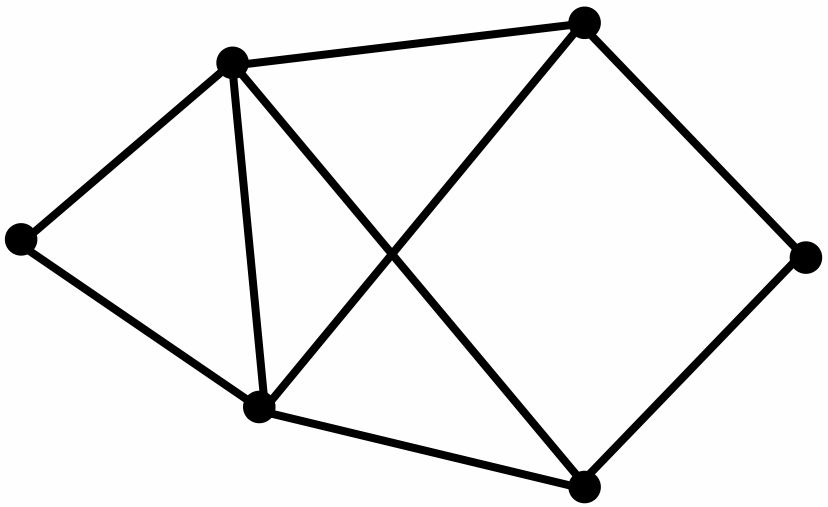
The expanding search ratio of a graph

Spyros Angelopoulos*
Christoph Dürr*
Thomas Lidbetter**

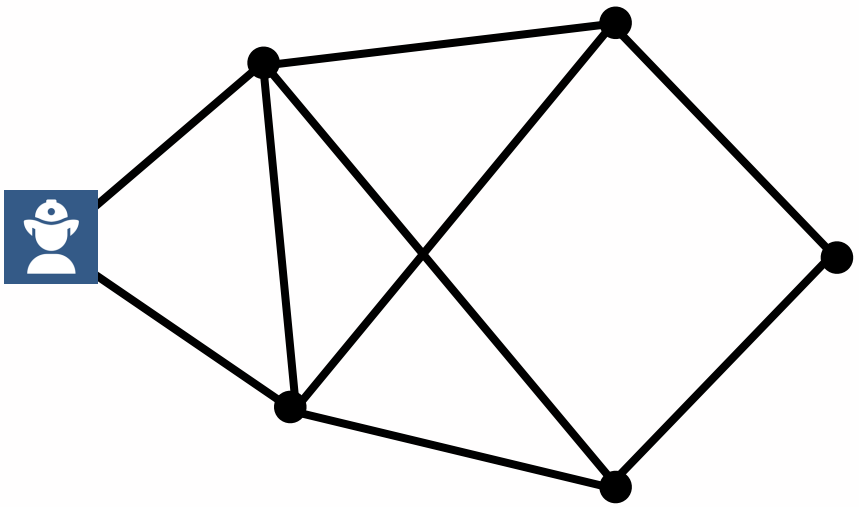
*CNRS and Sorbonne Universités, UPMC
**Rutgers Business School and LSE

Background

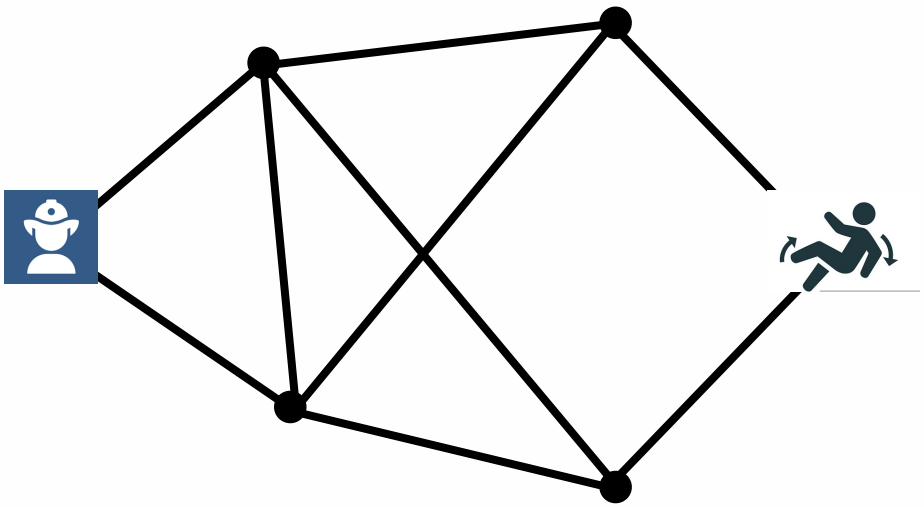
Background



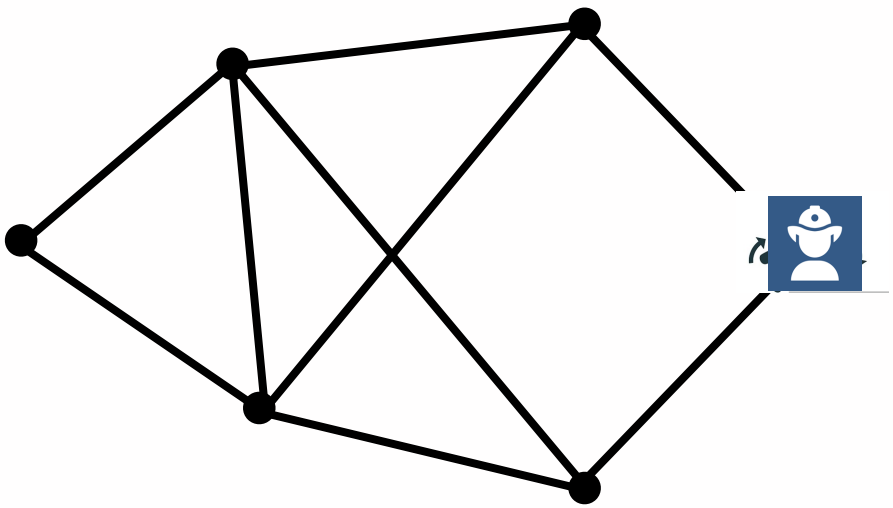
Background



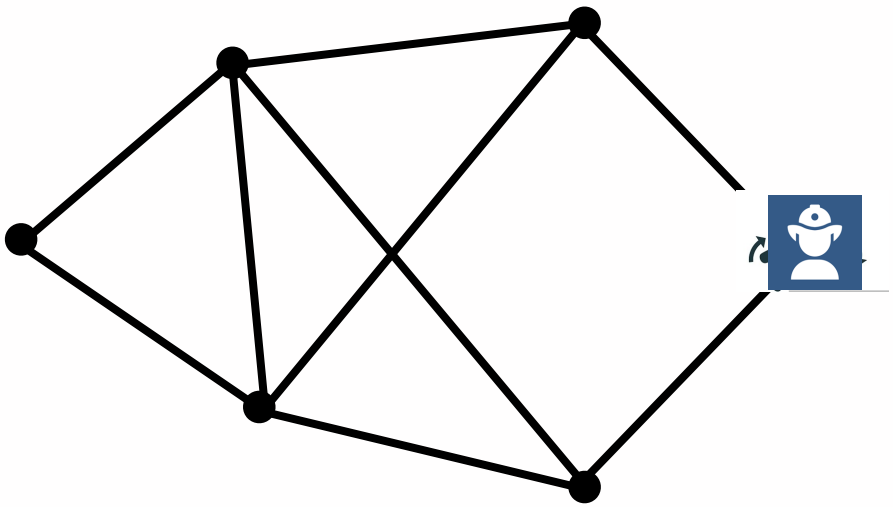
Background



Background



Background



How do we define the search cost?

What is the performance measure?

Background

Background

Searching a Fixed Graph
[Koutsoupias, Papadimitriou, Yiannakakis 1996]

Background

Searching a Fixed Graph
[Koutsoupias, Papadimitriou, Yiannakakis 1996]

Pathwise search

Background

Searching a Fixed Graph
[Koutsoupias, Papadimitriou, Yiannakakis 1996]

Pathwise search

Competitive (or search) ratio

Background

Searching a Fixed Graph
[Koutsoupias, Papadimitriou, Yiannakakis 1996]

Pathwise search

Competitive (or search) ratio

Mining coal and finding terrorists: the expanding search paradigm
[Alpern and Lidbetter 2013]

Background

Searching a Fixed Graph
[Koutsoupias, Papadimitriou, Yiannakakis 1996]

Pathwise search

Competitive (or search) ratio

Mining coal and finding terrorists: the expanding search paradigm
[Alpern and Lidbetter 2013]

Expanding search

Background

Searching a Fixed Graph
[Koutsoupias, Papadimitriou, Yiannakakis 1996]

Pathwise search

Competitive (or search) ratio

Mining coal and finding terrorists: the expanding search paradigm
[Alpern and Lidbetter 2013]

Expanding search

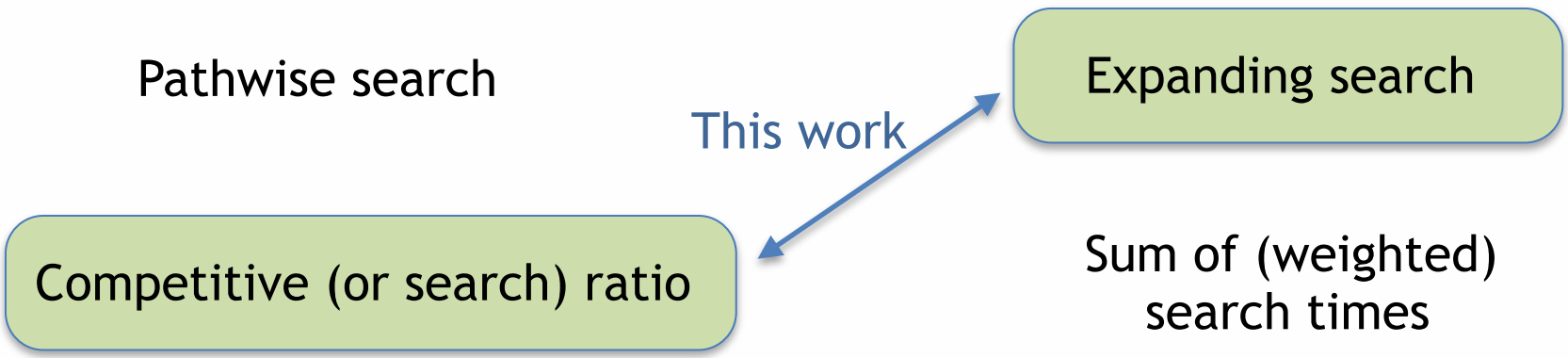
Sum of (weighted) search times

Background

Searching a Fixed Graph
[Koutsoupias, Papadimitriou, Yiannakakis 1996]

Mining coal and finding terrorists: the expanding search paradigm
[Alpern and Lidbetter 2013]

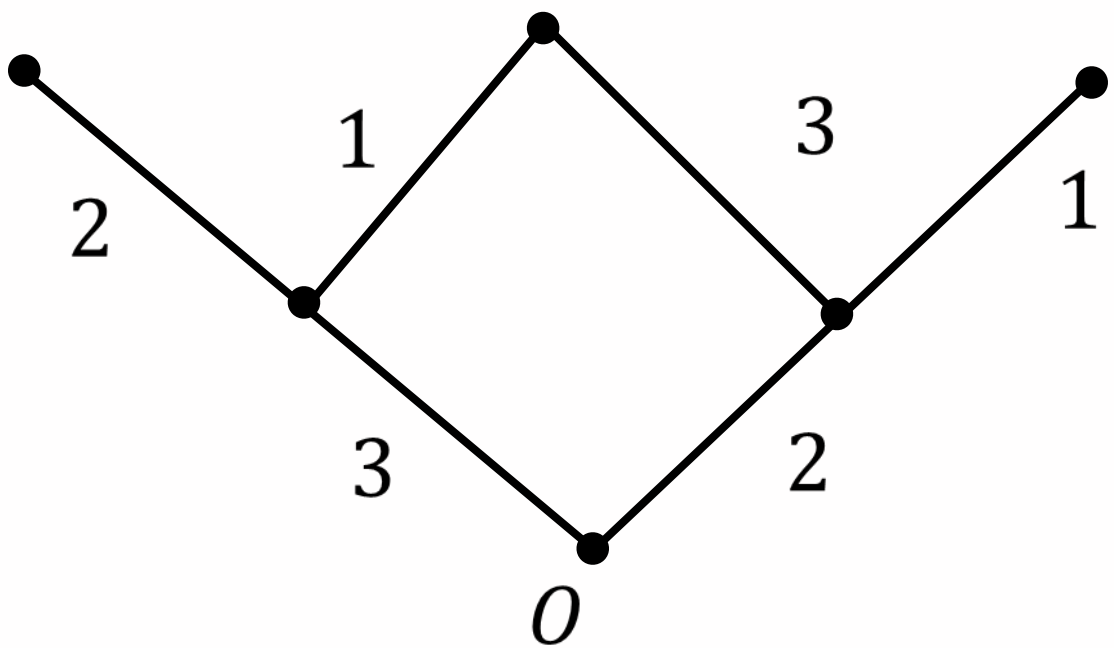
Pathwise search



Sum of (weighted) search times

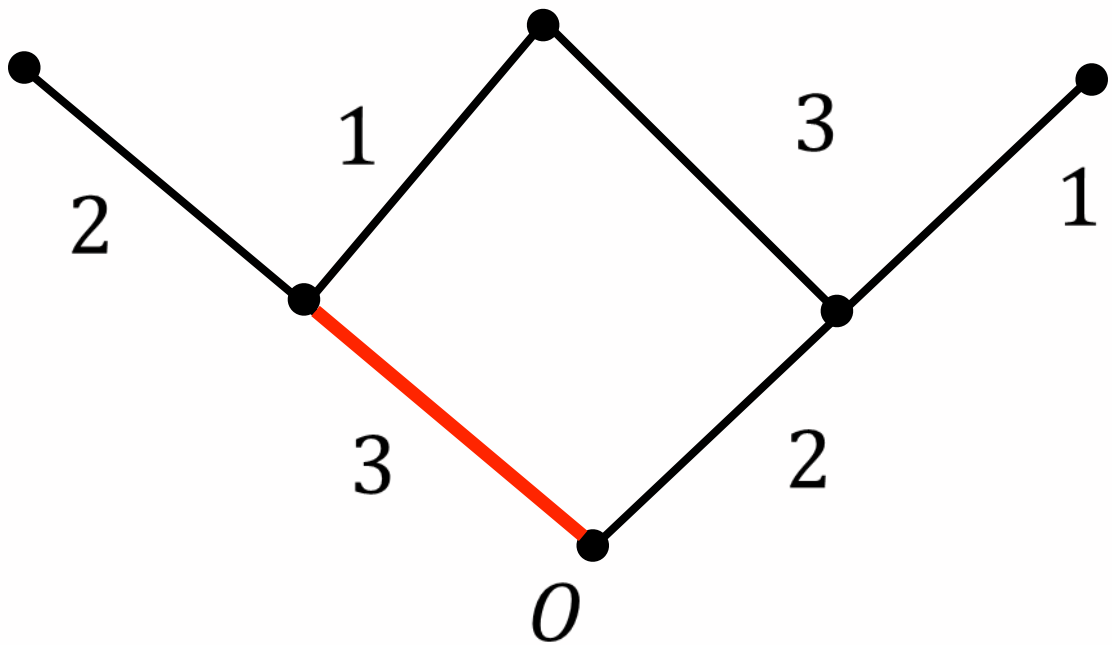
Expanding search

An expanding search of a (weighted, connected) graph with origin O is a sequence of edges each one of which is incident to a previously searched vertex



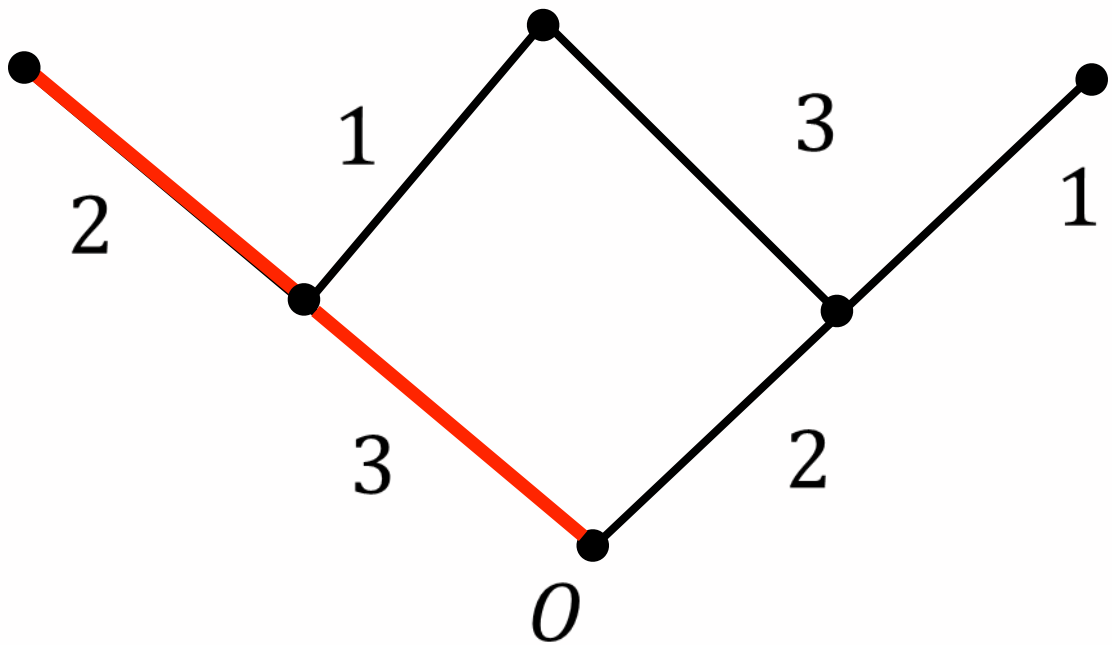
Expanding search

An expanding search of a (weighted, connected) graph with origin O is a sequence of edges each one of which is incident to a previously searched vertex



