Recent Developments in Algorithm Design (Spring 2025)

Lecture 3: Min-Sum Set Cover and its variants, Intro to Stochastic Boolean Function Evaluation

(Hellerstein)

NYU | TANDON SCHOOL OF ENGINEERING

Min-Sum Set Cover

Min-Sum Set Cover [Feige et al. 02, 04]

- Input: Ground set (universe) $\mathscr{U} = \{e_1, \ldots, e_n\}$ and family of subsets $\mathscr{F} = \{S_1, \ldots, S_m\}$ where each $S_i \subseteq \mathscr{U}$, such that $\bigcup_{i=1}^{n} S_i = \mathscr{U}$

- Task: Find the permutation of the subsets that minimizes the sum of the *covering times* of the ground elements
  - If an element is covered by the jth element in the permutation, we say that it is covered at time j

Greedy Algorithm

- Greedy Rule: Choose subset that covers the maximum number of uncovered elements
- Same greedy algorithm we used for “classical” set cover problem.
- What approximation factor does it achieve for Min-Sum Set Cover problem?

Greedy Algorithm

- Thm [Feige et al. 02, 04] :
  Greedy Algorithm is a 4-approximation algorithm for Min-Sum Set Cover

Proof: Based on histograms.

Original proof was LP-based, later proof was histogram based. We'll present version of histogram proof used in [Happach et al. 2022] to prove a more general result.

Reformulation of Min-Sum Set Cover Problem in terms of "utility"

- Let $I = \{1, \ldots, m\}$
- Utility: Define $u : 2^I \to \mathscr{R}^{\geq 0}$ s.t. for $I' \subseteq I$,

  $u(I') = |\bigcup_{k \in I'} S_k|$

  = # elts of $\mathscr{U}$ covered by subsets S_k such that $k \in I'$
- For permutation π of I, $i \geq 0$, define π^i = set of first i items in π
- Min-Sum Set Cover: Find permutation π of elements of I minimizing

$$\sum_{i=1}^{m} i \times (u(\pi^i) - u(\pi^{i-1}))$$

Greedy Algorithm

- Greedy Rule: If I' is the set of $i \in I$ already chosen for permutation, next add the item k maximizing increase in utility

$$u(I' \cup \{k\}) - u(I')$$

- Thm: Greedy Algorithm is a 4-approximation algorithm for Min-Sum Set Cover Problem

Proof based on histograms

Taxicab story

- Group of friends who get in taxicab together, not all going to same location
- Taxi goes to desired locations in some order, dropping off some passengers at each location
- Suppose each passenger has to pay for for travel before they are dropped off (!) at a rate of \$1 per stop, e.g., if passenger is dropped off at the 3rd stop, their fare is \$3.
- Driver could have each passenger pay their total fare when dropped off. So at stop i, total amount paid to the driver is

  $i \times$ (number of people dropped off at stop i)
- Alternatively, driver could have the passengers pay as they go: each time the taxi stops, everyone in the taxi has to give the driver \$1. So at stop i, amount paid to driver is

  1 x number of people still in taxi when arrive at stop i
- Either way, driver earns the same amount of money for trip!

Pf of Theorem:

Given a permutation π of the elements in $I = \{1,2,\ldots,m\}$ and a prefix π^k of π, consisting of the first k sets in π,

let $r(\pi^k) = \sum_{i=1}^{k} i \times (u(\pi^i) - u(\pi^{i-1}))$

= sum of covering times of the first k items of π

- We're seeking a permutation π of the elements in $I = \{1, \ldots, m\}$ that minimizes the sum of the covering times, $r(\pi)$
- Total utility: $n = u(I)$ [number of elements in universe]

  Increase in utility at step i: $u(\pi^i) - u(\pi^{i-1})$
- Analogy to taxi: elements in $\mathcal{U}$ are passengers, each time step is a stop of taxi, being covered at step i is leaving taxi at stop i
- Pay when covered

$$r(\pi) = \sum_{i=1}^{m} i \times (u(\pi^i) - u(\pi^{i-1}))$$

- Pay as you go

$$r(\pi) = \sum_{i=1}^{m} 1 \times (u(I) - u(\pi^{i-1}))$$

Let G be permutation produced by running the greedy algorithm.

Let T be the permutation minimizing $r()$, i.e., T is an optimal solution to our Min-Sum problem

We will prove

Claim: $r(G) \leq 4r(T)$

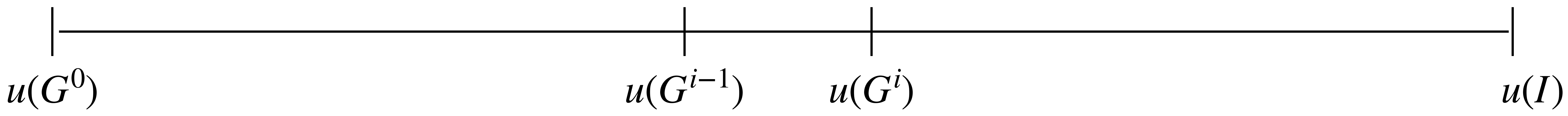
which implies G is a 4-approx solution to our Min-Sum problem, proving the theorem.

- Think of each permutation as adding subsets, step by step
- In each step, cover more elements of $\mathcal{U}$, utility increases
- At start, $u(G^0) = 0$ and at end, $u(G^m) = u(I) = n$
- View G as a journey from utility 0 to utility $u(I)$ [NOT as a journey through time!]
- In ith step of G, utility increases from $u(G^{i-1})$ to $u(G^i)$
- and analogously for optimal permutation T
- We will compare utility at start of ith step of G to utility at end of jth step of T

- We will use the following lemma.

  Lemma:

  If $$\frac{u(G^i) - u(G^{i-1})}{u(I) - u(G^{i-1})} < \frac{1}{2j}$$ then $$u(I) - u(T^j) > \frac{1}{2}(u(I) - u(G^{i-1}))$$



- Lemma: If $\dfrac{u(G^i) - u(G^{i-1})}{u(I) - u(G^{i-1})} < \dfrac{1}{2j}$ then $u(I) - u(T^j) > \dfrac{1}{2}(u(I) - u(G^{i-1}))$

- Pf: Assume $\dfrac{u(G^i) - u(G^{i-1})}{u(I) - u(G^{i-1})} < \dfrac{1}{2j}$. By Greedy Rule, if add single item to G^{i-1}, can't increase utility (number of covereed elements) by more than $u(G^i) - u(G^{i-1})$.

