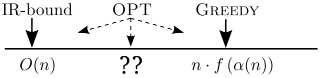
A different application of the technique...

Independent Rectangle bound [Demaine et al. '09] [Wilber '89]

$\rightarrow$ Lower bound on the cost of **any** BST algorithm

$\rightarrow$ Conjectured to be $\Theta(OPT)$

We show:
If X **avoids** P, then **IR-bound** for X is $O(n)$, for any constant-sized P.



Consequence: "something's gotta give ..." ♪ ♩ ♫

? GREEDY is in fact linear on all pattern-avoiding input

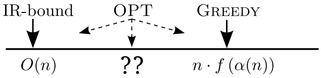
A different application of the technique...

Independent Rectangle bound [Demaine et al. '09] [Wilber '89]

$\rightarrow$ Lower bound on the cost of **any** BST algorithm

$\rightarrow$ Conjectured to be $\Theta(OPT)$

We show:
If X **avoids** P, then **IR-bound** for X is $O(n)$, for any constant-sized P.



Consequence: "something's gotta give ..." ♪ ♩ ♫

- ? GREEDY is in fact linear on all pattern-avoiding input
- ? GREEDY is not $O(1)$-competitive

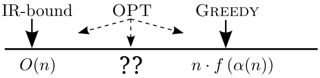
A different application of the technique...

Independent Rectangle bound [Demaine et al. '09] [Wilber '89]

$\rightarrow$ Lower bound on the cost of **any** BST algorithm

$\rightarrow$ Conjectured to be $\Theta(OPT)$

We show:
If X **avoids** P, then **IR-bound** for X is $O(n)$, for any constant-sized P.



Consequence: "something's gotta give ..." ♪ ♩ ♫

- ? GREEDY is in fact linear on all pattern-avoiding input
- ? GREEDY is not $O(1)$-competitive
- ? Conjecture is false (IR-bound not tight)

Conclusion:

Conclusion:

On inputs that avoid an arbitrary pattern, GREEDY is linear*

Conclusion:

On inputs that avoid an arbitrary pattern, GREEDY is linear*

For traversal conjecture, GREEDY is linear*†

Conclusion:

On inputs that avoid an arbitrary pattern, GREEDY is linear*

For traversal conjecture, GREEDY is linear*†

* up to $f(\alpha(n))$ factor, **or**
† with preprocessing

Conclusion:

On inputs that avoid an arbitrary pattern, $\textsc{Greedy}$ is linear*

For traversal conjecture, $\textsc{Greedy}$ is linear*†

* up to $f(\alpha(n))$ factor, **or**
† with preprocessing

Open Question 1
Prove traversal conjecture unconditionally for an online algorithm.

Conclusion:

On inputs that avoid an arbitrary pattern, GREEDY is linear*

For traversal conjecture, GREEDY is linear*†

* up to $f(\alpha(n))$ factor, **or**
† with preprocessing

Open Question 1
Prove traversal conjecture unconditionally for an online algorithm.
Even for preorder sequence of a **path**

Conclusion:

On inputs that avoid an arbitrary pattern, $\textsc{Greedy}$ is linear*

For traversal conjecture, $\textsc{Greedy}$ is linear*†

* up to $f(\alpha(n))$ factor, **or**
† with preprocessing

Open Question 1
Prove traversal conjecture unconditionally for an online algorithm.
Even for preorder sequence of a **path** $\rightarrow$ sequence avoiding both 231 and 213.

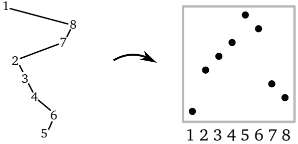
Conclusion:

On inputs that avoid an arbitrary pattern, GREEDY is linear*

For traversal conjecture, GREEDY is linear*†

* up to $f(\alpha(n))$ factor, **or**
† with preprocessing

Open Question 1
Prove traversal conjecture unconditionally for an online algorithm.
Even for preorder sequence of a **path** $\rightarrow$ sequence avoiding both 231 and 213.



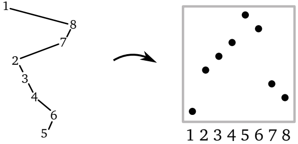
Conclusion:

On inputs that avoid an arbitrary pattern, $\textsc{Greedy}$ is linear*

For traversal conjecture, $\textsc{Greedy}$ is linear*†

* up to $f(\alpha(n))$ factor, **or**
† with preprocessing

Open Question 1
Prove traversal conjecture unconditionally for an online algorithm.
Even for preorder sequence of a **path** $\rightarrow$ sequence avoiding both 231 and 213.



Open Question 2

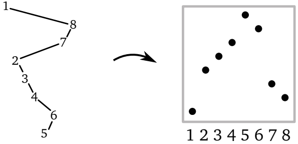
Conclusion:

On inputs that avoid an arbitrary pattern, $\textsc{Greedy}$ is linear*

For traversal conjecture, $\textsc{Greedy}$ is linear*†

* up to $f(\alpha(n))$ factor, **or**
† with preprocessing

Open Question 1
Prove traversal conjecture unconditionally for an online algorithm.
Even for preorder sequence of a **path** $\rightarrow$ sequence avoiding both 231 and 213.



Open Question 2
Prove $o(\log(n))$-competitiveness for $\textsc{Greedy}$ or Splay Tree, or

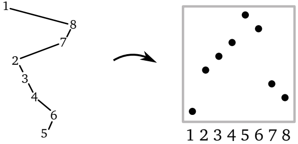
Conclusion:

On inputs that avoid an arbitrary pattern, GREEDY is linear*

For traversal conjecture, GREEDY is linear*†

* up to $f(\alpha(n))$ factor, **or**
† with preprocessing

Open Question 1
Prove traversal conjecture unconditionally for an online algorithm.
Even for preorder sequence of a **path** $\rightarrow$ sequence avoiding both 231 and 213.



Open Question 2
Prove $o(\log(n))$-competitiveness for GREEDY or Splay Tree, or $o(\log\log(n))$-competitiveness for any algorithm.