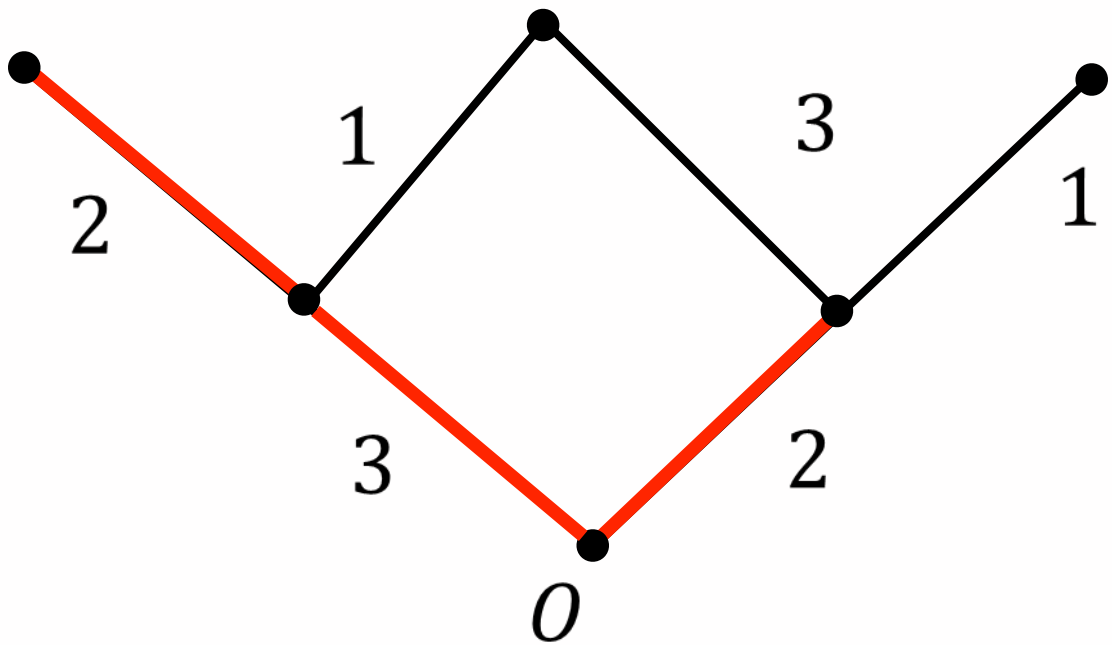
Expanding search

An expanding search of a (weighted, connected) graph with origin O is a sequence of edges each one of which is incident to a previously searched vertex



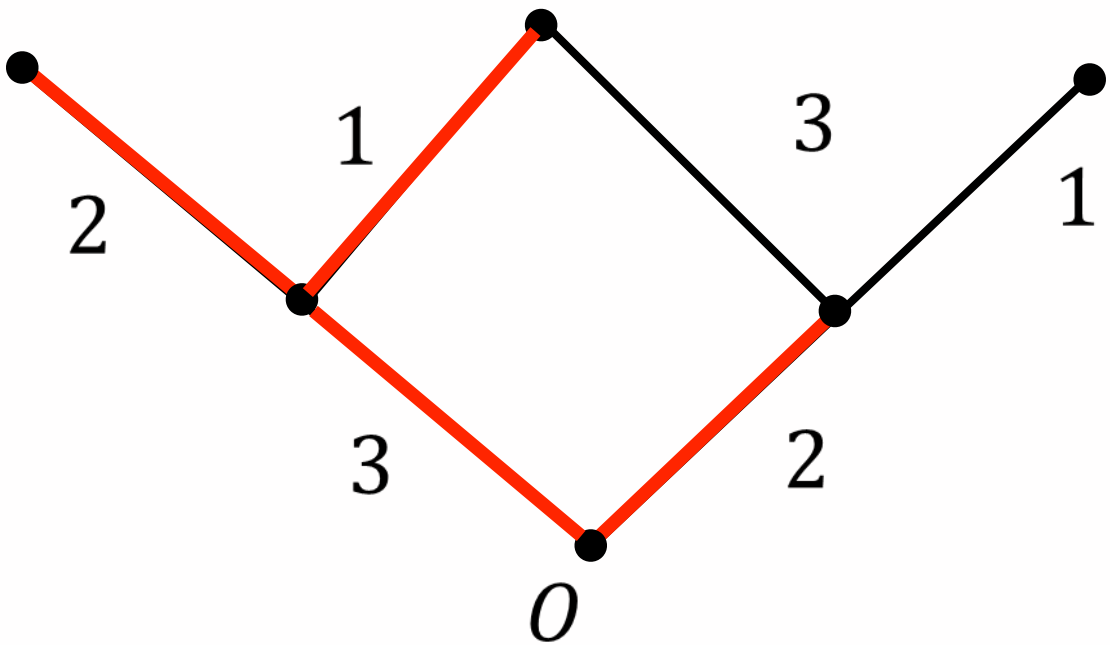
Expanding search

An expanding search of a (weighted, connected) graph with origin O is a sequence of edges each one of which is incident to a previously searched vertex



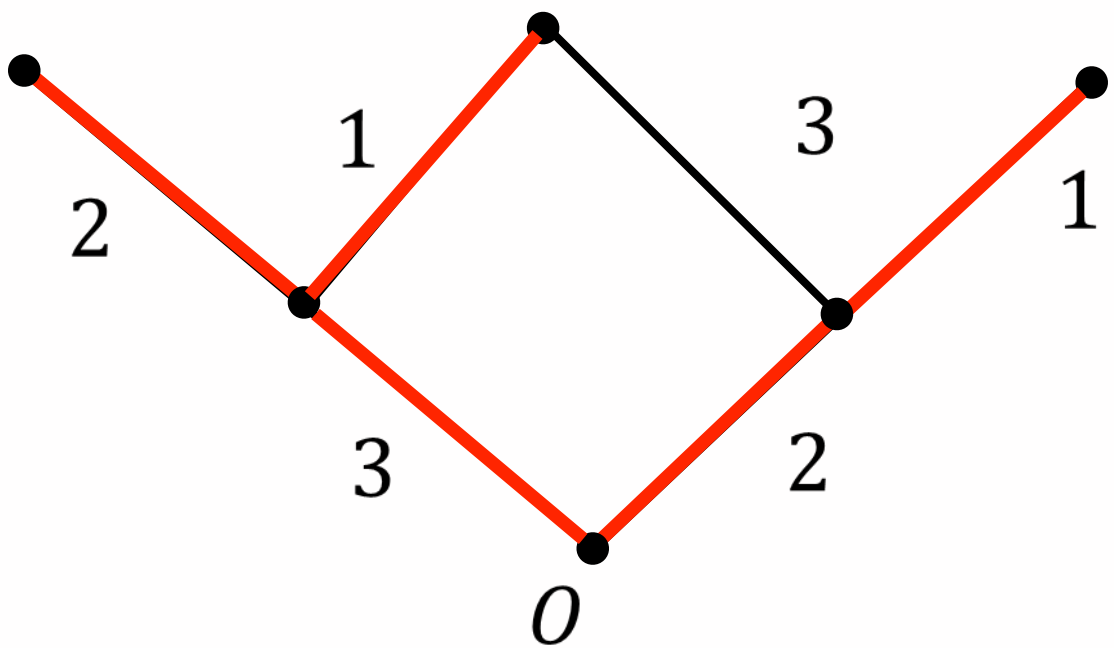
Expanding search

An expanding search of a (weighted, connected) graph with origin O is a sequence of edges each one of which is incident to a previously searched vertex

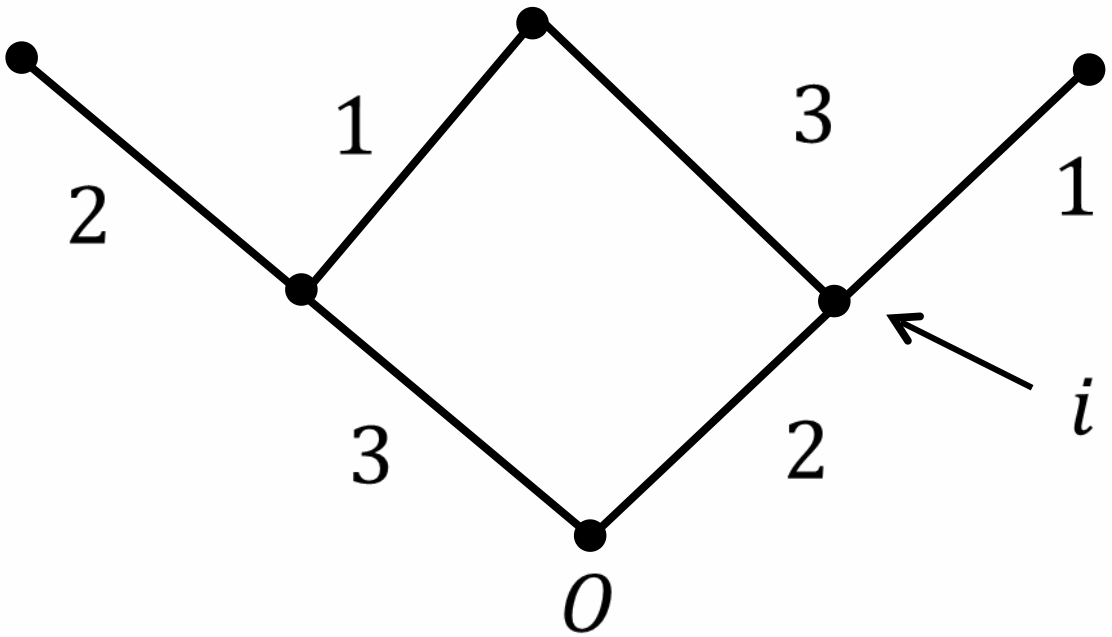


Expanding search

An expanding search of a (weighted, connected) graph with origin O is a sequence of edges each one of which is incident to a previously searched vertex

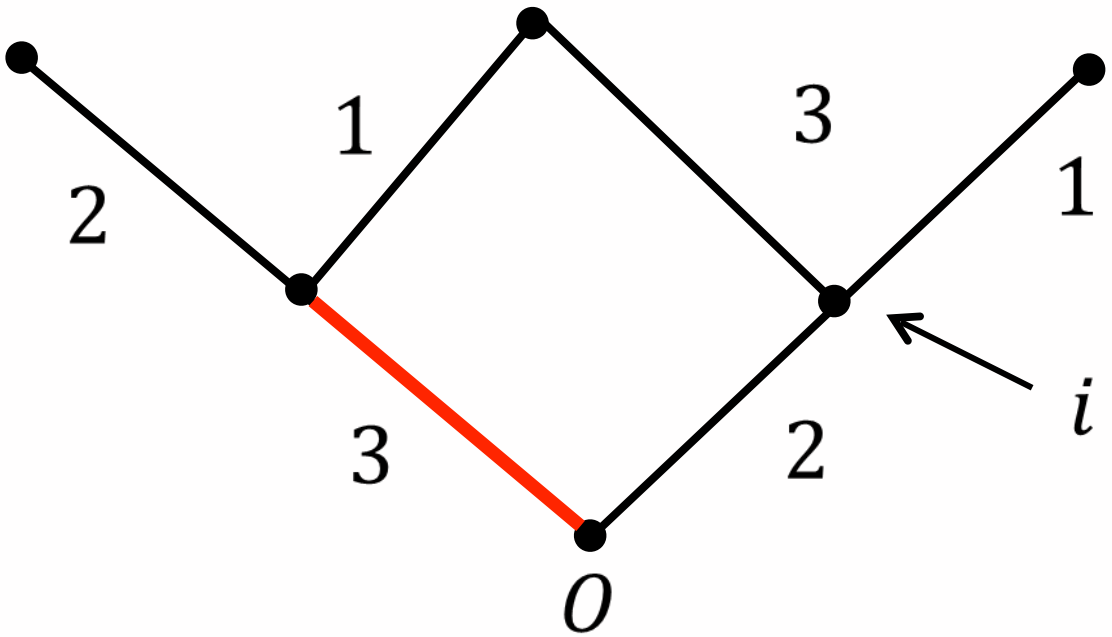


Search time



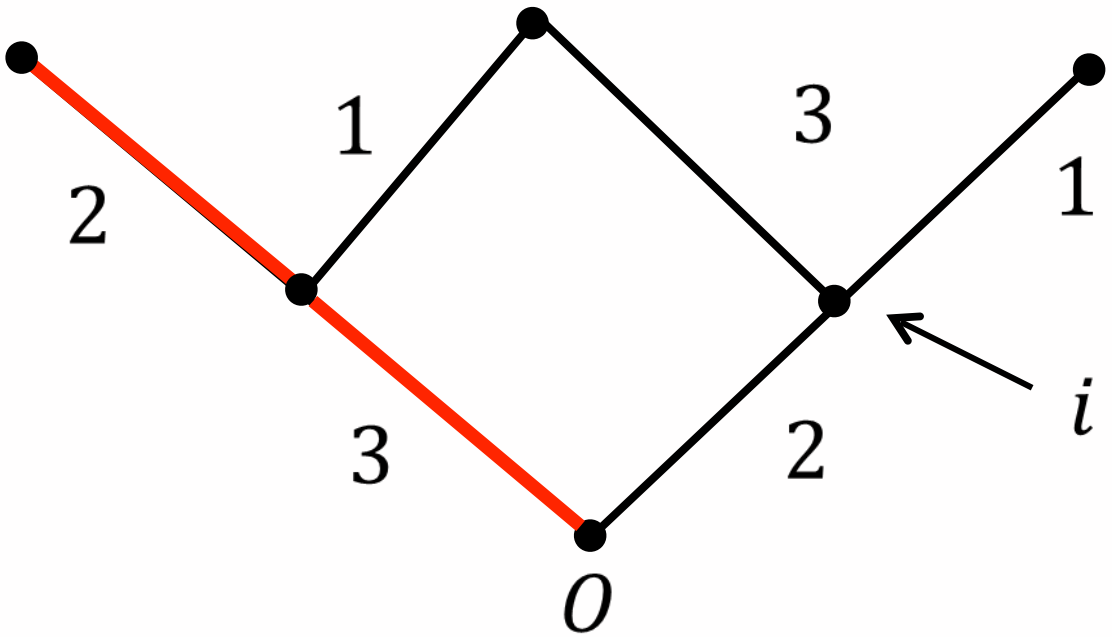
For a search S and vertex i, the search time $T(S, i)$ is the time i is first discovered E.g., $T(S, i) =$

Search time



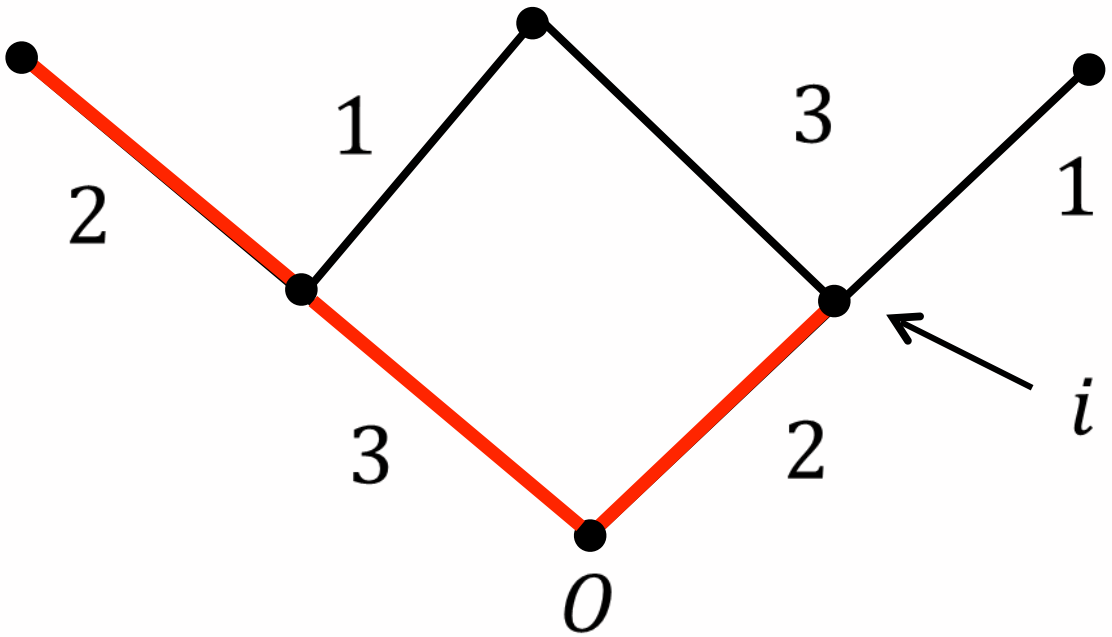
For a search S and vertex i, the search time $T(S, i)$ is the time i is first discovered E.g., $T(S, i) = 3$