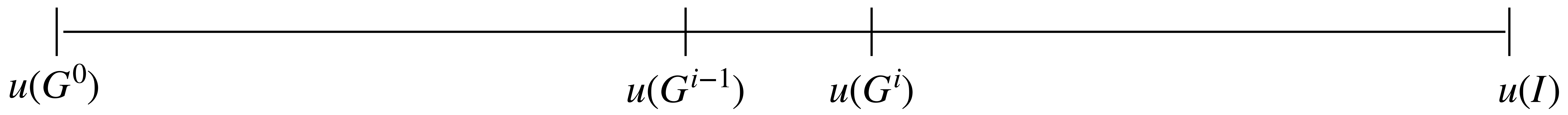
- So if add all j items in T^j to G^{i-1}, can't increase utility by more than $j * (u(G^i) - u(G^{i-1}))$

$u(G^{i-1} \cup T^j) - u(G^{i-1}) < j * (u(G^i) - u(G^{i-1}))$

$\Rightarrow u(T^j) - u(G^{i-1}) < j * (u(G^i) - u(G^{i-1})$ because $u(T^j) \leq u(T^j \cup G^{i-1})$

$< \dfrac{1}{2}(u(I) - u(G^{i-1}))$ by assumption

$\Rightarrow u(I) - u(T^j) > \dfrac{1}{2}(u(I) - u(G^{j-1}))$ because previous inequality meant that $u(T^j)$ was located to the left of the midpoint between $u(G^{i-1})$ and $u(I)$, or equivalently, by subtracting both sides of previous inequality from $u(I) - u(G^{i-1})$



Claim: $r(G) \leq 4r(T)$

Pf of claim:

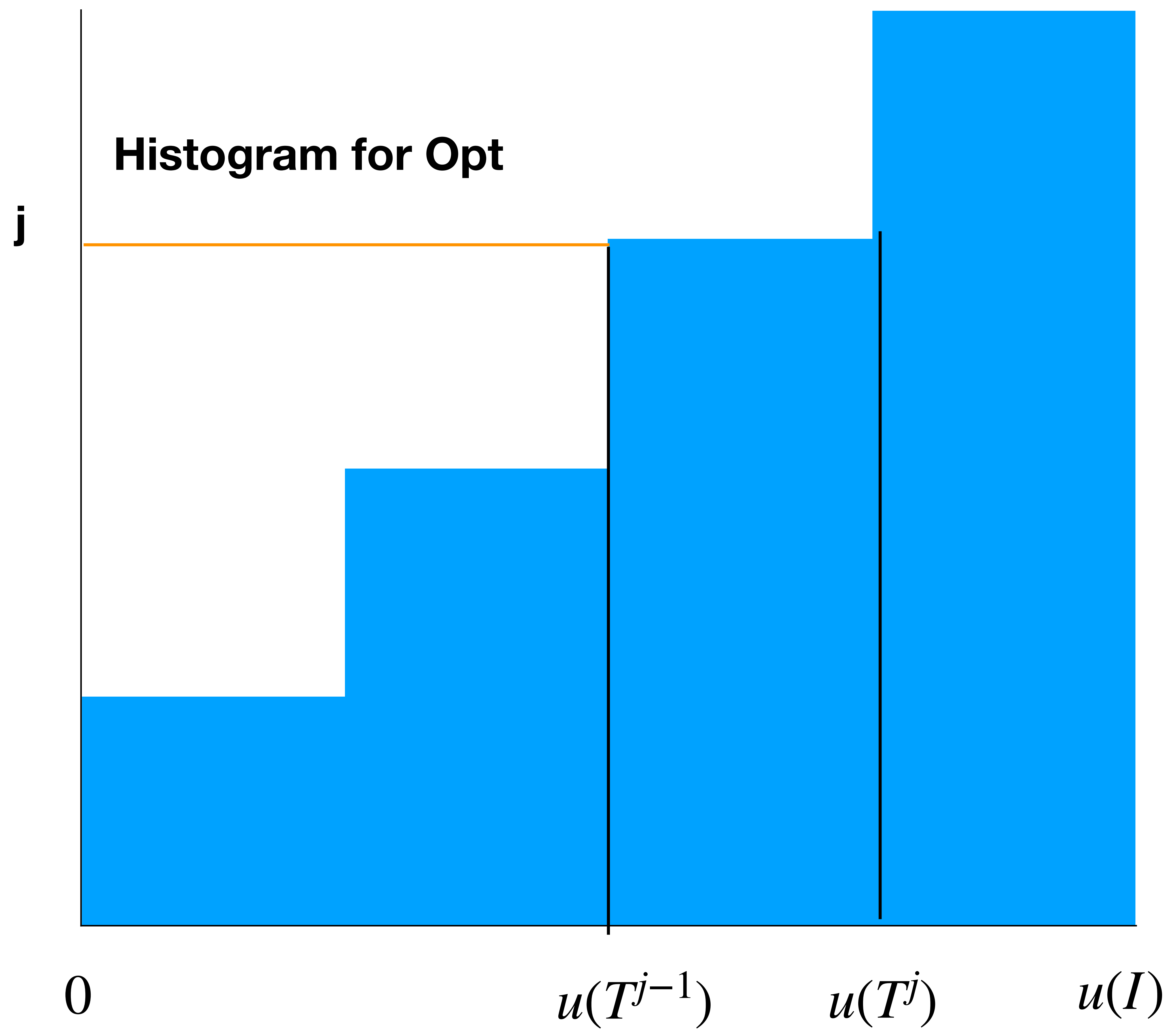
Consider a histogram for T.

Horizontal axis: labeled from 0 to $u(I)$

Bar corresponding to T^j goes from

$u(T^{j-1})$ to $u(T^j)$

of height j



Area under histogram is $r(T) = \sum_{j=1}^{m} j \times (u(T^j) - u(T^{j-1}))$

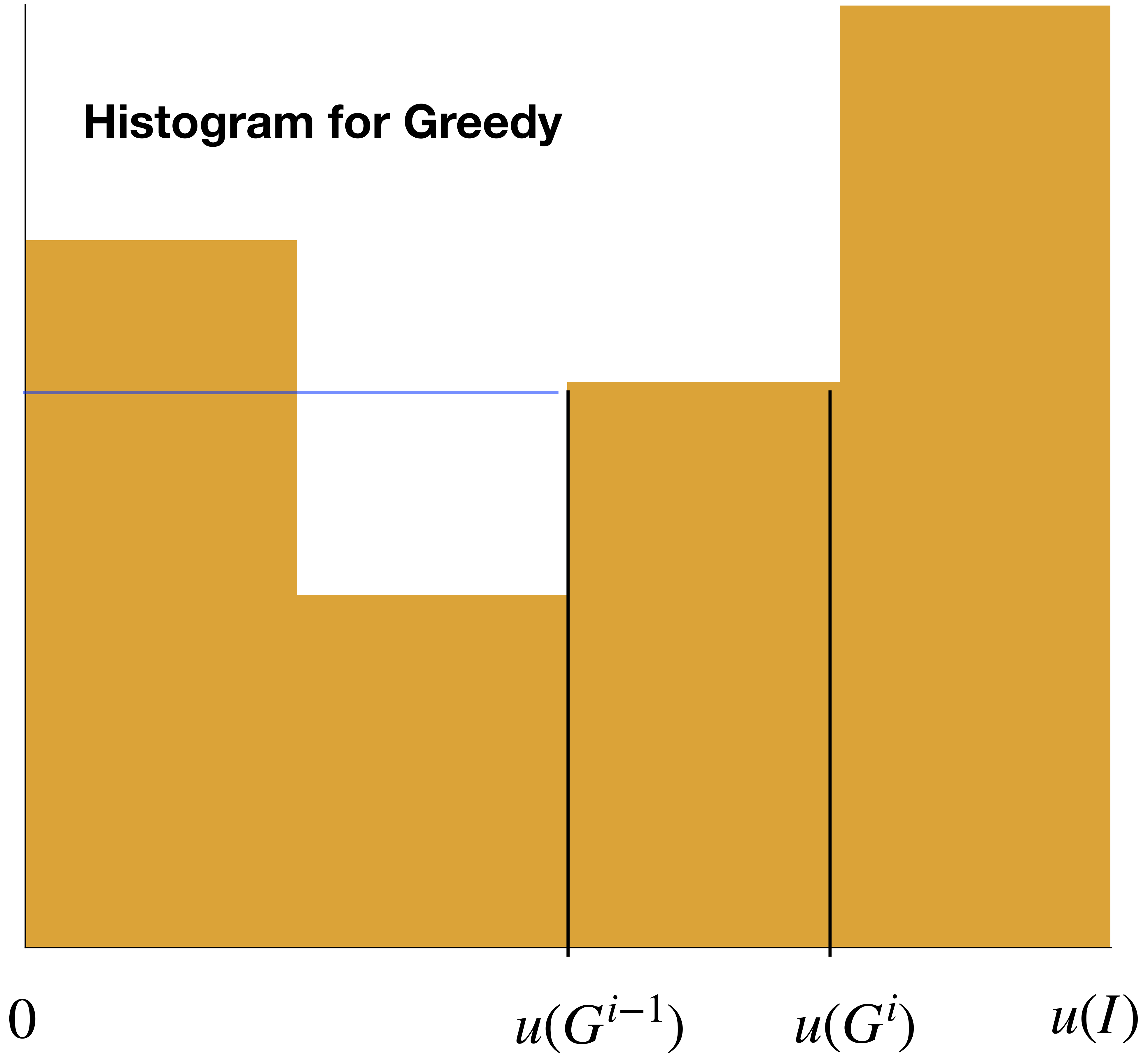
Consider a histogram for G.