Search time



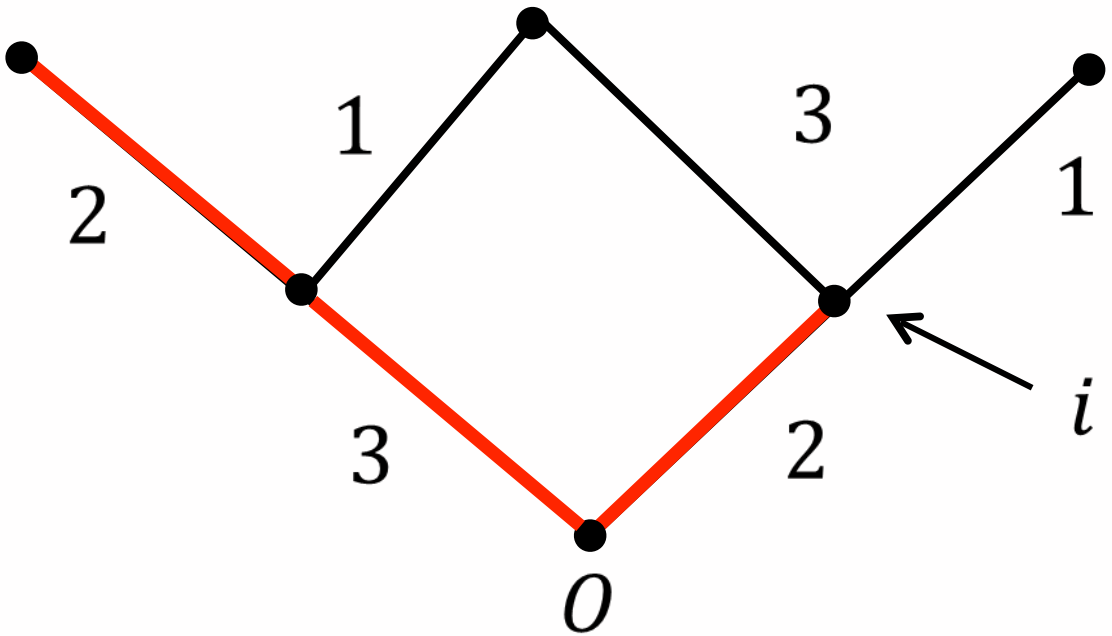
For a search S and vertex i, the search time $T(S, i)$ is the time i is first discovered E.g., $T(S, i) = 3+2$

Search time



For a search S and vertex i, the search time $T(S, i)$ is the time i is first discovered E.g., $T(S, i) = 3+2+2 = 7$

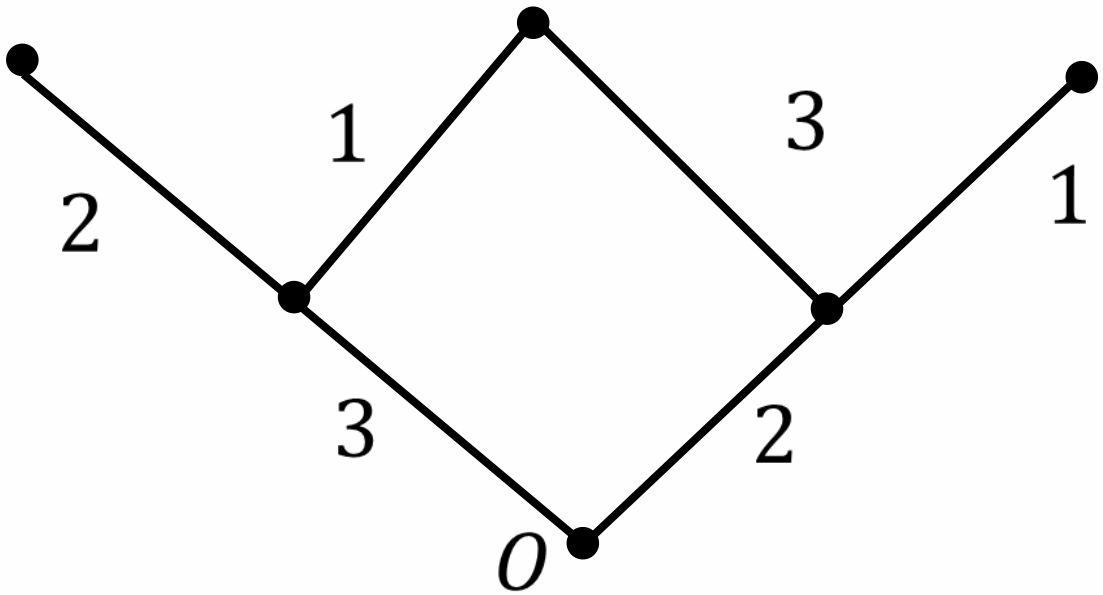
Search time



For a search S and vertex i, the search time $T(S,i)$ is the time i is first discovered E.g., $T(S,i) = 3+2+2 = 7$

The normalized search time $\hat{T}(S,i)$ is defined as $T(S,i)/d(O,i)$
E.g., $\hat{T}(S,i) = 7/2$