Horizontal axis: labeled from 0 to $u(I)$

Bar corresponding to G_i goes from

$$u(G^{i-1}) \text{ to } u(G^i)$$

of height $(u(I) - u(G^{i-1})) \times \dfrac{1}{u(G^i) - u(G^{i-1})}$



Area under histogram is $\sum_{i=1}^{m} 1 \times (u(I) - u(G^{i-1})) = r(G)$

pay as you go

Consider histogram for G.

Horizontal axis: labeled from 0 to $u(I)$

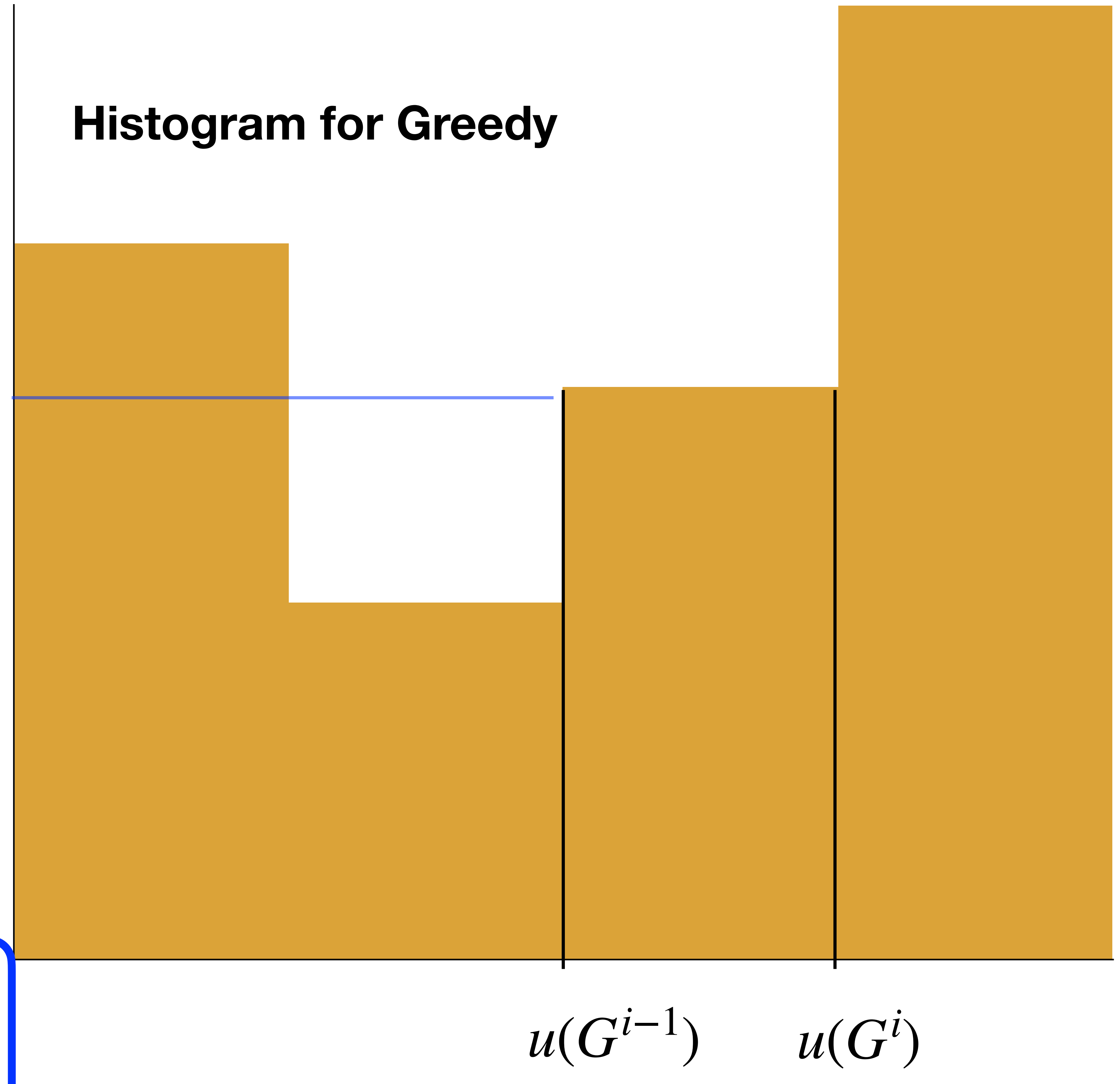
Bar corresponding to G_i goes from

$$u(G^{i-1}) \text{ to } u(G^i)$$

and height $(u(I) - u(G^{i-1})) * \boxed{\dfrac{1}{u(G^i) - u(G^{i-1})}}$

Inverse of utility gained by Greedy in step i

Area under histogram is $\sum_i 1 \times (u(I) - u(G^{i-1})) = r(G)$



Consider histogram for G.