Search ratio

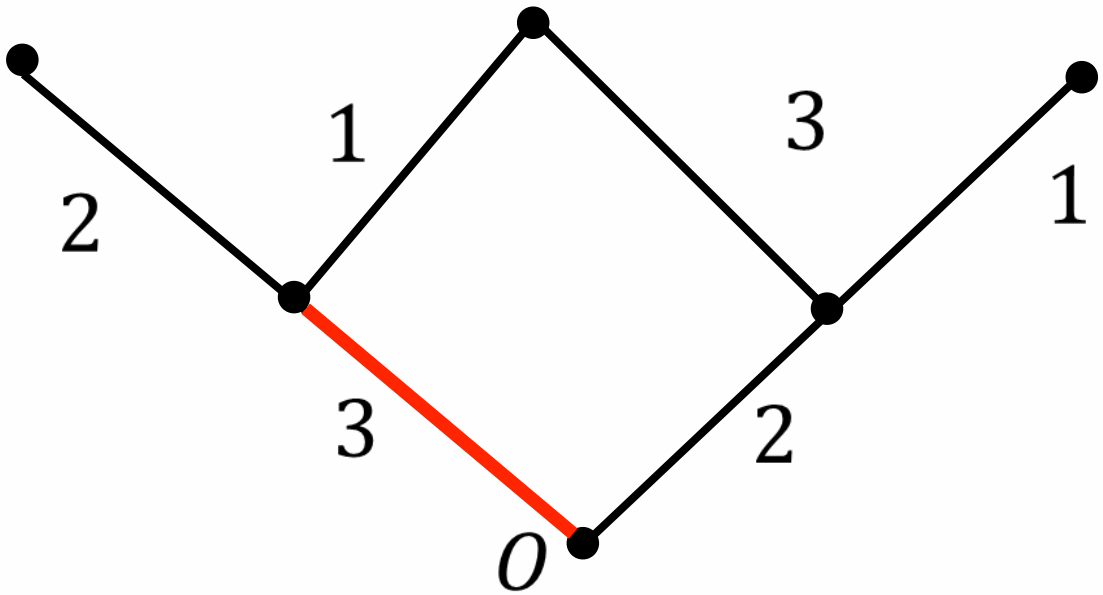


The search ratio σ_S of S is $\max_i \hat{T}(S,i)$

The search ratio σ of G is $\min_S \sigma_S = \min_S \max_i \hat{T}(S,i)$

If a strategy S minimizes the search ratio it is optimal

Search ratio

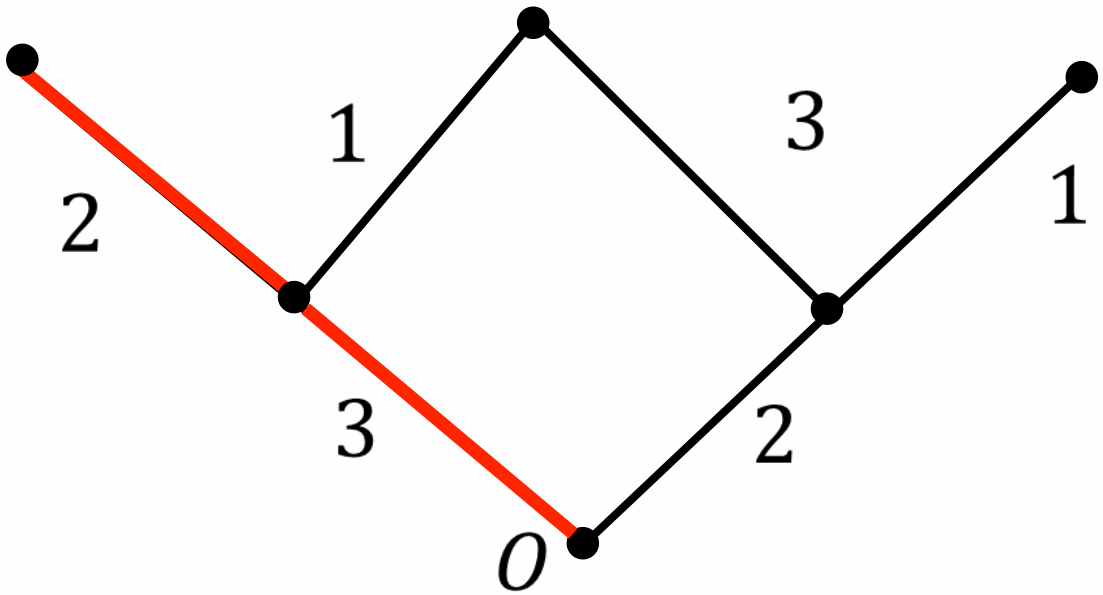


The search ratio σ_S of S is $\max_i \hat{T}(S,i)$

The search ratio σ of G is $\min_S \sigma_S = \min_S \max_i \hat{T}(S,i)$

If a strategy S minimizes the search ratio it is optimal

Search ratio

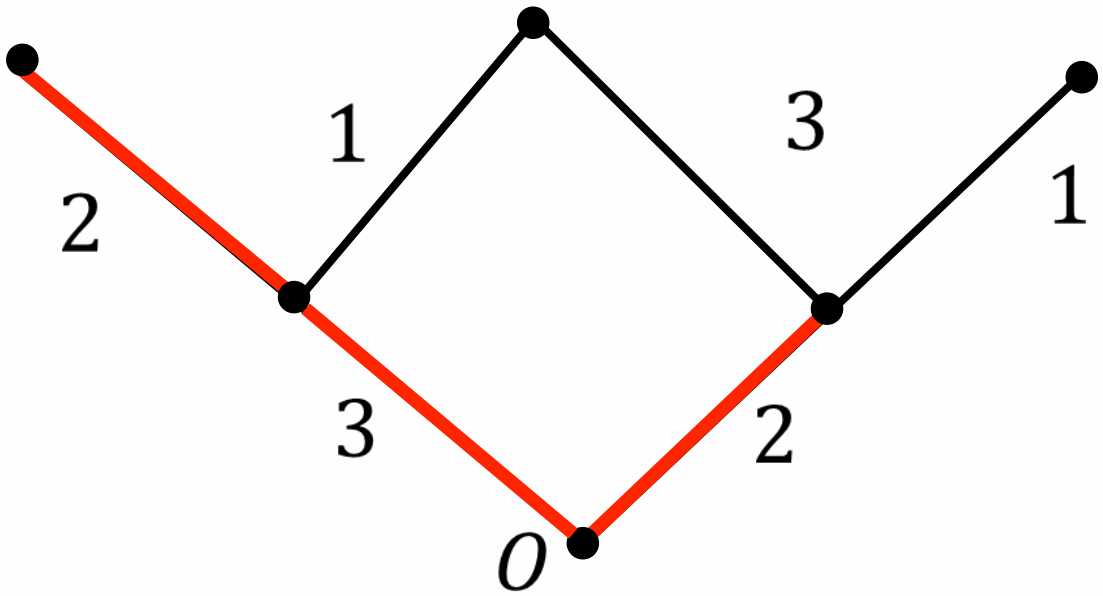


The search ratio σ_S of S is $\max_i \hat{T}(S,i)$

The search ratio σ of G is $\min_S \sigma_S = \min_S \max_i \hat{T}(S,i)$

If a strategy S minimizes the search ratio it is optimal

Search ratio

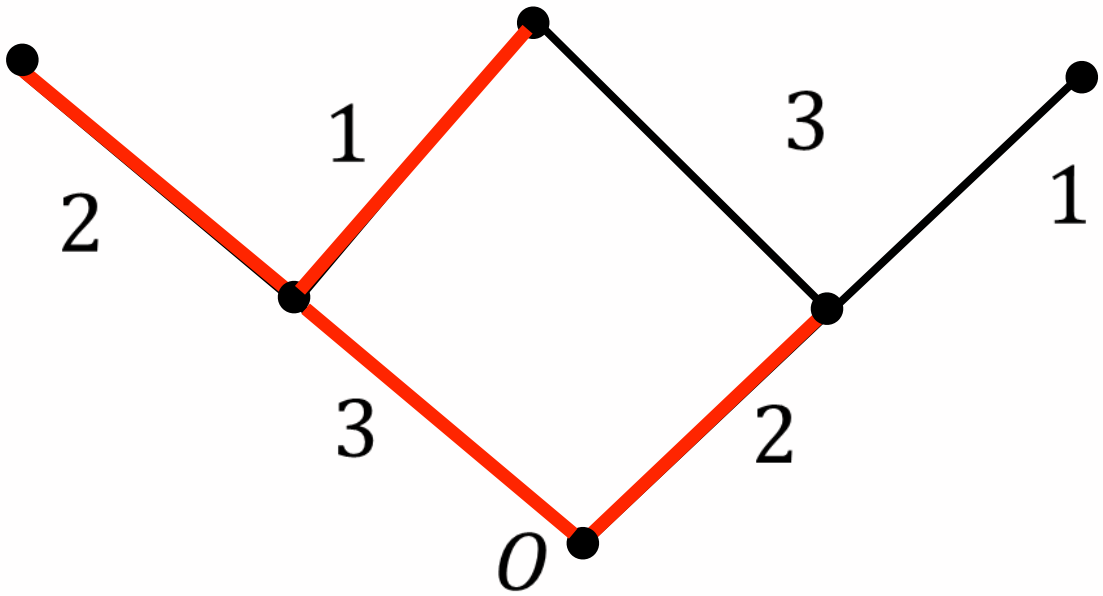


The search ratio σ_S of S is $\max_i \hat{T}(S,i)$

The search ratio σ of G is $\min_S \sigma_S = \min_S \max_i \hat{T}(S,i)$

If a strategy S minimizes the search ratio it is optimal

Search ratio

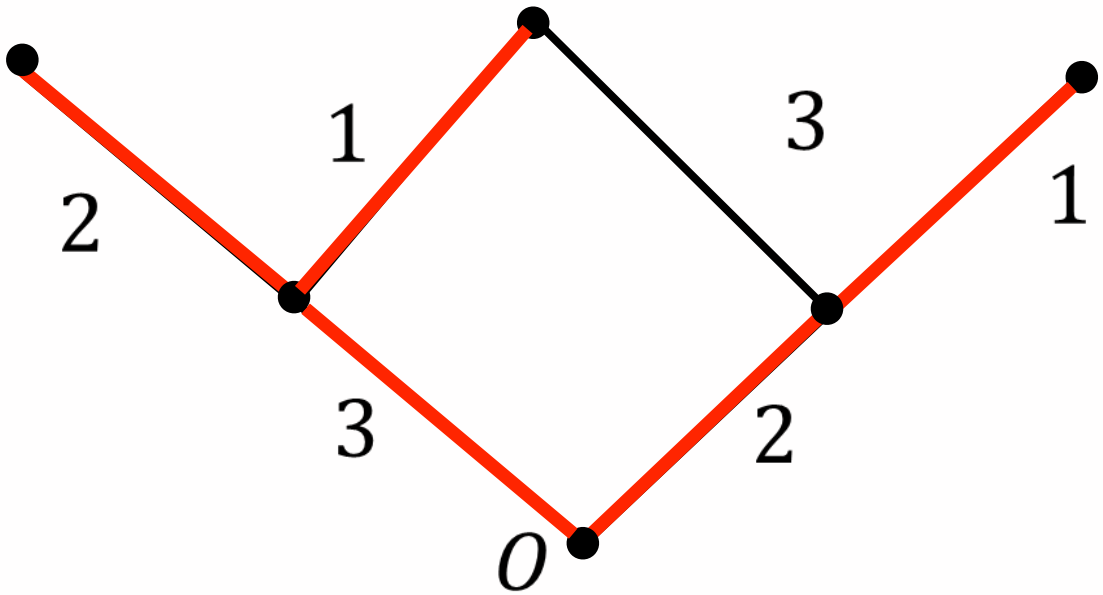


The search ratio σ_S of S is $\max_i \hat{T}(S,i)$

The search ratio σ of G is $\min_S \sigma_S = \min_S \max_i \hat{T}(S,i)$

If a strategy S minimizes the search ratio it is optimal

Search ratio

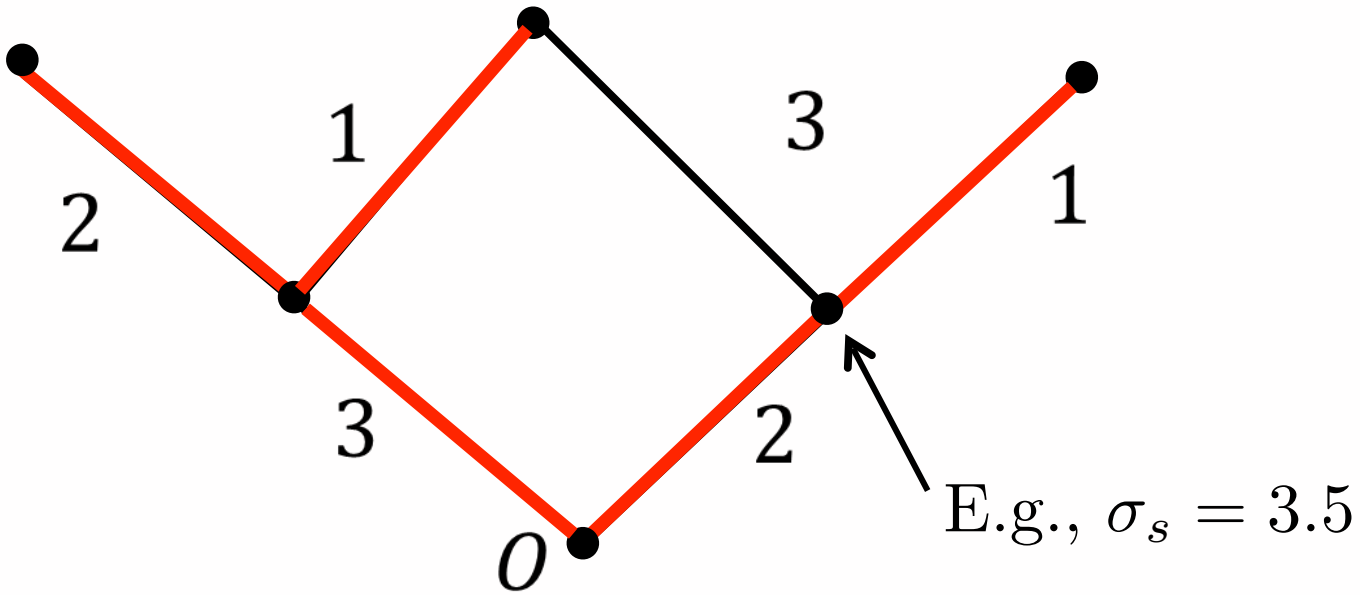


The search ratio σ_S of S is $\max_i \hat{T}(S,i)$

The search ratio σ of G is $\min_S \sigma_S = \min_S \max_i \hat{T}(S,i)$

If a strategy S minimizes the search ratio it is optimal

Search ratio



The search ratio σ_S of S is $\max_i \hat{T}(S, i)$

The search ratio σ of G is $\min_S \sigma_S = \min_S \max_i \hat{T}(S, i)$

If a strategy S minimizes the search ratio it is optimal

Easy and hard cases

For trees or graphs with unit edge weights, it is optimal to search the vertices in order of their distance from *O*.

For general graphs deciding whether $\sigma \leq R$ is NP-hard

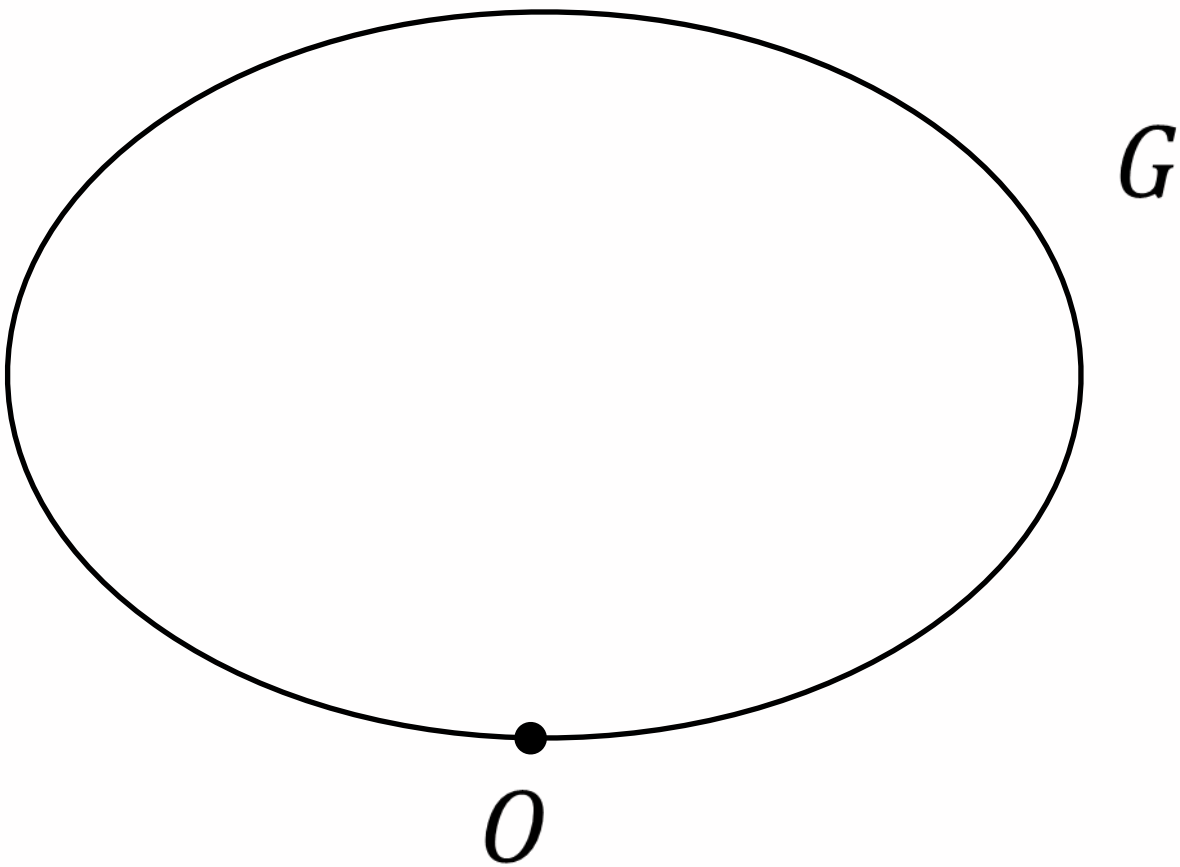
Proof: Reduction from 3-SAT

General graphs

There is a polynomial-time algorithm that approximates the search ratio within a factor of $4 \ln 4 + \epsilon$

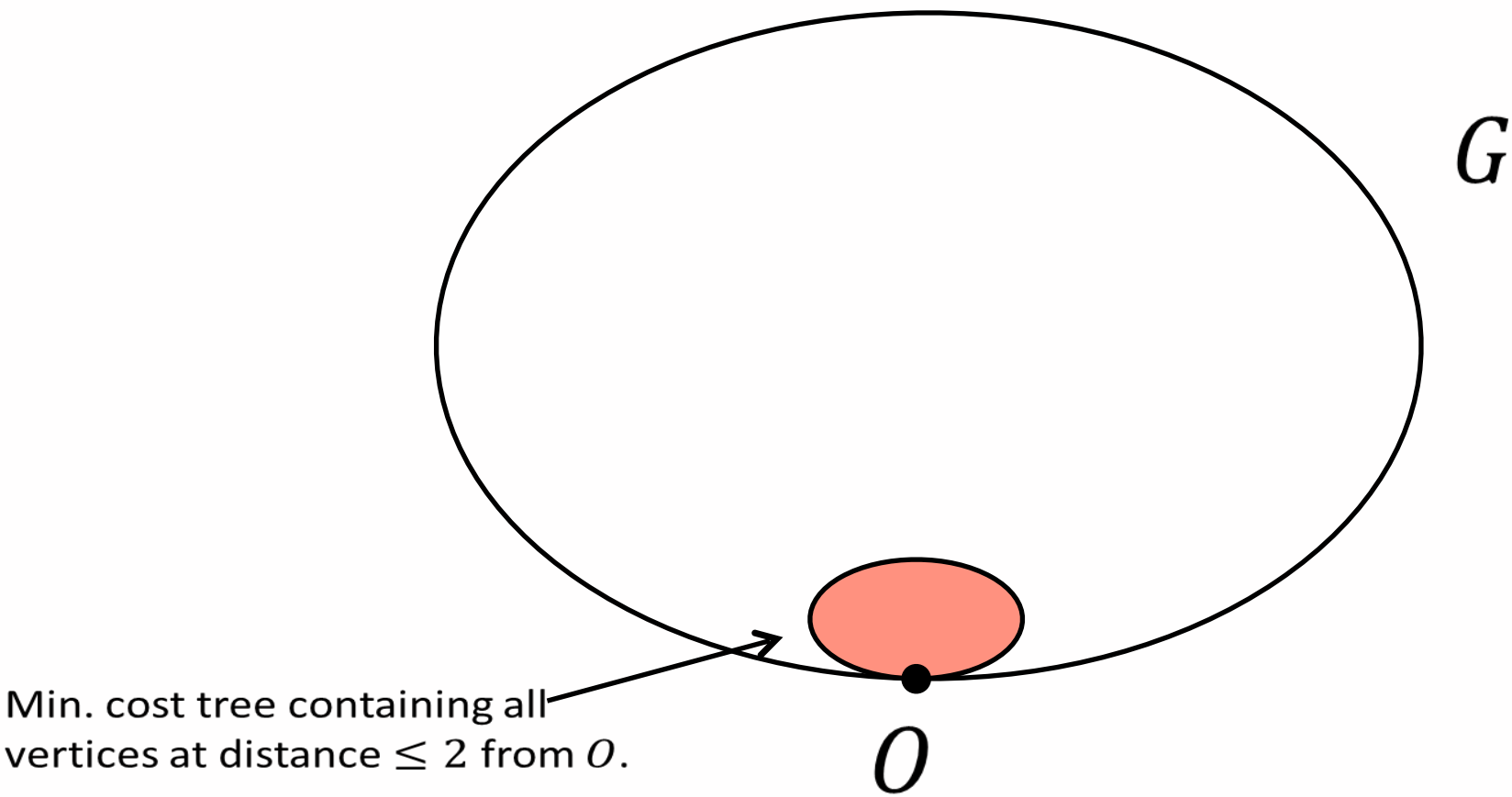
General graphs

There is a polynomial-time algorithm that approximates the search ratio within a factor of $4 \ln 4 + \epsilon$



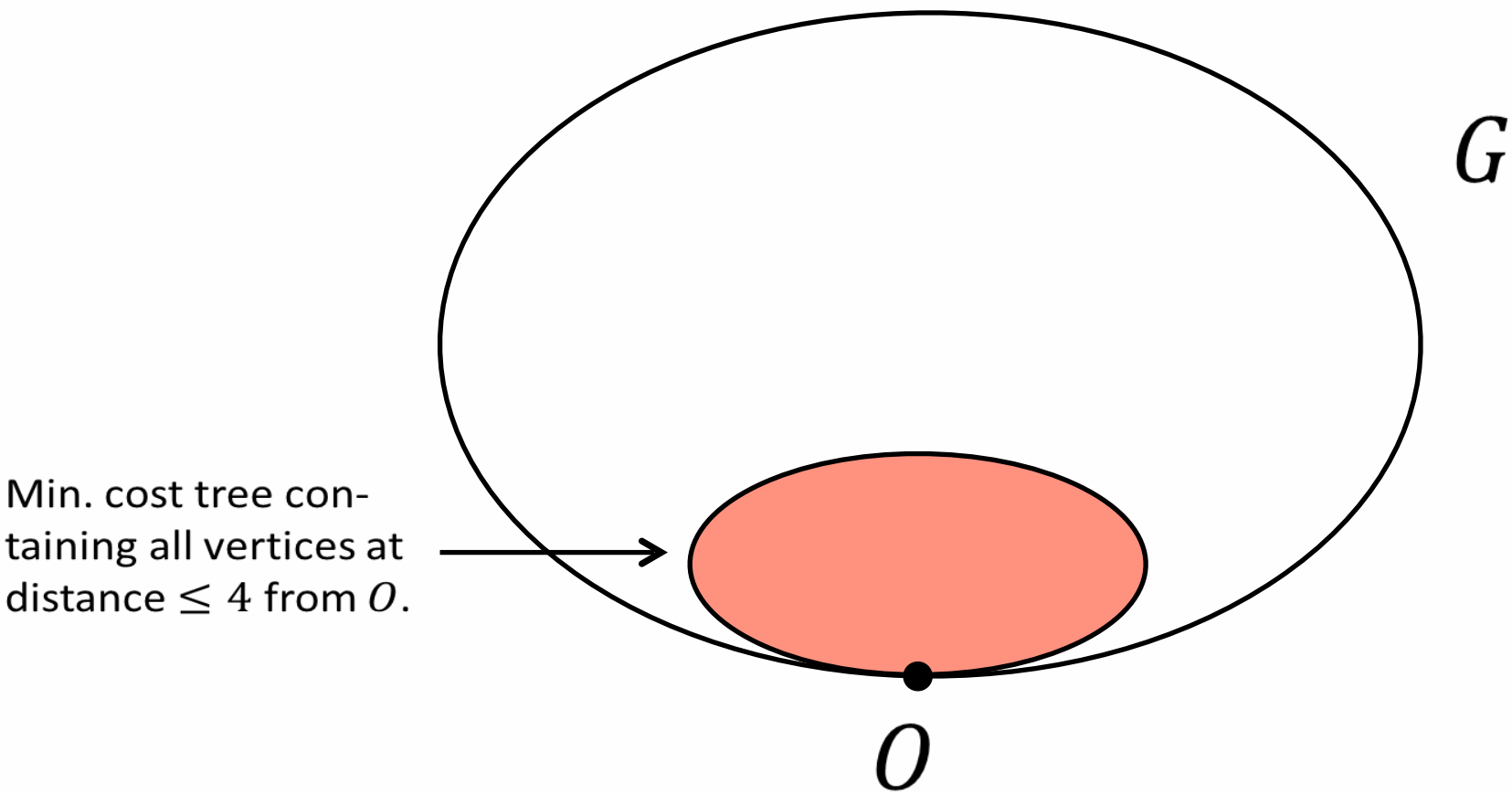
General graphs

There is a polynomial-time algorithm that approximates the search ratio within a factor of $4 \ln 4 + \epsilon$



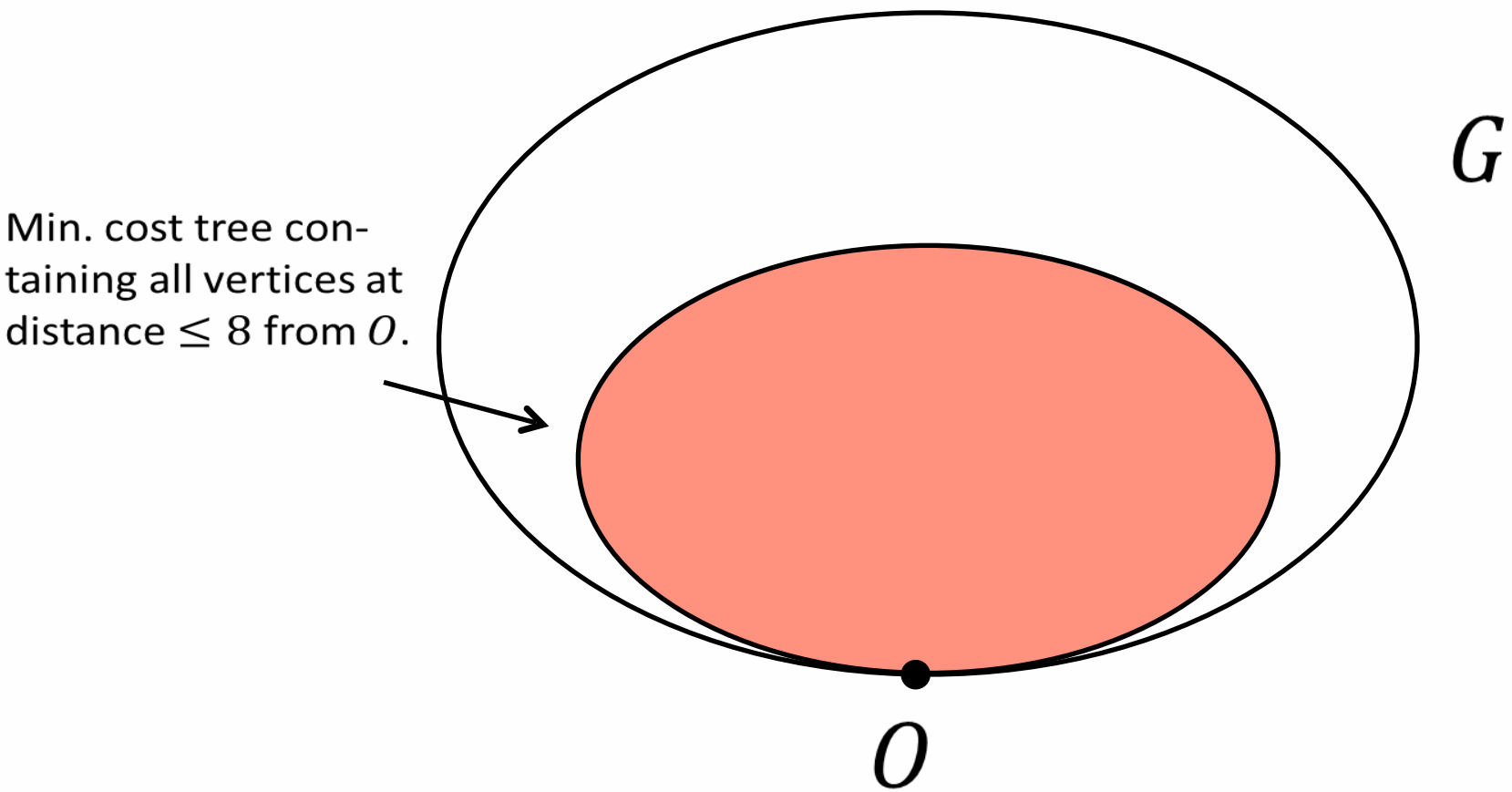
General graphs

There is a polynomial-time algorithm that approximates the search ratio within a factor of $4 \ln 4 + \epsilon$