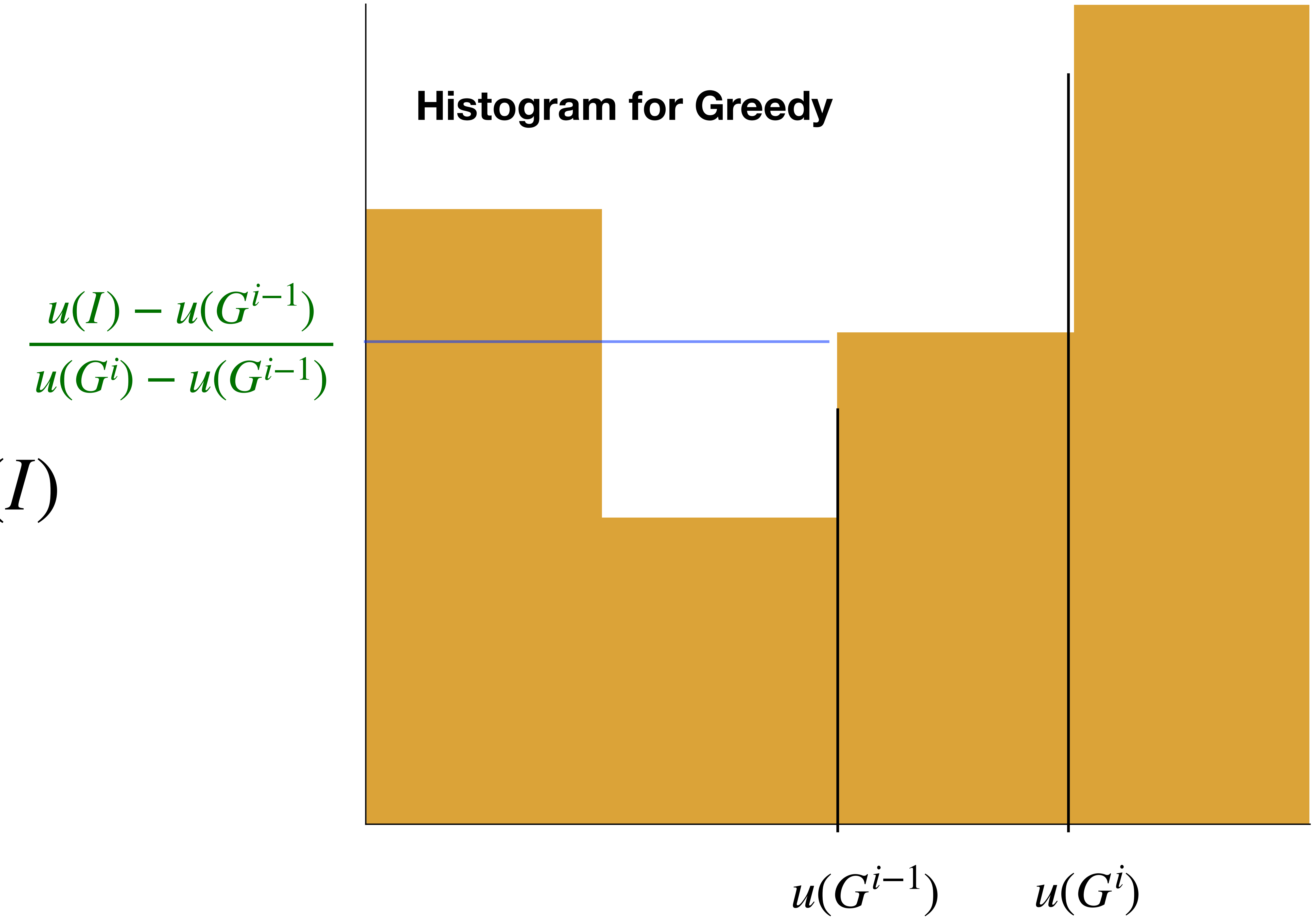
Horizontal axis: labeled from 0 to $u(I)$

Bar corresponding to G_i goes from

$$u(G^{i-1}) \text{ to } u(G^i)$$

of height $\dfrac{u(I) - u(G^{i-1})}{u(G^i) - u(G^{i-1})}$

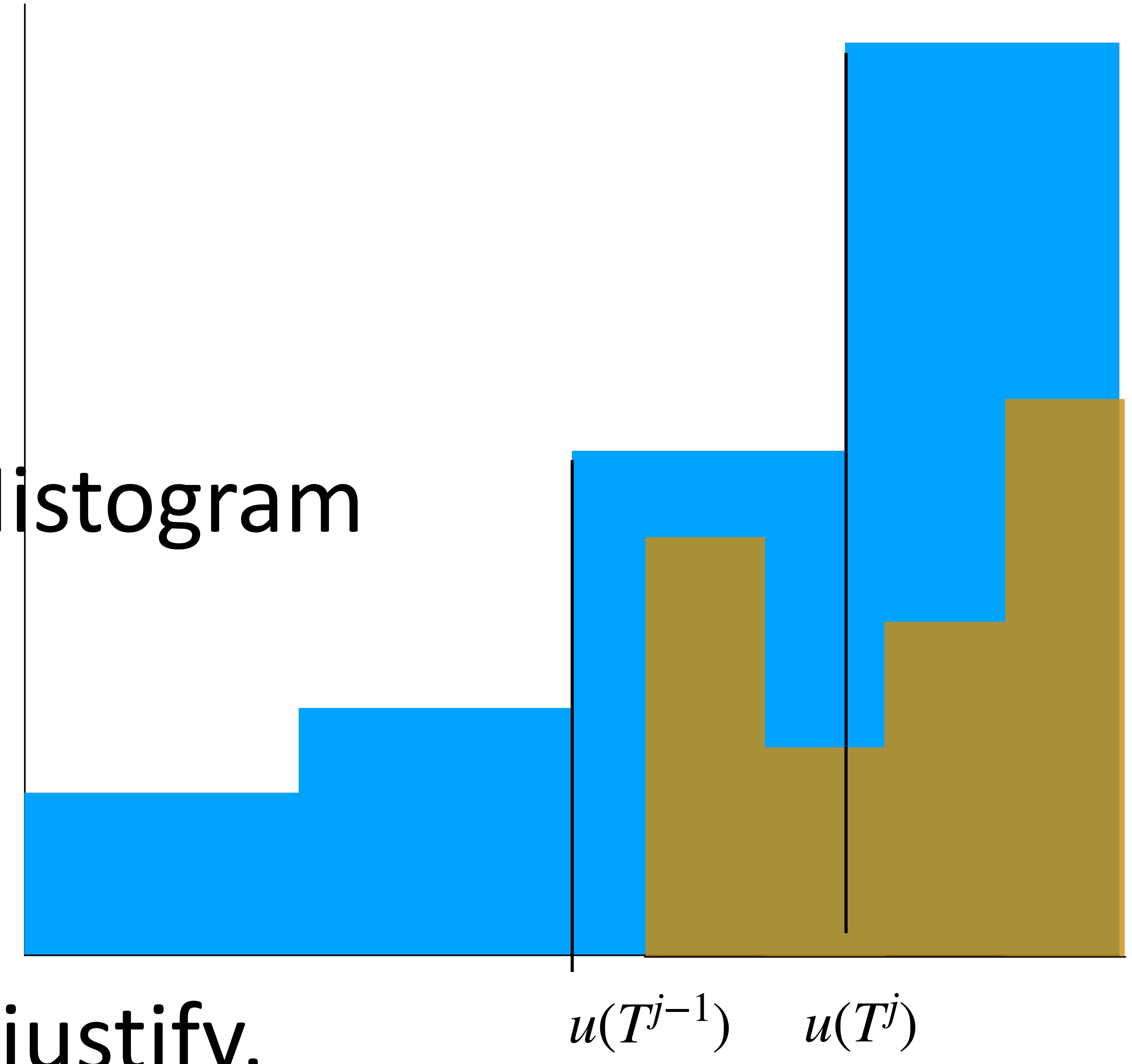
Area under histogram is $\sum_i 1 \times (u(I) - u(G^{i-1})) = r(G)$



Claim: $r(G) \leq 4r(T)$

Pf of claim (continued):