General graphs

There is a polynomial-time algorithm that approximates the search ratio within a factor of $4 \ln 4 + \epsilon$



Randomized search ratio

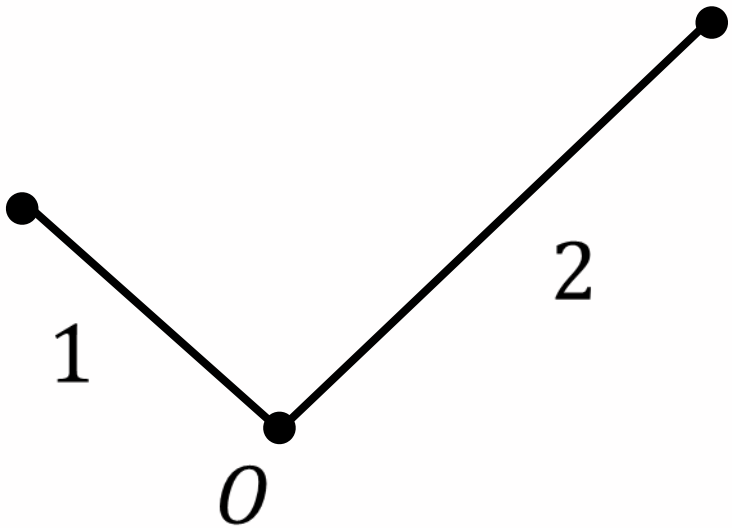
For a randomized search s and a vertex i, the expected search time and expected normalized search time are denoted by $T(s,i)$ and $\hat{T}(s,i)$

The randomized search ratio ρ_s of a random search s is $\max_i \hat{T}(s,i)$

The randomized search ratio ρ of a graph is $\min_s \rho_s = \min_s \max_i \hat{T}(s,i)$

Game theoretic interpretation

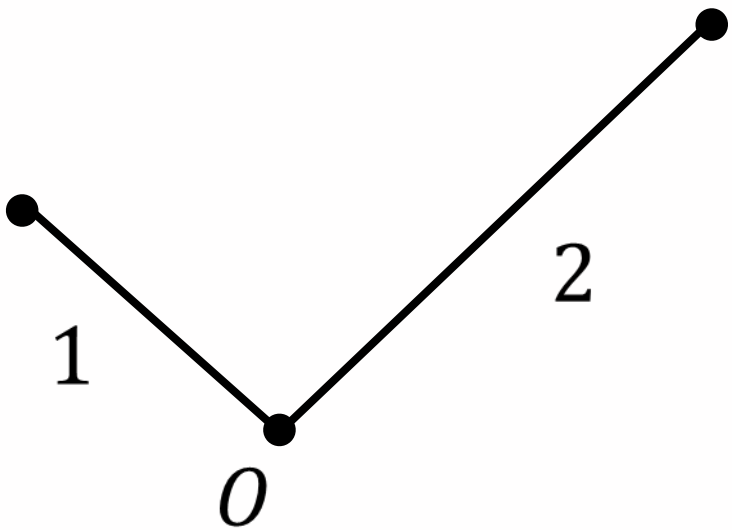
Finding the optimal randomized search is equivalent to finding the optimal strategy in a zero-sum search game between a Searcher and a Hider.



Hider/	1,2	2,1
1	1	3
2	3/2	1

Game theoretic interpretation

Finding the optimal randomized search is equivalent to finding the optimal strategy in a zero-sum search game between a Searcher and a Hider.



Hider/	1,2	2,1
1	1	3
2	3/2	1

Optimal randomized search: start with short edge with probability $4/5$ and long edge with probability $1/5$.

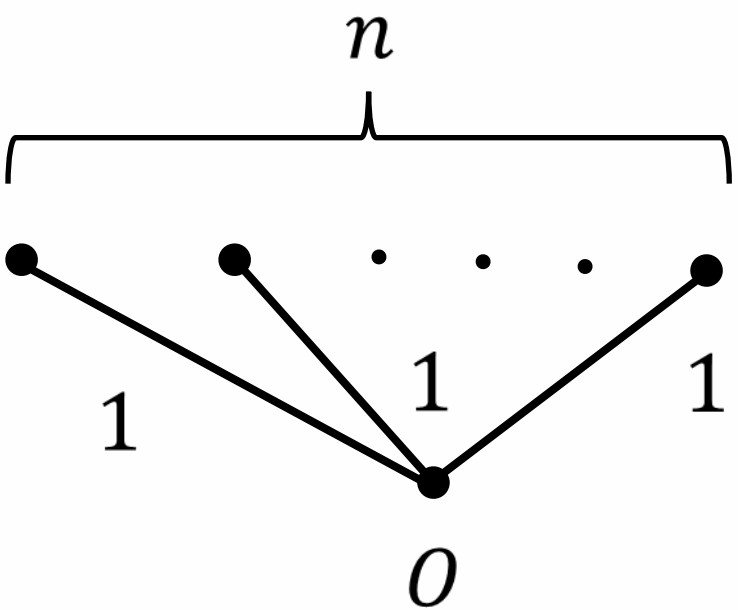
Randomized search ratio, $\rho = 7/5$.

2-approximate strategy

For trees or graphs with unit-length edges, the optimal deterministic strategy is a 2-approximation of the optimal randomized strategy

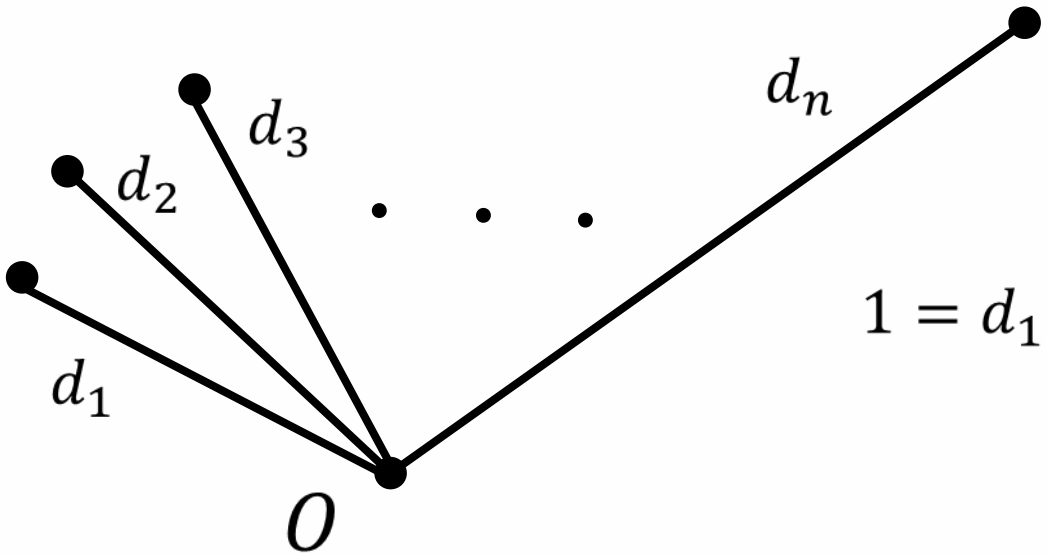
2-approximate strategy

For trees or graphs with unit-length edges, the optimal deterministic strategy is a 2-approximation of the optimal randomized strategy



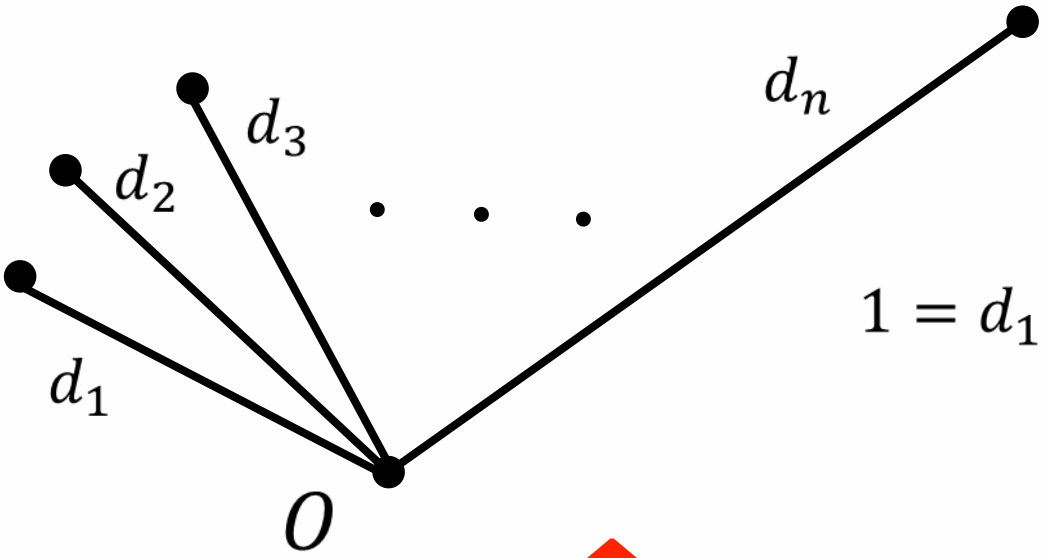
$$\sigma = n \text{ and } \rho \approx n/2$$

Randomized star search



$$1 = d_1 \leq d_2 \leq \cdots \leq d_n$$

Randomized star search



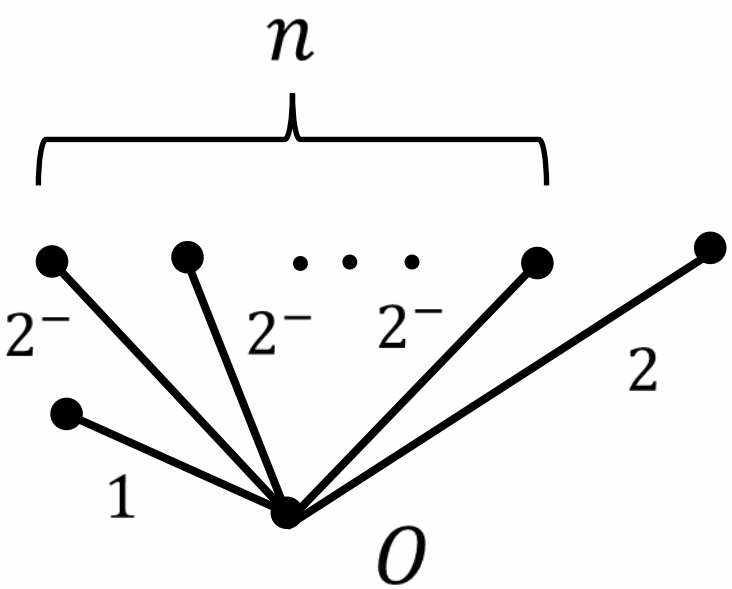
$$1 = d_1 \leq d_2 \leq \cdots \leq d_n$$

Idea: randomize in “stages”

Idea: randomize in "stages"

First approach: Randomize between all edges with length l satisfying $2^j \leq l < 2^{j+1}$ for $j = 0, 1, \ldots$

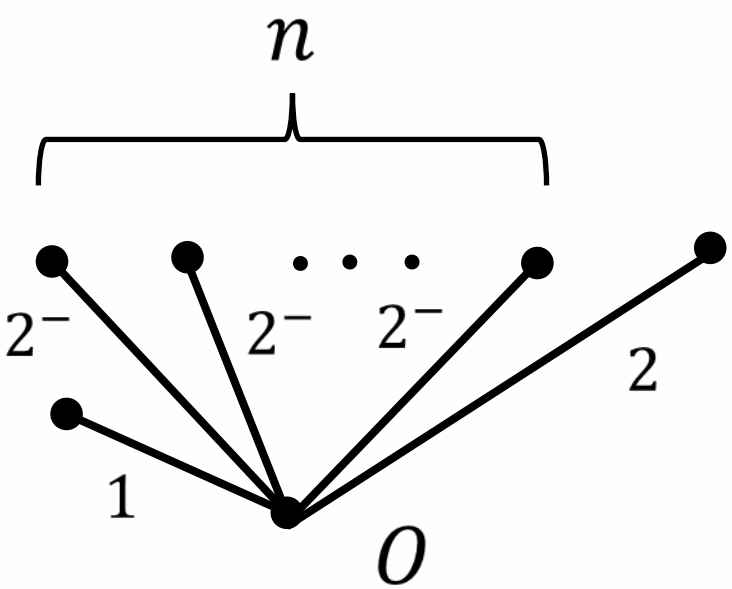
Unfortunately it does not work.....



Idea: randomize in "stages"

First approach: Randomize between all edges with length l satisfying $2^j \leq l < 2^{j+1}$ for $j = 0, 1, \ldots$

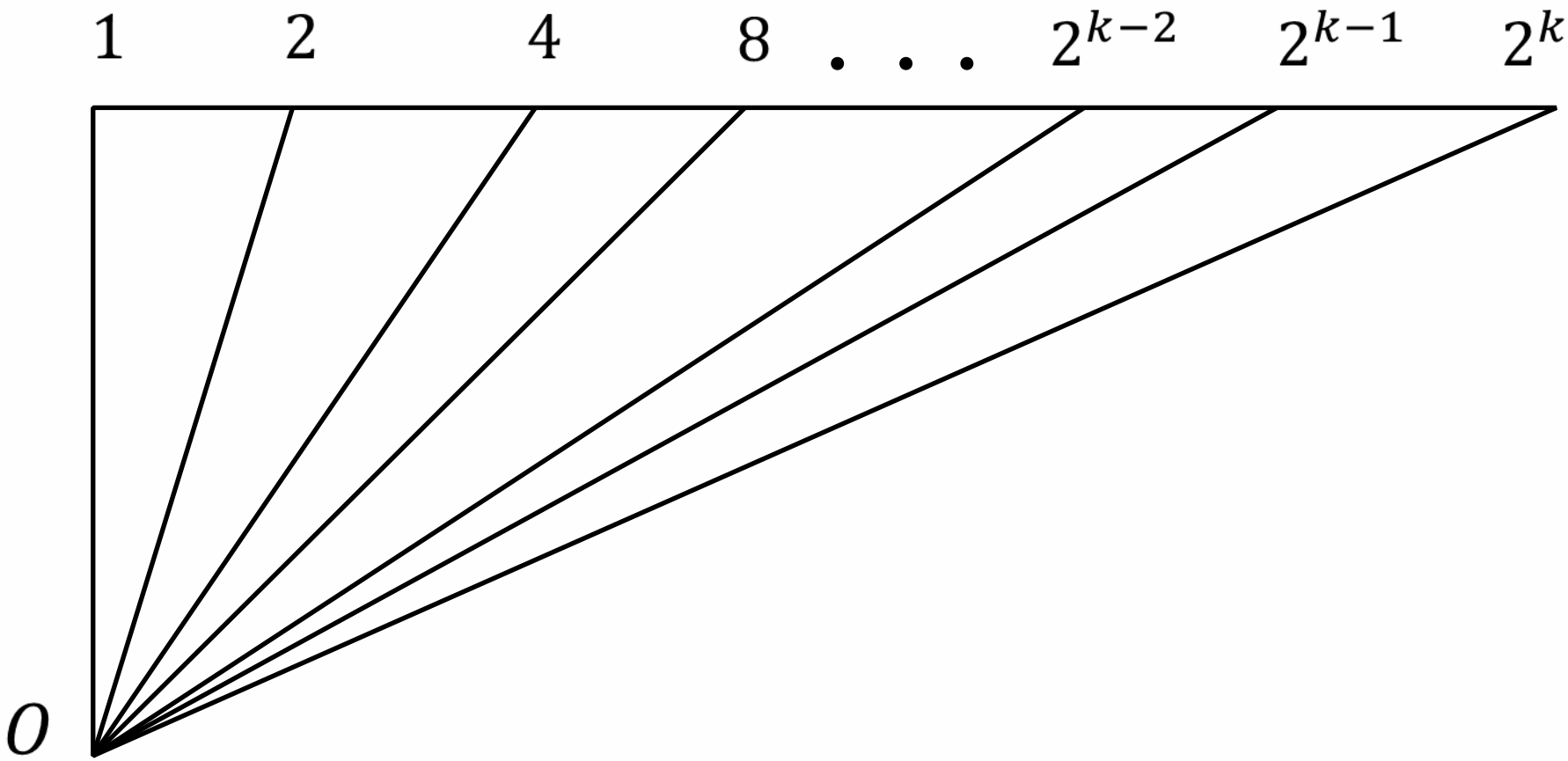
Unfortunately it does not work.....