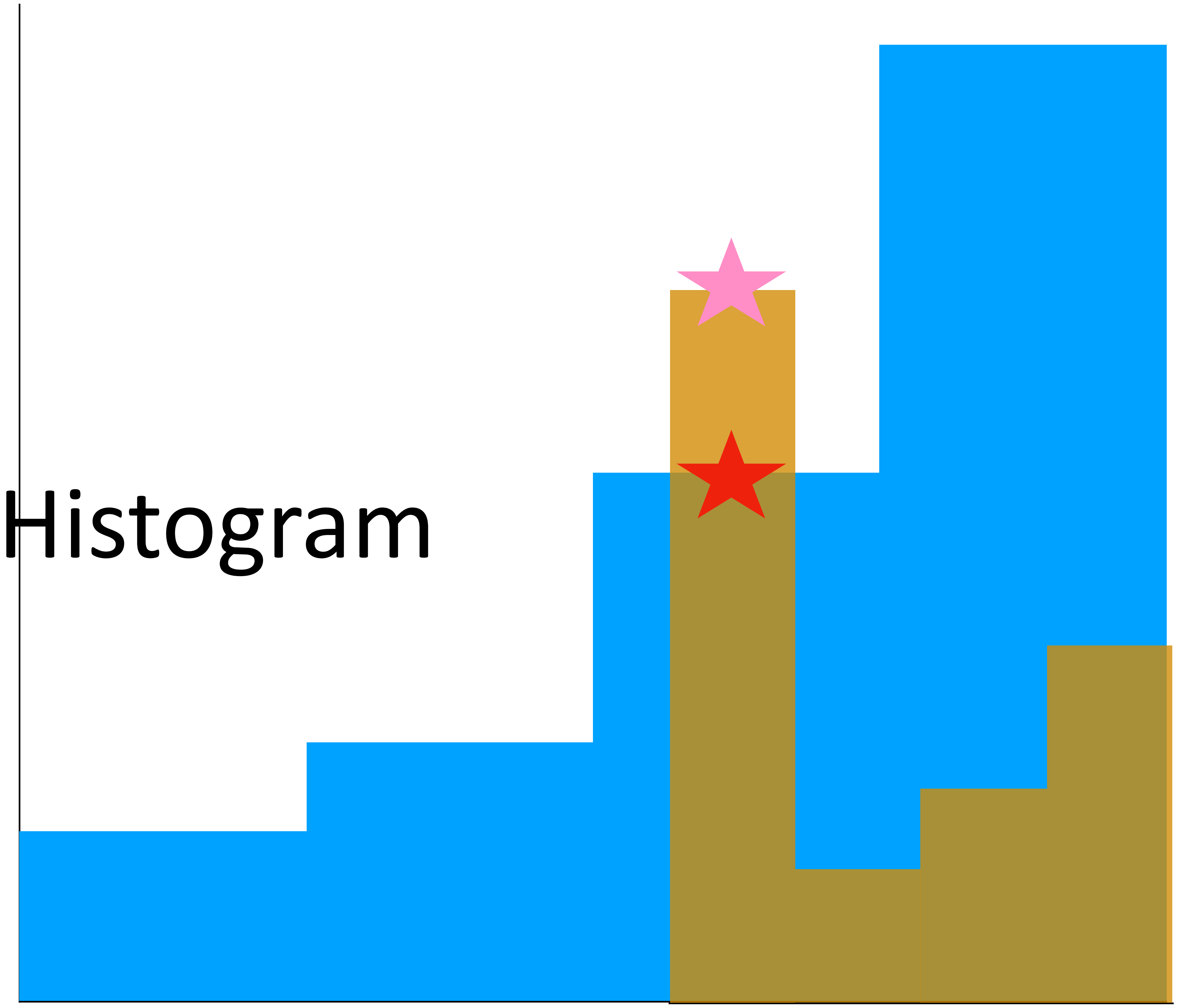
We'll show if shrink Greedy (brown) Histogram by factor of 2 in both horizontal and vertical directions (decreasing its area by 4), overlay it on optimal histogram, right justify, then it "fits" into Optimal (blue) Histogram



Claim: $r(G) \leq 4r(T)$

Pf of claim (continued):

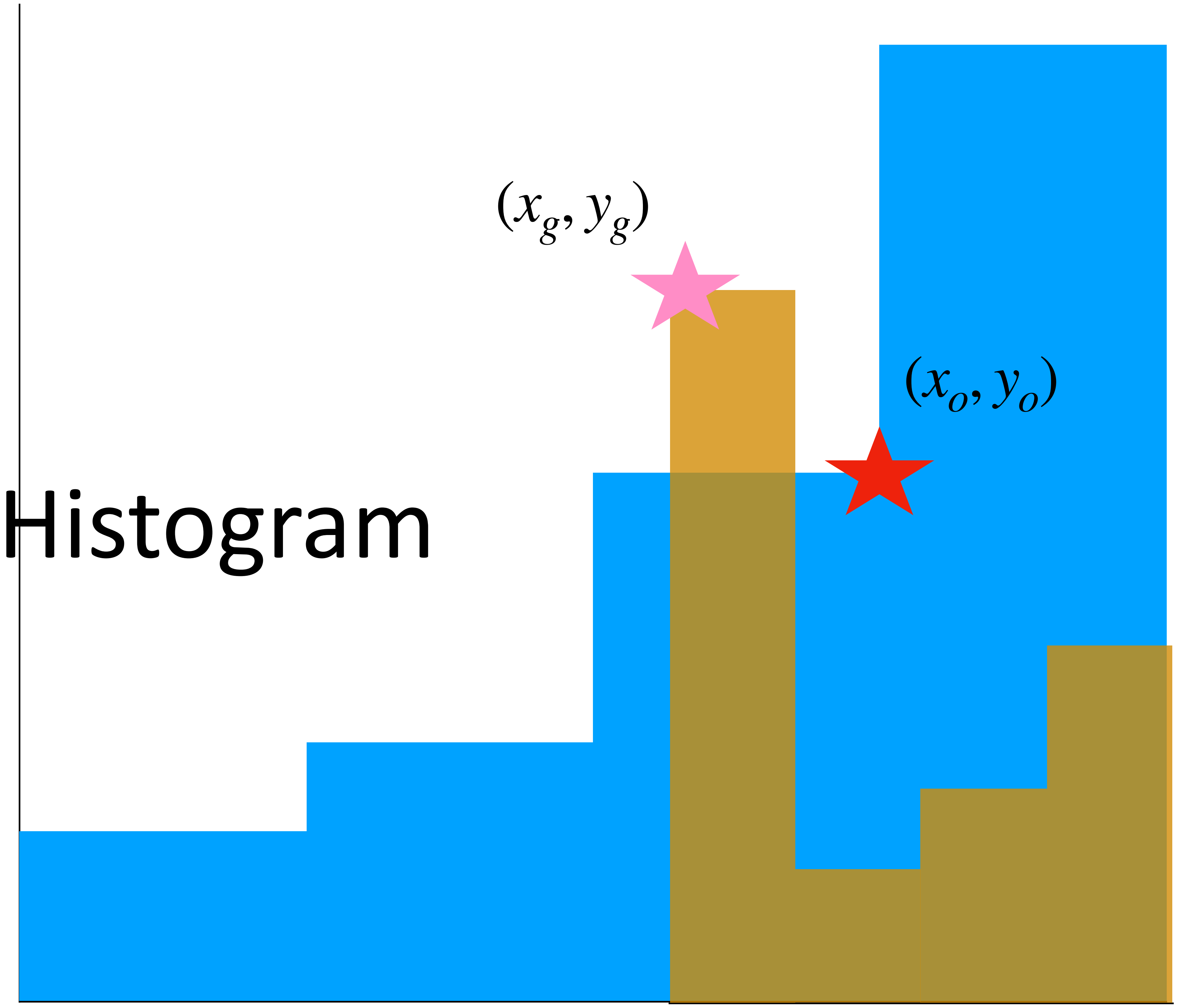
Suppose not. Suppose shrink Greedy Histogram by factor of 2 in horizontal and vertical directions, overlay it on optimal histogram, right justify, and it *doesn't* fit into Optimal Histogram