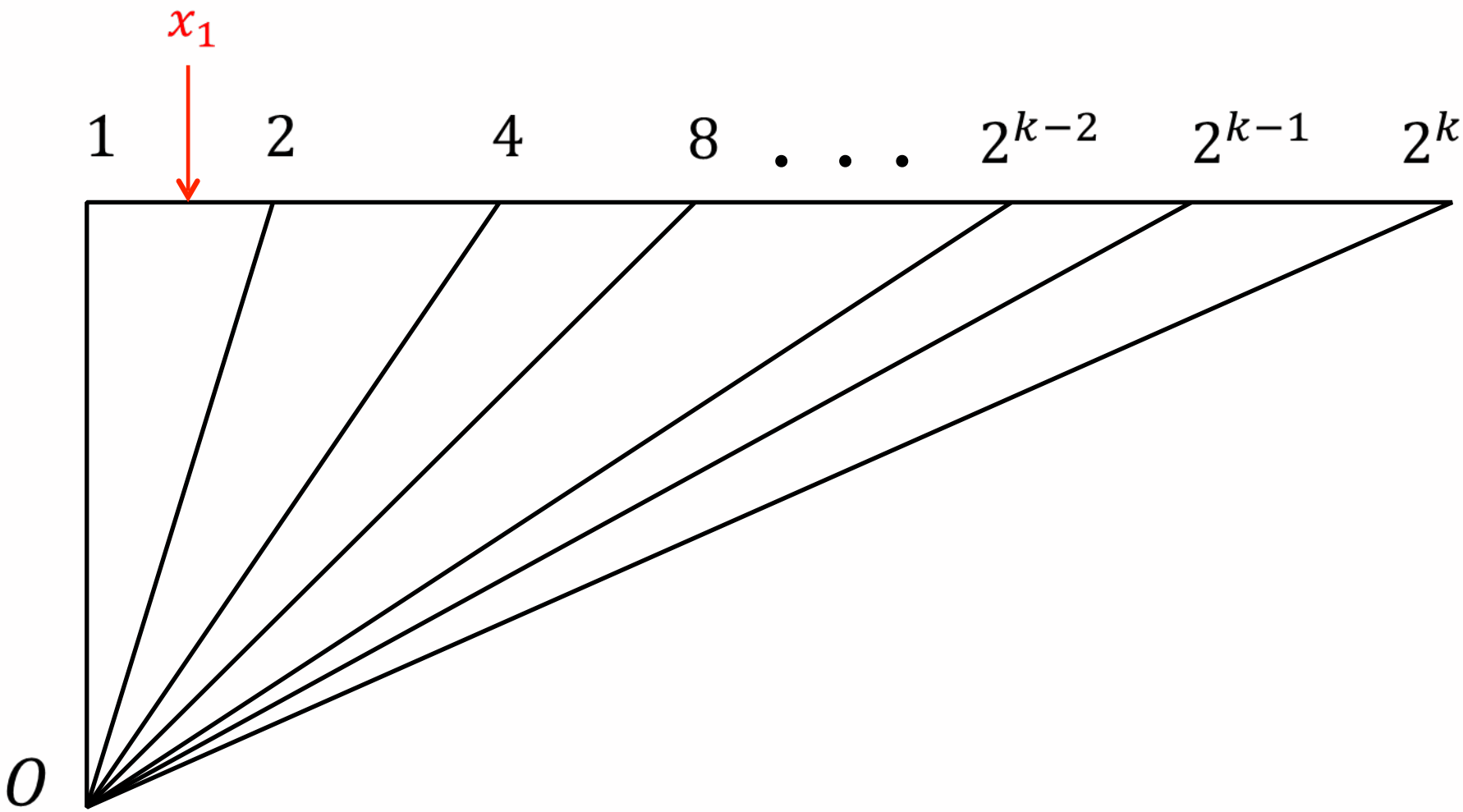
This has search ratio $\approx \frac{2n}{2} = n.$

But $\rho \approx \frac{n}{2}$.

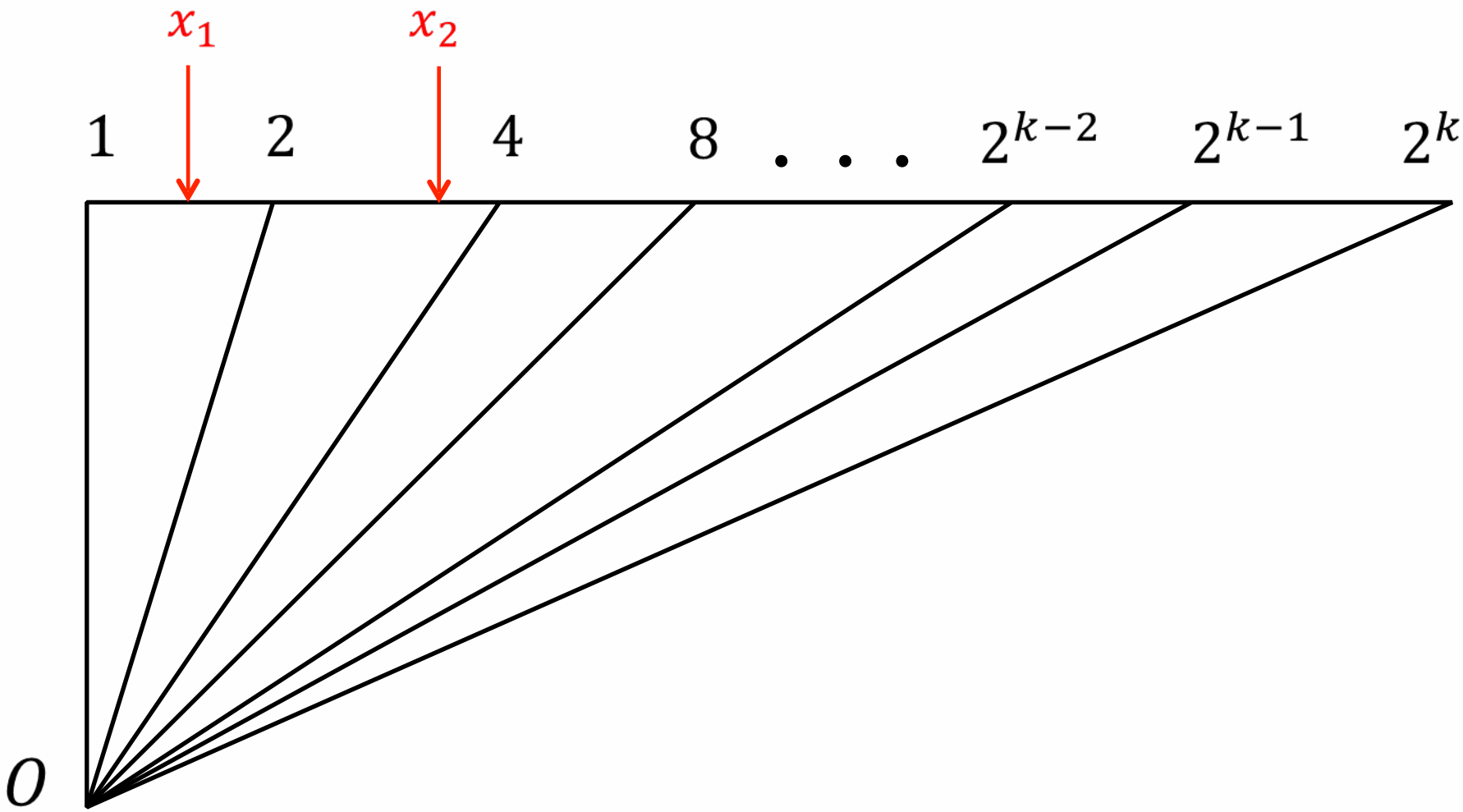
Better idea: randomize in "random stages"



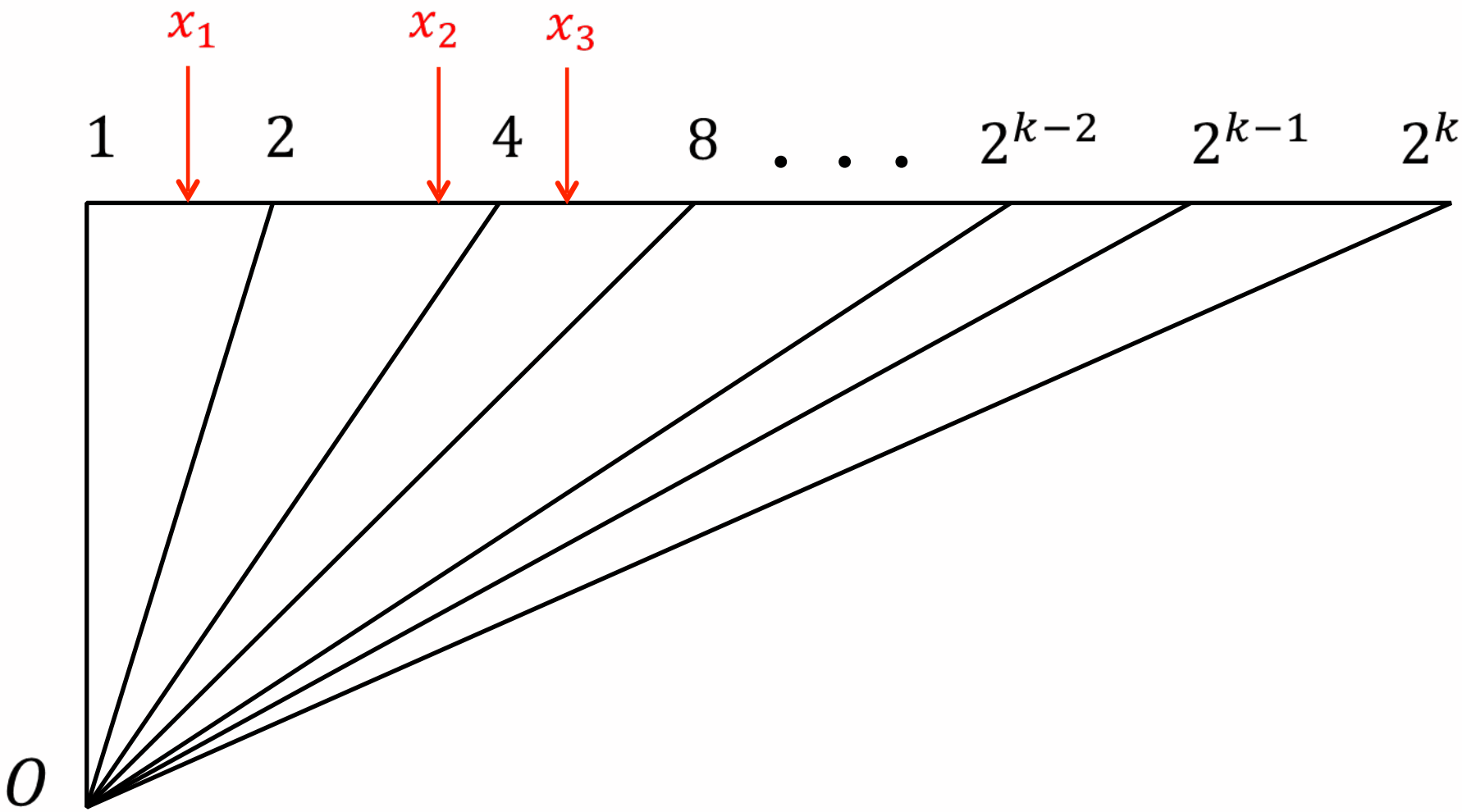
Better idea: randomize in "random stages"



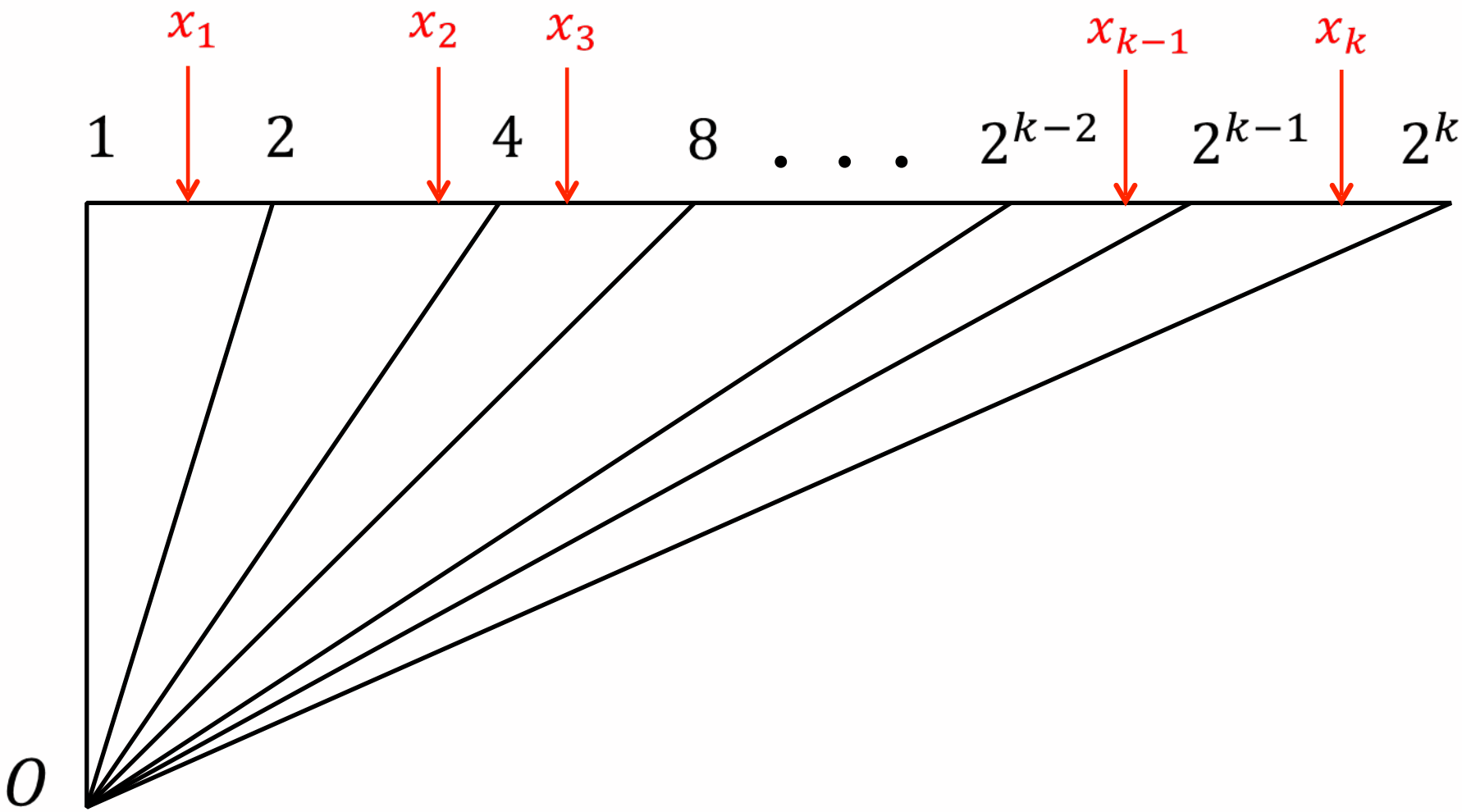
Better idea: randomize in "random stages"



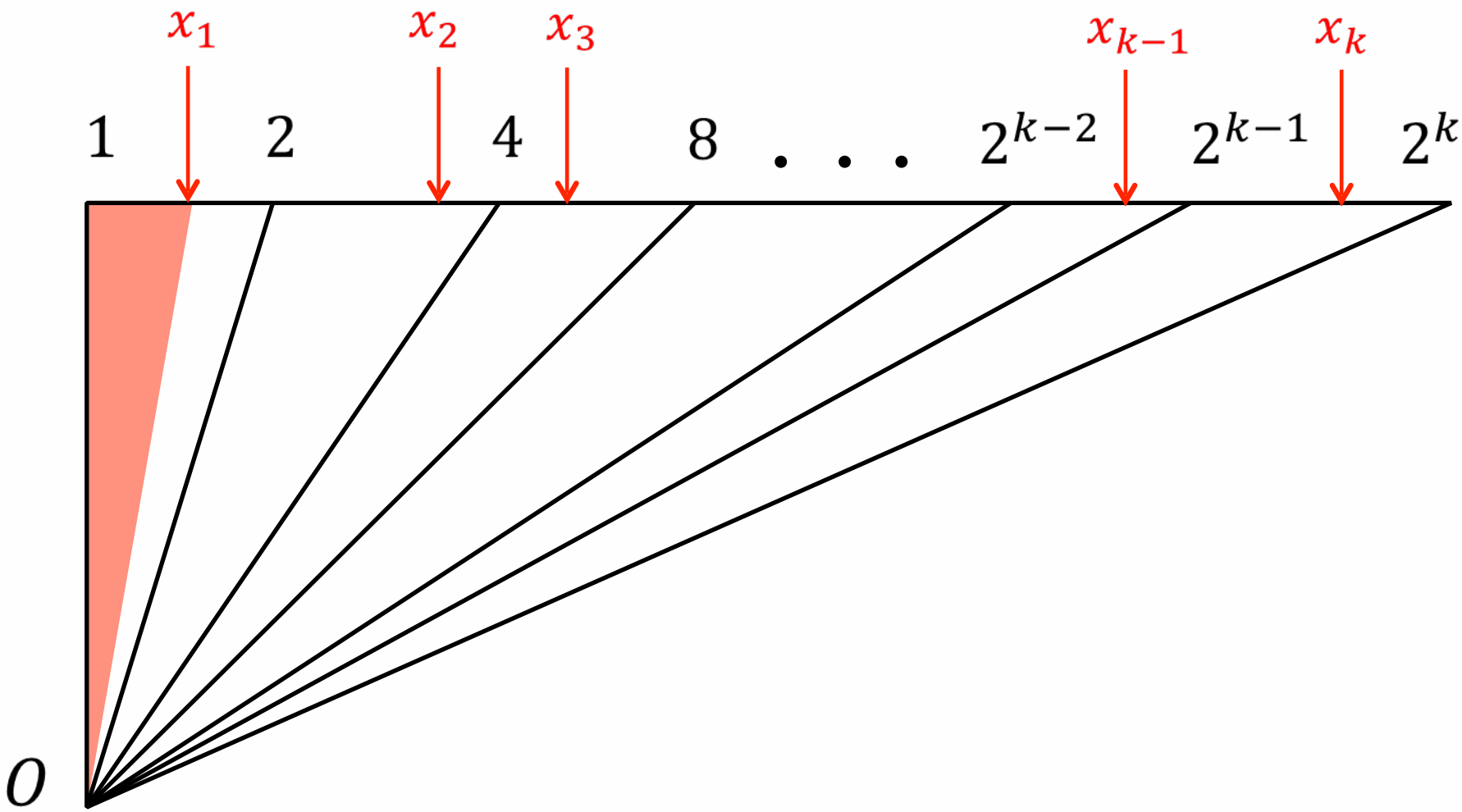
Better idea: randomize in "random stages"