Claim: $r(G) \leq 4r(T)$

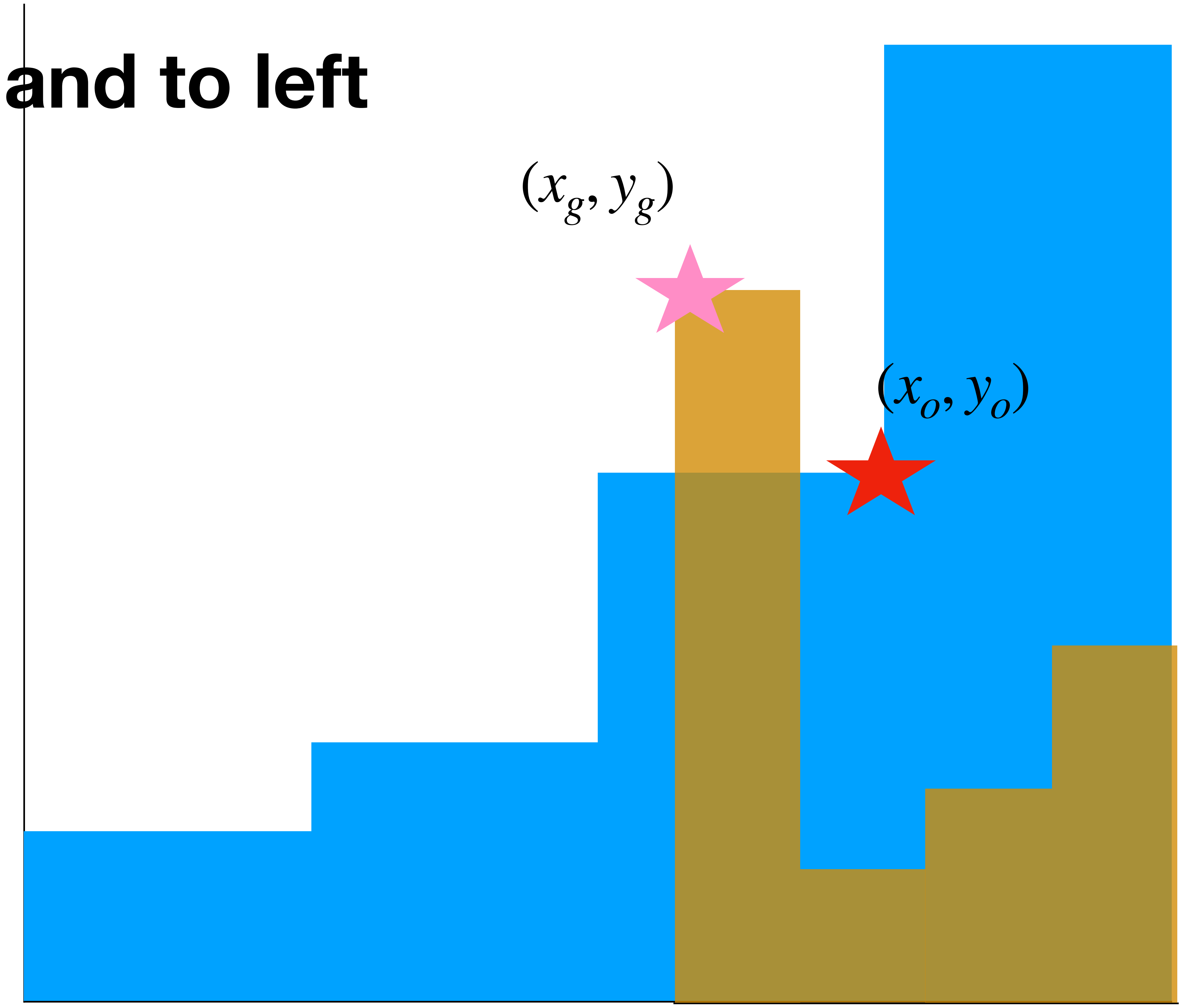
Pf of claim (continued):

Suppose not. Suppose shrink Greedy Histogram by factor of 2 in horizontal and vertical directions, overlay it on optimal histogram, right justify, and it *doesn't* fit into Optimal Histogram



Then a vertical bar of the shrunken Greedy (brown) histogram rises above the Optimal (blue) Histogram at some point. Top left corner of the brown bar is above and to left of top right corner of lower blue bar.

Then top left corner of the brown bar is above and to left of top right corner of lower blue bar.



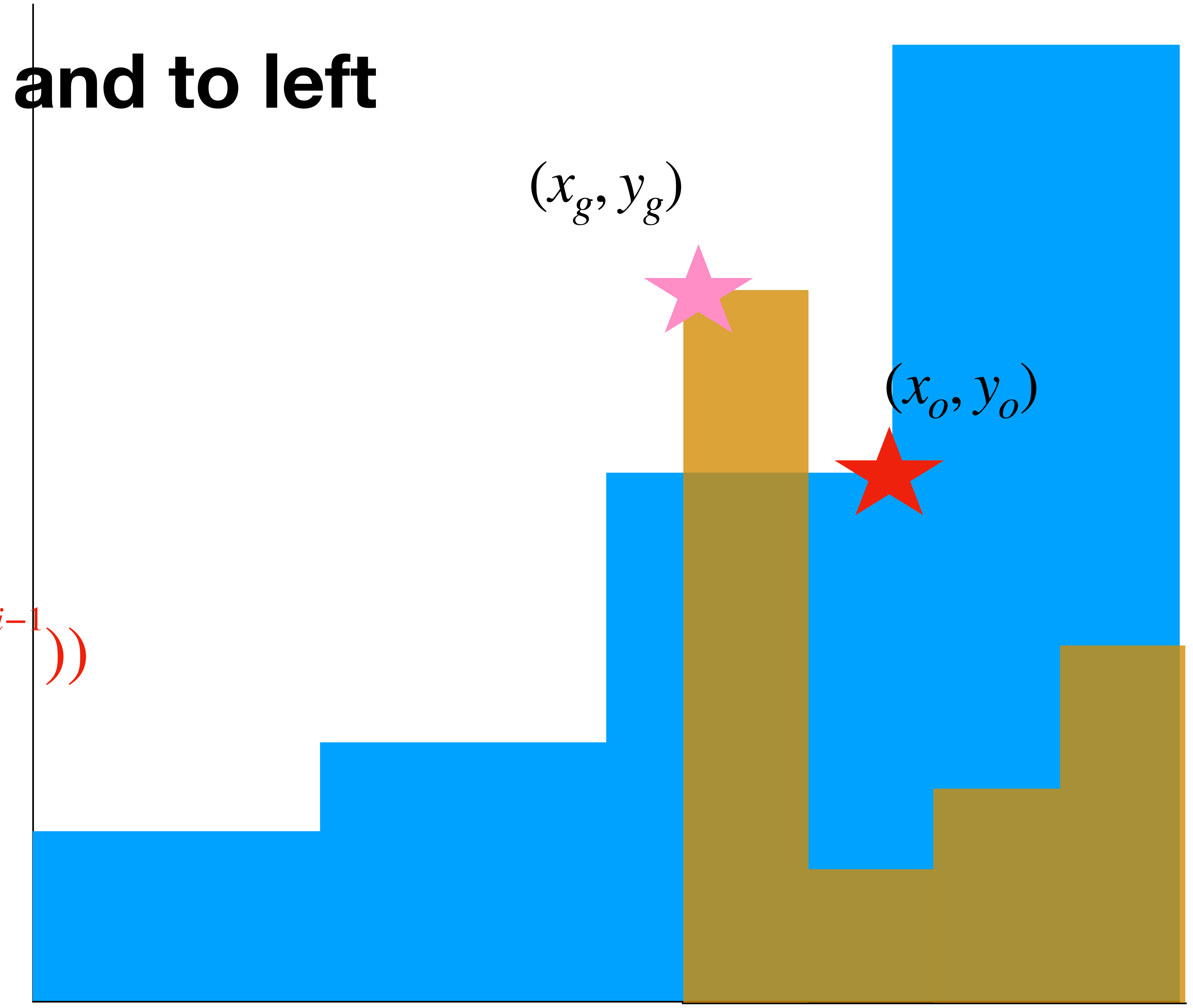
$$\frac{1}{2} * \frac{u(I) - u(G^{i-1})}{u(G^i) - u(G^{i-1})} > j$$

$$y_g > y_o$$

$$u(I) - u(T_j) \leq \frac{1}{2}(u(I) - u(G^{i-1}))$$

$x_o \geq x_g$ or equivalently $u(I) - x_o \leq \frac{1}{2}(u(I) - u(G^{i-1}))$

Then top left corner of the brown bar is above and to left of top right corner of lower blue bar.



But recall Lemma:

If $\boxed{\dfrac{u(G^i) - u(G^{i-1})}{u(I) - u(G^{i-1})} < \dfrac{1}{2j}}$ then $u(I) - u(T^j) > \dfrac{1}{2}(u(I) - u(G^{i-1}))$

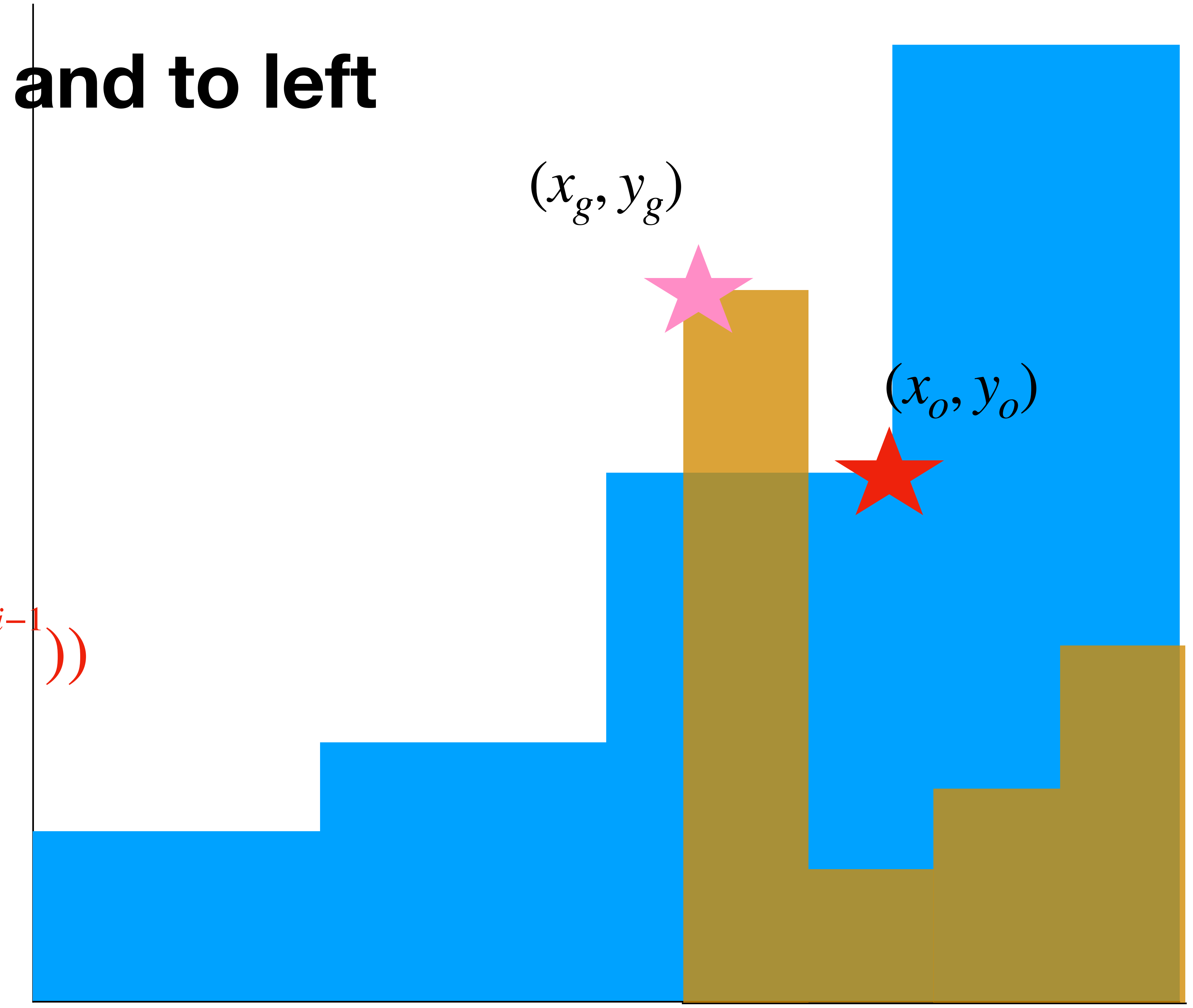
Equivalent

$$\boxed{\frac{1}{2} * \frac{u(I) - u(G^{i-1})}{u(G^i) - u(G^{i-1})} > j}$$

$$y_g > y_o$$

$$u(I) - u(T_j) \leq \frac{1}{2}(u(I) - u(G^{i-1}))$$

Then top left corner of the brown bar is above and to left of top right corner of lower blue bar.