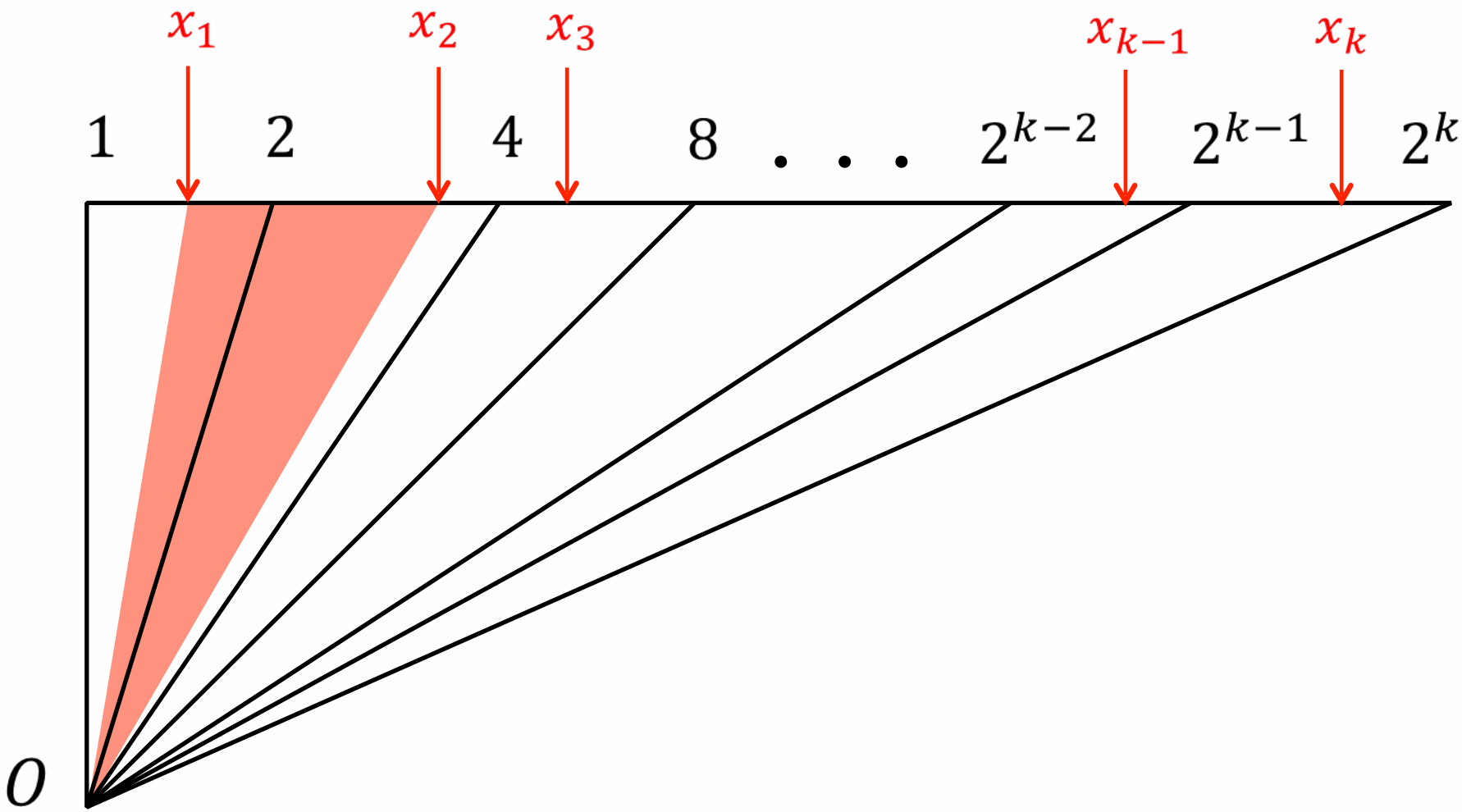
Better idea: randomize in "random stages"



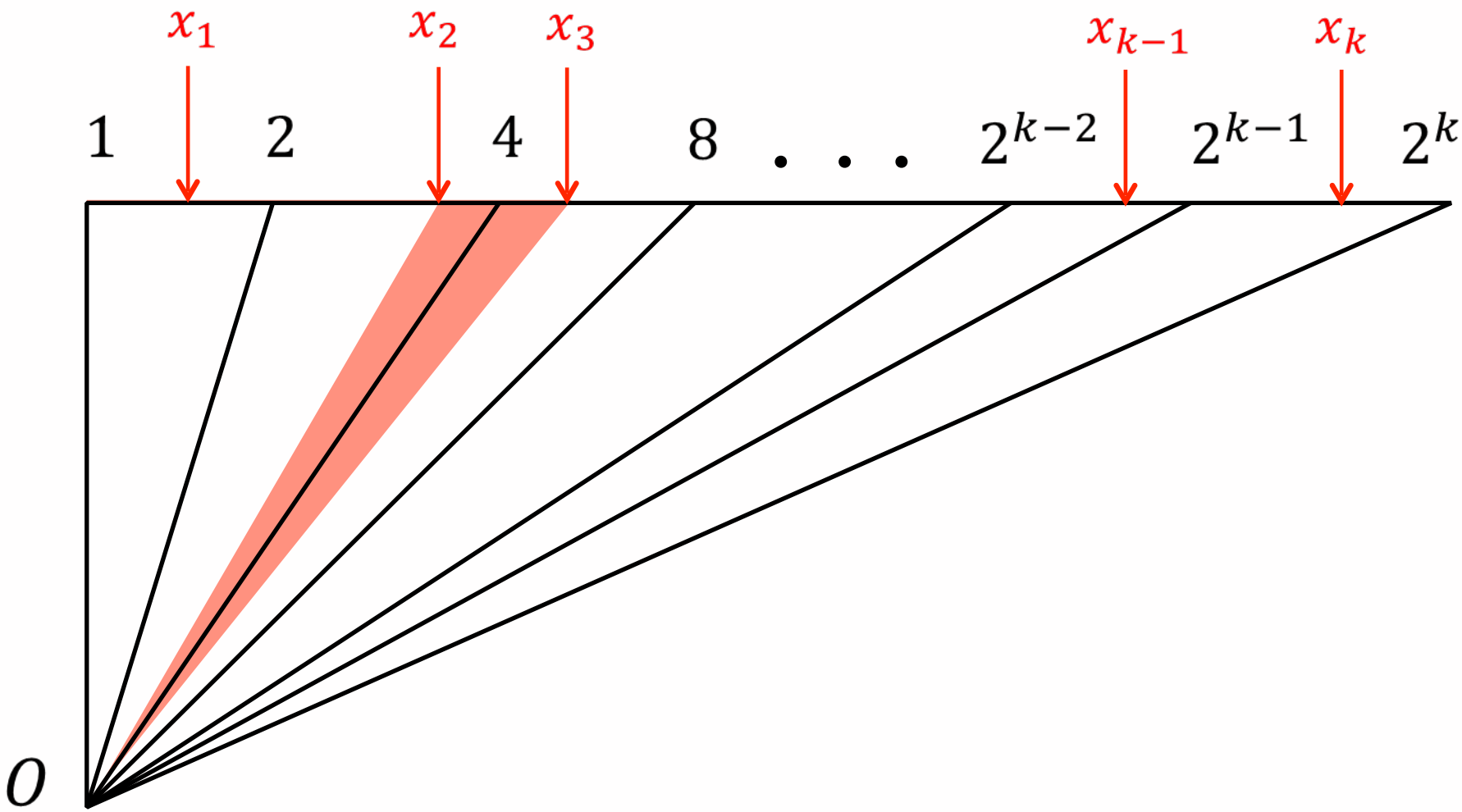
Better idea: randomize in "random stages"



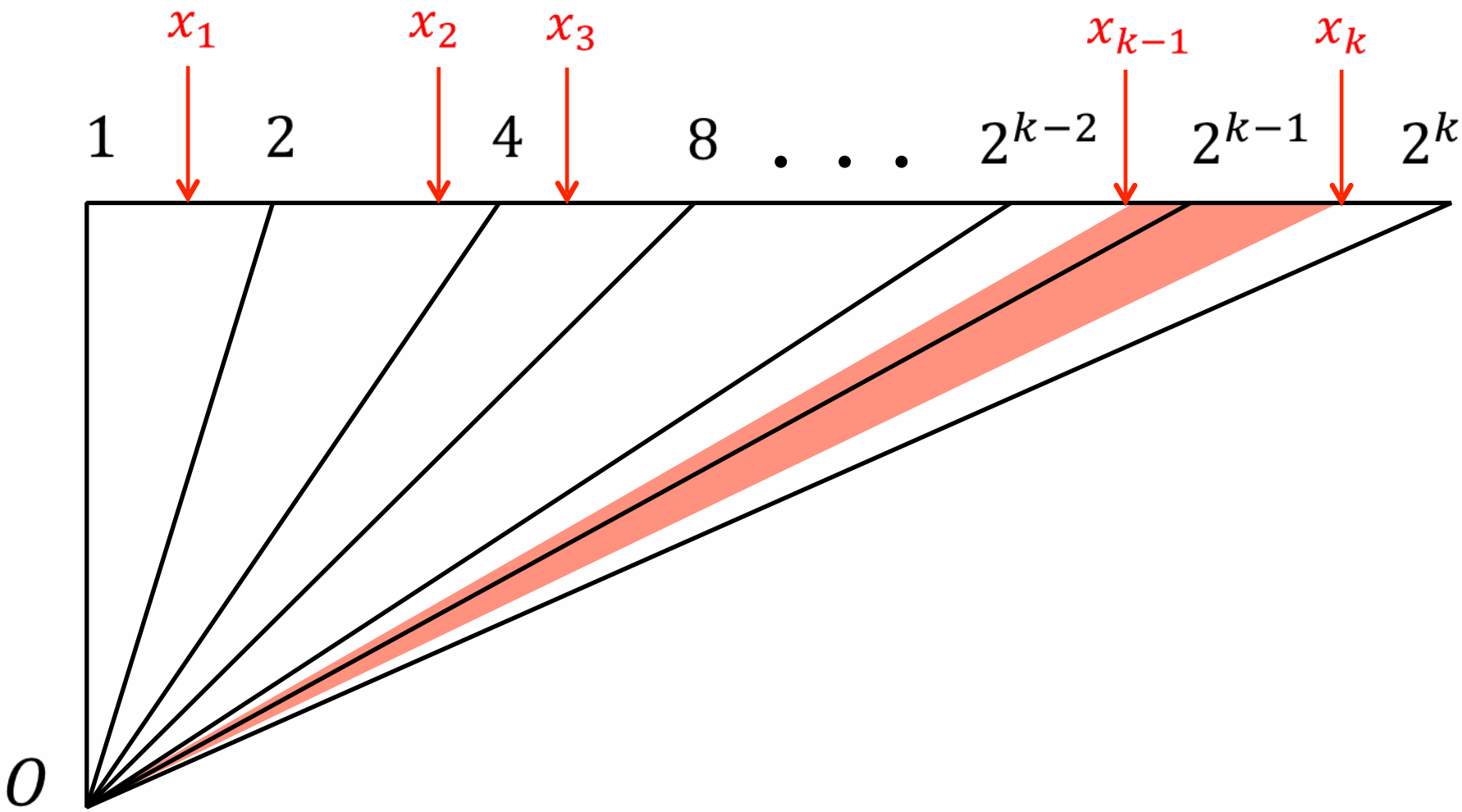
Better idea: randomize in "random stages"



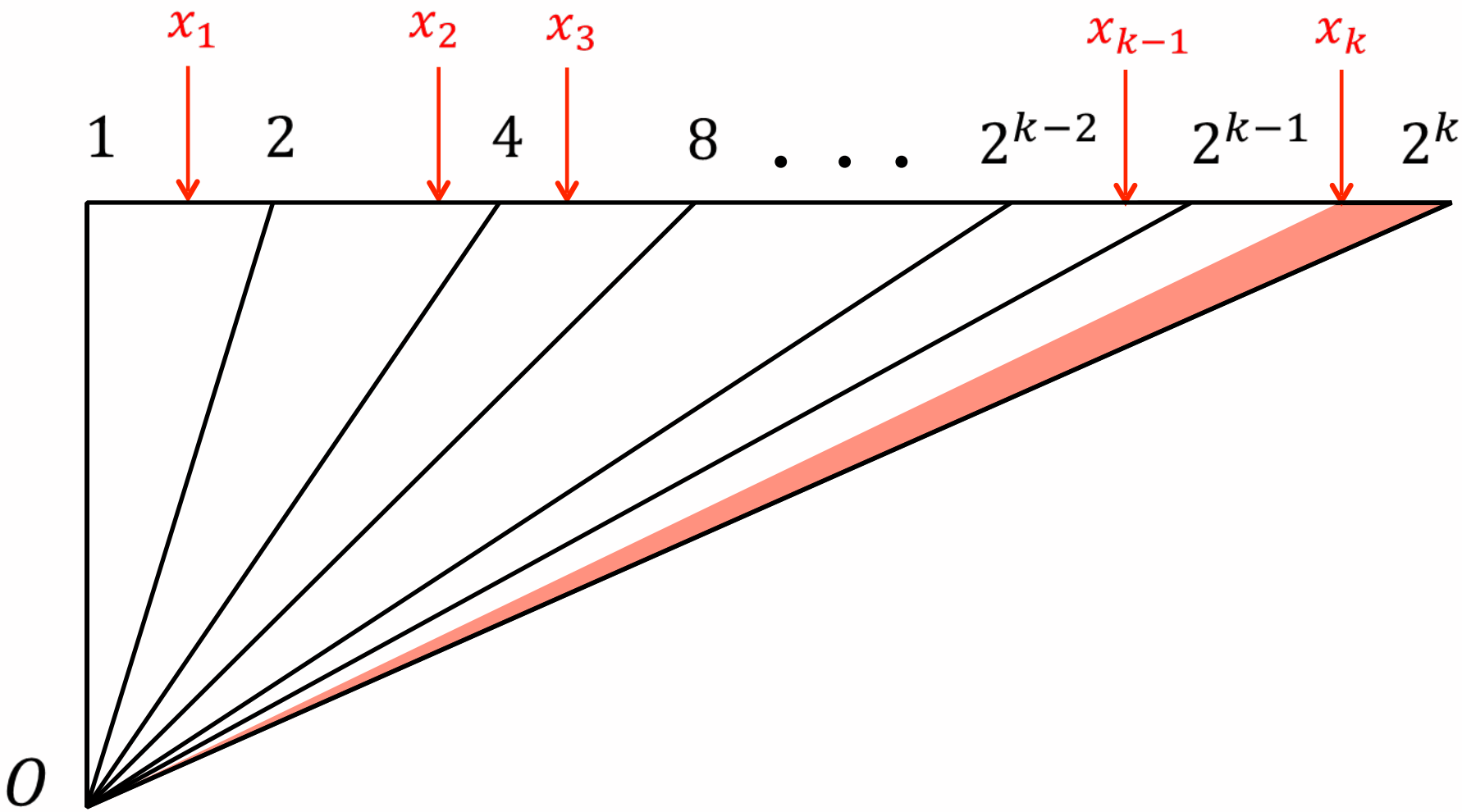
Better idea: randomize in "random stages"