But recall Lemma:

If $\dfrac{u(G^i) - u(G^{i-1})}{u(I) - u(G^{i-1})} < \dfrac{1}{2j}$ then $u(I) - u(T^j) > \dfrac{1}{2}(u(I) - u(G^{i-1}))$

CONTRADICTION!

$$\frac{1}{2} * \frac{u(I) - u(G^{i-1})}{u(G^i) - u(G^{i-1})} > j$$

$$y_g > y_o$$

$$u(I) - u(T_j) \leq \frac{1}{2}(u(I) - u(G^{i-1}))$$

- QED, proved that Greedy Algorithm produces a solution with sum of covering time $\leq 4\times$ OPT
- NP-hard to achieve $(4-\epsilon)\times$ OPT, for any $\epsilon > 0$
- Weighted version of Min-Sum Set Cover:
    - each subset S_j of elements of the universe has weight (cost) c_j , find π minimizing sum of weighted covering times:

$$\sum_{i=1}^{m} c(\pi^i)\times(u(\pi^i)-u(\pi^{i-1})) \quad \text{where } c(I') = \sum_{j\in I'} c_j$$

  - Weighted Greedy algorithm (bang-for-the-buck) produces a solution with value $\leq 4\times OPT$
  - also called “pipelined set cover”   [Widom et al. 2005]

More general functions u

- Min-Sum Submodular Cover
- Again, want to maximize
  - $\sum_{i=1}^{m} c(\pi^i) \times (u(\pi^i) - u(\pi^{i-1}))$ where $c(I') = \sum_{j \in I'} c_j$
  - But function u can be arbitrary monotone submodular utility function $u : 2^I \to \mathscr{R}^{\geq 0}$
  - analogous weighted greedy algorithm:

    Greedy rule chooses item j that maximizes bang-for-buck $\dfrac{u(I' \cup \{j\}) - u(I')}{c_j}$
  - where I' is set of items chosen so far
  - also produces a solution with value $\leq 4 \times$ OPT [Streeter and Golovin, 2008]

Min-sum ordering problems [Happach et al. 2022]

- Very general class of problems that includes all of the above min-sum problems
- Again, want to maximize
  - $\sum_{i=1}^{m} c(\pi^i) \times (u(\pi^i) - u(\pi^{i-1}))$ where $c(I') = \sum_{j \in I'} c_j$
- Don't require u to be submodular
  just monotone and $u(\emptyset) = 0$
- Greedy algorithm
  - Greedy rule: Chooses ***subset*** S that maximizes bang-for-buck $\dfrac{u(I' \cup S) - u(I')}{c(I' \cup S) - c(I')}$
  - Elements in S are added to currrent permutation in arbitrary order
- produces a solution with value $4 \times$ OPT (also holds if c is monotone submodular and $c(\emptyset) = 0$)
- But may not be able to implement greedy rule in polynomial time

Other min-sum set cover variants

- Minimum-Latency Set Cover (introduced by [Hassin and Levin, 2005])
  - Change definition of covering time of element $e \in \mathcal{U}$ in a permutation π of I. It's the earliest step in the permutation at which ALL j such that $e \in S_j$ have appeared.
  - Studied in the scheduling literature. Shown to be special case of scheduling problem with precedence constraints.
  - A number of poly-time approximation algorithms, approximation is $2 \times$ OPT
- Generalized Min-Sum Set Cover
  - Have a *covering requirement* $k(e)$ for each $e \in \mathcal{U}$. Change definition of covering time of element e in a permutation π of N. It's the earliest step in the permutation at which $k(e)$ of the j such that $e \in S_j$ have appeared.
  - (no weights)
  - Generalizes Min-Sum Set Cover and Min-Latency Submodular Cover
  - Current best approximation achieved by poly-time algorithm is 4.642 $\times$ OPT [Bansal et al. 2023]
- and there are many others generalizations of the Min-Sum problem

Stochastic Boolean Function Evaluation

SBFE Problems

- Stochastic Boolean Function Evaluation (SBFE)
  - aka Sequential Testing of Boolean Functions
- Given representation of Boolean Function
  - e.g., $f(x_1, \ldots, x_n) = x_1 \vee x_2 \vee \ldots \vee x_n$
- Need to evaluate f on initially unknown random input $x = (x_1, \ldots, x_n)$
  - \`x_i values are independent
  - p_i = $P[x_i = 1]$ (assume 0 < p_i < 1)
- Only way to determine value of x_i is to perform "test" which has cost c_i > 0
  - Need to continue testing until have enough info to determine value of $f(x_1, \ldots, x_n)$
- SBFE Problem: Given representation of f, the p_i, and the c_i, determine the order in which to perform the tests so as to minimize the expected testing cost.
  - Testing order can be adaptive (choice of next test can depend on outcomes of previous tests)

- Testing strategy corresponds to a decision tree
  - Don’t need to output full testing strategy, just need to be able to determine next test to perform at each step
- Algorithmic problem
  - Easy to do with unlimited computational time
  - Question is whether it can be done efficiently

Motivation for SBFE problems

- Database query optimization
- Aggregating information from network of sensors
- Testing components of computer chip
- Testing network connectivity
- Medical diagnosis
- . . .

Evaluation of OR function

Example: Boolean OR

- $f(x_1, \ldots, x_n) = x_1 \vee x_2 \vee x_3$
  - $c_1 = c_2 = c_3 = 1$ (unit costs)
  - $p_1 = 0.8, \;\; p_2 = 0.5, \;\; p_3 = 0.999$
  - Optimal test ordering?

Example: Boolean OR

- Repeat with different costs
- $f(x_1, \ldots, x_n) = x_1 \vee x_2 \vee x_3$
  - $c_1 = 5 \quad c_2 = 1 \quad c_3 = 1000$
  - $p_1 = 0.8, \quad p_2 = 0.5, \quad p_3 = 0.999$
  - Optimal test ordering? And in general?

Example: Boolean OR

- Thm: Optimal to test in increasing order of the ratio

$$\frac{c_i}{p_i}$$

- Pf: (Adjacent interchange argument.)

Supppose Thm doesn't hold. Then there exists a different ordering π that is optimal and has lower expected cost. w.l.o.g., assume π is $x_1, x_2, \ldots, x_n$ (can renumber).

There must be an index $i$such that $\frac{c_i}{p_i} > \frac{c_{i+1}}{p_{i+1}}$

Consider testing using this ordering. Let T_j be indicator random variable:

$T_j = 1$ if x_j is tested

$T_j = 0$ otherwise

- Expected cost of testing using ordering π is

$$E[\text{cost of } \pi] = E[\sum_{j=1}^{n} c_j T_j] = \sum_{j=1}^{n} c_j E[T_j] \qquad \text{by linearity of expectation}$$

$$= \sum_{j=1}^{n} c_j \text{ P[test } j \text{ is performed when using } \pi]$$

$$= \sum_{j=1}^{n} c_j \prod_{k=1}^{j-1} (1 - p_k)$$

- Now consider new ordering π' that reverses order of x_i and x_{i+1}. Analogous expression for expected cost.

- Since π is optimal, $E[\text{ cost of } \pi] \leq E[\text{cost of } \pi']$
- Expressions differ only in ith term and $(i+1)$th term. Subtract off common terms.
- $c_i[(1-p_1)(1-p_2)\ldots(1-p_{i-1})] + c_{i+1}[(1-p_1)(1-p_2)\ldots(1-p_{i-1})(1-p_i)]$

  $\leq c_{i+1}\ [(1-p_1)(1-p_2)\ldots(1-p_{i-1})] + c_i\ [(1-p_1)(1-p_2)\ldots(1-p_{i-1})](1-p_{i+1})$
- Divide both sides by $(1-p_1)\ldots(1-p_{i-1})$ get

  $c_i + c_{i+1}(1-p_i) \leq c_{i+1} + c_i(1-p_{i+1})$

  $\Rightarrow -c_{i+1}p_i \leq -p_{i+1}c_i$

  $\Rightarrow \dfrac{c_i}{p_i} \leq \dfrac{c_{i+1}}{p_{i+1}}$
- But $\dfrac{c_i}{p_i} > \dfrac{c_{i+1}}{p_{i+1}}$. Contradiction! $\square$