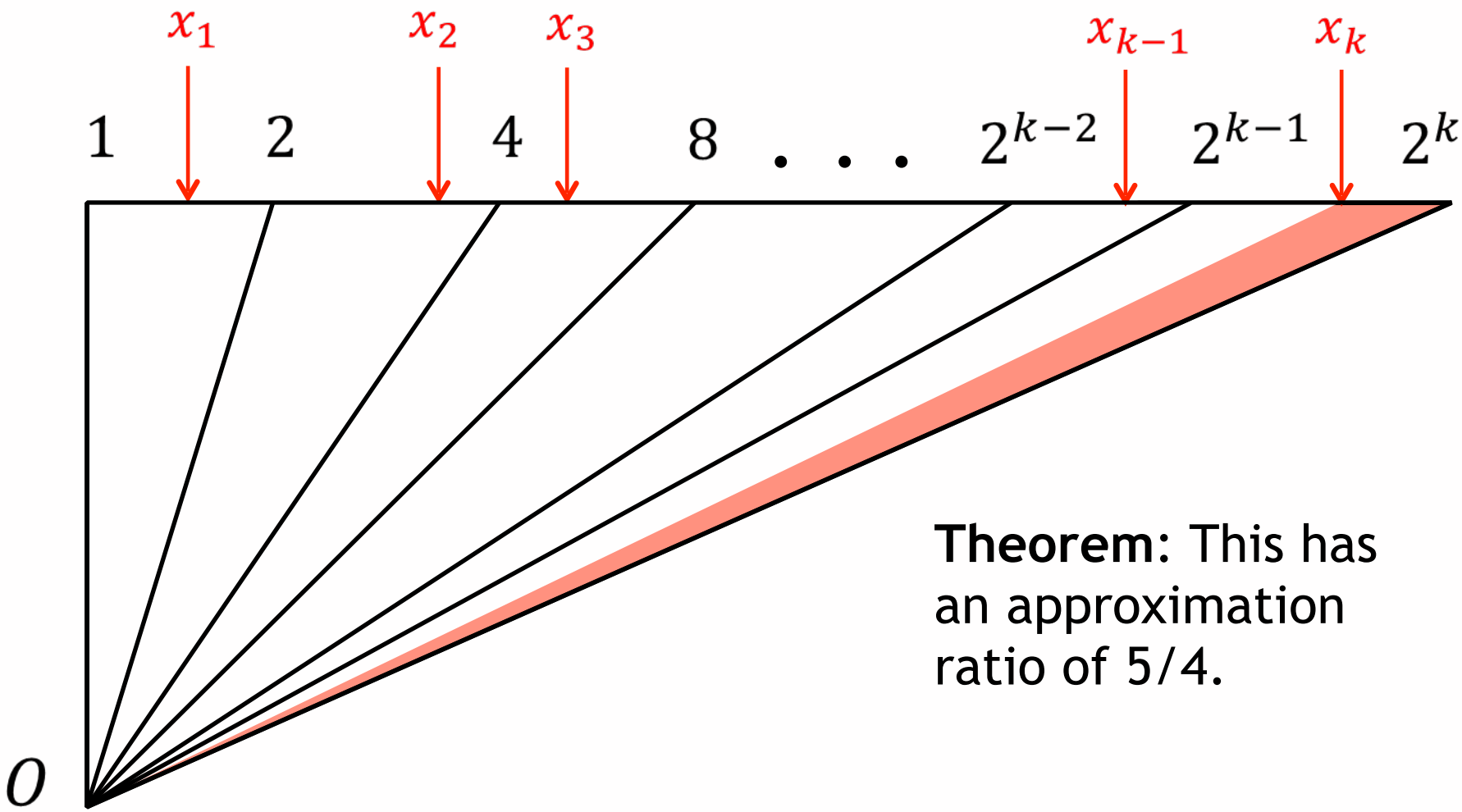
Better idea: randomize in "random stages"



Better idea: randomize in "random stages"

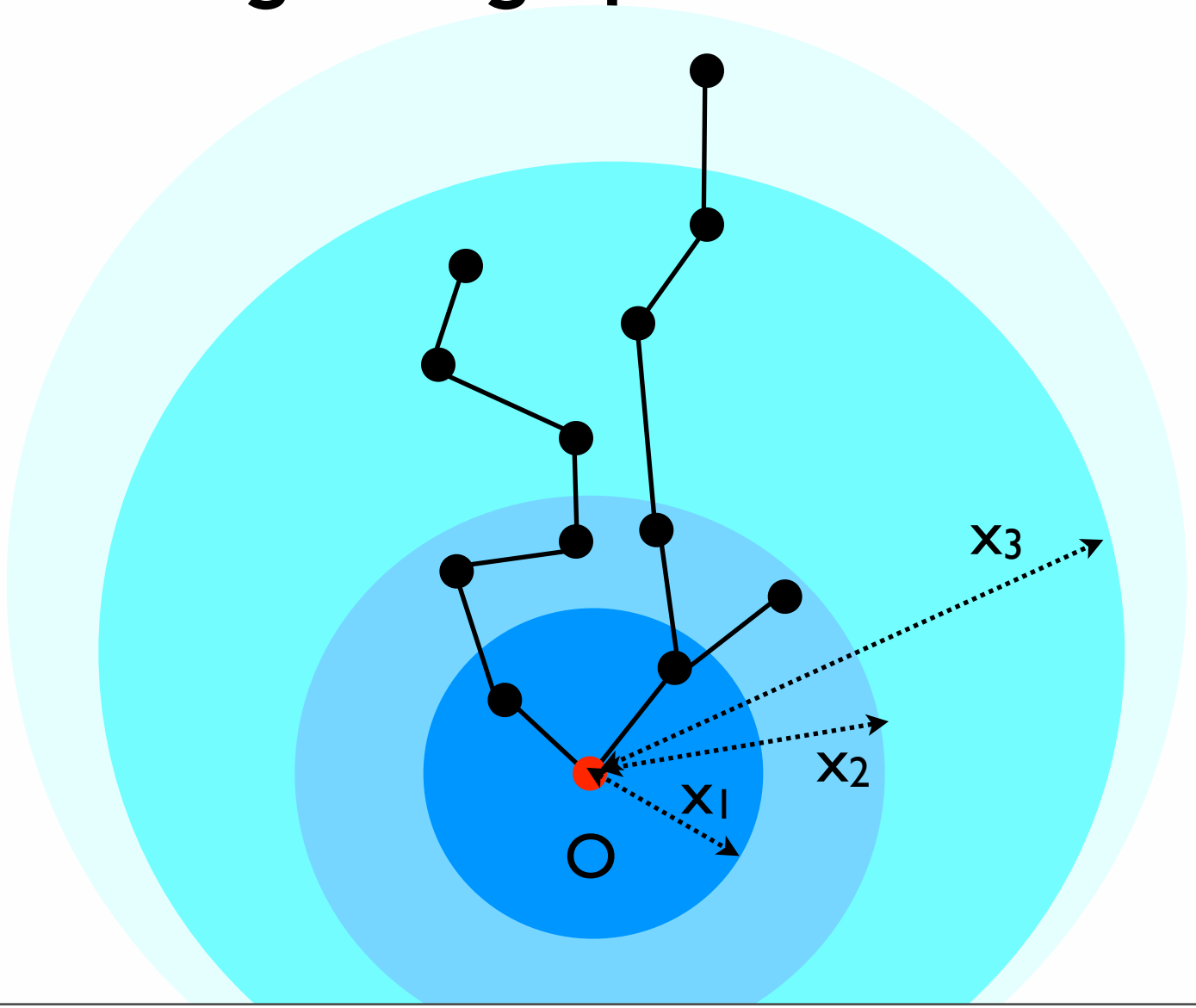


Better idea: randomize in "random stages"



Theorem: This has an approximation ratio of 5/4.

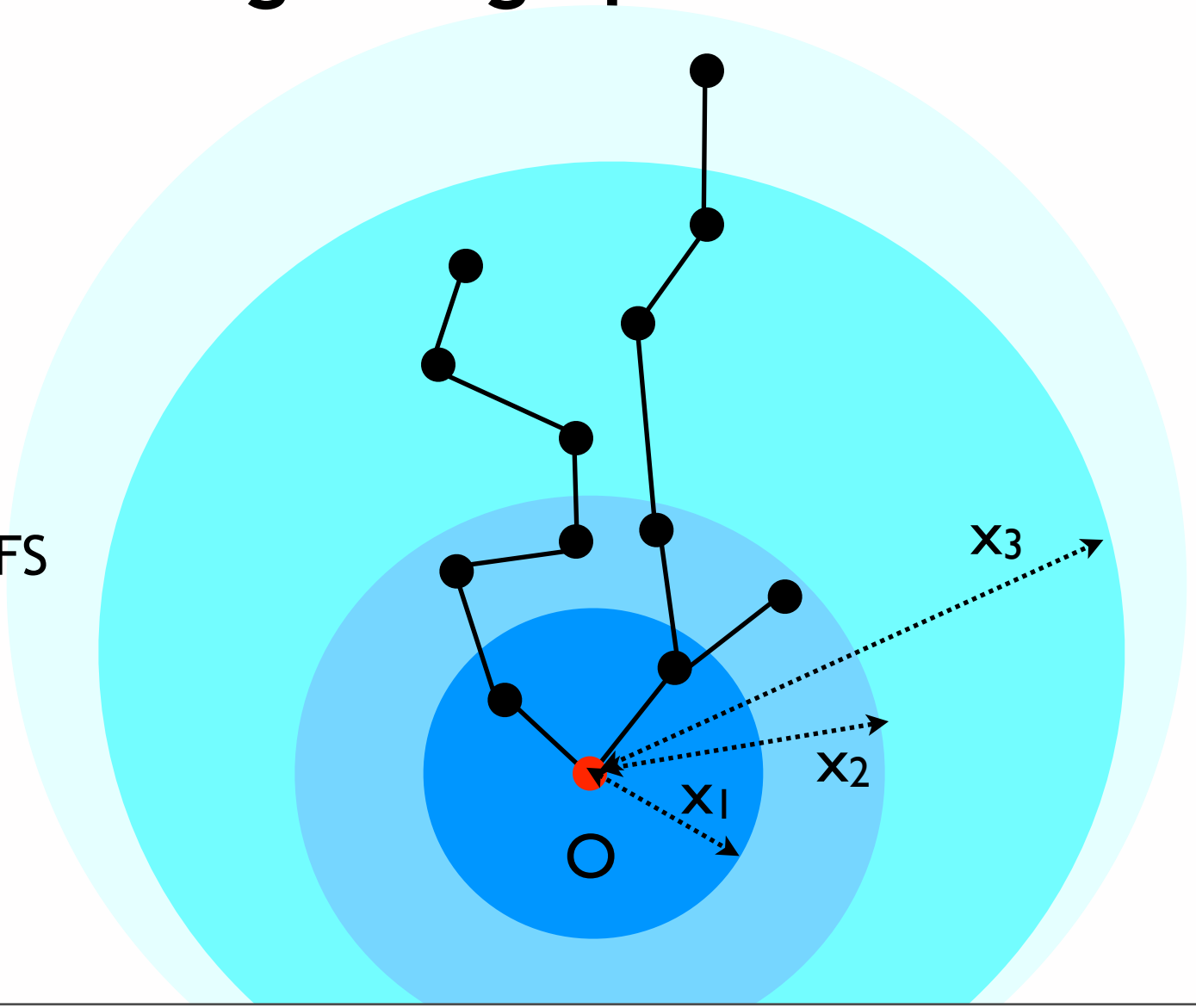
Extension to trees and unweighted graphs



Extension to trees and unweighted graphs

key idea:

Randomized DFS



A tight bound as function of n

A tight bound as function of n

On a star graph with n edges $\rho \leq (n+1)/2$,
and this bound is tight

In particular, $\rho = (n+1)/2$ iff all edges have same length

A tight bound as function of n

On a star graph with n edges $\rho \leq (n+1)/2$,
and this bound is tight

In particular, $\rho = (n+1)/2$ iff all edges have same length

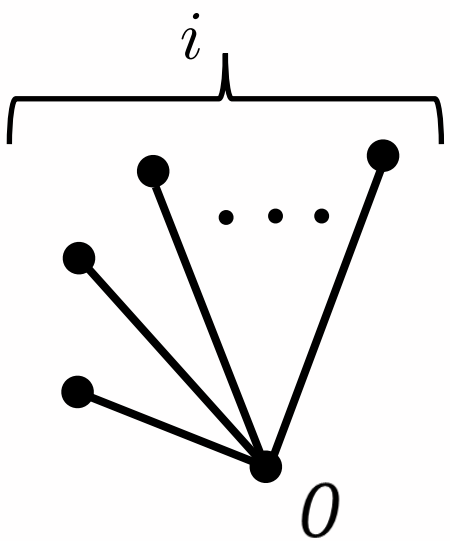
Proof idea: Inductively defined mixed strategy

A tight bound as function of n

On a star graph with n edges $\rho \leq (n+1)/2$,
and this bound is tight

In particular, $\rho = (n+1)/2$ iff all edges have same length

Proof idea: Inductively defined mixed strategy

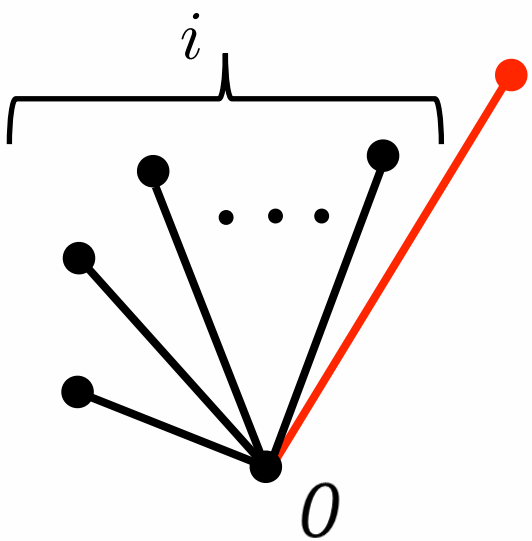


A tight bound as function of n

On a star graph with n edges $\rho \leq (n+1)/2$,
and this bound is tight

In particular, $\rho = (n+1)/2$ iff all edges have same length

Proof idea: Inductively defined mixed strategy

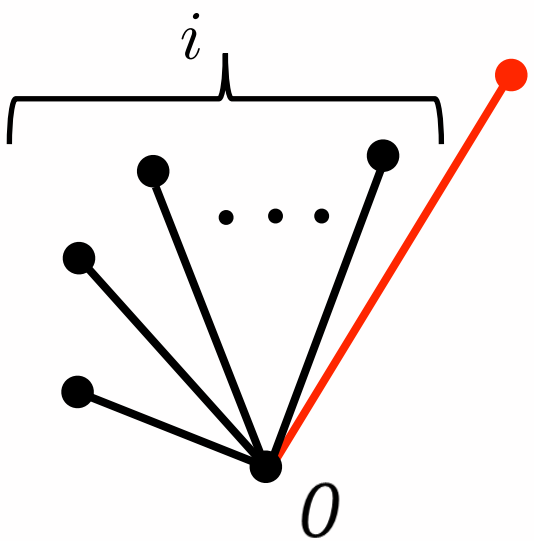


A tight bound as function of n

On a star graph with n edges $\rho \leq (n+1)/2$,
and this bound is tight

In particular, $\rho = (n+1)/2$ iff all edges have same length

Proof idea: Inductively defined mixed strategy



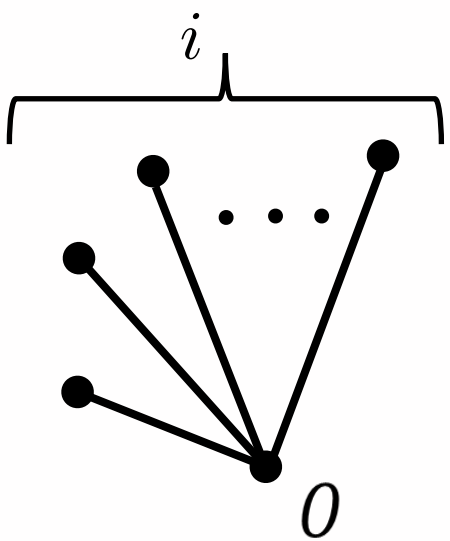
strategy 1: leave it at the end

A tight bound as function of n

On a star graph with n edges $\rho \leq (n+1)/2$,
and this bound is tight

In particular, $\rho = (n+1)/2$ iff all edges have same length

Proof idea: Inductively defined mixed strategy



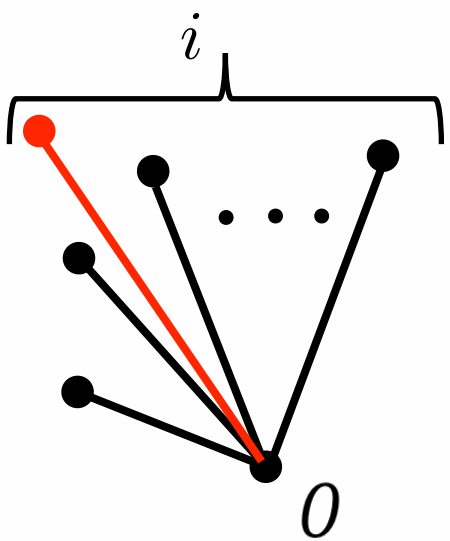
strategy 1: leave it at the end

A tight bound as function of n

On a star graph with n edges $\rho \leq (n+1)/2$,
and this bound is tight

In particular, $\rho = (n+1)/2$ iff all edges have same length

Proof idea: Inductively defined mixed strategy



strategy 1: leave it at the end

strategy 2 : place among the i edges u.a.r according to their total length

Outlook

Outlook

Continuous setting: Easy for deterministic search, quite hard for randomized search

Outlook

Continuous setting: Easy for deterministic search, quite hard for randomized search

What if the graph is revealed on-line?

Outlook

Continuous setting: Easy for deterministic search, quite hard for randomized search

What if the graph is revealed on-line?

Long-term objective: Algorithmic search theory

Outlook

Continuous setting: Easy for deterministic search, quite hard for randomized search

What if the graph is revealed on-line?

Long-term objective: Algorithmic search theory

Thank you!